**České vysoké učení technické v Praze**

Fakulta elektrotechnická

Katedra řízení

# Bakalářská práce

## Sběr dat ze senzorů formulového vozu Cartech FS01

Tomáš Jiřinec

**Vedoucí bakalářské práce: Ing. Lukáš Čarek**

**Studijní program: Elektrotechnika a informatika bakalářský**

**Obor: Kybernetika a měření**

**červenec 2009**

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra řídicí techniky

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Tomáš Jiřinec**

Studijní program: Elektrotechnika a informatika (bakalářský), strukturovaný
Obor: Kybernetika a měření

Název tématu: **Sběr dat ze senzorů formulového vozu Cartech FS01**

Pokyny pro vypracování:

1. Seznamte se s procesory ARM7 LPC21xx a moduly vyvinuté pro soutěž Eurobot.
2. Proveďte rešerši v oblasti senzorů používaných pro závodní účely a vyberte vhodné senzory.
3. Zvažte využití GPS modulu.
4. Navrhněte elektronický modul, který umožní zpracování informací z těchto senzorů.
5. Implementujte základní software a ověřte funkci nasazením ve voze FS01.

Seznam odborné literatury:

Dodá vedoucí práce

Vedoucí: Ing. Lukáš Čarek

Platnost zadání: do konce letního semestru 2009/10

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE Fakulta elektrotechnická L.S.

doc. Ing. Boris Šimák, CSc.
děkan

V Praze dne 13. 3. 2009

**Prohlášení**

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady ( literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne 17.7.09 .....

podpis

## Abstrakt

Sběr dat ze senzorů přestavuje v automobilech důležitou roli. Díky těmto datům je možné udržovat automobil v ideálních pracovních podmínkách, což vede k větší efektivitě, spolehlivosti a bezpečnosti. V moderních závodních autech je běžné sbírat telemetrická data. Tyto data představují důležité vodítko pro mechaniky při hledání optimálního nastavení vozu. Nicméně stejně tak jsou tyto údaje důležité pro jezdce, neboť jim umožňují detailní rozbor jejich činnosti. To umožňuje analyzovat jezdcův jízdní styl a dále ho vylepšovat nebo přizpůsobovat daným podmínkám. Předmětem této práce je navrhnout zařízení pro sběr takovýchto dat.

## Abstract

Data gathering plays an important role in car industry. These data helps to keep the whole in optimal working environment. This leads to improvement in efficiency, reliability and security. In modern race cars telemetry data gathering is wide spread out. These data enables mechanics and engineers to find the optimal setup for the car. However these same data are vital for drivers. It helps them to improve, adopt or compare their driving style. The main topic of this thesis is design of the device which is capable of gathering these data.

## Obsah:

*pozn. Odkazy na použitou literaturu jsou ve tvaru [x], kde x je celé číslo. Odkaz na dokumentaci je v kapitole Seznam použité literatury. Pokud je odkaz zdrojový kód, pak je odpovídající soubor k nalezení v Seznam zdrojových kódů.*

*pozn. Odkazy na internetové adresy jsou ve tvaru [ox], kde x je celé číslo. Odpovídající internetová adresa je v kapitole Seznam odkazů.*

## Seznam obrázků

# Úvod:

Sběr dat z vhodných senzorů závodního vozu vede ke zlepšení spolehlivosti a bezpečnosti vozu. Sběr telemetrických dat je velmi důležitý pro dosažení maximální výkonnosti vozu. Pro závodní inženýry jsou tyto data velmi důležitá při hledání optimálního nastavení, pro jezdce tyto data poskytují možnost analýzy jejich jezdeckého stylu a možnost srovnání s jinými piloty. To vše má za následek zlepšení výkonnosti celého týmu.

Cílem této práce je navrhout zařízení pro sběr vybraných dat. V první části práce se zabývám popisem hardwaru, který pro toto řešení použiji.

V druhé části práce navrhnu schéma samotného zařízeni a jeho plošný spoj.

Ve třetí části práce vytvořím program zajišťující zvolenou funkčnost.

# 1. Výběr hardware

## 1.1 Seznámení s deskou LPCEUROBOT

Jádro celého systému tvoří deska LPCEUROBOT (obrázek 1.1). Deska obsahuje mikrokontrolér LPC2119 (U3), výkonný 32-bitový mikrokontrolér s jádrem ARM7, budič sběrnice CAN PCA82C250 (U4), převodník USB-UART FT232R (U8). Deska má dva stabilizátory napětí : LE50 (U5) - stabilizace na 5V a TPS73HD301 (U6) – stabilizace na 3,3V, které umožňují připojit externí napájení. Ochranu pro dodržení správné polarity napájecího napětí tvoří dioda D6. Pro programování a komunikaci s mikrokontrolérem se využívá rozhraní USB, připojené přes USB MINI konektor (U7). USB rozhraní je pomocí U8 převedeno na UART sériovou linku. Její datové signály Rx a Tx jsou připojeny na modul UART0 mikrokontroléru. Signál DTS je využit pro resetování mikrokontroléru (resetuje U6). Desku je možné přes tento konektor napájet, pro oddělení 5V potenciálu desky a USB portu je použita dioda D8. Dále deska obsahuje 5 LED diod, D2-D5 jsou přímo připojeny k výstupům mikrokontroléru, LED dioda D7 signalizuje příjem dat mikrokontrolérem z USB. Mikrokontrolér je taktovaný krystalem U2 o frekvenci 14,745 MHz. Na desce dál najdeme pomocné rezistory, stabilizační a blokovací kondenzátory. Na pin lišty J16 a J17 jsou vyvedeny všechny vstupy a výstupy mikrokontroléru, uzemnění desky, stabilizovaná napětí 5V a 3,3V a také vstup pro napájecí napětí.

Samotný mikrokontrolér má 46 vstupně/výstupních pinů. 30 na PORTu 0 a 16 na PORTu 1. Dále má periférie pro sériovou komunikaci UART, SPI, $I^2C$, CAN, vše po dvou kanálech, čtyřkanálový 10-ti bitový A/D převodník, hodiny reálného času RTC a dva 32-bitové čítače/časovače, umožňující funkce capture, match a pulsně šířkovou modulaci PWM.

Bohužel se mi nepodařilo na toto místo vložit schéma LPCEUROBOT desky. Při vkládání pdf stránky na toto místo, došlo k pokažení diakritiki ve zbytku dokumentu. Stránka která sem patří je v souboru **dokumentace/SCHEMATIC_LPC2129_mainb.pdf**

*Obrázek 1.1 - schéma desky LPCEUROBOT*

*Pozn. Výše uvedené schéma jsem měl k dispozici pouze ve formátu pdf. Bohužel se mi nepodařilo vložit jeho popisek přímo k němu.*

## 1.2 Výběr vhodných senzorů

Aby bylo možné analyzovat jízdu na závodním okruhu, má jednotka měřit následující data:

- natočení volantu
- natočení plynového a brzdového pedálu
- zrychlení v příčném a podélném směru
- otáčky motoru
- okamžitou rychlost
- pozici na okruhu

Pro možnosti diagnostiky má jednotka měřit:

- teplotu oleje
- tlak oleje
- teplotu chladící kapaliny
- tlak brzdové kapaliny

### 1.2.1 Natočení pedálů a volantu:

Vzhledem k málo častému používání auta jsem pro snímání natočení volantu, plynového a brzdového pedálu se rozhodl použít potenciometry. Toto řešení je velmi jednoduché, ale na druhé straně je potřeba přihlédnout k prostředí, kde budou potenciometry používány. V závodním, otevřeném autě budou tyto senzory vystaveny velkému množství vibrací, nečistot a také nebezpečí v podobě vlhkosti. Abych zabezpečil spolehlivost těchto senzorů za výše uvedených podmínek, rozhodl jsem se použít potenciometry MP20IP [1] americké společnosti P3 America [o1]. Tyto potenciometry splňují normu IP65 [o2] a mají dobrou odolnost proti vibracím.

*Obrázek 1.2 - potenciometr mp20ip*

### 1.2.2 Zrychlení v příčném a podélném směru

Pro měření zrychlení ve dvou směrech jsem se rozhodl použít integrovaný obvod s několika akcelerometry a jejich podpůrnými obvody. Při hledání vhodných obvodů jsem se rozhodoval mezi produkty firem Analog Devices [o3] a STMicroelectronics [o4]. Nakonec jsem se rozhodl pro obvod LIS3LV02DQ [2] druhé výše jmenované firmy. Jde o tříosý akcelerometr s digitálním rozhraním $I^2C$ nebo SPI napájený až 3,6V a měřící v rozsazích ±2g/4g/8g. Jediným problémem je pouzdro standartu QFN, které není snadné ručně pájet. Bohužel všechny ostatní podobné akcelerometry jsou ve stejných pouzdrech nebo v pouzdru LGA, které ručně pájet není o nic snazší.

### 1.2.3 Otáčky motoru

Pro měření otáček motoru je použit signál vystupující z řídící jednotky. V době výběru senzorů a navrhování plošného spoje nebyla známa přesná specifikace tohoto signálu. Věděl jsem pouze, že má jít o obdélníkový signál, modulovaný pulsně šířkovou modulací. Demodulaci tohoto signálu jsem se rozhodl implementovat do programu mikrokontroléru.

### 1.2.4 Okamžitá rychlost a pozice na okruhu

Viz kapitola 1.3 – Použití GPS

### 1.2.5 Teplota oleje

Měření teploty oleje je dobrý způsob měření teploty motoru. U běžných čtyřdobých spalovacích motorů se teplota pohybuje okolo 100°C. U vysokootáčkového závodního motoru bude provozní teplota vyšší. Pro měření teploty jsem proto vybral čidlo RTD-850 [3] od firmy Omega [o5]. Toto robustní čidlo má maximální teplotu 230°C.

*Obrázek 1.3 – senzor RTD-850*

### 1.2.6 Tlak oleje

Měření tlaku oleje vyžaduje tlakový senzor, který vydrží vysoké teploty tohoto média. Bohužel žádný tlakový senzor od firmy Omega nesplnil tyto požadavky. Nicméně našel jsem firmu Kulite [o6], která se specializuje na výrobu tlakových čidel a jedno z jejích odvětví působení je automobilový průmysl a motor sport. V jejím sortimentu je velké množství tlakových senzorů dostatečně robustních pro nasazení v takovýchto podmínkách. Pro své potřeby jsem zvolil tlakové čidlo XTEL-312 [4], které má teplotní pracovní rozsah -55-232°C.

*Obrázek 1.4 – senzor XTEL-312*

### 1.2.7 Teplota chladící kapaliny

Pro měření teploty chladící kapaliny jsem rozhodl použít opět čidlo RTD-850.

### 1.2.8 Tlak brzdové kapaliny

Pro tento účel jsem zvolil senzor XTL-123B [5] od firmy Kulite. Hlavní výhoda tohoto senzoru je garantovaná „kompatibilita“ s veškerými kapalina používanými v autech.

*Obrázek 1.5 – senzor XTL-123B*

## 1.3 Použití GPS

Pro měření okamžité rychlosti a vyhodnocení pozice auta na okruhu je možné pohodlně použít GPS přijímač. Aby bylo možné dobře analyzovat závodní stopu a další telemetrická data, bylo by vhodné mít vzorkován alespoň každý metr. Pokud budeme přihlížet k maximální rychlosti pouze 250 $kmh^{-1}$, tak pro splnění této podmínky potřebuji vzorkovací frekvenci:

$$f_{vz} = \frac{v}{d} = \frac{250}{3{,}6 \cdot 1} \approx 70\ Hz$$

kde $v$ – maximální rychlost

$d$ – úsek na kterém chci vzorkovat

Přesnost měření by měla být alespoň 10 cm.

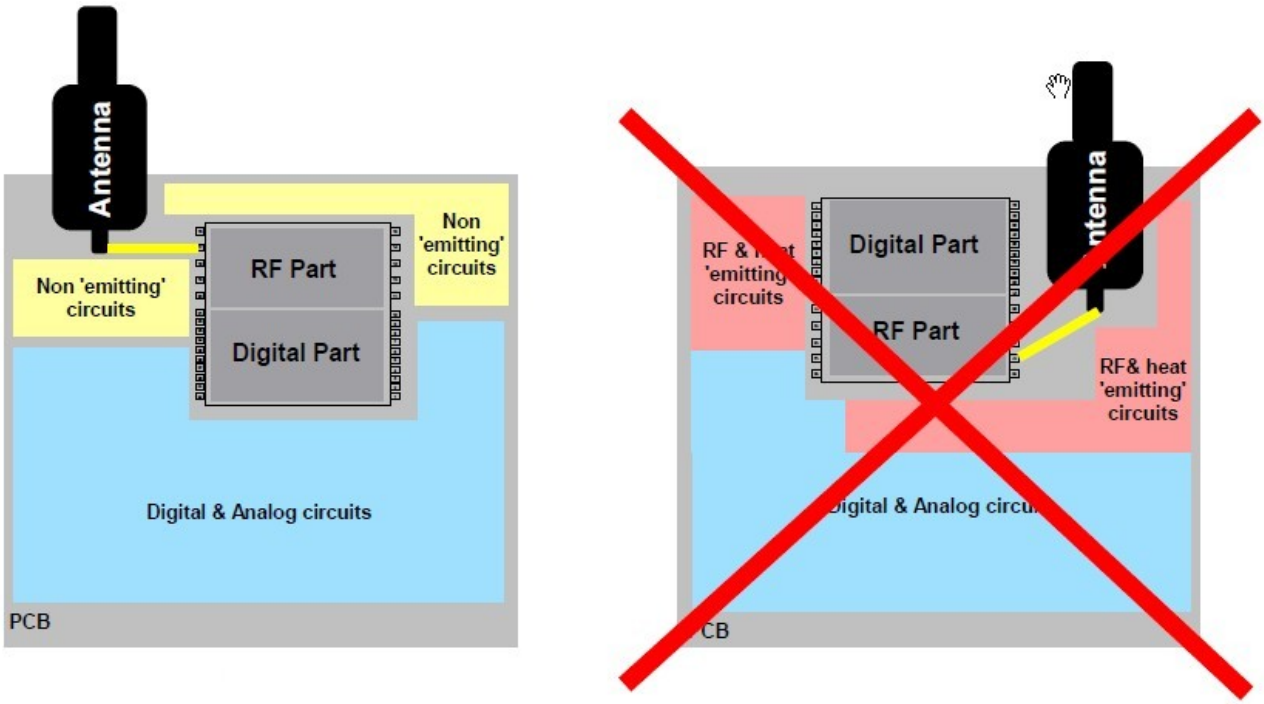
Při hledání vhodného, běžně dostupného modulu jsem ovšem narazil právě na tyto dvě omezující kritéria: nízkou vzorkovací frekvenci polohy (2-4Hz) a nedostatečnou přesnost (cca 2 m). [6] [7]

Řešení založená na GPS se v praxi používají. Například britská firma Racelogic [o7], nabízí zařízení VBOX [8], které má vzorkovací frekvenci až 100Hz a přesnost až 2 cm. Po podrobnějším prozkoumání jsem zjistil, že tato velmi vysoká přesnost je dosahována díky použití dalšího vysílače v blízkosti závodní dráhy.

Použití GPS mě přesto zaujalo a rozhodl jsem se udělat drobnou rešerši o instalaci GPS modulu do aplikace. Rozhodl jsem se použít modul TIM-5H [6] firmy ublox [o8].

GPS modul je obvod velmi citlivý na rušení a pro jeho správný chod je potřeba dodržet několik zásad při navrhování plošného spoje. Níže uvedené zásady platí přímo pro modul TIM-5H, nicméně podobná pravidla je potřeba dodržet i u jiných podobných zařízení. Informace byly čerpány z [9].

### 1.3.1 Vhodné umístění modulu

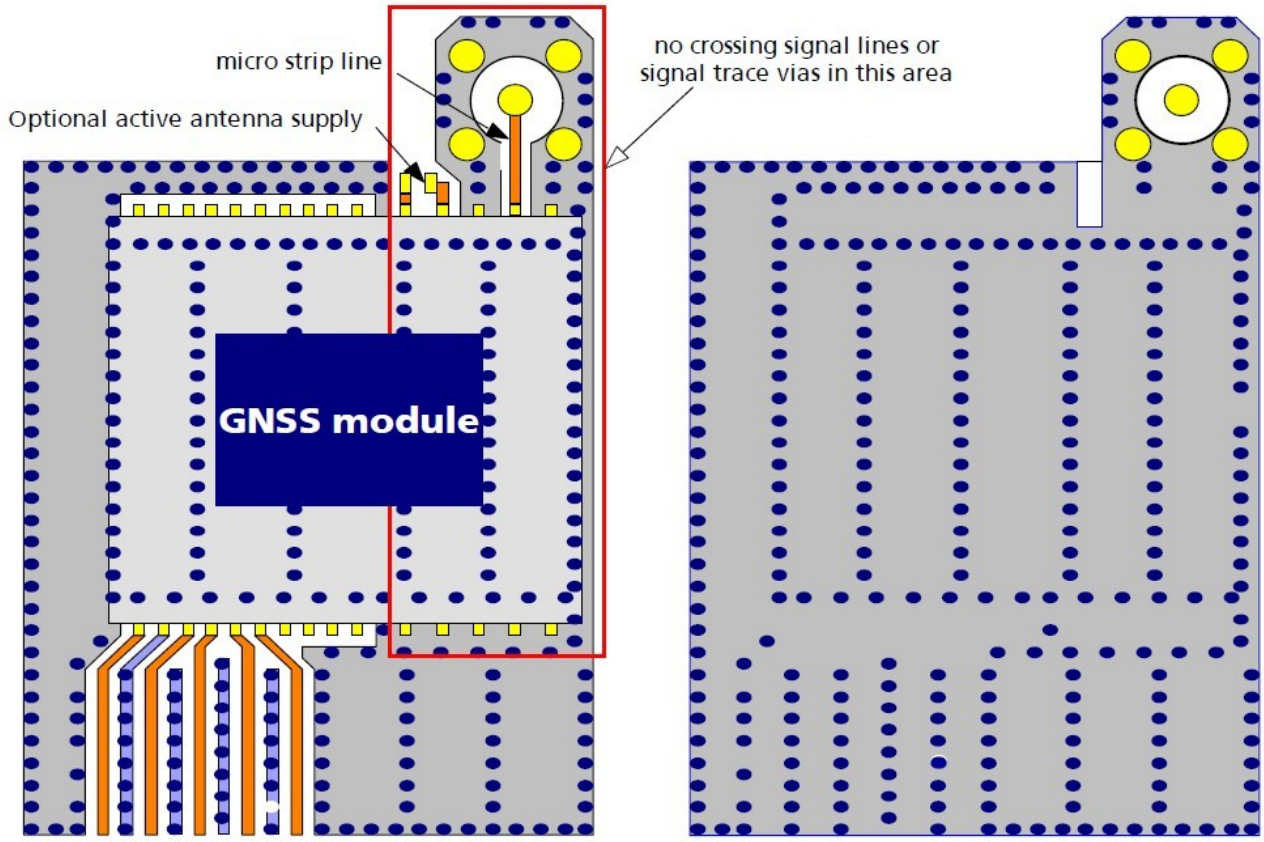



Modul TIM-5H je rozdělen na dvě části – digitální část a vf analogovou část. Při umisťování modulu na desku plošného spoje je potřeba umístit analogovou část co nejdále od jakýchkoliv zdrojů rušení. Dále je potřeba zajistit teplotní stabilitu analogové části, tzn. neumisťovat ji do blízkosti zdrojů tepla nebo naopak do míst, kde hrozí prudké ochlazování (např. proud vzduchu od ventilátorů). *Obrázek 1.6 – orientace GPS modulu*

### 1.3.2 Stínění modulu

Pro správnou funkci je potřeba zajistit dobré odstínění modulu od rušivých signálů. Všude v okolí modulu, digitálních linek sloužících ke komunikaci s modulem a konektoru pro připojení antény je nutné rozlít měď a propojit se zemí. Takovéto stínění udělat jak v BOTTOM vrstvě (modul je umístěn v TOP vrstvě), tak v TOP vrstvě. Dále je nutné tyto zemnící pláty hustě spojit prokovy, obzvlášť na jejich obvodu. Pro zlepšení spolehlivosti je vhodné prokovy v TOP vrstvě, které se nacházejí přímo pod modulem, zakrýt nepájivou maskou. Pod analogovou části modulu,

anténním konektorem a anténním spojem nesmí vést žádné signály a je nutné zajistit, že stínícími pláty nepotečou zemní proudy od digitálních obvodů, které by mohly modul rušit.

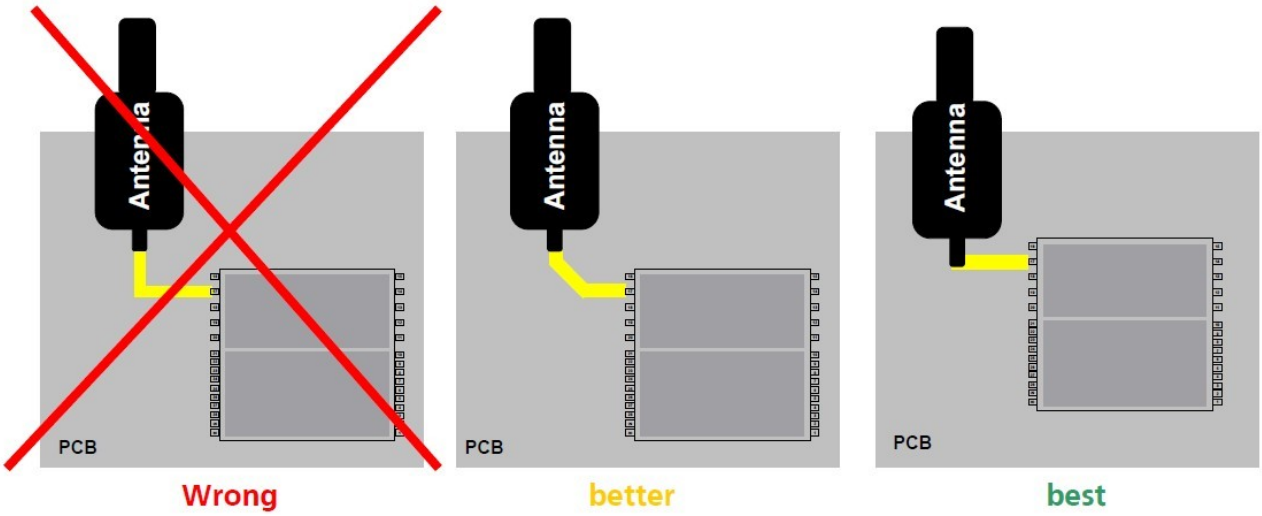



*Obrázek 1.7 – stínění GPS modulu*

### 1.3.3 Anténní spoj

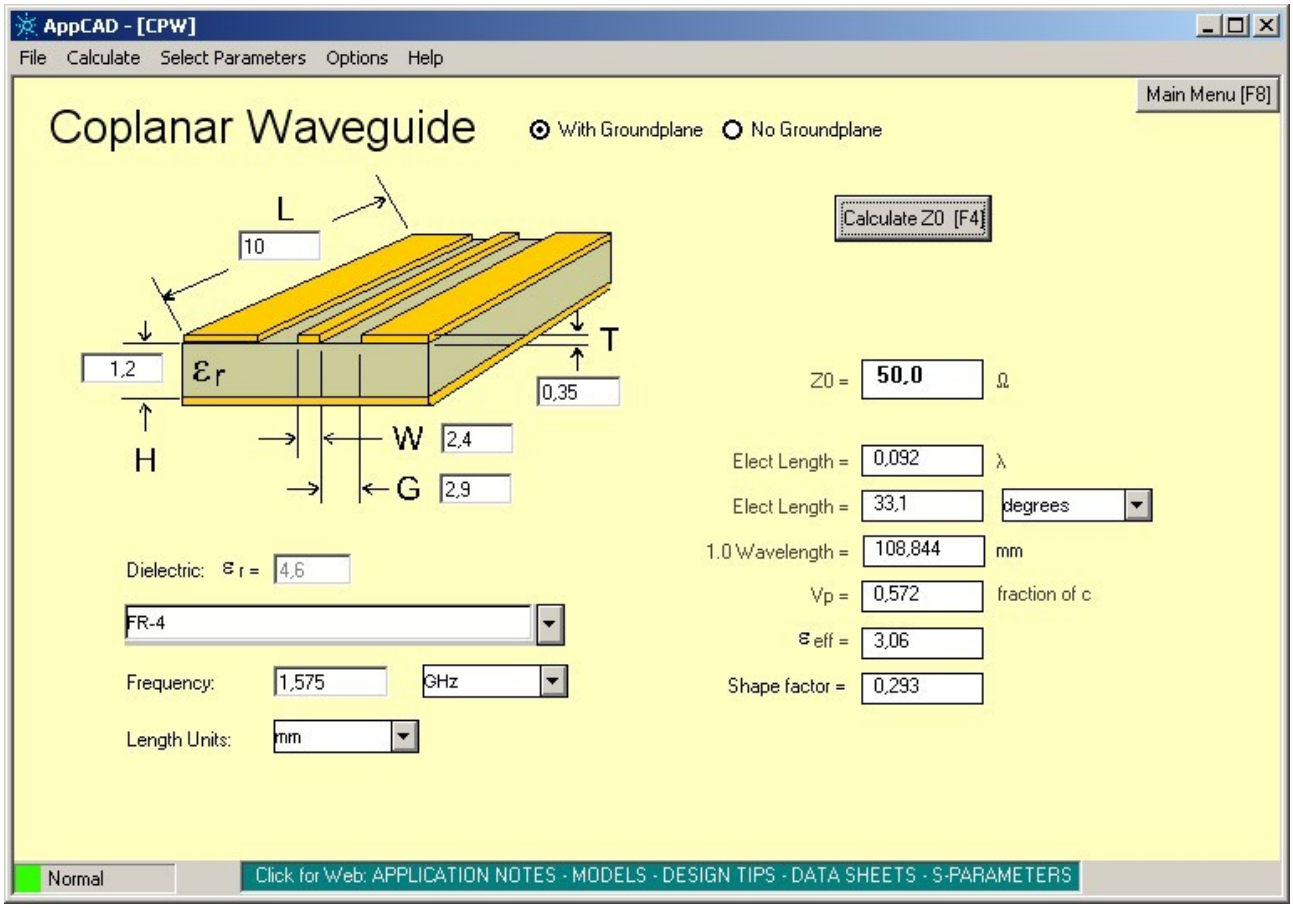
Pro zaručení správné funkce modulu musí být anténní spoj co nejkratší (max 2,5 cm), mít vhodný tvar a mít vhodnou impedanci. Anténní spoj nesmí být veden pod modulem.

Pro omezení napěťových a proudových odrazů je nutné navrhnout anténní spoj co nejrovnější.




*Obrázek 1.8 – vedení anténního spoje*

Pro zajištění správné impedance vodiče 50Ω je nutné tento spoj navrhnout jako vlnovod. Pro správný výpočet parametrů vlnovodu je možné použít například freewarový program AppCAD [o9] a jeho funkci Coplanar Waveguide.




*Obrázek 1.9 – AppCAD a výpočet vlnovodu*

## 2. Schéma a plošný spoj

*Pozn. Při navrhování obvodu ještě nebylo zcela jasné jaké senzory budou použity, proto jsem zapojení dělal pokud možno co nejuniverzálnější, aby ho bylo později možné přizpůsobit pro široké spektrum účelů.*

*Pozn. Pro návrh schématu a plošného spoje byl použit program EAGLE 5.6.0 Light [o10]*

Požadavky na plošný spoj:

- umožnit připojení a napájení desky LPCEUROBOT
- zajistit dostatečný počet analogových vstupů pro připojení analogových senzorů
- zajistit vstupy umožňující demodulaci PWM signálu
- zajistit měření zrychlení pomocí akcelerometru LIS3LV02DQ
- umožnit připojení dvou CANových sběrnic
- mít k dispozici několik vstupů a výstupů, které by mohly být někdy užitečné
- dodržet maximální rozměr 10x7 cm
- umožnit přimontování pomocí čtyř 3mm šroubů

### 2.1 Popis zapojení

Deska LPCEUROBOT je s mým plošným spojem propojena pomocí konektorů SV1 a SV2. Přes tyto konektory jsou desky napájeny a těmito konektory probíhá veškerá komunikace mezi deskami.
Veškeré vodiče jsou k desce připojeny pomocí svorkovnic.
Deska je napájena z automobilového 12V rozvodu. Toto napětí je rozvedeno po desce na několik svorek, což umožňuje snadné připojení napájení senzorů, nebo jiných zařízení. Dále toto napětí vede na pin 34 konektoru SV1. Tento konektor obstarává napájení desky LPCEUROBOT. Zbytek desky je napájen z z 5V stabilizátoru LE50, který je umístěn na desce LPCEUROBOT. Součástky vyžadující nižší napájení využívají 3,3V stabilizátor IC3, který je napájen z výše zmíněné 5V větve.

Deska poskytuje osm analogových stupů. Vstupy na svorkách X6-2, X7-2 a X8-2 jsou určeny pro připojení potenciometrů. Tyto vstupy jsou dále vedeny na ochranný obvod složený z schottkyho

dvojité diody BAT54S, odporu a kondenzátoru (pro vstup X6-2 jsou to součástky D1, R1 a C1). Dále jsou tyto vstupy připojeny na analogový multiplexer IC1.
Zbývajících pět analogových vstupů na svorkách X9-2, X10-2, X11-2, X12-2 a X13-2 jsou každý zvlášť vedeny na odporový dělič umožňující převod napěťové úrovně. Pak každý z těchto vstupů pokračuje na již výše zmíněný ochranný obvod a nakonec je připojen k multiplexeru IC1.

Obvod IC1 má za úkol multiplex analogových kanálů. Je napájen napětím 5V, jeho výstup je připojen na pin 14 konektoru SV1. Tento signál dále vede na pin P0.27/AIN0 mikrokontroléru, což je jeden ze čtyř kanálů integrovaného A/D převodníku. Multiplexer je ovládán piny 9, 10, 11, které jsou připojeny na piny 19, 23 a 25 konektoru SV2 a dále na piny P1.16, P1.18 a P1.22 mikrokontroléru. Pin INH multiplexeru je trvale připojen k zemi, což zajišťuje trvalý provoz multiplexeru.

Svorky X14-1 a X15-1 slouží k demodulaci PWM signálu. Stejně jako pět výše jmenovaných analogových signálů jsou ty to signály přivedeny na odporový dělič, dále na ochranný obvod a nakonec jsou přivedeny na piny 7 a 8 konektoru SV1. Zde jsou signály vedeny na piny P0.22 respektive P0.21, který umožňují funkci CAPTURE čítače TIMER1 respektive TIMER0.

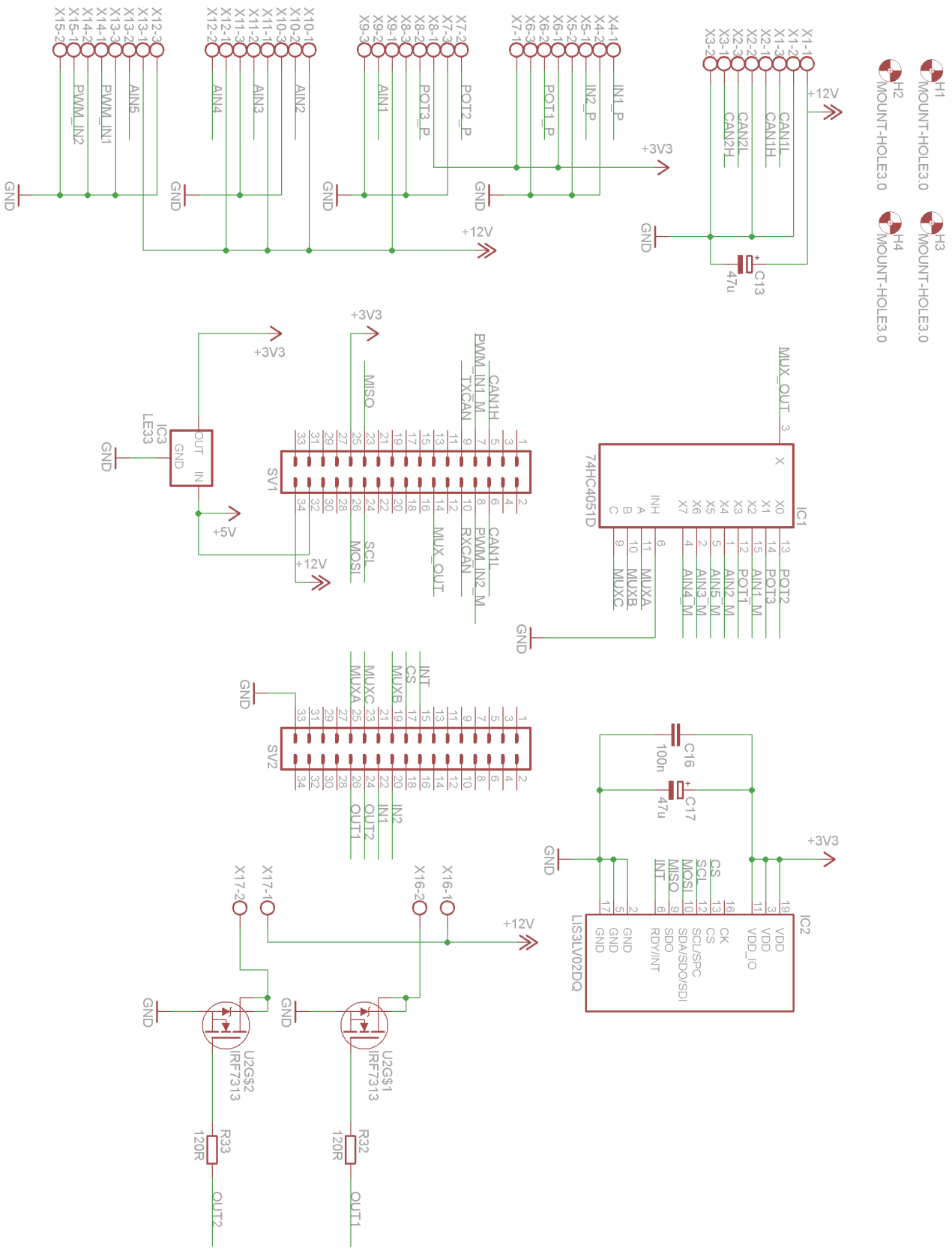
Akcelerometr LIS3LV02DQ (IC2) je napájen z 3,3V větve. Pro komunikaci s mikrokontrolérem je využívána třívodičová sběrnice SPI. Ta je připojena k odpovídajícím pinům SPI0 periferie mikrokontroléru. Dále je k pinu P1.24 mikrokontroléru připojen pin CS, kterým se zahajuje a ukončuje přenos po SPI. Signál RDY/INT akcelerometru je připojen k pinu P0.15/EINT0 mikrokontroléru. Tento signál umožňuje vyvolat přerušení v případě nově naměřených dat, nebo pokud se akcelerometr dostane do stavu nulového zrychlení – volného pádu. Za zmínku stojí ještě připojení 3,3V napětí k pinu 25 konektoru SV1. Tento pin je připojen k pinu P0.7/SSEL0. Funkce tohoto pinu je výběr role master/slave pro mikrokontrolér při SPI komunikaci. Připojení logické jedničky na tento pin vybere mikrokontrolér do režimu master.

Rozhraní CAN1 vedoucí ze svorek X1-3 a X2-1 je přímo vedeno na piny 5 a 6 konektoru SV1, odtud rovnou na budič PCA82C250, který je umístěn na desce LPCEUROBOT. Dále jsou tyto signály přivedeny na CAN0 periférii mikrokontroléru. Rozhraní CAN2 vede ze svorek X2-3 a X3-1 na budič PCA82C250 (U1), který je umístěn na mé desce. Odtud vedou signály na piny 9 a 10 konektoru SV1 a dále přímo na rozhraní CAN1 mikrokontroléru.

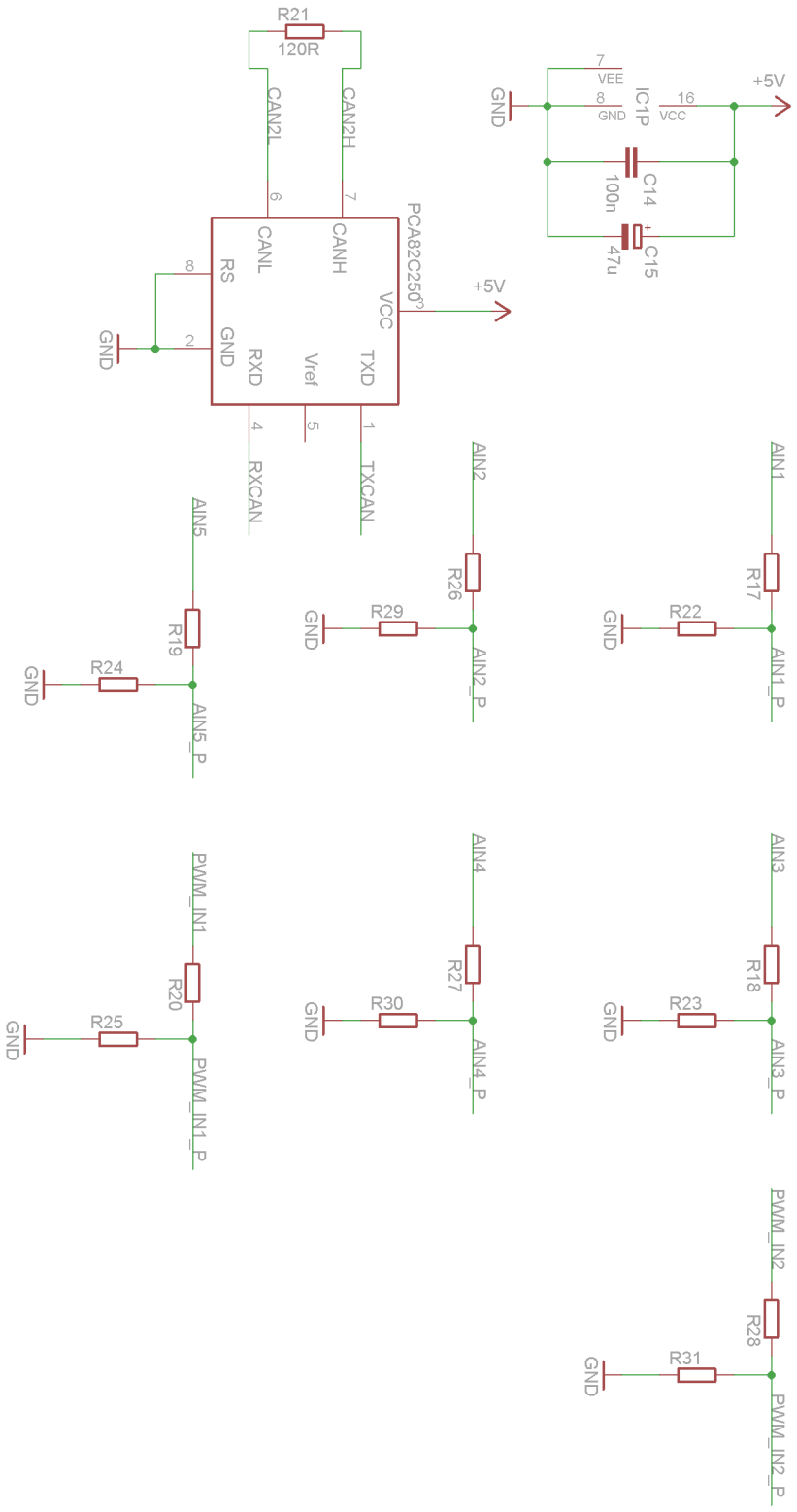
Svorky X4-1 a X5-1 jsou připraveny jako digitální vstupy k jakémukoliv použití. Signály jsou vedeny přes odporový dělič a dále přes výše zmíněný ochranný obvod. Nakonec vedou přes piny 20 a 22 konektoru SV2 na piny P1.21 a P1.23 mikrokontroléru.

Svorky X16-2 a X17-2 slouží jako výstupy typu otevřený kolektor. Výkonové spínání řeší obvod IRF7313 (U2), což je dvojice MOSFET tranzistorů. Každý je schopen spínat proud 6,5A při napětí až 30V. Hlavní výhoda těchto transistorů je nízké spínací napětí gate, které je 1V. Pro snazší připojení zátěže je na sousedních svorkách X16-1 a X17-1 vyvedeno napájecí napětí 12V.

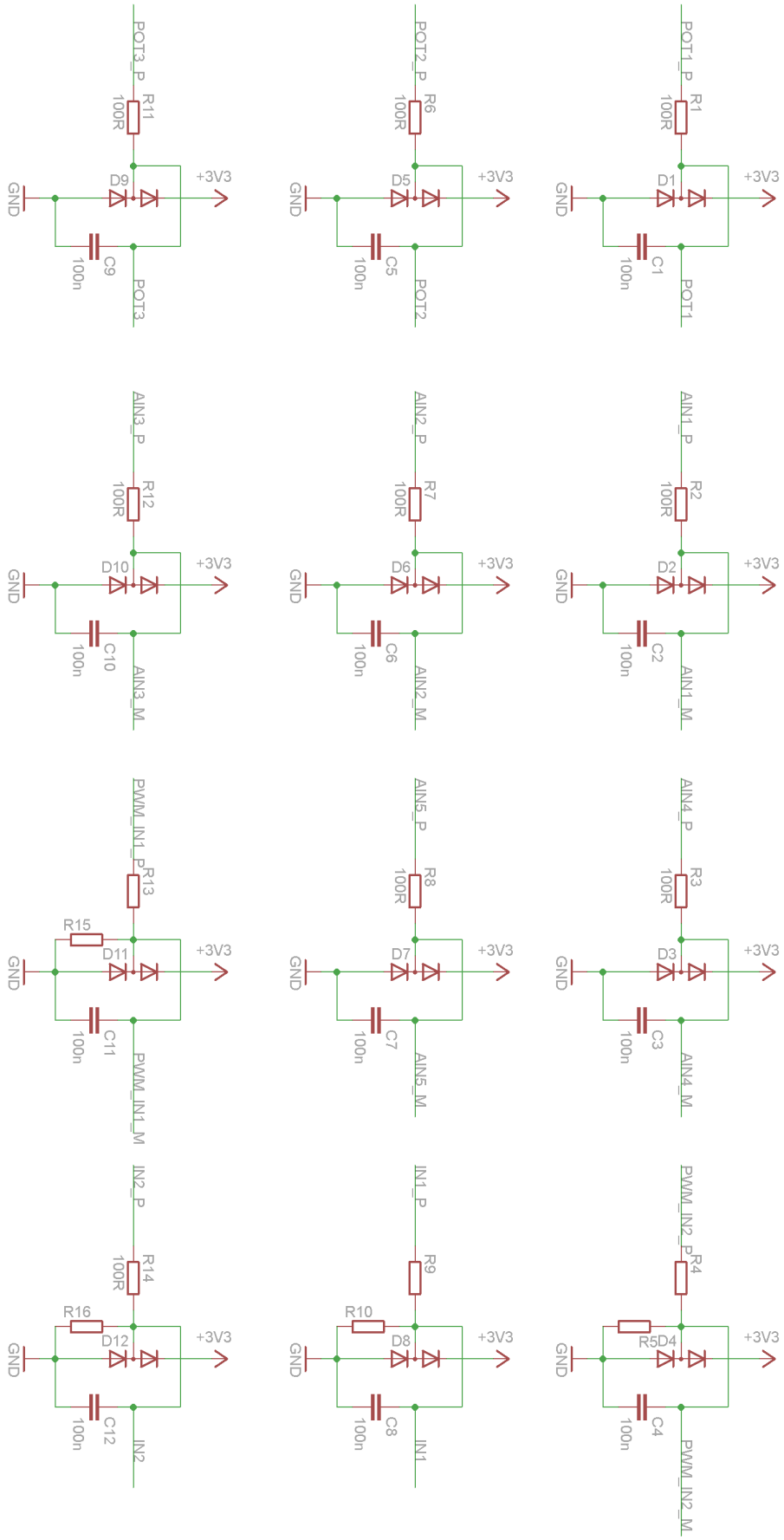
Dále deska obsahuje několik kondenzátorů pro stabilizaci a blokování napětí.




*Obrázek 2.1 – schéma zapojení 1. část*



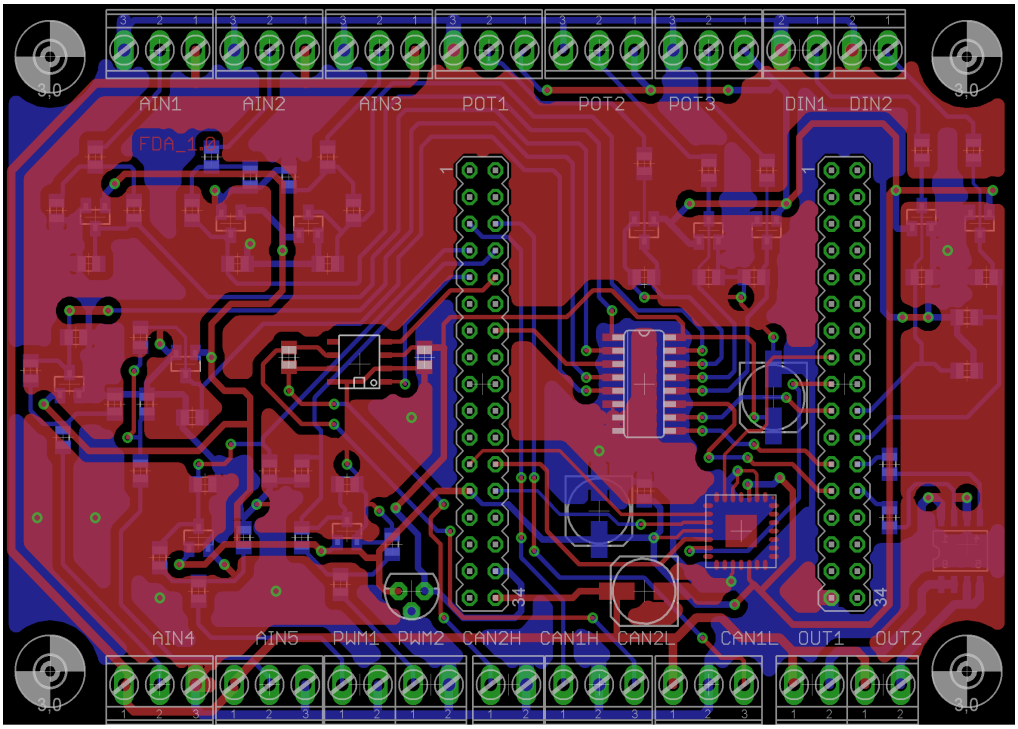

*Obrázek 2.2 – schéma zapojení 2. část*




*Obrázek 2.3 – schéma zapojení 3. část*

## 2.2 Plošní spoj

Plošný spoj je rozměru 10x7 cm. U vodičů, které budou více proudově namáhány, jsem volil odpovídající šířku. Připevnění je realizováno přes čtyři otvory průměru 3 mm v, které jsou umístěny v jeho rozích. Rozpis součástek a jejich pouzder jsou v kapitole *Výčet součástek a jejich pouzder*. Pro zlepšení odolnosti proti rušení jsem rozlil měď po obou stranách plošného spoje. Tyto plochy jsem následně připojil na zem. Výrobní podklady jsem exportoval pomocí CAM procesoru EXCELLON (vrtačka) a Extended Gerber (zbylé vrstvy). Desku zhotovila firma PragoBoard s.r.o. [o11]




*Obrázek 2.4 – plošný spoj*

## 2.3. Osazení plošného spoje

Bohužel kvůli nedostatku času a pracovním povinnostem jsem musel své působení v projektu CarTech ukončit. Pro demonstraci funkčnosti mého návrhu jsem se rozhodl pro následující funkčnost.

- Měření procentuálního natočení tří potenciometrů
- měření teploty pomocí NTC termistoru
- měření zrychlení pomocí akcelerometru LIS3LV02DQ
- demodulace PWM signálu – určení střídy signálu
- spínání odporové zátěže pomocí U2
- čtení dat a ovládání výstupů přes USB

Pro dosažení požadované funkčnosti jsem desku osadil následujícími součástky – *viz kapitola Výčet použitých součástek a pouzder*

Pájení pouzdra QFN obvodu LIS3LV02DQ:
Protože s pájením tohoto pouzdra jsem neměl žádnou zkušenost, konzultoval jsem tento problém s Doc. Ing. Janem Urbánkem Csc. z katedry elektrotechnologie. Na jeho doporučení jsem se rozhodl pro pájení horkým vzduchem. Za pomoci asistentky Ing. Ivany Beshajové-Pelikánové Ph.D. z téže katedry jsem se pokusil dané pouzdro zapájet. Při dodržení postupu v [10] se bohužel pájecí pasta nepřetavila a pouzdro zůstalo nepřipájené. Po zvýšení teploty se mi nakonec pouzdro zapájet podařilo, bohužel za cenu zničení obvodu a pájecích plošek na plošném spoji. Proto jsem se rozhodl zničené pouzdro odpájet a měření zrychlení vypustit.

# 3. Programování LPC2119

## 3.1. Úvod

Na stránce [o12] jsou informace pro instalaci a používání překladače, knihoven a nástroje pro nahrávání programu do mikrokontroléru . Program jsem psal v jazyce C a kompiloval pod operačním systémem Ubuntu 9.04 Jaunty Jackalope [o13].

Požadavky na program:

- Měření analogových vstupů pomocí vhodně nastaveného A/D převodníku a správného ovládání multiplexeru IC1
- Demodulace PWM signálu a určení jeho střídy.
- Konverze naměřených dat do vhodného formátu a jejich pravidelné odesílání přes USB. Data mají být zobrazitelná přímo na terminálu
- Ovládání obvodu UC2 pomocí jednoduchých příkazů zasílaných z terminálu přes USB

*Pozn. Pro napsání některých vlastních knihoven jsem vycházel z již napsaných zdrojových kódů. Kdykoliv je tak učiněno, je o tom zmínka v textu. Výjimku tvoří knihovna lpc21xx.h [11] a types.h [12]. První jmenovaná knihovna obsahuje definice registrů mikrokontroléru. Druhá obsahuje definici typů proměnných. Také neuvádím použití standardních knihoven jazyka C. Jejich použití je jasně patrné ze zdrojových kódů.*

## 3.2 UART:

Pro umožnění komunikace desky LPCEUROBOT přes rozhraní USB je deska vybavena převodníkem FT232R, který umožňuje převod USB↔UART. Tento převodník je připojen k rozhraní UART0 mikrokontroléru.

Pro zajištění správné komunikace jsem použil knihovnu uart.h [13], která byla použita v projektu eurorobot. Tato knihovna poskytuje následující funkce:

```
void init_uart0(int baudrate, char bits, char stopbit, char parit_en, char parit_mode );
```

*Zajišťující inicializaci periférie UART0 pro zadané parametry.*

```
unsigned char uart0GetCh(void);
```

*Přijme jeden znak vyslaný po sériové lince na UART0.*

```
void uart0SendCh(char ch);
```

*Vyšle jeden znak přes UART0.*

Pro mojí potřebu jsem připsal následující funkce:

```
void uart0SendString(const char *s);
```
*Odešle celý string.*

```
void int_to_string(char **s, int i);
```
*Převede integer na řetězec reprezentující ho v desítkové soustavě.*

*Moje knihovna uart.h je k nalezení zde [14].*

### 3.3 A/D převodník

Dále bylo potřeba napsat algoritmus, který bude pravidelně přepínat multiplexer IC1 a obstarávat správnou funkci A/D převodníku. Opět jsem použil knihovnu z projektu eurorobot. Použitou knihovnu adc.h [15] jsem ovšem velmi výrazně upravil.

```
#define ADC_MUL         1///< This value multiplies mesured value from ADC
#define ADC_OFFSET      0///< This value is added to mesured value from ADC

#define ADC_VPB_DIV     255   ///< VPB divisor for ADC

#define ADC_MUX_CHANNEL_ON    0xFFFFFFFF
#define ADC_MUX_CHANNEL_OFF   0
#define MUX_CHANNEL_COUNT     8

#define ADC_CHANNEL     0           //adc channel which is used for conversion

#define MUXA      16                //channel selection for multiplexer – bit 0
#define MUXB      22                //channel selection for multiplexer – bit 1
#define MUXC      18                //channel selection for multiplexer – bit 2

volatile unsigned int adc_val[MUX_CHANNEL_COUNT];
volatile unsigned int mux_sel[MUX_CHANNEL_COUNT];

/**   inicializes ADC lines and starts converion (use ISR)
 *  @param rx_isr_vect  ISR vector
 */
void init_adc(unsigned rx_isr_vect);
```

*Kód 3.1 – část adc_mux.h*

Moje knihovna je zde [16]

Pro použití této knihovny je potřeba nejdříve nastavit kanály multiplexeru IC1, na kterých chci měřit. K tomu použiji pole adc_sel[]. Do jeho políček vložím hodnoty ADC_CHANNEL_ON/ADC_CHANNEL_OFF. Pak už stačí jen zavolat funkci init_adc() a předat jí číslo

vektoru přerušení, který chci pro A/D převod zarezervovat. Funkce zajistí správné nastavení pinů, aktivace přerušení a jeho obslužné rutiny. Pak již stačí vyzvedávat naměřené hodnoty z pole adc_val[].

Příklad inicializace A/D převodu pro kanály 0, 1, 3 a 6 a vektor přerušení 1.

```
mux_sel[0] = ADC_MUX_CHANNEL_ON;
mux_sel[1] = ADC_MUX_CHANNEL_ON;
mux_sel[3] = ADC_MUX_CHANNEL_ON;
mux_sel[6] = ADC_MUX_CHANNEL_ON;
init_adc(1);
```

*Kód 3.2 – příklad nastavení a/d převodníku a přepínání multiplexeru*

### 3.4 Demodulace PWM signálu

Pro určení střídy PWM signálu jsem použil capture funkci TIMERu1. Při hraně vstupního signálu se vyvolá přerušení. Takto se měří doby, kdy má signál vysokou úroveň a kdy má signál nízkou úroveň. Potom se snadno spočítá střída vstupního signálu.

*Pozn. Makro BIT() je převzato od autora sysless-frameworku. Toto makro lze najít zde [17]*

```
IO0CLR = BIT(21);
IO0DIR &= ~BIT(21);                 //set output to zero and switch to input
PINSEL1 |= BIT(10) | BIT(11);       //set capture function for p0.22

//set t1 interupt handler
((uint32_t*)&VICVectAddr0)[depwm_isr_vector] = (unsigned)depwm_isr_timer1;

//assign t1 interupt to the handler
((uint32_t*)&VICVectCntl0)[depwm_isr_vector] = BIT(5) | T1_IRQ;
VICIntEnable |= BIT(T1_IRQ);        //enable t1 interupt

//select rising edge triggering and enable interupt generation
T1CCR |= BIT(9) | BIT(11);
T1TCR = BIT(1);                     //reset t1
T1TCR = BIT(0);                     //start t1
```

*Kód 3.3 – nastavení pinů a přerušení pro timer1*

```
if (T1CCR & BIT(9)){                //if rising edge occurs
      t1_low = T1TC;                //save value of timer
      T1TC = 0;                     //reset timer

      duty_cycle_t1 = t1_high / (t1_high + t1_low);//calcula tethe duty cycle
      period_t1 = (t1_high + t1_low);

      T1CCR &= ~BIT(9);             //toggle edges, rising -> falling
      T1CCR |= BIT(10);
      T1IR |= BIT(7);               //clear interupt flag

}else{
      t1_high = T1TC;               //falling edge occurs
      T1TC = 0;
      T1CCR &= ~BIT(10);
```

```
	T1CCR |= BIT(9);
	T1IR |= BIT(7);
}

VICVectAddr = 0;
```

*Kód 3.4 – isr rutina pro timer1*

Moje knihovna pro demodulaci PWM signálu [18]

## 3.5 Odesílání dat

Pro konverzi dat a jejich odesílání jsem napsal funkci send_data(), která je uložena ve stejnojmenném souboru [19].

```
int ntc_voltage[] = {19, 27, 36, 48, 64, 83, 107, 135, 168, 206, 248, 295,
344, 395, 448, 500, 550, 598, 643, 684, 722, 755, 785, 811, 835, 855, 872,
888, 901,    912, 922, 931, 938, 945, 950, 955, 959, 963, 966};

int ntc_temperature[] = {-40, -35, -30, -25, -20, -15, -10, -5, 0, 5, 10, 15,
20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100, 105, 110,
115, 120, 125, 130, 135, 140, 145, 150};

int volt_to_temp(int voltage)
{
	int i;
	for (i = 0; i < 39; i++)
		if (voltage < ntc_voltage[i]) break;

	if (ntc_voltage[i] - voltage > voltage - ntc_voltage[i - 1]) i--;

	return ntc_temperature[i];
}
```

*Kód 3.5 – konverze napětí ntc termistoru na tepltu*

Hodnoty ntc_voltage[] byli získány 10-ti bitovým převodem napětí vypočteného na děliči složeném z rezistorů R18, R23 a ntc rezistoru.

### 3.6 Ovládání výstupu

Poslední knihovnu kterou používám je powswitch.h [20], která obsluhuje spínání U2 a nastavování stavových informacích o sepnutých výstupech.

```
const char* switch_status[2];

// initializes power switch
void init_pow_switch(void);


/* switch on
* @param switch can be 1 or 2 to choose desired output
*/
void pow_switch_on(int switch_number);

/*switch off
* @param switch can be 1 or 2 to choose desired output
*/
void pow_switch_off(int switch_number);
```

*Kód 3.6 – ukázka headeru powswitch .h*

### 3.7 main.c [21]

Po startu procesoru dojde k volání funkcí knihoven desky. Tyto funkce mají za úkol inicializaci systémových hodin (SYSTEM CONTROL BLOCK), dále nastavení MEMORY ACCELERATOR MODULE a nakonec knihovna del_led.h inicializuje výstupy obsluhující čtveřici LED diod (LPCEUROBOT deska, součástky D2-D5). Zdrojové kódy těchto knihoven jsou v [22].

Dále je spuštěna funkce main(). V ní jsou volány inicializační funkce mých knihoven.

```
init_uart0((int) 9600, UART_BITS_8, UART_STOP_BIT_1, UART_PARIT_OFF, 0);
init_depwm_timer1(DEPWM_ISR_VEC);

mux_sel[0] = ADC_MUX_CHANNEL_ON;
mux_sel[1] = ADC_MUX_CHANNEL_ON;
mux_sel[3] = ADC_MUX_CHANNEL_ON;
mux_sel[6] = ADC_MUX_CHANNEL_ON;

init_adc(ADC_ISR_VEC);
init_pow_switch();

deb_led_set(LEDG);


unsigned char input;

while (TRUE) {
      input = uart0GetCh();

      switch (input) {
      case SW1ON: pow_switch_on(0);
            break;
      case SW1OFF: pow_switch_off(0);
            break;
      case SW2ON: pow_switch_on(1);
            break;
      case SW2OFF: pow_switch_off(1);
            break;
      }
}
```

*Kód 3.7 – ukázka main funkce*

Zde proběhne inicializace UART0, dále inicializace TIMER1 pro účely demodulace PWM signálu, poté vybrání kanálů multiplexeru IC1, které budu vzorkovat, pak se nastartuje A/D převodník a inicializují se výstupy ovládající U2. Nakonec se rozsvítí zelená dioda na desce LPCEUROBOT. Poté program skončí v nekonečné smyčce, která čte příchozí znaky z USB, respektive UART0, a podle toho zapíná nebo vypíná výstupy.

Ale pokud hlavní smyčka programu pouze přijímá a vyhodnocuje data z UART0, tak jak se data odesílají? Aby se program dostal z této nekonečné smyčky a mohl zavolat funkci `send_data()`, je potřeba přerušení. V tomto konkrétním případě jsem mohl použít zbývající čítač TIMER0. Protože TIMER0 může v jiných případech sloužit také k demodulaci PWM signálu, použil jsem jiné řešení. Jako čítač generující přerušení jsem použil A/D převodník. Po každém dokončení převodu A/D převodník vyvolá přerušení. Obslužní program zpracuje data a opět spustí konverzi. Pokud umístím následující kód do obslužné rutiny A/D převodníku, tak dosáhnu pravidelného volání

funkce `send_data()`.

```
if (!counter--){
      counter = SENDING_PERIOD;
      send_data();
}
```

*Kód 3.8 – volání funkce send_data*

## 4. Otestování zařízení.

K zařízení jsem připojil tři potenciometry, ntc termistor 6K8, astabilní klopný obvod generující signál o střídě 54% a dvě žárovky.

Zařízení jsem připojil k počítači, otevřel terminál (GtkTerm), nastavil komunikaci (port: /dev/ttyUSB0, rychlost 9600, parita žádná, bitů 8, stopbit 1, řízení toku žádné)

Po uvolnění RTS a DTR linek zařízení začalo vysílat data. Během níže zaznamenaných dat jsem jsem lehce pootáčel jedním potenciometrem, vypínal a zapínal výstupy.

```
strida: 54% pot 1: 16% pot 2: 95% pot 3: 72 ain1: 25 OFF  OFF
strida: 54% pot 1: 14% pot 2: 95% pot 3: 72 ain1: 25 OFF  OFF
strida: 54% pot 1: 19% pot 2: 95% pot 3: 72 ain1: 25 OFF  OFF
strida: 54% pot 1: 18% pot 2: 95% pot 3: 72 ain1: 25 OFF  OFF
strida: 54% pot 1: 21% pot 2: 95% pot 3: 72 ain1: 25 OFF  OFF
strida: 54% pot 1: 20% pot 2: 95% pot 3: 72 ain1: 25 OFF  OFF
strida: 54% pot 1: 23% pot 2: 95% pot 3: 72 ain1: 25 OFF  OFF
strida: 54% pot 1: 5% pot 2: 95% pot 3: 72 ain1: 25 OFF  OFF
strida: 54% pot 1: 22% pot 2: 95% pot 3: 72 ain1: 25 OFF  OFF
strida: 54% pot 1: 26% pot 2: 95% pot 3: 72 ain1: 25 OFF  OFF
strida: 54% pot 1: 26% pot 2: 95% pot 3: 72 ain1: 25 OFF  OFF
strida: 54% pot 1: 26% pot 2: 95% pot 3: 72 ain1: 25 ON  OFF
strida: 54% pot 1: 26% pot 2: 95% pot 3: 72 ain1: 25 ON  ON
strida: 54% pot 1: 26% pot 2: 95% pot 3: 72 ain1: 25 ON  ON
strida: 54% pot 1: 26% pot 2: 95% pot 3: 72 ain1: 25 ON  ON
strida: 54% pot 1: 26% pot 2: 95% pot 3: 72 ain1: 25 ON  ON
strida: 54% pot 1: 26% pot 2: 95% pot 3: 72 ain1: 25 ON  ON
strida: 54% pot 1: 26% pot 2: 95% pot 3: 72 ain1: 25 OFF  ON
strida: 54% pot 1: 26% pot 2: 95% pot 3: 72 ain1: 25 OFF  ON
strida: 54% pot 1: 26% pot 2: 95% pot 3: 72 ain1: 25 OFF  OFF
strida: 54% pot 1: 26% pot 2: 95% pot 3: 72 ain1: 25 OFF  OFF
strida: 54% pot 1: 26% pot 2: 95% pot 3: 72 ain1: 25 OFF  OFF
strida: 54% pot 1: 26% pot 2: 95% pot 3: 72 ain1: 25 OFF  OFF
```

*Obrázek 4.1 – výstup terminálu*

## Závěr

Vzhledem k tomu, že jsem skončil své působení v projektu CarTech, tak jsem zařízení neotestoval se všemi připravenými senzory. Zvolil jsem proto alternativní zapojení testovací zapojení v kterém jsem vyzkoušel funkčnost A/D převodu s multiplexováním, určení střídy PWM signálu, měření teploty pomocí ntc termistoru, obousměrnou komunikaci přes USB rozhraní spínání výkonového obvodu. Jediná škoda je, že jsem nemohl otestovat akcelerometr LIS3LV02DQ.

## Výčet součástek a jejich pouzder

| součástka | hodnota | pouzdro |
|---|---|---|
| C1-C12, C14, C16 | 100n | SMD 0805 |
| C13, C15, C17 | 47u | SMD C |
| D1-D12 | BAT54S | SOT23 |
| R1-R3, R6-R8, R11, R12, R14 | 100R | SMD 0805 |
| R21, R32, R33 | 120R | SMD 0805 |
| Ostatní R | zatím nespecifikováno | SMD 0805 |
| IC1 | 74HC4051 | 16.7.2009 |
| IC2 | LIS3LV02DQ | QFPN-28 |
| IC3 | LE33 | TO92 |
| U1 | PAC82C250 | |
| U2 | IRF7313 | SO8 |
| X1-X13 | svorkovnice 3 svorky, RM 3,5 mm | |
| X14-17 | svorkovnice 2 svorky, RM 3,5 mm | |
| SV1, SV2 | zásuvková lišta přímá. 2x17 kontaktů, RM 2,54mm | |

## Výčet použitých součástek a jejich pouzder

| součástka | hodnota | pouzdro |
|---|---|---|
| C1-C12, C14, C16 | 100n | SMD 0805 |
| C13, C15, C17 | 47u | SMD C |
| D1-D12 | BAT54S | SOT23 |
| R1-R3, R6-R8, R11, R12, R14 | 100R | SMD 0805 |
| R21, R32, R33 | 120R | SMD 0805 |
| R18 | 3K3 | SMD 0805 |
| R23 | 1K2 | SMD 0805 |
| R4, R28 | 56K | SMD 0805 |
| R5 | 220K | |
| IC1 | 74HC4051 | 16.7.2009 |
| IC2 | LIS3LV02DQ | QFPN-28 |
| IC3 | LE33 | TO92 |
| U1 | PAC82C250 | |
| U2 | IRF7313 | SO8 |
| X1-X13 | svorkovnice 3 svorky, RM 3,5 mm | |
| X14-17 | svorkovnice 2 svorky, RM 3,5 mm | |

## Seznam použité literatury

[1] – mp20ip.pdf

[2] – LIS3LV02DQ.pdf

[3] – RTD-850.pdf

[4] – XTEL-312.pdf

[5] – XTL-123.pdf

[6] – TIM-5H.pdf

[7] – CW20.pdf

[8] – VBOX.pdf

[9] – TIM-5H integration manual.pdf

[10] - J-STD-020C.pdf

## Seznam zdrojových kódů

[11] – lpc21xx.h

[12] – types.h

[13] – uart_orig.h

[14] – uart.h

[15] – adc_orig.h

[16] – adc_mux.h

[17] – config.h

[18] – depwm.h

[19] – send_data.h

[20] – powswitch.h

[21] – main.c

[22] – deb_led.h, hwinit.c, startcfg.h, error.h

## Seznam odkazů


[o1] - www.p3america.com

[o2] – en.wikipedia.org/wiki/IP_Code

[o3] - www.analog.com

[o4] – www.st.com

[o5] - www.omega.com

[o6] - www.kulite.com

[o7] - www.racelogic.co.uk

[o8] - www.ublox.com

[o9] - www.hp.woodshot.com

[o10] – www.cadsoft.de

[o11] - www.pragoboard.cz

[o12] – rtime.felk.cvut.cz/hw/index.php/LPC21xx

[o13] – www.ubuntu.cz

## Obsah přiloženého cd

```
bp.pdf – elektrotechnická forma této práce

adresář
      dokumentace – dokumentace a datasheety
      src – zdrojové kódy
      pcb – schéma a deska ve formátu eagle
```