Řízení motorů pomocí procesoru

Motor Control by the CPU

Bc. Lukáš Marcaník

Diplomová práce
2010

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2009/2010

ZADÁNÍ DIPLOMOVÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Lukáš MARCANÍK**
Osobní číslo:	**A08828**
Studijní program:	**N 3902 Inženýrská informatika**
Studijní obor:	**Počítačové a komunikační systémy**
Téma práce:	**Řízení motorů pomocí procesoru**

Zásady pro vypracování:

1. **Seznamte se s problematikou řízení stejnosměrných elektrických motorů a s problematikou řízení vzducholodi.**
2. **Navrhněte elektrický modul s mikroprocesorem, který na základě dat dodávaných procesoru ovládá dva motory a jedno servo.**
3. **Vytvořte příslušný řídicí algoritmus.**
4. **Realizujte navržený modul fyzicky a odzkoušejte jeho vlastnosti.**

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. LADMAN, Josef. Elektronické konstrukce pro začátečníky . 1. vyd. Praha : BEN technická literatura, 2002. 144 s. ISBN 978-80-7300-015-8.
2. BASTIAN, Peter. Praktická elektrotechnika . Karel Radil. 2. vyd. Brno : Europa Sobotáles, 2006. 303 s. ISBN 80-86706-15-X.
3. HÄBERLE, Heinz . Průmyslová elektronika a informační technologie. Jiří Handlíř. 1. Auflage. Praha : Europa Sobotáles, 2003. 719 s.ISBN 80-86706-04-4.
4. ROZEHNAL, Zdeněk. Mikrokontroléry Motorola HC11 . 1. vyd. Praha : BEN – technická literatura, 2001. 191 s. ISBN 80-86056-77-5.
5. VÁňA, Vladimír. Začínáme s mikrokontroléry Motorola HC08 Nitron . 1. vyd. Praha : BEN – technická literatura, 2003. 96 s., CD. ISBN 80-7300-124-1.
6. NOVÁK, Petr. Mobilní roboty – pohony, senzory, řízení. : 1. díl. Praha : BEN – technická literatura, 2005. 256 s. ISBN 80-7300-141-1.
7. CPU08 Central Processor Unit Reference manual. Motorola, 2001
8. VAŠEK, V., VAŠEK, L. Programování mikropočítačů. 1. vyd., VUT Brno, 1990. 97 s.

Vedoucí diplomové práce: **doc. Mgr. Milan Adámek, Ph.D.**
Ústav bezpečnostního inženýrství

Datum zadání diplomové práce: **19. února 2010**

Termín odevzdání diplomové práce: **7. června 2010**

Ve Zlíně dne 19. února 2010

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

prof. Ing. Vladimír Vašek, CSc.
děkan

prof. Ing. Karel Vlček, CSc.
ředitel ústavu

ABSTRAKT

Diplomová práce se zabývá návrhem a praktickou realizací modulu, jenž dokáže řídit směr a otáčky stejnosměrných motorů a serva, které jsou hnacími prvky vzducholodě. Základem řídícího obvodu je mikrokontrolér Freescale HCS08, kterému jsou dodávána data z tlačítek nebo také je schopen si je brát z LED indikace modulu ultrazvukových senzorů. Příslušný řídící algoritmus je tak rozdělen na manuální, ale i automatický režim reagující na překážky v blízkosti senzorů.

Klíčová slova: Freescale, HCS08, mikropočítač, EAGLE, stejnosměrný motor, servo

ABSTRACT

This thesis deals with design and practical realization of the module, which can control the direction and speed of DC motors and servos, which are the key driver of the airship. The basis of control circuit is Freescale HCS08 microcontroller, to which are delivered the dates of the buttons or it is also able to take them from the LED indication of module ultrasonic sensors. Competent control algorithm is divided into the manual, but also into an automatic mode responding to the barriers near the sensors.

Keywords: Freescale, HCS08, microcomputer, EAGLE, DC motor, servomechanism

Tímto bych chtěl poděkovat panu doc. Mgr. Milanu Adámkovi, Ph.D. za odborné vedení mé diplomové práce. Ale také panu Ing. Martinu Pospíšilíkovi za pomoc při návrhu obvodové části a Ing. Michalu Brázdovi za cenné rady při praktické realizaci.

„*Nikdo se moudrým nerodí, nýbrž stává se jím.*“

Lucius Annaeus Seneca

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně …….

podpis diplomanta

OBSAH

ÚVOD

S postupem času čím dál víc narůstá využívání mikroelektroniky, mezi kterou patří i mikropočítače, a to nejen v průmyslu, ale i v domácnostech a mnoha dalších odvětvích. Vyznačují se svou velkou spolehlivostí a kompaktností, proto jsou určeny především pro jednoúčelové aplikace jako je řízení nebo regulace. Jedním z jejich využití může být i řízení motorů a serva, které jsou využity k ovládání vzducholodě za letu.

V teoretické části diplomové práce se zabývám popisem jednotlivých ovládaných prvků, jako jsou stejnosměrné motory a servo. A také seznamuji s možnostmi a vlastnostmi mikropočítačů Freescale HCS08.

V praktické části je zahrnut návrh schémat řídících obvodů s jejich funkčním popisem a vlastní konstrukcí desek plošných spojů. V závěru je nastíněn princip řídících algoritmů a vysvětleno ovládání modulu.

I. TEORETICKÁ ČÁST

1 VZDUCHOLOĎ

Vzducholodě lze rozdělit do dvou skupin.

- Velké pro venkovní provoz
- Malé optimalizované pro vnitřní provoz

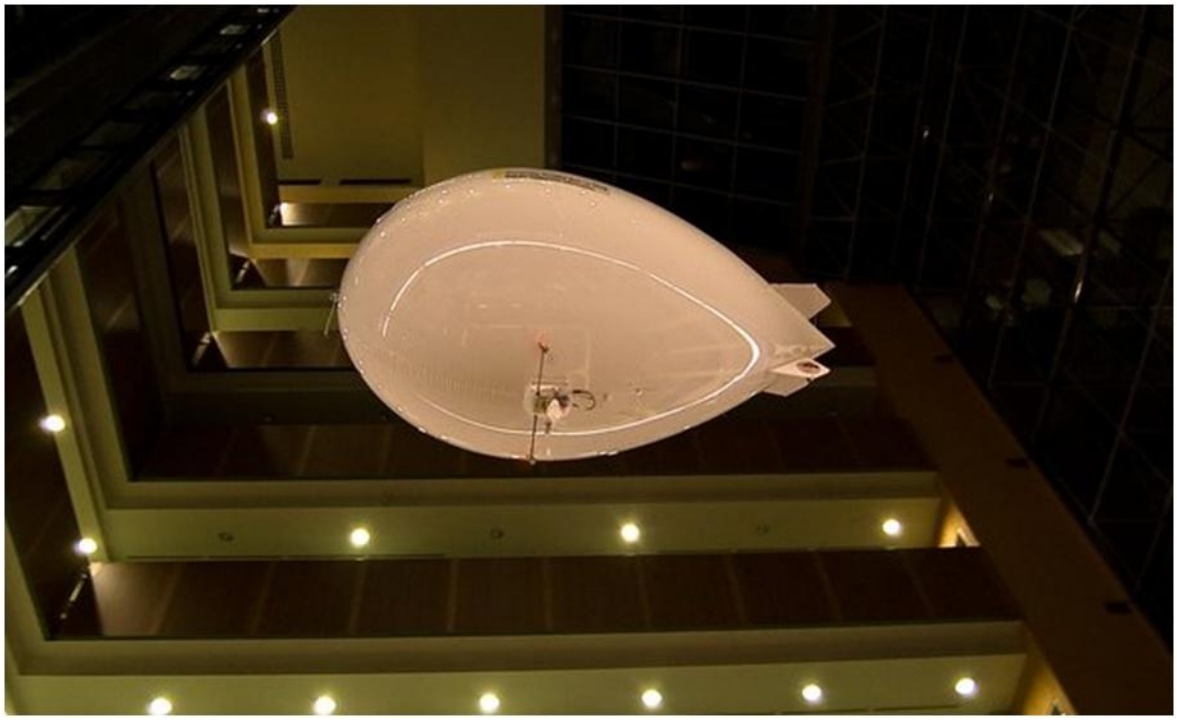
Velké vzducholodě se často využívají k meteorologickému měření v určitých výškách nebo k přepravě a to nejen osob, ale i rozměrných nákladů, které by se komplikovaně transportovali po zemi. Kdežto malé vzducholodě určené pro provoz v uzavřených prostorech a halách se spíše používají jako vznášející se reklama, která přitahuje pozornost. Díky schopnosti, že se vzducholoď dokáže vznášet ve vzduchu jen s minimální spotřebou energie, jí dává další zajímavou možnost, jako třeba nést doplňující monitorovací nebo řídicí systémy.

1.1 Charakteristika vzducholodě

Pro projekt Autonomně se řídící vzducholoď byla vytvořena malá vzducholoď na ovládání pro vnitřní provoz s rezervní nosností pro přídavná zařízení. K pohybu vpřed využívá dva stejnosměrné motorky, které je možné pomocí serva natáčet ve vertikálním směru a ovládat tak stoupání či klesání vzducholodě. Servo je umístěno v gondole pod balónem, kde je i místo pro další zařízení. O horizontální změnu směru se stará ocasní motorek, který musí zvládat otáčení oběma směry. Nosnost vzducholodi byla zvolena jako kompromis mezi jejími rozměry a je tím pádem dosti omezena, proto je třeba dbát na hmotnost přídavných systémů.

Parametry vzducholodě jsou následující:

- Plnící plyn: He
- Nosnost: 0,65 kg (teoretická)
- Napájení: Li-Pol 7,2 V, 2,4 Ah (později spínaný zdroj na 5V a 7V)
- Délka: 2,6 m
- Max. šířka: 1,45 m
- Objem: 2,7 m^3

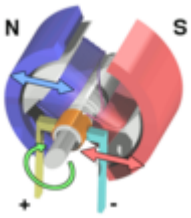


Obr. 1 Vzducholoď na Univerzitě Tomáše Bati [11]

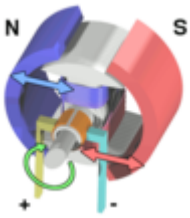
1.2 Stejnosměrný motor

Nejjednodušší motor na stejnosměrný proud má stator, tvořený permanentním magnetem a rotující kotvu ve formě elektromagnetu s dvěma póly. Rotační přepínač zvaný komutátor mění směr elektrického proudu a polaritu magnetického pole procházejícího kotvou dvakrát během každé otáčky. Tím zajistí, že síla působící na póly rotoru má stále stejný směr. V okamžiku přepnutí polarity (mrtvý úhel motoru) udržuje běh tohoto motoru ve správném směru setrvačnosti. Principiálně se tento motor trochu podobá střídavému synchronnímu motoru, kde rotační přepínání směru proudu a jím vytvářeného magnetického pole zajišťuje sama elektrorozvodná síť.

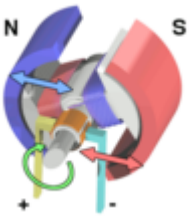
Komutátor zajišťuje, že se v cívce změní směr proudu + a − (− a +) po každém pootočení o 180° (u dvoupólového motoru). Takto dochází ke změně směru indukčních siločar v cívce. [2]



Rotor (kotva) je přes oranžový komutátor připojen ke zdroji stejnosměrného napětí. Stator je tvořen dvěma velkými permanentními magnety.



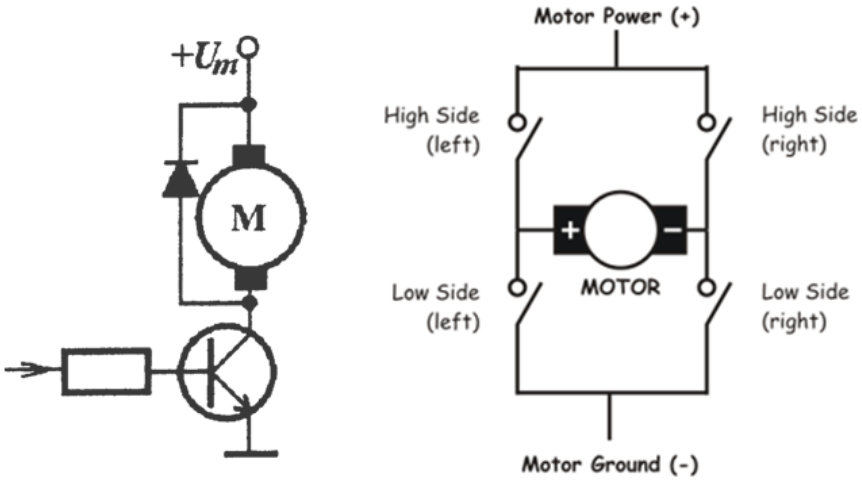
Vzhledem k polaritě statoru a rotoru se souhlasné póly (barvy) odpuzují a rotor se otáčí.



Opačné póly se přitahují, rotor se stále otáčí. V okamžiku, kdy se rotor dostane do vodorovné polohy, dojde na komutátoru k přepnutí polarity magnetického pole rotoru.

1.2.1 Řízení směru otáčení

Jako nejjednodušší způsob spínání stejnosměrného motoru lze použít tranzistor s diodou jako ochranu proti naindukovanému proudu. Tohle zapojení znázorněné na obrázku č. 2 se dá použít pouze k řízení motoru jedním směrem. Pokud je potřeba otáčení oběma směry, je nezbytné použít zapojení jako tzv. full-bridge, které je na obrázku č. 2 v pravé části.



Obr. 2 Spínání motoru a řízení směru otáček [10]

Toto zapojení nazývané H můstek je tvořeno čtveřicí spínacích prvků a princip spočívá v tom, že měníme polaritu napájení tím, že spínáme jednotlivé prvky v určitých kombinacích. V následující tabulce jsou znázorněny jednotlivé stavy můstku. Jediné, čemu se musí předejít je, aby nikdy nenastalo současné sepnutí obou spínačů na levé nebo pravé straně, což by znamenalo zapojení do zkratu.

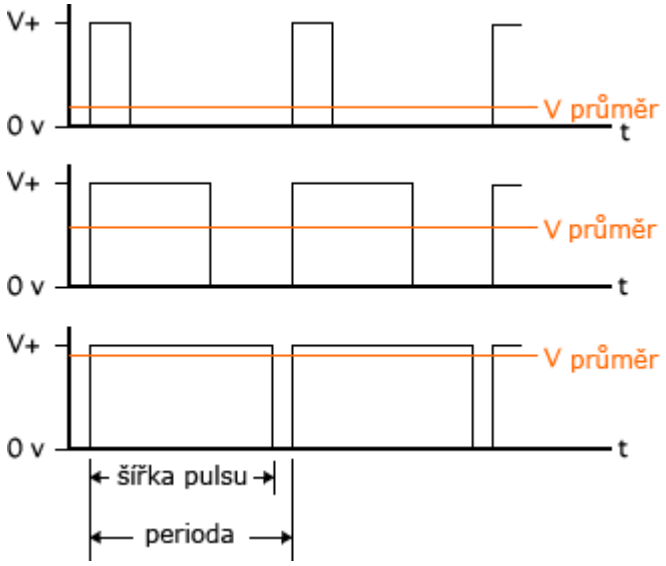
Horní levý spínač	Spodní levý spínač	Horní pravý spínač	Spodní pravý spínač	Motor
0	0	0	0	volně se otáčí
1	0	0	1	vpřed
0	1	1	0	vzad
0	1	0	1	brzda
1	0	1	0	brzda

Tab. 1 Jednotlivé stavy H můstku

1.2.2 Pulsně šířková modulace

Pulsně šířková modulace, neboli PWM (Pulse Width Modulation) je diskrétní modulace pro přenos analogového signálu pomocí dvouhodnotového signálu. Jako dvouhodnotová veličina může být použito například napětí, proud nebo světelný tok, kde signál je přenášen pomocí střídy. Vzhledem ke svým vlastnostem je pulsně šířková modulace často využívána ve výkonové elektronice pro řízení velikosti napětí nebo proudu. Kombinace PWM modulátoru a dolnofrekvenční propusti bývá rovněž využívána jako levná náhrada D/A převodníku.

Přenosový signál, který nese informaci o přenášené hodnotě, může nabývat hodnot zapnuto/vypnuto tj. log.1/log.0. Hodnota přenášeného signálu je v přenosu "zakódována" jako poměr mezi stavy zapnuto/vypnuto. Tomuto poměru se říká střída. Cyklu, kdy dojde k přenosu jedné střídy, se říká perioda. Omezením pro PWM je to, že přenos informace je vždy omezen na relativní vyjádření a to 0 - 100 %, to znamená, že musí být znán poměr mezi skutečnou hodnotou a procentuelním vyjádřením. Časové hodnoty střídy se pohybují v sekundách, v milisekundách pro přesnější řízení. Perioda je vždy součtem doby zapnuto a vypnuto.[3]



Obr. 3 Princip pulsně šířkové modulace

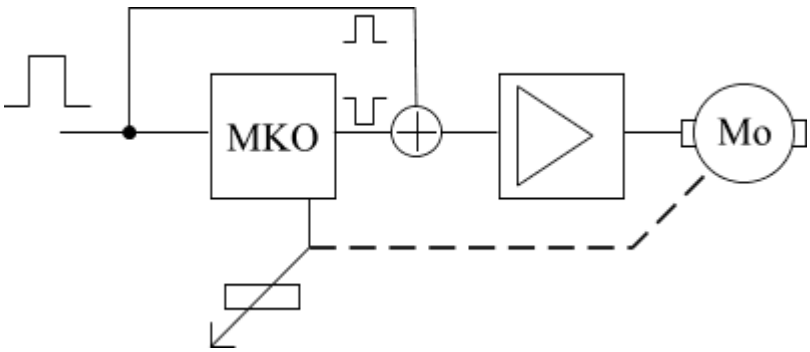
Pulsně šířkovou modulaci lze využít například při řízení otáček. U stejnosměrných motorů jsou otáčky úměrné napájecímu napětí a zátěži. Rychlost tedy můžeme řídit změnou napětí. V případě, kdy není k dispozici regulovatelný zdroj napětí, je ideální použít právě PWM. Princip regulace spočívá v rychlém spínání a vypínání napájení. Díky setrvačnosti motoru a dostatečně vysoké frekvenci spínání, rotor nestačí tyto změny sledovat. Motor se chová, jako kdyby byl napájen napětím o velikosti střední (průměrné) hodnoty, která je dána poměrem doby zapnutí a vypnutí.

1.3 Modelářské servo

Obyčejné levné modelářské servo např. Hitec HS-303 se vyrábí ve velikosti 40x20x36mm, s hmotností 45 g, rychlostí pohybu 60 stupňů za 0,16 s, krouticím momentem 3 kg na páce 1 cm. Přívodní kabel je dlouhý 25 cm, na konci je konektor, který má rozteč dutinek 2,54 mm a jejich velikost takovou, že se jako protikus dá použít špiček pro jumpery. Zapojení konektoru je následující: černý vodič 0V, červený vodič +4,8V, žlutý vodič vstup řídících impulsů. Samozřejmě typů serva je podstatně více, od různých výrobců, různé velikosti a provedení. Servo na obrázku je ukázka jen běžného typu.

Obr. 4 Modelářské servo

Dnešní modelářská serva obsahují elektromotor s převodovkou a řídící elektroniku. Zjednodušené zapojení elektroniky je na blokovém schématu.

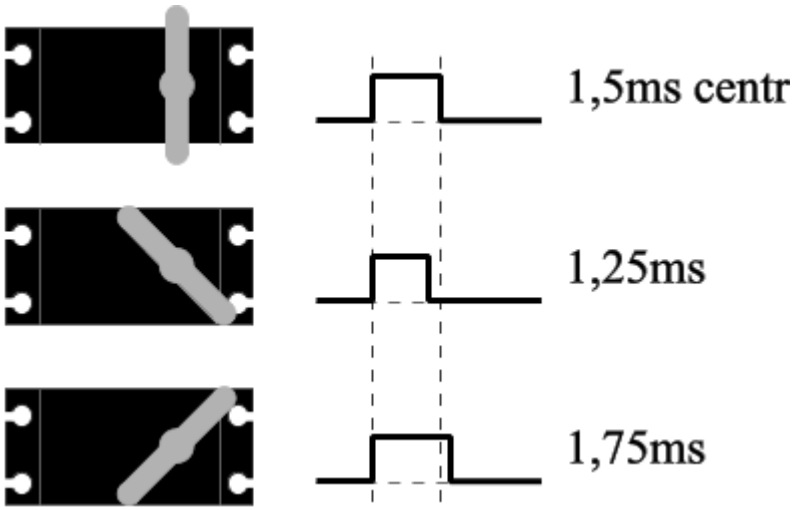


Obr. 5 Blokové schéma servomotoru [4]

Do vstupu přichází řídící impuls, který spustí monostabilní klopný obvod, ten vygeneruje impuls o délce odpovídající momentální poloze serva a opačné polarity než je vstupní řídící impuls. Tyto dva impulsy se porovnají a výsledkem je rozdílový impuls, který po zesílení přes můstkový spínač způsobí roztočení elektromotoru jedním nebo druhým směrem. Elektromotor přes převodovku otáčí výstupní hřídelí a současně i potenciometrem, který působí jako zpětná vazba do monostabilního klopného obvodu. Směr otáčení je takový, že impuls generovaný monostabilním klopným obvodem se svojí

délkou přibližuje délce vstupního řídícího impulsu a až jsou oba impulsy stejně dlouhé, elektromotor se zastaví. Servo dosáhlo polohu, která odpovídá momentálně přijímanému řídícímu impulsu.

Dnes používaná serva pracují s kladnými řídícími impulsy o délce 1-2 ms. Délce impulsu 1,5 ms odpovídá střední poloha serva, 1 ms je jedna a 2 ms druhá krajní poloha. Jeho mechanické provedení může být takové, že servo je schopno pohybu v rozsahu o něco větším než 180 stupňů, ale není to pravidlem. Většina jich má na koncích rozsahu pohybu mechanické blokování, na což je potřeba dávat pozor, protože při dojetí na doraz se výrazně zvýší proud, odebíraný servem a může to skončit tím, že shoří jeho elektronika.

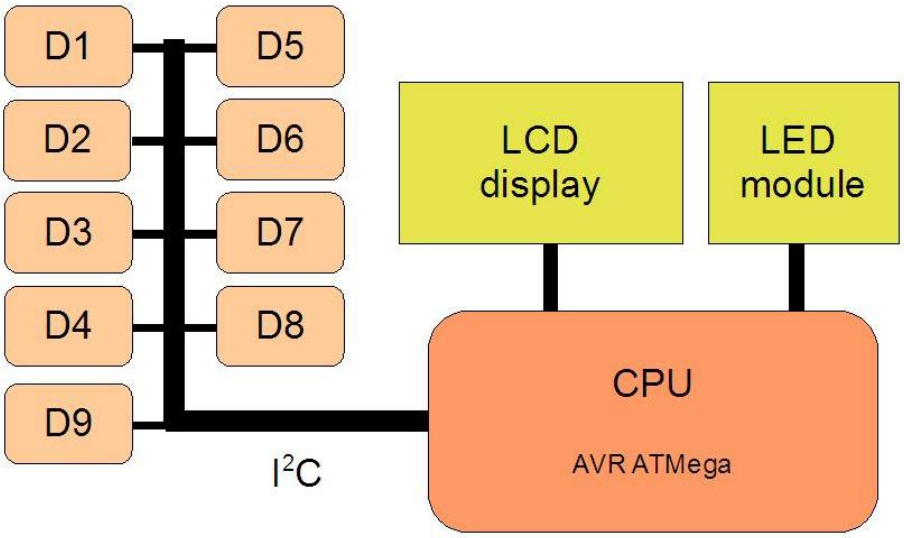


Obr. 6 Vliv délky pulsu na natočení serva [4]

Jak už bylo o něco výše napsáno, dnes používaná serva pracují s kladnými řídícími impulsy o délce 1-2 ms. Tyto impulsy se opakují většinou s frekvencí kolem 50 Hz. Rychlost opakování není kritická, pokud budou řídící impulsy přicházet méně často, pouze se zpomalí rychlost jeho pohybu. Velikost impulsů je rovná velikosti napájecího napětí serva, což je 4,8-6V. [4]

1.4 Modul ultrazvukových senzorů

Modul byl vytvořen jako samostatná část projektu s názvem Autonomně pracující vzducholoď. Konstrukce zařízení byla realizována v praktické části diplomové práce pana Bc. Josefa Voříška. Modul je navržen pro detekci překážek, za pomoci devíti ultrazvukových senzorů s ohledem na nízkou hmotnost kvůli případnému umístění na vzducholodi. Mezi vlastnosti senzorů SRF02 patří měřitelná vzdálenost, která se pohybuje od 16 cm do 6 m, hmotnost 4,6 g a napájecí napětí 5 V s odběrem 4 mA. Komunikace s procesorem probíhá na sběrnici I^2C, jež je charakteristická synchronní sériovou datovou linkou SDA a linkou hodinového signálu SCL. Jako řídící obvod celého modulu je 8mi bitový RISC mikroprocesor řady AVR od firmy Atmel. Výstupem je buď LCD displej udávající přesnou vzdálenost překážky, nebo také panel devíti LED diod indikujících překážku ve vzdálenosti 1 m. Tato vzdálenost je softwarově nastavena v programu a je možnost ji případně měnit. Na následujícím obrázku je vidět blokové schéma.



Obr. 7 Blokové schéma modulu ultrazvukových senzorů[11]

2 SOFTWARE

2.1 EAGLE

2.1.1 Charakteristika návrhového systému

EAGLE je to jeden z nejpoužívanějších, uživatelsky přívětivých a výkonných nástrojů pro návrh desek plošných spojů tzv. DPS. Název EAGLE je zkratka ze slov původního názvu Easily Applicable Graphical Layout Editor. Návrhový systém je složen ze tří hlavních částí a to schematický editor, editor plošného spoje a autorouter. [5]

Obr. 8 Logo programu EAGLE

Vlastnosti:

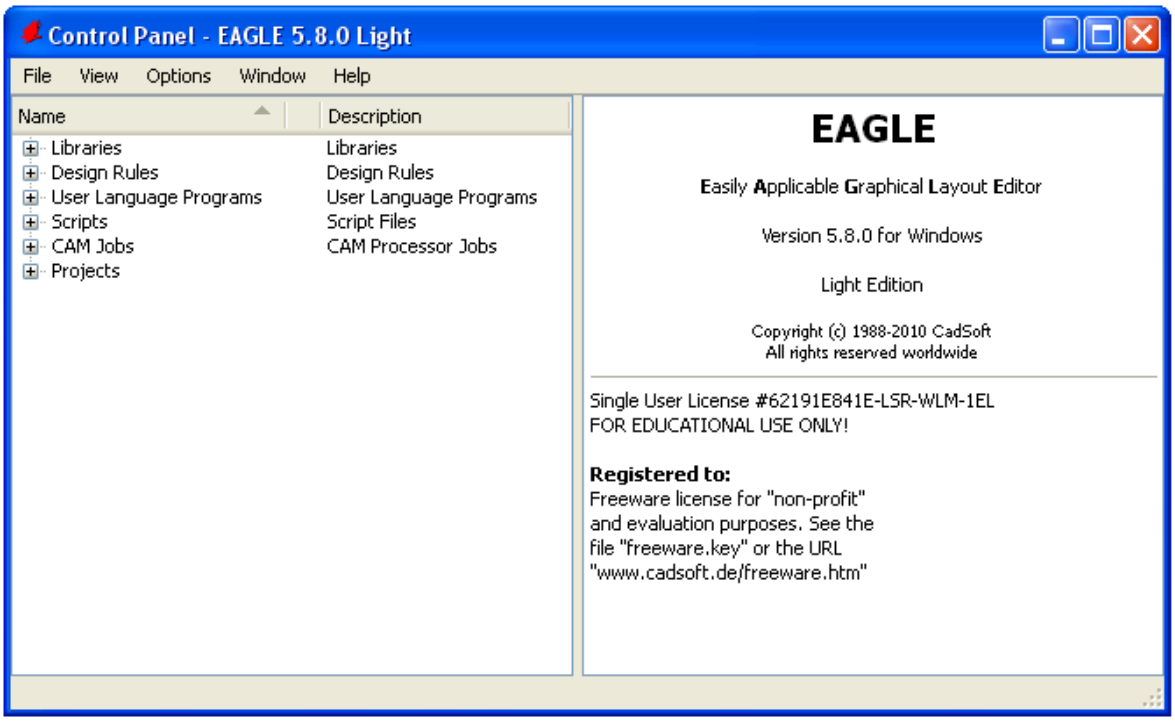
- Jednoduchý přechod mezi editory
- Žádná hardwarová ochrana
- Výkonný uživatelský jazyk
- Integrovaný textový editor
- Snadné vytváření vlastních součástek v editoru knihoven
- Kontrola elektrických návrhových pravidel
- Dostupný pro operační systémy Windows, Linux a MAC
- Není náročný na hardware počítače

2.1.2 Ovládání programu

Po spuštění programu se zobrazí hlavní okno s názvem Control Panel. V panelu je možno provést základní nastavení, přecházet mezi jednotlivými editory a složkami, popřípadě aktivovat knihovny součástek. [5]

Pro naši práci jsou podstatné následující dvě položky menu:

- File – otevírání editorů pro vytvoření nových projektů, schémat, desek a knihoven.
- Options – nastavení cest k souborům, počet vytvářených kopií a nastavení času automatického zálohování souborů, popř. nastavení pracovního prostředí.



Obr. 9 Control panel programu EAGLE

2.1.3 Založení projektu

Při návrhu jakéhokoliv schématu zapojení nebo DPS je vhodné založit nový projekt, který se vytvoří v Control Panelu, kliknutím na kontextové menu File, New a Project. Tímto se v Control Panelu ve složce Projects vytvoří adresář New_Project, který je dobré vhodně pojmenovat a nyní je možno přistoupit ke tvorbě schématu.

2.1.4 Tvorba schématu

U tvorby schématu se postupuje stejným způsobem jako při zakládání projektu. Po zadání v menu File – New – Schematic se zobrazí pracovní prostředí schematického editoru, ve kterém je možno realizovat schéma zapojení.

Veškeré součástky jsou uloženy v knihovnách. Knihovny jsou organizovány podle typu součástek nebo podle výrobců do stromové struktury. Pro použití jsou potřeba aktivovat knihovny do zásobníku, který se následně v editoru schémat vyvolá příkazem ADD.

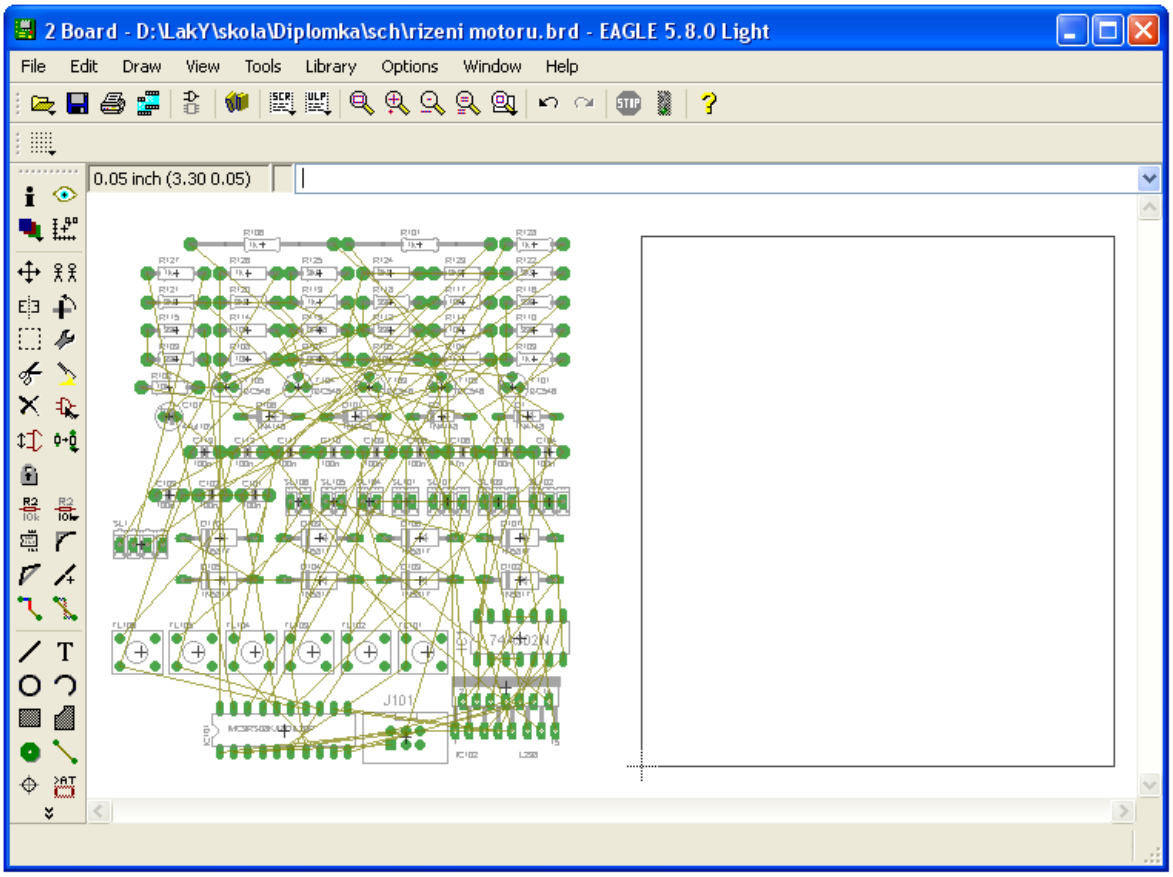
Pomocí příkazu ADD je možno vyhledávat součástky ze zásobníku s knihovnami a umísťovat je na pracovní plochu. Při výběru součástek je nutno dbát na správnou volbu

pouzder nazývaných package, jelikož budou následně použity při tvorbě DPS. Po vyhledání a rozmístění všech součástek na ploše je na řadě zapojení obvodu.

Propojení součástek je jednoduché, avšak vyžaduje cvik. Ikona NET slouží ke kreslení elektrických spojů mezi jednotlivými vývody součástek. První kliknutí je na počáteční vývod první součástky a klikem na vývod součástky druhé se ukončuje elektrický spoj. Po zapojení celého obvodu se schéma doplní o hodnoty součástek a servisní popisky. Nyní je schéma hotové a uloží se standardním způsobem. [5]

2.1.5 Tvorba desky plošného spoje

Po aktivaci ikony BOARD se ihned spustí editor plošných spojů. Pracovní prostředí a způsob ovládání je podobný schematickému editoru. Na pracovní ploše jsou již vidět pouzdra součástek, které obsahuje příslušné schéma. Jednotlivé vývody jsou propojeny tzv. „gumovými spoji“, které napovídají, co má být s čím spojeno. Automaticky se zobrazuje maximální rozměr obrysu desky, který se upraví na požadovanou velikost. Dále se postupuje vhodným rozmístěním součástek na DPS. Jednoznačný návod, jak pouzdra součástek rozmístit, neexistuje. Je to otázka citu a znalostí v oblasti elektroniky. Pouzdra součástek se přesouvají do obrysu desky za pomocí ikony MOVE, přičemž je dobré co nejvíce respektovat vedení gumových spojů. Po rozmístění všech pouzder, nastává další důležitá věc a to propojení všech součástek elektrickými spoji. [5]



Obr. 10 Editor plošného spoje programu EAGLE

K automatickému propojení slouží auto-router, který spustíme ikonou AUTO. Auto-router je potřeba nastavit (počet vrstev spoje, izolační vzdálenost mezi spoji, atd.). Po spuštění auto-routeru je obvod propojen a v drtivé většině případů je potřeba provést korekturu. Pro opravdu kvalitní návrh propojení obvodu s minimem propojek na horní straně je ideálním řešením vytvoření vodivých cest ručně. Tahle metoda je však značně obtížná a vyžaduje zkušenosti.

Ve většině případů se pracuje se dvěma vrstvami propojovacích cest. Hlavní vrstva se nazývá BOTTOM, je zobrazena modrou barvou a znázorňuje spodní vodivou stranu DPS. Červenou barvou se vykresluje vrstva TOP, která představuje propojení na horní straně např. pomocí drátu nebo odporu s nulovou hodnotou. Po dokončení všech spojů je schéma připraveno k vytištění a samotné výrobě desky plošného spoje fotocestou .

2.2 CodeWarrior

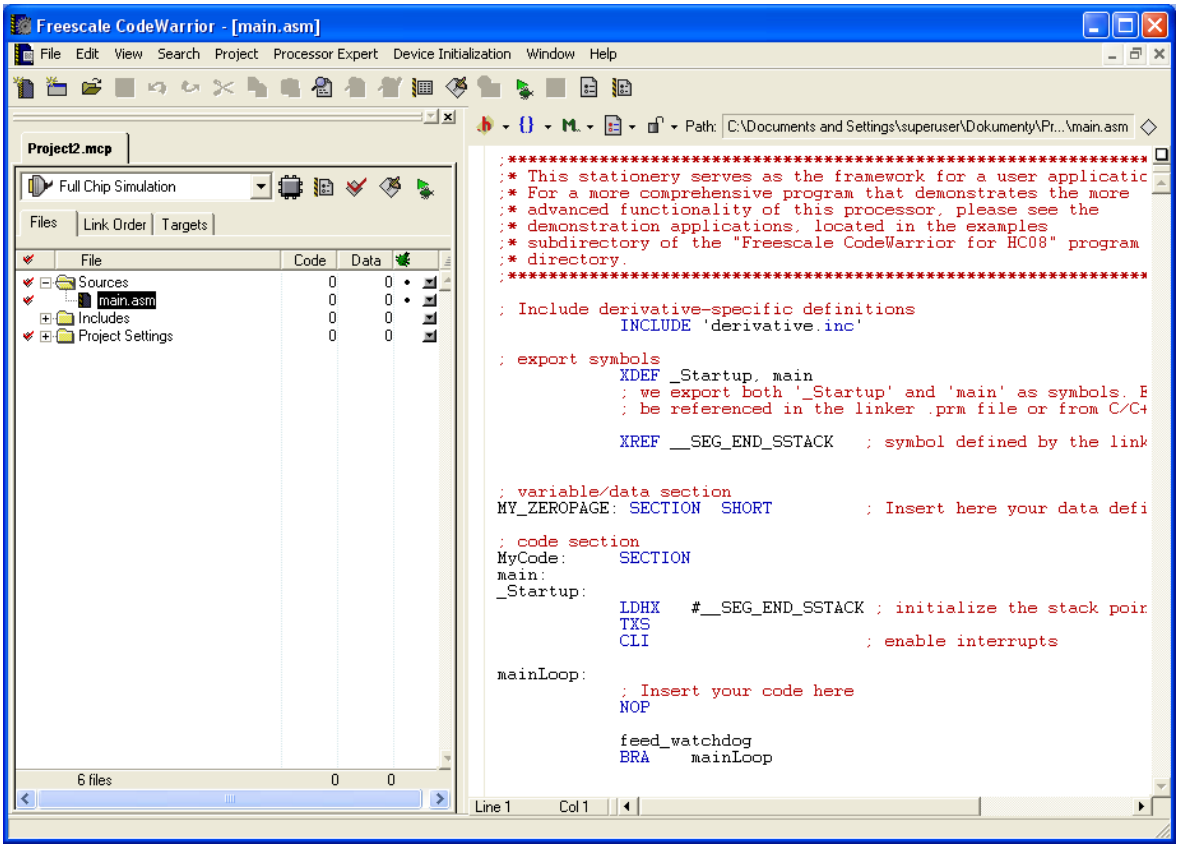
2.2.1 Charakteristika vývojového prostředí

CodeWarrior Development Studio je kompletně integrované vývojové prostředí (IDE), které poskytuje vysoce vizualizovaný a automatický framework k urychlení vývoje i těch nejsložitějších embedded aplikací. Verze nazvaná Special Edition: CodeWarrior for Microcontrollers je určena pro programování mikropočítačů freescale RS08, HC(S)08 a ColdFire V1. V CodeWarrioru je možno programovat v jazyce symbolických adres – assembleru, ale také v jazyce C a C++. Součástí vývojového prostředí je i grafický debugger, jež umožňuje spustit, zastavit, či krokovat vytvořenou aplikaci v simulaci nebo přímo v modulu. Ovládání programu je celkově velmi intuitivní.

2.2.2 Vytváření nového projektu

Ihned po spuštění CodeWarrioru se zobrazí úvodní stránka, jež nabídne uživateli na výběr mezi vytvořením nového projektu nebo například načtením projektu stávajícího.

Pro založení nového projektu je nutno vybrat *Create New Project,* načež vyskočí wizardovské okno s výběrem konkrétního mikroprocesoru a typu připojení jako např. *Full Chip Simulation*, které umožňuje simulovat aplikaci i bez reálného zařízení nebo připojení pomocí P&E Multilink programátoru. Po stisku tlačítka *Next* se zadává název projektu s cestou umístění a výběr programovacího jazyka. V dalších krocích se pokračuje jen tehdy, pokud je potřeba vložit do projektu knihovny například pro ovládání displeje. K úspěšnému vytvoření projektu už stačí jen kliknutí na tlačítko *Finish* a objeví se vývojové prostředí s prázdným projektem.



Obr. 11 Screenshot vývojového prostředí CodeWarrior

2.2.3 Ovládání programu

V levé části vývojového prostředí je zobrazena adresářová struktura projektu včetně všech souborů, jež projekt obsahuje. Ve složce *Sources* se nachází soubor *main.asm* (popřípadě *main.c*), který obsahuje hlavní zdrojový kód. Soubor *main* má již předdefinovanou strukturu a je určen pro tvorbu vlastního programu pro mikropočítač. Pro tvorbu programu je předpokládána alespoň základní znalost vybraného programovacího jazyku.

Po napsání programu je nutno program zkompilovat (Compile - CTRL + F7), sestavit (Make - F7) a odladit (Debug - F5). Tím je spuštěn debugger, který slouží k odladění aplikace buď v simulaci nebo přímo na reálném modulu.

3 MIKROPOČÍTAČ

3.1 Jednočipový mikropočítač

Koncem 70. let se ve vývoji mikroprocesorů objevují i první monolitické mikropočítače, které pak dále představují samostatnou větev vývoje. Tyhle jednočipové mikropočítače jinak též označované jako mikrokontroléry, jsou integrované obvody, které v sobě zahrnují zpravidla vše potřebné k tomu, aby mohly obsáhnout celou aplikaci, aniž by potřebovaly další podpůrné obvody.

Mezi první patřil například osmibitový obvod 8048 firmy Intel, nebo mikropočítač 6801 firmy Motorola. Pokud se porovná vývoj v oblasti procesorů pro osobní počítač a vývoj v oblasti jednočipových mikropočítačů, tak rozvoj osobních počítačů je velmi rychlý a každým rokem dochází k výraznému nárůstu výkonu a složitosti obvodů. Kdežto v oblasti mikrokontrolérů není tento trend vývoje tak výrazný. Velké množství mikropočítačů jsou jen zdokonalenými verzemi obvodů, které byly navrženy často před více než 20 lety. Vývoj se zaměřuje především na periferie umístěné na čipu a zvýšení spolehlivosti. Výrazným krokem vpřed je také použití vnitřních pamětí typu FLASH místo dříve používaných ROM nebo EEPROM.

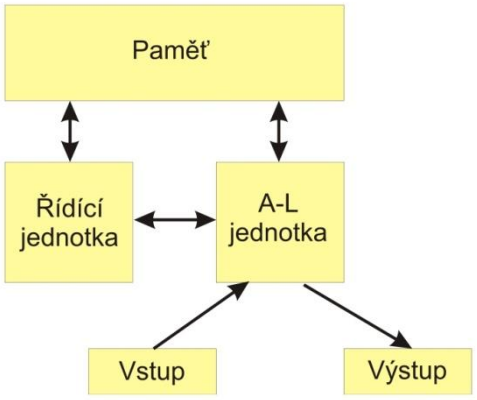
Hlavním rozdílem mezi mikroprocesorem a jednočipovým mikropočítačem je ten, že mikropočítač obsahuje na jediném čipu kromě mikroprocesoru zastoupeného řadičem, aritmetickologickou jednotkou a pamětí, také generátor hodinového signálu a vstupně výstupní brány (periferie), které umožňují alespoň v malé míře samostatnou činnost.

Mikrokontroléry můžeme dělit podle různých kritérií jako je architektura nebo velikosti instrukční sady.

3.1.1 Architektury mikroprocesoru

Von Neumannova architektura

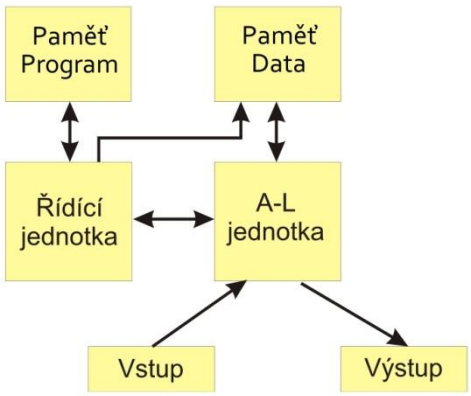
Von Neumannova architektura je architektura, pro kterou je typická společná paměť pro data i program. Toto uspořádání má výhody v tom, že není potřeba rozlišovat instrukce pro přístup k paměti dat a paměti programu, což vede k zjednodušení čipu. Z čehož plyne, že je použita jen jedna datová sběrnice, která kromě toho, že umožňuje použití externích pamětí tak má tu nevýhodu, že přenos obou typů dat po jedné sběrnici je pomalejší.



Obr. 12 Von Neumannova architektura počítače

Harvardská architektura

Harvardská architektura je typická oddělením paměti programu a paměti dat. Hlavní nevýhodou je větší technologická náročnost daná nutností vytvořit dvě sběrnice. Avšak naproti tomu je možnost použití různé šířky programové a datové sběrnice, ale také výhoda čtení instrukcí a potřebných dat v jeden okamžik.



Obr. 13 Harvardská architektura počítače

3.1.2 Instrukční sady mikroprocesoru

CISC

CISC označuje procesor se „složitým instrukčním souborem“. Procesor podporuje mnoho formátů a druhů instrukcí. Na jednu stranu to znamená úsporu místa v programové paměti (vyšší hustotu kódu), na druhé straně to však znamená komplikovanější dekodér instrukcí ve vlastním mikrokontroléru a pomalejší zpracování instrukcí. [1]

RISC

RISC označuje procesor s redukovaným instrukčním souborem. Základní myšlenkou je omezení počtu a zjednodušení kódování instrukcí, což vede ke zjednodušení instrukčního dekodéru. Hlavní výhodou tohoto přístupu je rychlost a jednoduchost. Nevýhodou je, že pro zakódování instrukce je potřeba více místa, někdy se musí použít dvě instrukce místo jedné, takže klesá hustota kódu. [1]

3.2 Programování mikropočítačů

3.2.1 Jazyky symbolických adres

Jazyky symbolických adres patří mezi nejstarší programovací jazyky, se kterými se setkáváme již u počítačů první generace. Dříve než se u počítačů objevily, programovalo se přímo ve strojovém kódu, tj. většinou v binární, oktalové nebo hexadecimální reprezentaci instrukcí počítače. Tento způsob programování byl velmi obtížný a nepřehledný a vyhovoval pouze v úplných začátcích, kdy programy byly poměrně krátké a jednoduché.

S rozvojem techniky však bylo třeba najít způsob, který by programování zjednodušil a zrychlil. Vznikly tak první jazyky symbolických adres. U nich zavádíme symbolické označování objektů s tím, že vazbu symbolů na jejich číselné vyjádření nebude provádět programátor, ale speciální program - překladač (assembler). Symboly jsou nejčastěji tvořeny mnemotechnickými zkratkami slov z běžného jazyka, nejčastěji angličtiny. Program se tak stává čitelnější a přehlednější.

Možnosti jazyků symbolických adres byly dále rozšířeny doplněním nových prostředků (např. makrojazyk, makra). Makrojazyk je založen na možnosti pojmenovat celé

posloupnosti instrukcí. Bez makrojazyka bylo možné pojmenovat pouze objekty v instrukcích.

Dalším krokem k tomu, aby se zjednodušila práce programátora, je zbavit jej závislosti na instrukční sadě mikropočítače, se kterou je úzce spjat při programování v jazyce symbolických adres. Při programování mikropočítačů se tak objevují programovací jazyky používané při práci s velkými počítači, jako jsou jazyk C, C++, Pascal, Delphi, apod. Přesto však jazyky symbolických adres nezanikly. Je to způsobeno tím, že některé úlohy nelze ve vyšších programovacích jazycích řešit. Při programování v jazyce symbolických adres se totiž dostáváme do úzkého styku se systémem a s hardwarem. Dalším důvodem je také to, že pokud je kód programu napsán efektivně, může být program v jazyce symbolických adres rychlejší než stejný program napsaný ve vyšším programovacím jazyce.

Při programování v jazyce symbolických adres vzniká jeden zásadní problém a to, že existuje velké množství různých assemblerů. Jednak se liší instrukční soubory mikropočítačů různých výrobců, ale velmi často také existují rozdíly i mezi assemblery určenými pro stejný typ mikropočítače. Také je třeba mít na paměti, že programy napsané pro určitý typ mikroprocesoru jsou většinou nepřenosné. To znamená, že bez úprav je nelze použít pro jiný typ mikropočítače, a to ani tehdy, jedná-li se o mikropočítač pocházející ze stejné typové řady od stejného výrobce. U HC(S)08 se například jedná o názvy jednotlivých registrů periferií, rozsah paměti a periferie samotné. [1]

3.2.2 Programování v jazyce C

Pokud potřebujeme vytvořit jednoduchý program pro mikroprocesor, je použití assembleru ještě únosné. Avšak s rozvojem schopností malých počítačů potřebují k jejich využití konstruktéři vytvářet programy poměrně rozsáhlé a složité a jejich tvorba v assembleru se stává již neúnosná. Kód postupně začíná být nepřehledný, programování trvá dlouho a to výsledný program prodražuje. Proto je použití některého vyššího jazyka nezbytnost.

Vyšší jazyk se označuje programovací jazyk, s vyšší úrovní abstrakce než je assembler. Jazyk, ve kterém nemusíme program zapisovat jako posloupnost instrukcí procesoru, ale můžeme použít abstraktnější konstrukce jako např. c = a + b.

Dnešní mikroprocesory mají dostatek výpočetního výkonu, aby si mohly dovolit vykonat některé instrukce, které překladač generuje navíc. Proto byly pro jednočipové mikropočítače vytvořeny překladače z vyšších programovacích jazyků. Velké obliby dosáhl zejména jazyk C, což je dané tím, že má nejenom vlastnosti, které očekáváme od vyšších programovacích jazyků, ale i vlastnosti očekávané spíše u assemblerů. Z vyšších programovacích jazyků má jazyk C "nejblíže" k hardware. Proto se i u velkých počítačů používá při vytváření operačních systémů.

Programování mikropočítačů v jazyce C je v podstatě stejné jako vytváření programu pro osobní počítač (PC) jen s malými odlišnostmi. Při psaní programu se využívá jen velmi malé množství knihovních funkcí jazyka C. Dalo by se říci, že se převážně vystačí se základními konstrukcemi, jako jsou cykly a podmínky. Stejně jako u Assembleru, tak i v jazyku C je vytvořený program obtížně přenositelný na jiný mikropočítač, z důvodu, že každý mikropočítač obsahuje různé periferie a velikosti pamětí nebo přinejmenším jiné názvy portů. Avšak stále je lépe přenositelný program napsaný v jazyce C, díky své lepší čitelnosti.

Není-li to nezbytně nutné, není dobré používat reálná čísla (datové typy float a double). Mikropočítače nebývají vybaveny procesorem pro práci s reálnými čísly a veškeré operace s nimi musejí být prováděny pomocí knihovních funkcí, čehož důsledkem je jednak zpomalení programu, ale také nárůst jeho velikosti. U programů, kde záleží na časování, je nevýhodou jazyků vyšší úrovně to, že provedení jedné instrukce netrvá jeden takt procesoru jako je tomu u Assembleru, zde je nutné použít hardwarové časovače.

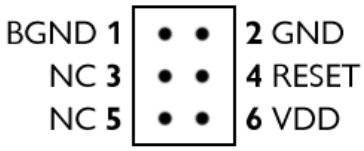
3.3 Programátor USB Multilink BDME

Rozhraní USB HCS08/HCS12 MULTILINK poskytuje přístup do BDM mikrokontrolérů HCS08, HC12 a HCS12. Pomocí USB a počítače pak lze přímo řídit čtení/zápis do registrů, hodnoty paměti, ladit kód procesoru, programovat zařízení s interní nebo externí Flash pamětí, atd. USBMULTILINK může komunikovat s procesory HCS08, HC12 nebo HCS12 s frekvencí sběrnice mezi 32KHz a 35MHz.

Základní vlastnosti

- Real-time In-Circuit Debug pomocí BDM rozhraní HCS08 nebo HCS12
- Rychlé obvodové programování Flash

- Rozměry 3“ x 2“ x 3/4“
- Rozhraní USB-to-BDM



Obr. 14 BDM konektor programátoru

3.4 Mikropočítač Freescale MC9S08SH4

Mikropočítač MC9S08SH4 spadá do velmi výkonné a levné rodiny 8-bitových mikropočítačů HCS08 postavené na stejnojmenném jádře. Jejich struktura odpovídá von Neumannově architektuře, kdy data i program jsou umístěny ve stejném paměťovém prostoru a řadí se mezi procesory typu CISC, tedy s úplnou instrukční sadou.

Vysoce integrovaná řada SH je charakteristická 40MHz taktem jádra ve spojení s nízko pinovým pouzdrem, silnými analogovými schopnostmi a kompletním vybavením pro sériovou komunikaci. A je určen pro 5V, takže se hodí i pro TTL logiku.

Rodina HCS08 obsahuje velké množství mikropočítačů s širokým výběrem periferií, velikostí paměti, typem paměti a použitým pouzdrem. Z označení mikropočítače lze vyčíst mnoho informací.

Popis značení mikroprocesorů Freescale:

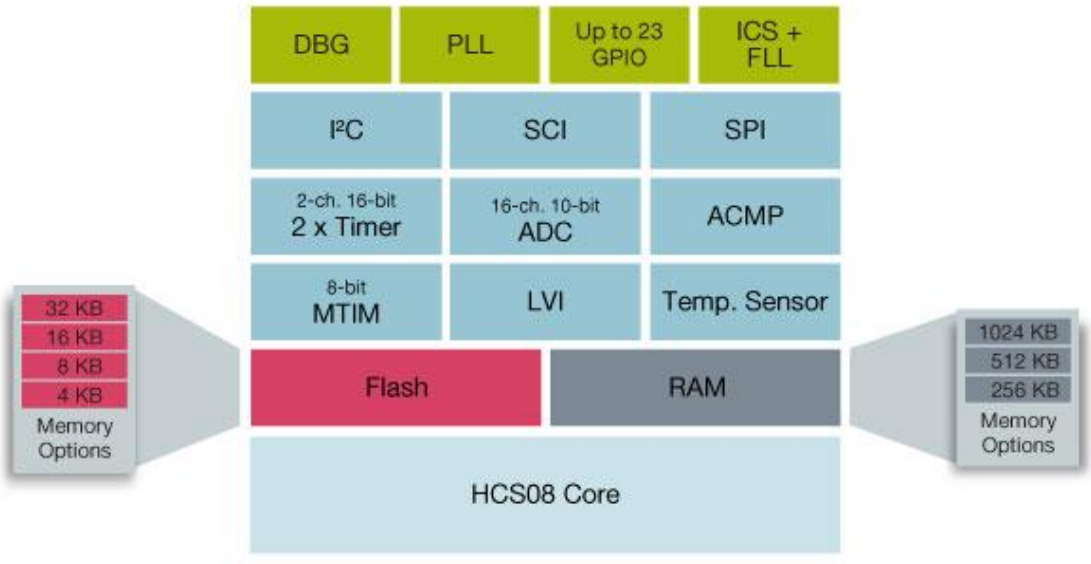
MC9S08SH4CPJ

- MC – Označení výrobce (MOTOROLA)
- 9 – Typ použité paměti (FLASH)
- S08 – Označení jádra procesoru
- SH – Označení řady
- 4 – Velikost paměti (FLASH)
- C – Rozsah pracovních teplot
- PJ – Označení pouzdra

3.4.1 Vlastnosti mikropočítače MC9S08SH4

- 8 bitová CPU HCS08 s maximální taktovací frekvencí 40MHz
- 20MHz maximální frekvence sběrnice
- 4Kb paměti FLASH
- 256 bytes paměti RAM
- 17 I/O linek na 3 portech
- 2 x 2 kanálový 16 bitový časovač
- 12 kanálový, 10 bitový analogově-digitální převodník
- Analogový komparátor
- Synchronní sériové periferní rozhraní (SPI)
- Asynchronní sériové komunikační rozhraní (SCI)
- Multi-masterová sériová sběrnice (I2C)
- Interní generátor hodinového kmitočtu
- Watchdog systém s nastavitelnou časovou prodlevou

Kromě těchto základních vlastností ještě disponuje systémem kontroly poklesu napájecího napětí pod stanovenou mez, podporou režimu se sníženou spotřebou, teplotními senzory nebo také systémem ladění programu přímo za chodu (DBG). Parametry mikropočítače MC9S08SH4 tak naznačují, že je vhodný téměř pro jakoukoliv jednodušší aplikaci v řídicím systému. [6]



Obr. 15 Blokové schéma rodiny mikropočítačů S08SH [6]

3.4.2 CPU

CPU (Central Processing Unit) zkráceně mikroprocesor se nazývá hlavní výkonná část mikropočítače. Na základě dodávaných instrukcí z programu, kterými je řízen, vykonává zadané úkoly. CPU je tvořena řadičem a aritmetickologickou jednotkou. Řadič má za úkol řídit všechny části mikropočítače a generuje potřebné signály pro jejich spolupráci. V aritmetickologické jednotce se provádějí všechny aritmetické (sčítání, násobení, bitový posuv, …) a logické (logický součin, negace, …) výpočty. Jádro mikroprocesoru HCS08 obsahuje následující pracovní registry:

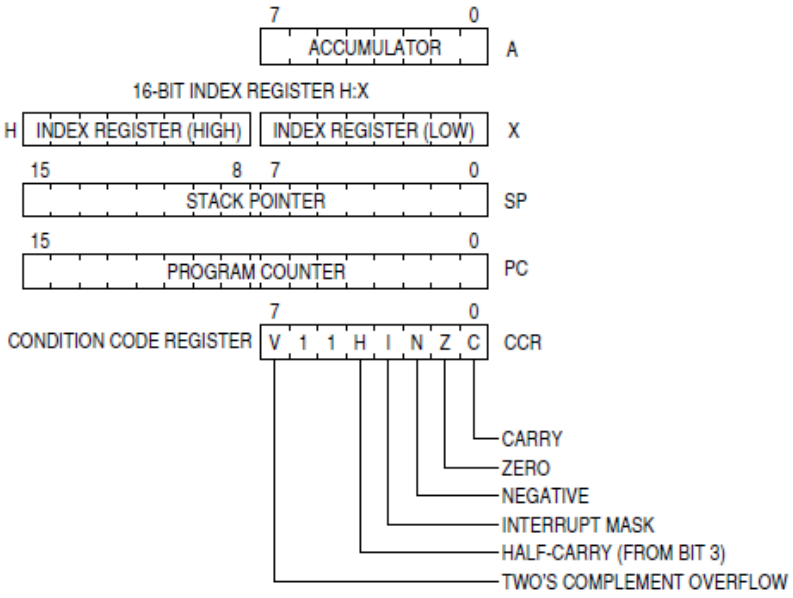
Akumulátor (A) – je 8-bitový registr, kterým se provádí aritmetické a logické operace.

Indexový registr (H:X) – je 16-bitový registr, který se ve skutečnosti skládá ze dvou 8-bitových registrů H a X, kde H je horní bajt a X dolní bajt adresy.

Ukazatel zásobníku (SP) – tenhle 16-bitový registr ukazuje na další volné místo v zásobníku a při zápisu dat je inkrementován a při čtení dekrementován. Zásobník může být umístěn kdekoliv v paměti RAM.

Programový čítač (PC) – je 16-bitový registr, do kterého se ukládají adresy dalších instrukcí, které se mají vykonat.

Příznakový registr (CCR) – je 8-bitový příznakový registr, ve kterém se nastavují příznaky výsledku a přenosu.

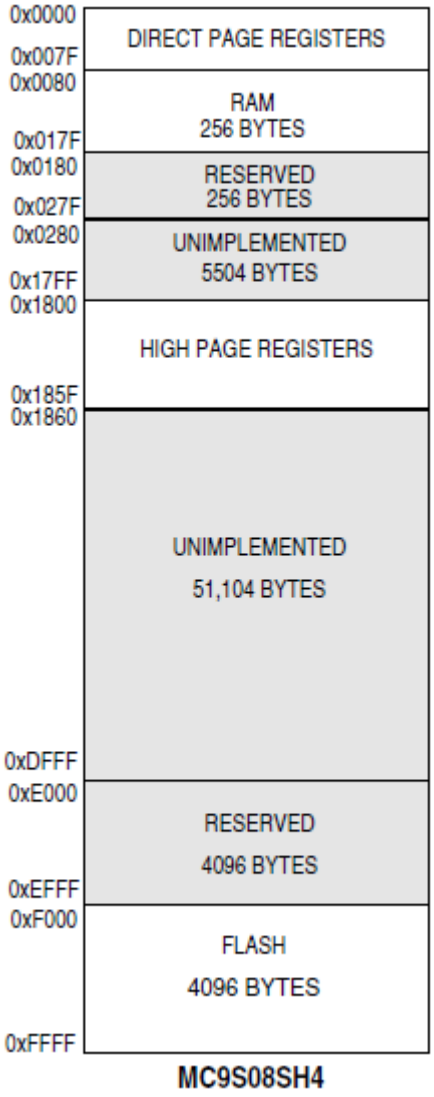


Obr. 16 Pracovní registry HCS08[6]

3.4.3 Paměť

Paměťová mapa

Pamětí se rozumí jakékoliv zařízení, do kterého je možné uložit a poté kdykoliv opět získat informace. Na obrázku je znázorněna paměťová mapa obsahující RAM, FLASH, programovou paměť pro permanentní uchování dat a vstupně/výstupní a kontrolní registry.



Obr. 17 Paměťová mapa S08SH4 [6]

Adresový prostor procesorů HCS08 by se dal rozdělit na:

- Registry (zápisník)
- Zásobník
- Obecnou datovou paměť

Registry

Registry slouží pro dočasné uchovávání zpracovávaných dat. Za pomoci registrů mikropočítač přijímá a předává důležité informace pro chod programu, adresy a kódy právě vykonávaných instrukcí, aktuální nastavení, stav periferií, žádosti o přerušení, příznaky přetečení, atd. Jsou i registry, které jsou určeny pouze pro čtení.

Zásobník

Zásobník u rodiny mikropočítačů HCS08 je koncipován jako paměť typu LIFO (Last In First Out). Princip spočívá v tom, že informace uložená na vrcholu zásobníku bude přečtena jako první. Vrcholem zásobníku se rozumí aktuální paměťová buňka, do které se ukládají, nebo načítají data. Adresa vrcholu zásobníku se uchovává tzv. SP – Stack Pointer.

Zásobník nemusí být využíván jen v uživatelském programu, ale může jej použít i jednotka CPU pro ukládání návratové adresy při skocích do podprogramu nebo obsluhy přerušení. Během přerušení se automaticky ukládají na zásobník obsahy všech pracovních registrů (s výjimkou registru H).

Obecná datová paměť

Tahle oblast paměti je kapacitně nejrozsáhlejší a slouží k ukládání programu a větších datových struktur. Je tvořena přepisovatelnou pamětí FLASH pro uchování programu a napětí RAM, která je většinou typu RWM popřípadě ROM a data v ní nejsou ovlivněna, dokud napětí neklesne pod minimální hodnotu nutnou pro uchování dat.

3.4.4 Přerušení

V mikropočítači se často pracuje s komponenty, u kterých nastávají tzv. neočekávané události. Pro představu je možné uvést takové typické stisknutí tlačítka. K vyřešení těchhle neočekávaných stavů se nabízejí dva způsoby.

První by se dalo charakterizovat jako řízení v čekací smyčce. Princip spočívá v tom, že se testuje stav zařízení, je- li například stisknuto tlačítko. Pokud není podmínka splněna, test se neustále opakuje ve smyčce a čeká na splnění podmínky. Po splnění podmínky se přechází do jiné části programu, kde se provede funkce pro danou komponentu, v tomhle případě pro příslušné stisknuté tlačítko. Po dokončení akce se program vrací zpět do smyčky a čeká na stisknutí dalšího tlačítka. Tahle metoda však není příliš vhodná při práci s více komponenty, které generují neočekávané události nebo pro časově náročné programy. Nevýhoda je, že při čekání ve smyčce mikroprocesor nemůže obvykle vykonávat jinou činnost a tím se zatěžuje jeho činnost.

Tahle nevýhoda se dá odstranit metodou přerušení od daného zařízení. Přerušovací systém umožňuje, aby mikroprocesor reagoval okamžitě na zařízení vyvolávající neočekávané události a nečekal v cyklu na ukončení činnosti těchto zařízení. Žádosti o přerušení jsou asynchronní vůči operacím v CPU. Při splnění podmínek přerušení procesor uloží právě běžící program na zásobník a provede se speciální podprogram, přiřazený danému přerušení, nazývaný obsluha přerušení. S žádostí o přerušení se vybere to s nejvyšší prioritou a přejde se na pevně danou adresu v paměti na tzv. vektor přerušení. Každý typ mikropočítače má vektory přerušení umístěny na jiném místě a obsahují zpravidla adresu v paměti se začátkem podprogramu pro obsluhu daného zařízení. Umístění vektorů přerušení lze zjistit z dokumentace. V podprogramu je nejprve potvrdí příjem přerušení (aby nedošlo k zacyklení) a následně se vykoná vlastní část programu, která má nastat po události. Po ukončení obsluhy se obnoví původní obsah pracovních registrů a pokračuje se v provádění přerušeného programu.

Využití přerušení:

- Tlačítka – změna log úrovně na portu
- Časovač – uplynutí určitého času
- A/D převodník – dokončení převodu
- Komunikace – příchod dat na sériovém rozhraní

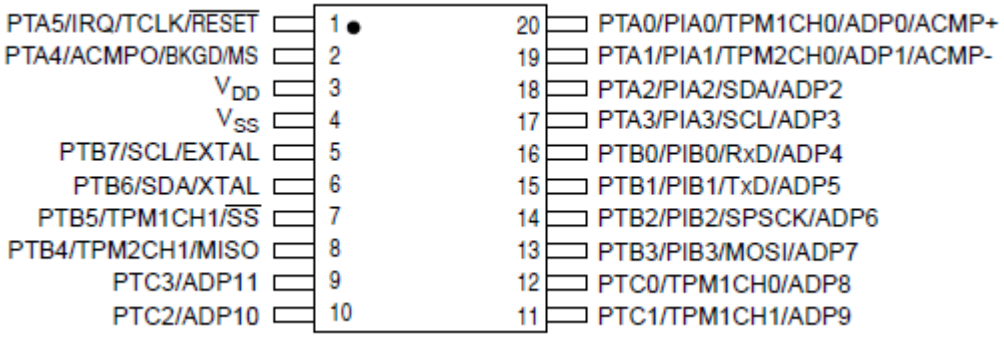
3.4.5 COP (Computer Operating Properly)

Funkce je známá spíš pod názvem Watchdog (nebo také krmení hlídacího psa). Je to systém založený na časovači s nastavenou hodnotou 2^{18} nebo 2^{13} cyklů procesoru s funkcí přetečení. Jeho úkolem je hlídání korektního chodu programu. Princip spočívá v jeho pravidelném softwarovém resetování a tím by mělo být zajištěno, že nikdy nedojde k jeho přetečení. Pokud by nastala situace, kdy program uvízne např. v nekonečné smyčce, tak po nějaké době dojde k přetečení, které následně vyvolá restart mikropočítače. Existuje i možnost watchdog softwarově vypnout.

3.4.6 Porty

Mikropočítač MC9S08SH4 má 17 vstupně/výstupních linek, které jsou rozděleny do 3 portů, z čehož linka BKGD se používá k programování, ale také k ladění programu přímo za chodu, proto pokud je to možné, je dobré ji nechat pro tento účel rezervovanou. Většina těchto vývodů (pinů) je sdílena s dalšími periferiemi mikropočítače. Můžou to být generátory PWM, vnější přerušení, analogové převodníky nebo komparátory. Pokud však tyto moduly nejsou aktivovány, piny jsou nastaveny do režimu I/O linek. Data a nastavení jednotlivých pinů respektive portů se prování přes stejnojmenné registry. Např. přiřazením hodnoty 0xFF do registru PTAPE se zapnou pull-up rezistory na všech pinech portu A. Samozřejmě by měli být splněný podmínky, že celý port A je nastaven jako vstup, což se provádí přes registr PTADD a je že obsahuje všech 8 linek.

Namapování periferií na jednotlivé piny je znázorněno na následujícím obrázku. Popis zkratek je k dispozici v dokumentaci k mikropočítači zde [6].



Obr. 18 Popis pinů S08SH4 [6]

3.4.7 Časovač

Časovače už patří ke standardní výbavě mikropočítačů. Používají se ke generování a měření přesných časových intervalů. Přerušení vždy nastane ve stanovený okamžik, čímž je možné docílit volitelné frekvence nebo konkrétní délky časového úseku, což se dá dobře využít kupříkladu u řízení serva. Obvody čítačů a časovačů jsou vybaveny speciálními funkcemi jako generovaní obdélníkového průběhu na určité frekvenci, ale také slouží jako zdroj hodinového signálu pro obvod COP (Watchdog).

Mikropočítač MC9S08SH4 je vybaven dvěma dvoukanálovými časovači TPM1 a TPM2. Nastavení časovače se provádí pomocí registru TPMxSC, kde první 3 bity zleva určují velikost děličky (1, 2, 4, 8,…, 128), následujícími dvěma bity volíme zdroj hodinového kmitočtu a dále pak po bitu, náběžnou hranu, povolení přerušení a nulování příznaku. Délka časového intervalu se zadává pomocí 16-bitového modulo registru, jehož velikost se počítá následujícím vzorcem.

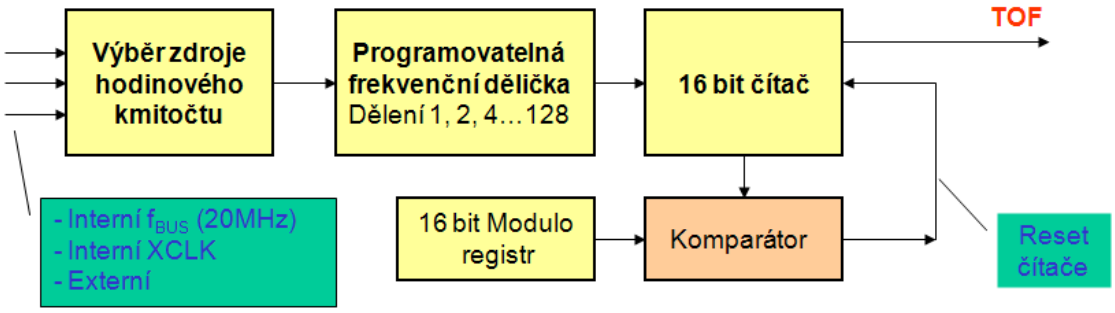
$$\text{Modulo} = \frac{t_{TOF} \cdot f_{source}}{\text{Prescaler}}$$

Modulo – údaj, který zapíšeme do modulo registru

t_{TOF} – požadovaný čas do přetečení časovače [s]

f_{source} – frekvence zdroje hodinového kmitočtu [Hz]

Prescaler – nastavení vstupní frekvenční děličky



Obr. 19 Blokové schéma časovače v mikropočítači

II. PRAKTICKÁ ČÁST

4 NÁVRH MODULU ŘÍZENÍ MOTORŮ

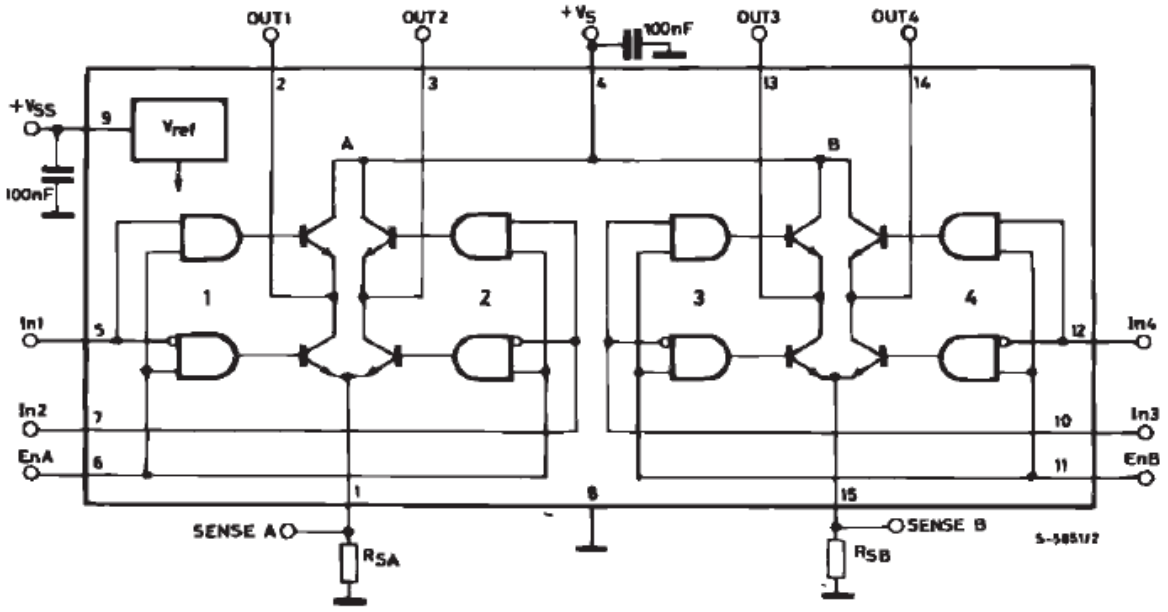
Předpokladem bylo navržení elektrického obvodu, který bude řídit rychlost otáčení hlavních dvou výškových motorů, popřípadě i motoru ocasního. U zadního směrového motoru není až tak důležitá možnost regulace otáček, jako spíš nutnost otáčení oběma směry. Posledním ovládaným prvek je servo, kterému se nastavuje poloha natočení pomocí velikosti řídících impulsů o stejné frekvenci většinou 50Hz.

Obvod nazvaný Řízení motorů obsahuje hlavní dva prvky. Ten nejdůležitější je mikroprocesor, který v první řadě přijímá veškerá vnější data, jako jsou signály od tlačítek nebo externí komunikace s jiným zařízením, ale také pomocí interního časovače a pulsně šířkové modulace (PWM) softwarově generuje signály pro řízení otáček motorů nebo řídící impulsy pro servo.

Jelikož se proud na vývodech procesoru pohybuje v jednotkách, maximálně desítkách mA, není tak možné připojit motory přímo k procesoru, ale je nutné je spínat pomocí H můstku, který tak představuje druhý nejdůležitější prvek v obvodu.

4.1 H – můstek L298

Součástka nazvaná DUAL FULL-BRIDGE DRIVER je 15 pinový integrovaný obvod obsahující dva H – můstky, určené pro řízení směru otáček motoru.



Obr. 20 Blokové schéma integrovaného obvodu L298

Maximální parametry obvodu:

- Napájení motorů: 50V
- Proud do motoru: 2A
- Napájení obvodu: 4,5V - 7V

Každý můstek je tvořen čtyřmi tranzistory, které jsou spínány čtveřicí hradel AND. Pokud je na vstup IN1 přivedena log. 1 a na IN2 log. 0, motor se začne otáček směrem doprava. Pokud se logické úrovně na vstupech vymění, změní se i směr otáčení motoru. Pokud však nastavíme stejnou log. úroveň na oba vstupy, motor se rychle zastaví. Všechny tyhle operace lze provádět jen pokud je připojen vstup ENABLE. Pokud není, tak jsou všechny tranzistory rozepnuty a motor tak pomalu setrvačností dokončí své otáčení.

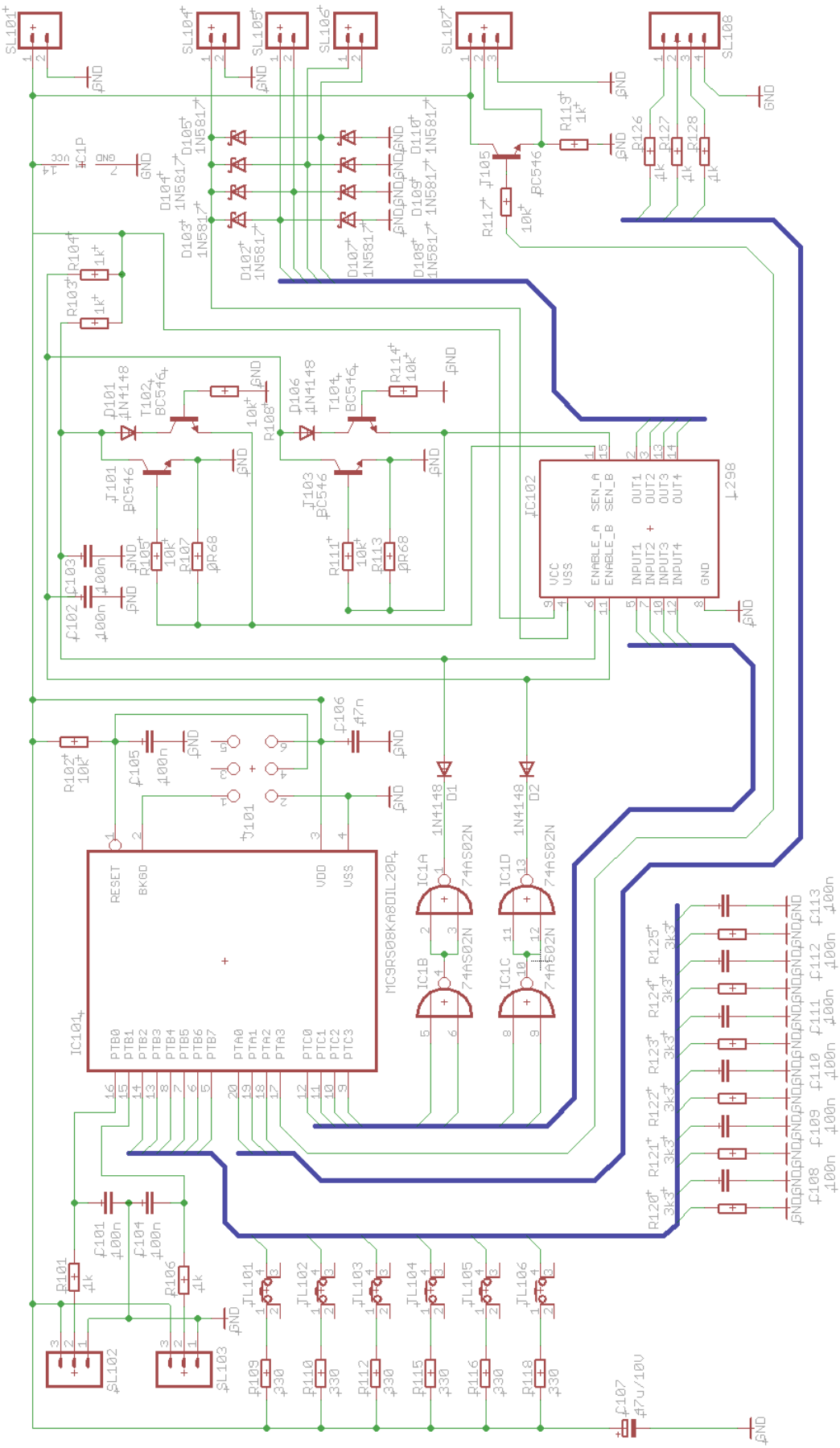
Integrovaný obvod má ještě výstupy OUT pro připojení motorů a výstupy SENSE pro připojení proudové ochrany.

4.2 Schéma obvodu řízení motorů

4.2.1 Ovládání a konektory

Ovládání modulu je možné dvěma způsoby. První možností je 6 tlačítek připojených proti napájení 5V přes rezistory 330 Ω. Procesor tak při stisknutí tlačítka indikuje log. 1 a není proto nutné zapínat interní pull-up rezistory. Tlačítka jsou také uzemněna přes větší rezistor a kondenzátor, aby se co možná nejvíce zabránilo zákmitům při stisku. Druhou možností jak získávat data k řízení motorů je připojení se k modulu ultrazvukových senzorů. Propojení obstarává další obvod s názvem Komunikace, který je připojen na konektor SL108 a je popsaný v jedné z dalších kapitol.

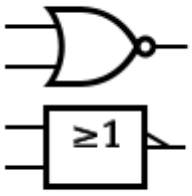
Ve schématu jsou umístěny také dva konektory SL102 a SL103 pro případné připojení akcelerometrů, které jsou spojeny s piny 15 a 16 mikroprocesoru, na kterých se nachází A/D převodním. Část obvodu kolem šesti-kolíkového BDM konektoru J101 je minimální konfigurace pro připojení programátoru. Poslední částí obvodu netýkající se řízení motorů je ovládání serva řídícími impulsy pomocí tranzistoru spínaného procesorem s výstupem na konektoru SL107.



Obr. 21 Schéma modulu řízení motorů

4.2.2 Logika řízení motorů

Zbylá část obvodu je určena k řízení motorů a jejím hlavním prvkem je integrovaný obvod L298. Původním návrhem bylo propojení vývodu SENSE přes proudovou ochranu zpět do vstupu ENABLE a zbylými dvěma vstupy IN1 a IN2 řídit směr otáčení. Při testování nižších otáček pomocí PWM na nepájivém poli však nastal problém. Za předpokladu, že by se měl motor otáčet jen 60 % svého výkonu, se nastaví log.1 na vstup IN1 po dobu 60 % jedné periody a po zbytek času do konce periody jej vrátíme do úrovně log. 0. Problémem však je, že pokud jsou na obou vstupech IN nuly (popřípadě jedničky), tak se motor dostává do stavu brzdění. Zapojení s touhle logikou by s největší pravděpodobností fungovalo, ale modul by neustále zrychloval a brzdil a zbytečně tak spotřebovával velké množství energie, které má vzducholoď kvůli nosnosti už tak málo. Řešením je tak po zbytek periody odpojovat vstup ENABLE, který uzavírá celý H můstek a dovolí motorům setrvačností dokončit otáčky. Jelikož má ale procesor omezený počet I/O linek, řešením je odpojování (uzemňování) ENABLE za pomoci dvou logických hradel NOR zapojených za sebou.



X_1	X_2	Y
0	0	1
0	1	0
1	0	0
1	1	0

Obr. 22 Hradlo NOR a pravdivostní tabulka

Popis všech tří stavů je následující:

1. Na vstupech IN1 a IN2 je [0 1] nebo [1 0] podle směru otáčení.

Po průchodu prvním hradlem je na výstupu log. 0, která se dále posílá do obou vstupů hradla druhého a na konci obou hradel se objevuje log. 1, která tak nedovolí v opačném směru přes diodu uzemnit vstup ENABLE. Motor se tak otáčí.

2. Na vstupech IN1 a IN2 jsou nuly [0 0]

První hradlo udělá ze dvou nul jedničku a následně ze dvou jedniček nulu. Což uzemní vstup ENABLE a odpojí celý můstek. Motor tak setrvačností zpomaluje.

3. Na vstupech IN1 a IN2 jsou jedničky [1 1]

Poměry na hradlech jsou stejné jako v prvním případě s tím rozdílem, že sice vstup ENABLE není uzemněn, avšak na vstupech IN1 a IN2 jsou jedničky, což způsobí rychlé zastavení motoru.

4.2.3 Napěťová a proudová ochrana

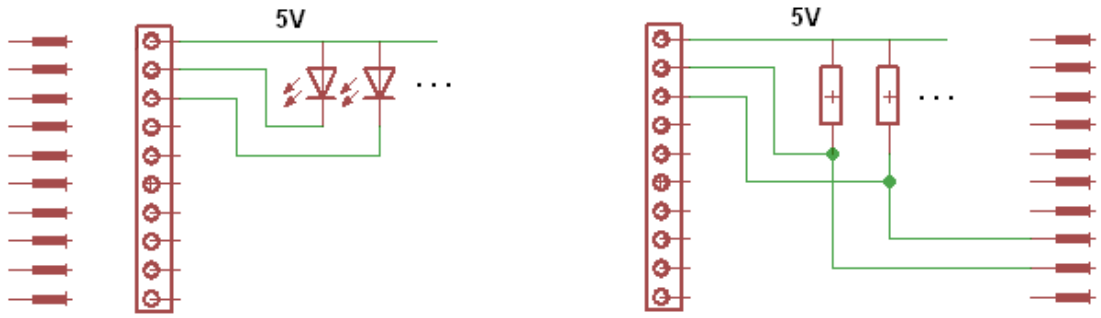
Ochrana spočívá ve dvou částech. První tvoří osm Schottkyho diod 1N5817, které tak chrání budič proti naindukovanému napětí, které vzniká na komutátorech motorů. Skokové impulsy by tak mohli dosáhnout amplitudy až ke 100V a poškodit tak výstupní tranzistory.

Druhou částí je proudová ochrana obvodu L298, která je připojena na výstup SENSE a po překročení proudu daného velikostí rezistoru tak tranzistor uzemní horní logickou úroveň na vstupu ENABLE. Ochrana je tvořena i druhou částí pro opačnou polaritu, která však nakonec není využívána.

Všechny výše popisované části řízení motorů a ochrany jsou realizovány ve dvojici pro připojení dvou motorů na konektory SL105 a SL106. Napájení celého obvodu 5V je na konektoru SL101 a zdrojem pro motory je 7V napětí na konektoru SL104.

4.3 Obvod komunikace

Vstupní data řídícího programu je možno brát buď standardně z šesti tlačítek, které tak slouží k manuálnímu ovládání řízené soustavy a posléze by mohlo být nahrazeno ovládáním na dálku z počítače např. přes grafické rozhraní při zachování stejného algoritmu. Další možností získávání dat o překážkách je připojení se k modulu ultrazvukových senzorů. Modul senzorů má koncovou oddělitelnou část devíti LED diod, které signalizují překážku před každým senzorem ve vzdálenosti 1m. Přes napětí 5V jsou všechny LED diody připojeny zpět k obvodu SAA1064, který získává data sběrnicí I^2C a spíná tak jednotlivé piny, ke kterým jsou přivedeny LED diody, a tím je rozsvěcuje. K získávání signálů bylo nutné nahradit LED diody devíti rezistory, za kterými je možné sledovat logickou úroveň signálů.



Obr. 23 Nahrazení 9 LED diod rezistory

Těchto 9 signálů je nutno přivést do mikroprocesoru. Jelikož má procesor jen omezený počet vstupů, tak se k zredukování devíti linek na tři využívá principu časového multiplexu.

4.3.1 Časový multiplex

Časový multiplex (TDM) je princip přenosu více signálů jedním společným přenosovým médiem. Jednotlivé signály jsou odděleny tím, že se každý z nich vysílá (přenáší) pouze krátký pevně definovaný časový okamžik. Prakticky ve všech případech se používá rámcové struktury, která je rozdělena na stejně velké timesloty, časové intervaly pro vysílání, pro každý signál jeden. Tento rámec se v čase neustále opakuje, a tedy každý signál se přenáší stále se stejnou pravidelností.

4.3.2 Čítač a multiplexor

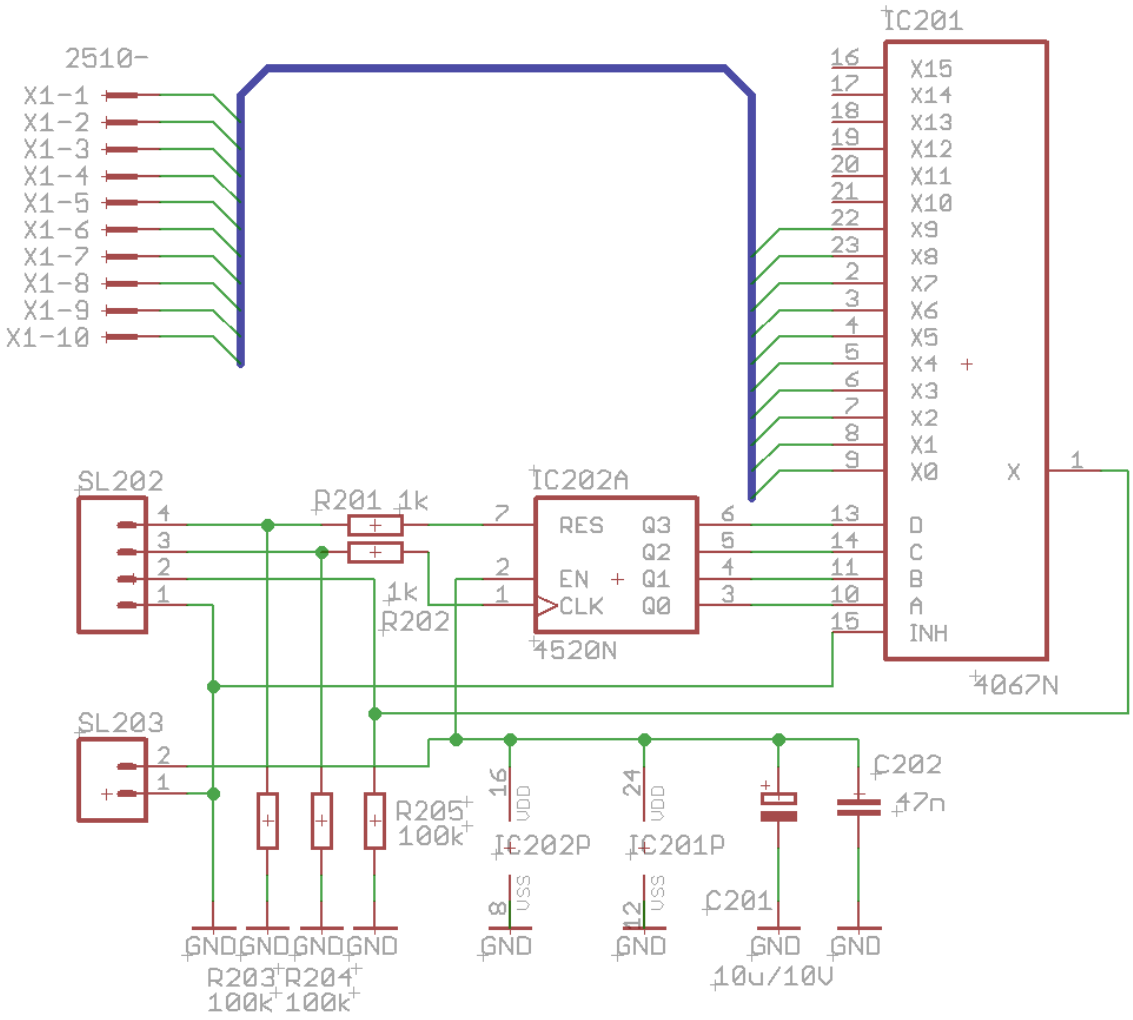
Obvod komunikace je postaven na spolupráci binárního čítače a 16-kanálového analogového multiplexoru. Devět signálů reprezentujících stavy na LED diodách jsou přivedeny na vstupy multiplexoru spolu s jedním vodičem z jejich napájení, který tak podává informaci o tom, jestli je modul zapnut.

Multiplexor má kromě 16 vstupů a INHIBITORU ještě čtyři piny pro binární ovládání a jednu výstupní linku. Princip spočívá v tom, že na ovládací piny A B C D se nastaví binární kombinace, podle které tak multiplexor přesměruje jeden ze vstupních kanálů na výstup. Kombinace stavů s příslušnými kanály je uvedeny v tabulce č. XXX

A	B	C	D	INH	Vybraný kanál
X	X	X	X	1	Žádný
0	0	0	0	0	0
1	0	0	0	0	1
0	1	0	0	0	2
1	1	0	0	0	3
0	0	1	0	0	4
1	0	1	0	0	5
0	1	1	0	0	6
1	1	1	0	0	7
0	0	0	1	0	8
1	0	0	1	0	9

Tab. 2 Výběr kanálu multiplexoru

K vytvoření čtyřmístné binární kombinace je použit čítač, který má kromě zmíněných čtyř výstupů i tři řídící piny CP_0 , CP_1 a RESET. Pokud má čítač reagovat na náběžnou hranu, tak na CP_1 musí být neustále log. 1 a každým impulsem na vstupu CP_0 se zvyšuje číslo, které je k dispozici na výstupu v binárním stavu. Pokud je třeba čítat jen do určitého čísla, tak impulsem na RESET začíná zas od nuly. O řízení čítače a zpracování dat se už stará procesor, jen třemi I/O linkami. Obvod je doplněn ochrannými rezistory a kondenzátory pro eliminaci napěťových špiček.



Obr. 24 Schéma obvodu komunikace

4.4 Praktická realizace modulu

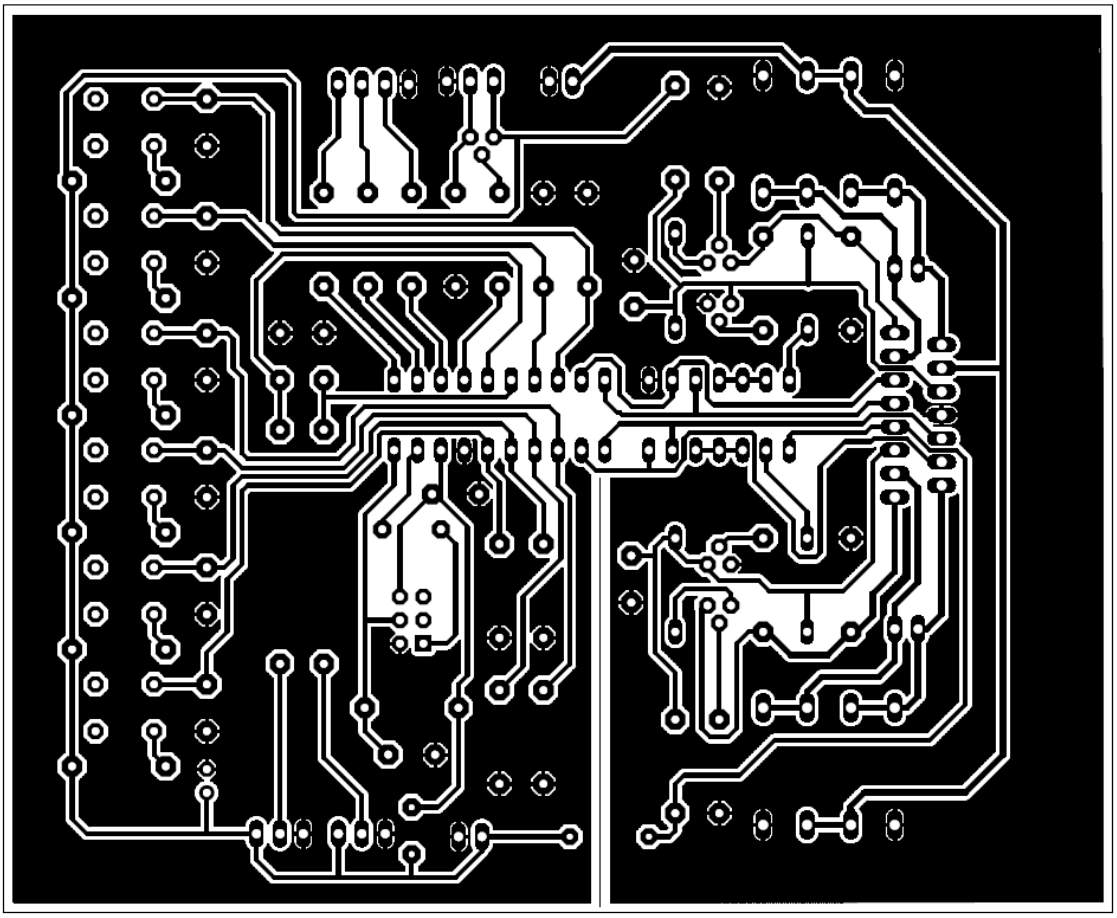
4.4.1 Výroba plošných spojů pomocí fotocesty

V dřívějších letech a převážně u domácí výroby desek plošných spojů zkráceně DPS se používala základní technika využívající lihové fixy případně leptuvzdorného laku. Vesměs jednoduché schéma se tužkou zrcadlově převrácené lehce navrhnulo na spodní stranu cuprextitu, což je izolační deska s nanesenou vrstvou mědi na spodní straně. A poté obtáhlo lihovou fixou a dalo se leptat. Tahle technika je však značně komplikovaná pokud se kreslí cestičky blízko u sebe, nebo je schéma dost složité. Daleko sofistikovanější řešení je výroba DPS pomocí fotocesty.

Co je k tomu potřeba:

- UV lampa, nebo deska s LED UV diodami
- Pauzovací papír nebo folie
- Laserovou tiskárnu
- Program EAGLE nebo jiný podobný na vytváření plošných spojů
- Hydroxid sodný
- Chlorid železitý nebo kyselina chlorovodíková s peroxidem vodíku

Postup výroby je následující. Po dokončení návrhu desky v programu EAGLE se nechají zapnuté vrstvy Bottom, Pads, Vias, Dimensiona vytiskne se návrh na pauzovací papír nebo folii na laserové tiskárně s dostatečnou vrstvou toneru. Vytištěnou stranu přiložíme k fotocitlivému cuprextitu, čímž se docílí zrcadlového převrácení, zatíží se sklem a vloží pod UV lampu.

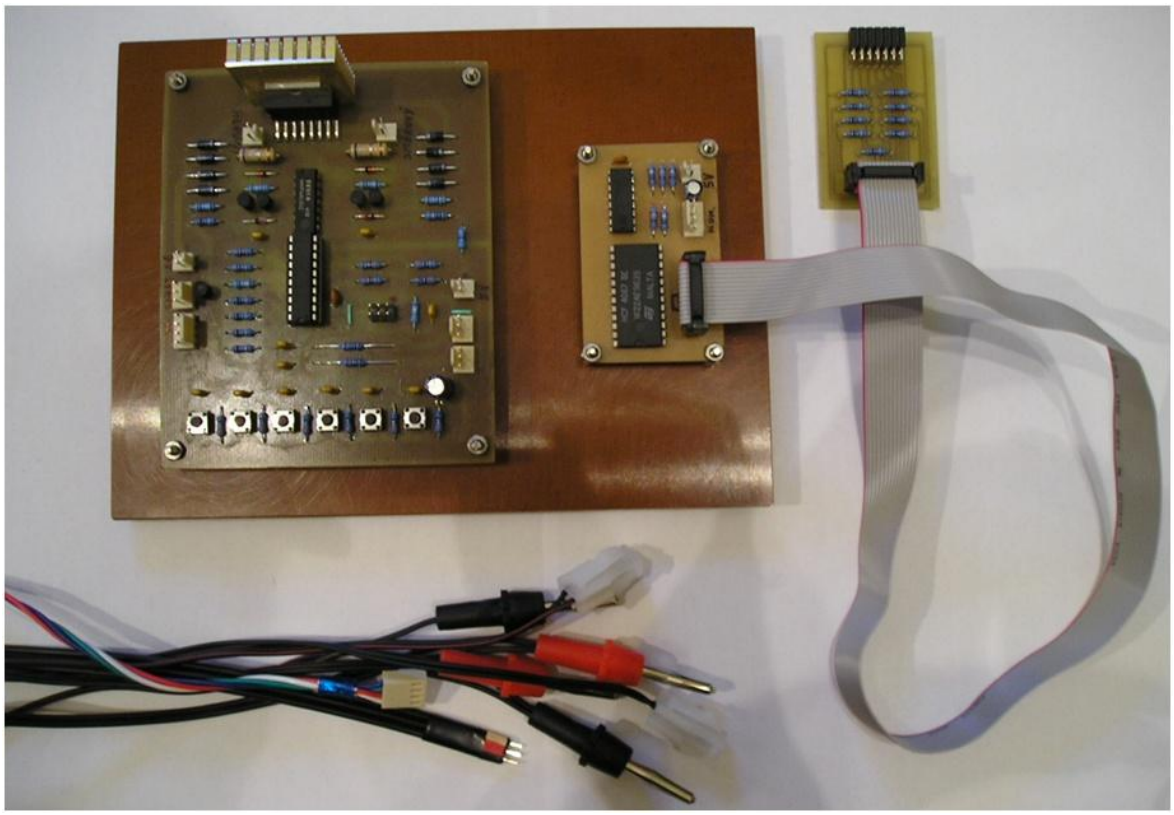


Obr. 25 Návrh desky obvodu řízení motorů

Doba osvěcování trvá podle příkonu výbojky nebo podle toho, jak vysoko je výbojka nad sklem, pro orientaci se čas pohybuje v jednotkách až desítkách minut. Po osvícení se cuprextit ponoří do tzv. vývojky, což je hydroxid sodný rozpuštěný ve vodě v poměru asi 10 – 15 gramů na litr. Po odstranění laku a opláchnutí vodou přichází na řadu leptání. Leptá se například v chloridu železitém. Samozřejmě stejně jako chlorid, tak i hydroxid jsou žíraviny a mohou způsobit poleptání, takže je nezbytné používat vhodné pomůcky. Nakonec se vyleptaný plošný spoj umyje, zkontroluje, zda není některá cestička odleptaná. Na konec se vyvrtají díry a celý spoj se potře v lihu rozpuštěnou kalafunou, která pak pomáhá při pájení a vytváří ochrannou vrstvu. Deska je tak připravena k osazení součástkami. [7]

4.4.2 Odzkoušení vlastností

Důležitým aspektem úspěšné spolupráce modulu řízení motorů se senzory bylo propojení společné země obou obvodů. Také motory, pokud jely delší dobu s odběrem kolem 1A, zahřívaly integrovaný obvod L298 natolik, že bylo nutné na něj umístit pasivní chladič. Po úspěšném otestování obvod splňoval všechny předpokládané vlastnosti.



Obr. 26 Praktická realizace modulu

5 VYTVOŘENÍ ŘÍDÍCÍHO ALGORITMU

5.1 Popis algoritmu

Základním pilířem řídícího algoritmu byla činnost na dvou konkrétních frekvencích a generování časového impulsu pohybujícího se mezi 1 – 2 ms. Délka pulsu určuje natočení serva, ale je nutné ho opakovat s frekvencí 50Hz. Druhá frekvence se využívá k buzení motorů pomocí PWM a měla by být vyšší, v ideálním případě by se měla pohybovat okolo 1kHz. Mikropočítač MC9S08SH4 však obsahuje jen dva časovače. První bylo nutné přidělit proměnlivým časovým impulsům a druhý se stará o obě frekvence tím způsobem, že přerušení bylo nastaveno na frekvenci 1kHz a v obsluze přerušení je cyklus, který se vykonává jen jednou za 20 průchodů. 1000 Hz / 20 = 50Hz. Tímto způsobem je dosažena druhá frekvence.

Obdobným způsobem je řešeno i snižování otáček pulsně šířkovou modulací. Na kmitočtu 1kHz běží cyklus, který s maximální velikostí dvaceti průběhů představuje jednu periodu buzení motorů.

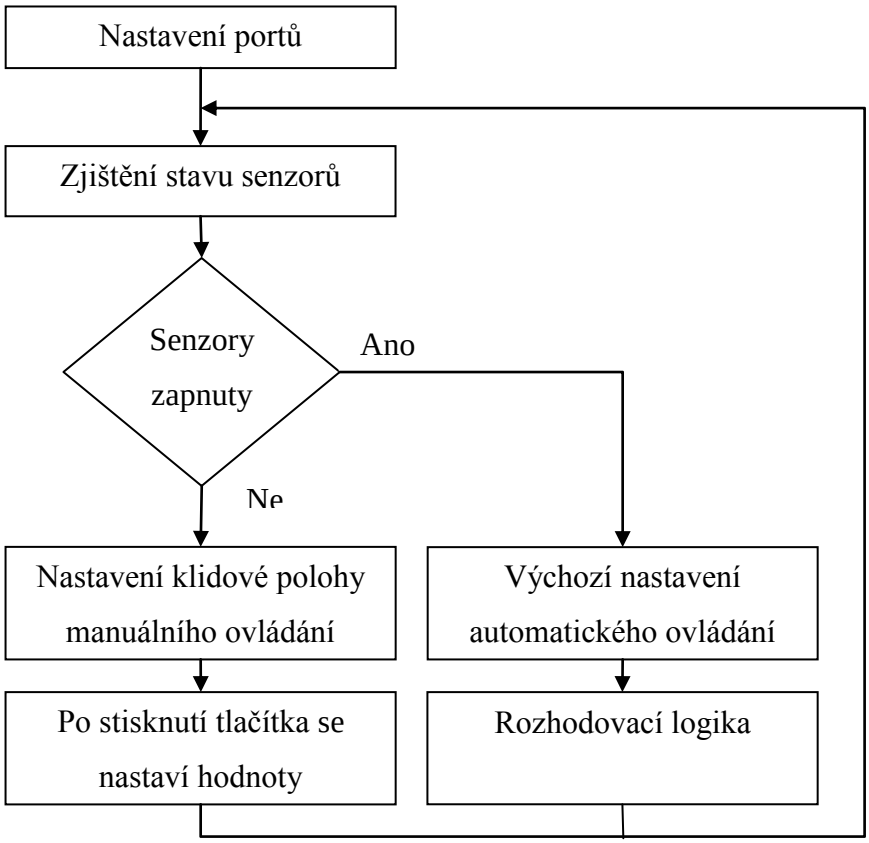
Výše zmíněnou činnost je nutno vykonávat ve víceméně přesných a pravidelných intervalech. Zbytek programu, jako je obsluha tlačítek, zjišťování stavu senzorů nebo rozhodovací algoritmy, je umístěn v hlavní smyčce programu a provádí se ve zbytku procesorového času.

Informace o stavech senzorů se zapisuje do pole o devíti prvcích typu integer. Aktualizace pole probíhá v cyklu, kde se zjišťuje logická úroveň na portu vstupních dat, načež je pokaždé vyslán impuls čítači pro inkrementaci a s tím spojené přepnutí kanálu. Po dokončení cyklu se čítač vyresetuje.

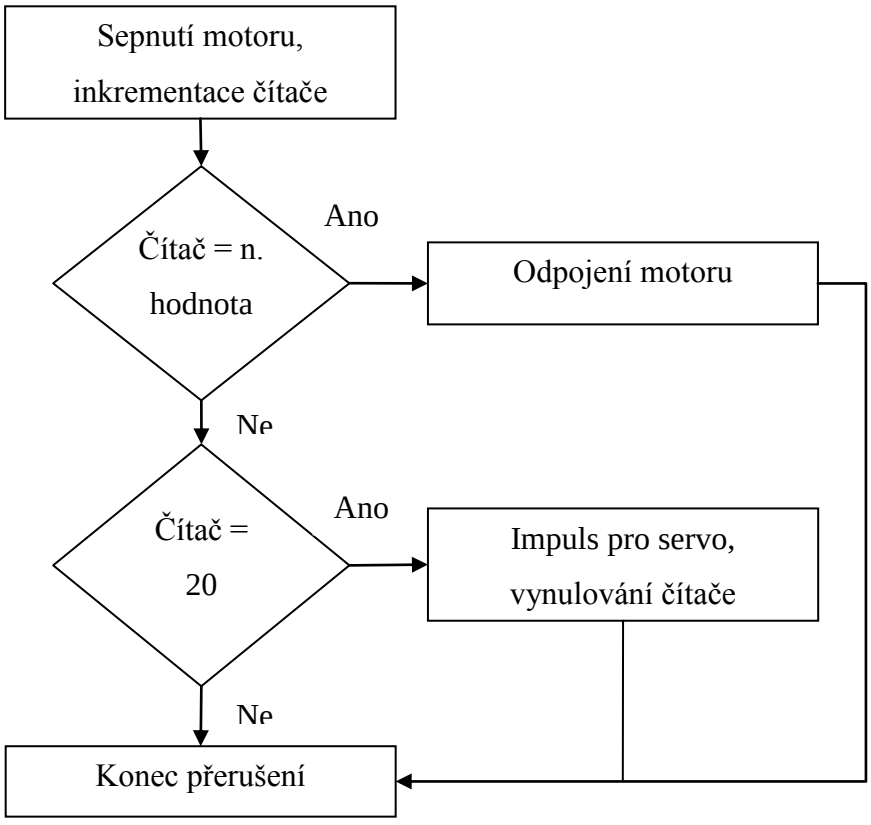
Zbylá část programu se větví podle podmínky, zdali je vypnut modul senzorů na manuální a automatické řízení, které však není nijak dokonalé, ale k základním úhybným manévrům by teoreticky mohlo stačit.

Program také obsahuje např. vylepšení kódu napomáhající rozbíhání motorů, avšak další detailní popisování by bylo nad rámec práce. Lepší představu o fungování snad napoví grafické schéma struktury algoritmu, nebo samotné okomentované zdrojové kódy umístěné v příloze.

5.2 Vývojový diagram



Přerušení 1 kHz



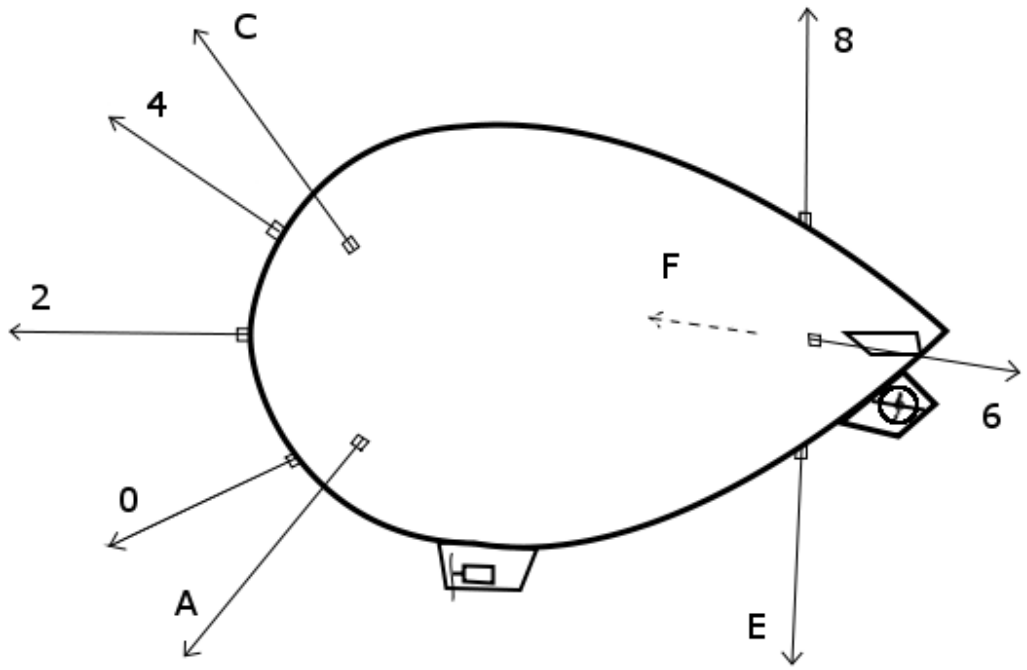
5.3 Ovládání modulu

5.3.1 Manuální

Manuální ovládání spočívá v řízení dvou motorů a serva šesti tlačítky. První dvě tlačítka ovládají polohu serva. První tak směruje hlavní motory dolů a druhé zpět nahoru. Pohyb serva je krokový, nelineární s dvaceti polohami. Dalšími dvěma tlačítky se mění otáčky hlavních vertikálních motorů. Regulace otáček probíhá také krokově v deseti stupních, ale přibližně první čtyři kroky budí motory natolik málo, že nejsou schopny udržet otáčky v chodu. Algoritmus navíc zajišťuje pomocné startovací impulsy plného výkonu pro prvních šest stupňů. Poslední dvě tlačítka ovládají ocasní motor oběma směry. Řízení už však není krokové, ale otáčí motorem na plný výkon po dobu stisknutí.

5.3.2 Automatické

Automatické ovládání je závislé na datech přicházejících od senzorů. Důležitým aspektem při rozhodovací logice programu je rozmístění devíti senzorů po plášti vzducholodě. Pokud by se senzory rozmístily chaoticky v přibližně stejných odstupech, program by pak musel obsahovat řešení na signalizaci všech kombinací senzorů, což by odpovídalo hodnotě 9! pohybující se v řádech statisíců.



Obr. 27 Předpoklad rozmístění senzorů na vzducholodi

Předpoklad rozmístění tak rozděluje okolí vzducholodě na čtyři, respektive pět oblastí. Nahoře, dole, vlevo, vpravo a vpředu. Každá z prvních čtyř oblastí je reprezentována součtem senzorů, detekují překážku ze tří možných. Pro představu např. oblast nahoře je složena ze senzorů 4, C, 8 a oblast vlevo zas C, A, 6. Podle převahy oblastí pak program řídí motory. Pokud je aktivní přední senzor sám nebo se všemi senzory v přední části, vzducholoď pak přejde do režimu zpětného chodu.

ZÁVĚR

Cílem diplomové práce bylo navržení a zrealizování elektrického modulu, jehož základem je mikroprocesor. Vytvořené obvody a algoritmy tak dokážou řídit motory a servo a ovládat tak vzducholoď při letu. Ovládání je možno dvěma způsoby. Základem je nezbytné manuální řízení tlačítky, ale modul také nabízí automatické řízení na základě dodávaných dat z ultrazvukových senzorů, jež dokážou detekovat překážku v určité vzdálenosti.

Praktické odzkoušení přímo za letu prozatím nebylo možné z důvodu chybějícího spínaného zdroje na 5V a 7V, který by byl umístěn také na vzducholodi. Další otázkou je, zdali vůbec projekt Autonomně se řídící vzducholoď jako celek unese veškeré své moduly a nebude třeba je vyrobit SMD provedení. Teoreticky po kalibraci by však vzducholoď měla být schopna samostatného letu.

ZÁVĚR V ANGLIČTINĚ

The aim of thesis was design and realization of power module based on a microprocessor. There are generated circuits and algorithms so they can drive motors and servo and control the airship in flight. Control is possible in two ways. The basis is a necessary manual controlling by buttons, but the module also provides automatic control based on data supplied from the ultrasonic sensors that can detect an obstacle at a distance.

Practical testing direct of the flight was not yet possible because of missing the Schwitched-mode power supply to 5V and 7V, which would be located also on the airship. The next question is, whether the project Autonomously controlling the airship as a whole piece sustain all of its modules and whether it don‘t need to be manufactured SMD. Theoretically, after calibration, however, the airship should be capable of autonomous flight.

SEZNAM POUŽITÉ LITERATURY

[1] VÁŇA, Vladimír. *Začínáme pracovat s mikrokontroléry HC08 Nitron.* 1. vydání, Praha: BEN - technická literatura, 2003, 96 s, ISBN 80-7300-124-1

[2] Wikipedia. *Elektromotor* [online].[cit. 2010-03-07]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Elektromotor>.

[3] Wikipedia. *Pulsně šířková modulace* [online].[cit. 2010-04-09]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Pulsně šířková modulace>.

[4] Vlastíkd. *Jak fungují modelářská serva* [online].[cit. 2010-03-21]. Dostupné z WWW: < http://vlastikd.webz.cz/bastl/serva.htm >.

[5] JURÁNEK, A., HRABOVSKÝ, M. *EAGLE – uživatelská a referenční příručka.* BEN – technická literatura, Praha 2005.

[6] MC9S08SH8 Data Sheet, Rev.3 Freescale Semiconductor, 2006

[7] ELEKTRO LUKIS. *Výroba plošných spojů pomocí fotocesty* [online].[cit. 2010-05-02]. Dostupné z WWW: <http://www.elektrolukis.estranky.cz/stranka/vyroba-plosnych-spoju-pomoci-fotocesty>.

[8] BASTIAN, Peter. *Praktická elektrotechnika.* 2. vydání. Europa Sobotáles, Brno 2006. 303 s. ISBN 80-86706-15-X.

[9] LADMAN, Josef. *Elektronické konstrukce pro začátečníky.* 1. vydání. BEN – technická literatura, Praha 2002. 144 s. ISBN 978-80-7300-015-8.

[10] *Řízení stejnosměrného motoru* [online].[cit. 2010-04-18]. Dostupné z WWW: < http://www.eurobot.cz/Workshop2006/Rizeni_stejnosmerneho_motoru.ppt >.

[11] POSPÍŠILÍK, M.; ADÁMEK, M. *Autonomous airship*. 2010.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

PWM Pulse Width Modulation

I/O Input/Output

DPS Deska plošného spoje

IDE Integrated Development Environment

CPU Central Processing Unit

TTL Transistor-transistor-logic

A/D Analog/Digital

USB Universal serial bus

TDM Time division multiplex

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: ZDROJOVÝ KÓD

```
#include <hidef.h>          /* for EnableInterrupts macro */
#include "derivative.h"     /* include peripheral declarations */

/* --- Deklarace promenych a funkci --- */
unsigned int zmacknute[8];
unsigned int poleDiod[10];
unsigned int hlavni_pwm = 0;
unsigned int hlavni_pwm_tmp;
unsigned int smerovy_pwm = 0;
unsigned int smer_vpravo;
unsigned int senzory_zapnuto = 1;
unsigned int citac2 = 0;
unsigned int i, j;
unsigned int modulo = 3000;
unsigned int p_nahore, p_dole, p_vlevo, p_vpravo, zpet;

interrupt void kiloHerz(void);
interrupt void ridiciImpuls(void);

/* --- Definujeme globalni promennou jako ukazatel na funkci a to na dane místo pameti
   (na adresu vektoru preruseni) a do teto promenne nastavime adresu nasi funkce --- */
void (*const obsluha) (void) @0xFFE8 = kiloHerz;
void (*const obsluha2) (void) @0xFFE2 = ridiciImpuls;

void main(void) {
/* --- nastaveni portu --- */
    PTBDD = 0;                      // port B jako vstup   (1 = vystup)
    PTBPE = 0;                      // pull-up na portu B vypnuty
    PTCDD = 0xF;                     // 4 bity portu C jako vystup
    PTADD_PTADD2 = 0;                // vstup komunikace
    PTAPE_PTAPE2 = 0;                // pull-up na portu A vypnuty
    PTADD_PTADD1 = 1;               // vystup citac
    PTADD_PTADD0 = 1;                // vystup reset
    PTADD_PTADD3 = 1;                // vystup servo

    /* --- Nastaveni casovace 1 --- */
    TPM1SC = 0x48;                  // source fbus, delicka 1
    TPM1MOD = 8000;                 // modulo registr, TOF perioda 1 ms
    /* --- Nastaveni casovace 2 --- */
    //TPM2SC = 0x4A;                // source fbus, delicka 4
    //TPM2MOD = modulo;             // modulo registr, TOF perioda 1,5 ms

    EnableInterrupts;
    for (;;) {
      if (PTBD_PTBD2 == 0) zmacknute[2] = 0;
      if (PTBD_PTBD3 == 0) zmacknute[3] = 0;
      if (PTBD_PTBD4 == 0) zmacknute[4] = 0;
      if (PTBD_PTBD5 == 0) zmacknute[5] = 0;
```

```
      if (PTBD_PTBD6 == 0) zmacknute[6] = 0;
      if (PTBD_PTBD7 == 0) zmacknute[7] = 0;
/* --- Zjisteni stavu senzoru --- */
      for (i = 0; i < 10; i++) {

         if (PTAD_PTAD2 == 0) {
            poleDiod[i] = 1;
         } else if (PTAD_PTAD2 == 1) {
            poleDiod[i] = 0;
         }
         PTAD_PTAD1 = 1;                  // impuls pro inkrementaci citace
         asm nop;
         PTAD_PTAD1 = 0;
      }
      PTAD_PTAD0 = 1;                     // impuls pro reset citace
      asm nop;
      PTAD_PTAD0 = 0;

/* --- Ultrazvukove senzory aktivni --- */
      if (poleDiod[9] == 0) {                        // napajeni senzoru
         if (senzory_zapnuto != poleDiod[9]) {       // provede se jen jednou po zapnuti
            senzory_zapnuto = poleDiod[9];
            j = 2000;
         }
         // ve kterem smeru se nachazi dioda
         p_nahore = poleDiod[5] + poleDiod[3] + poleDiod[2];      //senzory C,4,8
         p_dole = poleDiod[1] + poleDiod[4] + poleDiod[6];        //senzory A,0,E
         p_vlevo = poleDiod[5] + poleDiod[1] + poleDiod[0];       //senzory C,A,6
         p_vpravo = poleDiod[7] + poleDiod[4] + poleDiod[3];      //senzory F,0,4

/* --- Rozhodovaci logika --- */
         if (p_vpravo > p_vlevo) {
            smer_vpravo = 0;
            smerovy_pwm = 12;
         } else if (p_vpravo < p_vlevo) {
            smer_vpravo = 1;
            smerovy_pwm = 12;
         } else {
            if (poleDiod[8] == 1) {              //blizi se prekazka zepredu, senzor 2
               zpet = 1;
            } else {
               smerovy_pwm = 0;
               zpet = 0;
            }
         }

         if (p_nahore > p_dole) {
            hlavni_pwm = 18;
            modulo = 3600;
         } else if (p_nahore < p_dole) {
```

```
            hlavni_pwm = 18;
            modulo = 2500;
        } else if (zpet == 1) {
            modulo = 1800;
            hlavni_pwm = 20;
            smerovy_pwm = 5;
        } else {
            hlavni_pwm = 10;
            modulo = 3000;
        }
/* --- Senzory nejsou aktivni – Manualni ovladani --- */
    } else {
        if (senzory_zapnuto != poleDiod[9]) {          // provede se jen jednou po vypnuti
            senzory_zapnuto = poleDiod[9];
            modulo = 3000;
            hlavni_pwm = 0;
            smerovy_pwm = 0;

        }

/* --- Jednotliva tlacitka --- */
        if (PTBD_PTBD2 == 1 && zmacknute[2] == 0 && modulo < 4000) {
            modulo = modulo + 100;
            zmacknute[2] = 1;
        }
        if (PTBD_PTBD3 == 1 && zmacknute[3] == 0 && modulo > 2000) {
            modulo = modulo - 100;
            zmacknute[3] = 1;
        }

        if (PTBD_PTBD4 == 1 && zmacknute[4] == 0 && hlavni_pwm < 20) {
            hlavni_pwm = hlavni_pwm + 2;
            zmacknute[4] = 1;
        }

        if (PTBD_PTBD5 == 1 && zmacknute[5] == 0 && hlavni_pwm > 0) {
            hlavni_pwm = hlavni_pwm - 2;
            zmacknute[5] = 1;
        }

        if (PTBD_PTBD6 == 1) { //otaceni doleva
            smerovy_pwm = 20;
            smer_vpravo = 0;
        } else if (PTBD_PTBD7 == 1) { //otaceni doprava
            smerovy_pwm = 20;
            smer_vpravo = 1;
        } else {
            smerovy_pwm = 0;
        }
    }
```

```
      __RESET_WATCHDOG();
   }
}

/* --- Preruseni 1 kHz --- */
interrupt void kiloHerz(void) {
   TPM1SC &= 0x7F;                                          // nuluj priznak preruseni
   if (hlavni_pwm_tmp < hlavni_pwm && hlavni_pwm < 13)      // rozbehovy impuls
      j = 300;
   hlavni_pwm_tmp = hlavni_pwm;
   if (j > 0) {
      j--;
   }

/* --- Hlavni motory PWM --- */
   if (citac2 == 0 && hlavni_pwm > 0)                       // zacatek pulsu
      PTCD_PTCD1 = 1;
   if (citac2 == hlavni_pwm && j == 0)                      // konec pulsu
      PTCD_PTCD1 = 0;

/* --- Smerovy motor PWM a smer --- */
   if (citac2 == 0 && smerovy_pwm > 0) {                    // zacatek pulsu
      if (smer_vpravo == 1) {
         PTCD_PTCD2 = 1;
         //PTCD_PTCD2 = 0;
      } else {
         PTCD_PTCD3 = 1;
         //PTCD_PTCD3 = 0;
      }
   }
   if (citac2 == smerovy_pwm) {                             // konec pulsu
      PTCD_PTCD2 = 0;
      PTCD_PTCD3 = 0;
   }

   citac2++;

   if (citac2 == 20) {                                      // konec periody
      TPM2MOD = modulo;                                     // nastaveni casovace 2
      TPM2SC = 0x4A;                                        // source fbus, delicka 4
      PTAD_PTAD3 = 1;                                       // ridici impuls pro servo
      citac2 = 0;                                           // vynulovani citace
   }
}

interrupt void ridiciImpuls(void) {                         // konec impulsu servu
   TPM2SC = 0;
   PTAD_PTAD3 = 0;
}
```