České vysoké učení technické v Praze
Fakulta elektrotechnická

Bakalářská práce

Vícekanálový oddělovací A/D převodník s BlueTooth výstupem

Šach Jaroslav

Vedoucí práce: Ing. Jan Havlík
Studijní program: Elektronika a informatika strukturovaný bakalářský
Obor: Kybernetika a měření
červen 2009

Prohlášení

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

V Praze dne …..... …....

podpis

Poděkování

Chtěl bych vyjádřit své poděkování panu Ing. Janu Havlíkovi za odborné konzultace a cenné připomínky v průběhu zpracování mé bakalářské práce. Dále pak děkuji všem, kteří mi pomohli k zdárnému dokončení práce a v neposlední řadě rovněž své rodině za podporu v průběhu celého studia.

ANOTACE

Tato bakalářská práce se zabývá návrhem a realizací vícekanálového analogového převodníku umožňujícího bezdrátový přenos naměřených dat PC. Bezdrátové propojení musí být realizováno pomocí standardu bluetooth. Cílem projektu je také vytvoření jednoduchého obslužného software, který bude s naší aplikací komunikovat a získaná data zobrazovat.

ANOTATION

Aim of this work is designing and realization of multi-channel analog to digital converter. This application will provide wireless connection with personnely computers. This connection have to by realised via BlueTooth standard. Prototype of such device is only one half of this work. The second is writing service program for PC, which will be communicate with our device and then will by display obtained data.

Obsah

1 ÚVOD....7
2 VSTUP DO PROBLEMATIKY....8
2.1 Základní přiblížení technologie BlueTooth....8
2.1.1 Topologie sítí Bluetooth....9
2.1.2 Profily Bluetooth....10
2.1.3 Výhody a nevýhody....11
2.2 Vzorkování a kvantizace....11
3 HARDWAROVÉ ŘEŠENÍ, VÝBĚR KOMPONENT....14
3.1 Napájení....15
3.2 Napěťová reference....16
3.3 Uživatelské rozhraní....16
3.4 Vysokofrekvenční obvody....18
3.5 Řídící mikroprocesor....21
3.6 Anologové obvody....23
4 VÝVOJ SOFTWARE PRO MIKROKONTROLER....26
4.1 Programovací jazyky pro mikrokontrolery....26
4.2 VinAVR....27
4.3 Firmware....28
4.4 Přerušení....35
5 POPIS ZPŮSOBU KOMUNIKACE S PC....36
6 VÝVOJ SOFTWARE PRO WINDOWS....39
6.1 Přiblížení problematiky....39
6.2 Seznámení s platformou .NET Framework....40
6.3 Knihovny třídy systému .NET Framework....41
6.4 MSIL (Microsoft Intermediate Language)....42
6.5 CLR (Common Language Runtime)....43
6.6 JIT (Just In Time)....43
7 POPIS ŘÍDÍCÍ APLIKACE A JEJÍ OVLÁDÁNÍ....44
8 ZÁVĚR....47
9 POUŽITÁ LITERATURA A ZDROJE....49
[1] PUNČOCHÁŘ, J., Operační zesilovače v elektronice....49
[2] VÁŇA, V., Mikrokontroléry ATMEL AVR - programování v jazyce C, Nakladatelství BEN – technická literatura, Praha 2003, ISBN 80-7300-102-0. 49
10 POUŽITÝ SOFTWARE....50
12 PŘÍLOHY....52
12.1 Schéma zapojení....52
12.2 Obrazce plošných spojů....53
12.3 Osazovací plány plošného spoje....54

1 ÚVOD

Hlavní myšlenkou mé bakalářské práce je seznámení se s bezdrátovou technologií BlueTooth a následné její využití v mikroprocesorových aplikacích. Zejména pak v konstrukcích zabývajících se měřením a sběrem dat. V posledních několika letech zaznamenal tento standard bezdrátového připojení opravdu veliký vzestup. Díky řadě výhod, které poskytuje, se rozšířil do veškerých oblastí našeho každodenního života. Nejčastější uplatnění nalezlo BlueTooth ve spojení s mobilními telefony (bezdrátové náhlavní soupravy). Rovněž se často uplatní při vzájemném propojení laptopů, notebooků a kapesních počítačů PDA nebo jejich spojení s periferními zařízeními (např. tiskárnami). Jak ukázala měření v praxi, jsou bezdrátová spojení realizovaná standardem Bluetooth velmi dobře odolná proti rušení, a proto si tato technologie své místo vydobyla také v oblasti automatizace v těžkých provozech.

Tento standard si osvojila drtivá většina firem zabývajících se výrobou elektronických prvků a komponent. Vyrábí se tak nepřeberné množství technologicky různorodých periferií, integrovaných obvodů, či modulů. Jejich cena je zásluhou sériové výroby velmi nízká, což podporuje další rozšiřování BlueTooth.

Má práce je věnována návrhu a vytvoření modulu zařízení, které provádí na několika kanálech měření analogových veličin a následně získaná data odesílá k dalšímu zpracování do nadřazeného PC. Výsledkem mé činnosti nebude pouze teoretický návrh možného řešení, ale rovněž i fyzické provedení fungujícího elektronického zařízení. Rovněž, tak jak je vymezeno oficiálním zadáním, vytvořím obslužný software, jenž bude změřená data schopen zobrazovat, případně i dále zpracovávat.

V následujících kapitolách se budu zabývat nejdříve stručným popisem technologií s konstrukcí úzce spjatých. Provedu analýzu svého řešení a objasním funkci jednotlivých částí návrhu. Pokusím se nastínit způsob komunikace mého zařízení s aplikačním programem v uživatelském počítači. Závěrem uvedu možná rozšíření a případná využití vzniklé konstrukce.

2 VSTUP DO PROBLEMATIKY

2.1 Základní přiblížení technologie BlueTooth

Cílem kapitoly není čtenáři poskytnout podrobné informace o bezdrátové síti technologie bluetooth (dále jen bluetooth), ale pouze nastínit hlavní rysy, které bylo nutné mít v podvědomí při tvorbě programu obsluhujícího komunikaci mezi naším převodníkem a okolním světem. Úvodem bych rád zmínil literaturu, ze které je možné nabít další poznatky o popisované problematice. Jedná se především o kompletní specifikaci bluetooth [1], nebo lze doporučit i dřívější diplomové práce, které se problematiky bluetooth úzce dotýkají [2] [3].

Bluetooth byla navrhována jako osobní bezdrátová síť s nízkým dosahem (ve svém nízko-výkonovém režimu má dosah pouze 10 metrů). Její hlavní využití se nabízí v domácích sítích mezi mobilními zařízeními (PDA, telefon atd.) a periferními zařízeními, zejména pak tiskárnami, nebo ke komunikaci se spotřební elektronikou. Zařízení bluetooth poprvé představilo na trh sdružení Bluetooth SIG (Special Industry Group) v roce 1998. Toto sdružení vzniklo, jako neziskové spojení firem Ericsson, IBM, Intel, Nokia a Toshiba. Komunikace mezi dvěma bluetooth zařízeními probíhá v bezlicenčním kmitočtovém pásma 2,4 GHz. Použití volně využivatelného pásma zároveň předurčuje použití bluetooth v osobní sféře. Rychlost přenosu dat na fyzické vrstvě je 1Mbit/s, ale skutečná propustnost dat se pohybuje maximálně na hranici 720 kbit/s. Pro radiovou komunikaci, bluetooth využívá metod rozprostřeného spektra s přeskakováním kmitočtů (Frequency Hopping Spread Spectrum, FHSS). To si můžeme představit, jako signál, který 1600 krát/s náhodně změní frekvenční kanál. Každý kanál má frekvenční rozsah 1MHz a bluetooth zařízení jich může využívat
celkem 79. V každé síti bluetooth je vždy pouze jedna hlavní stanice tzv. master a několik, maximálně však sedm, podřízených stanic tzv. slave. Veškerou komunikace v síti bluetooth řídí hlavní stanice. Podřízená stanice může tedy komunikovat s dalšími podřízenými stanicemi pouze prostřednictvím stanice hlavní.

Komunikace mezi dvěma zařízeními může být buď synchronní nebo asynchronní, přičemž hlavní stanice alokuje časové úseky prostřednictvím vícenásobných přístupů s časovým dělením (Time Division Multiple Access, TDMA).

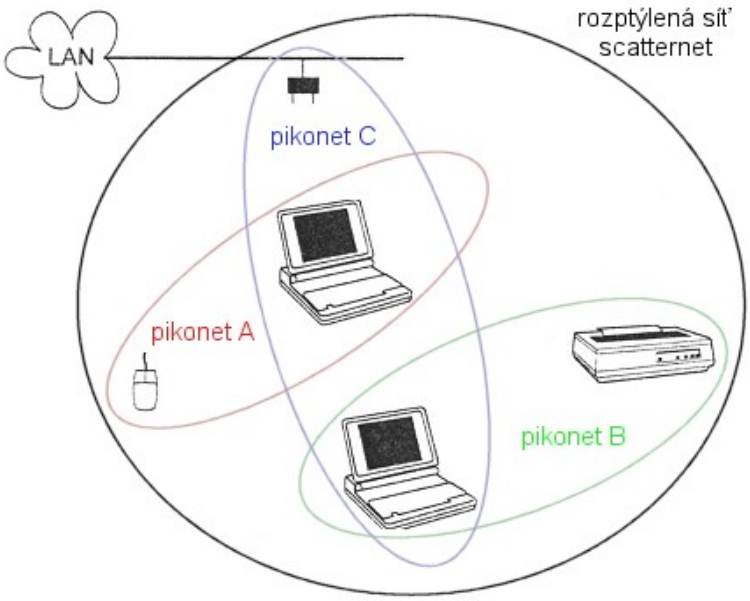
Bluetooth používá stejné kmitočty pro vysílání a příjem s využitím Time Division Duplexing (TDD) nebo multi-slot. Časování TDD odpovídá situaci v níž se vysílací a přijímací jednotka postupně střídají ve vysílání. Oproti tomu časování multi-slot odpovídá využití přeskokové sekvence pro přenos tak, že paket může obsadit více než jeden časový rámec (maximálně 5). Jak už bylo zmíněno, komunikace může probíhat synchronně (SCO, Synchronous Connection Oriented) a asynchronně (ACL, Asynchronous Connectionless).

Oba způsoby komunikace se od sebe výrazně odlišují v přenosových vlastnostech. Kanál ACL používá časování multi-slot, přičemž je možné dosáhnout přenosové rychlosti 721kb/s v jenom směru a 57,6 kb/s v opačném směru (asymetrický kanál), popř. 433 kb/s v obou směrech (symetrický kanál). Tento způsob komunikace se nejčastěji využívá pro přenos dat mezi stanicemi. Kanál typu SCO dovoluje realizovat přenos dat rychlostí 64 kb/s v synchronním režimu a to až po třech různých kanálech najednou. Tento způsob přenosu dat je vhodnější pro přenos zvuku nebo obrazu.

2.1.1 Topologie sítí Bluetooth

Standard Bluetooth umožňuje vytvářet tzv. pikosítě (piconet). V jedné síti mohou být minimálně dva přístroje (např. mobilní telefon a notebook), což představuje vlastně spojení od bodu k bodu. Pikosíť je možné kdykoli zvětšit a zmenšit, a vytvářet tak rychle se měnící sítě (ad hoc), které se po každém zapnutí znovu samostatně konfigurují. Obvykle je pikosíť tvořena osmi stanicemi, z nichž jedna je nadřazená (master) a ostatní jsou podřízené (slave). Propojením až deseti pikosítí vytvoříme scatternet (rozložená síť), v níž mohou mohou vzájemně komunikovat všichni účastníci.

Rozsáhlejší pikosíť obsahuje až 255 účastníků, přičemž v jednom okamžiku může být aktivních osm přístrojů. Pro zvýšení počtu aktivních přístrojů Bluetooth jsou nasazovány přístupové body BAP, kterými lze vybudovat další pikosítě, které se mohou i překrývat (a to i v okruhu 10 m). Přístupové body mohou spojovat systém Bluetooth se stávajícími lokálními sítěmi LAN, a tvoří tak jejich bezdrátové zakončení směrem k uživateli. V místnostech mohou být umístěny jak na stropě, tak i ve zdi (podobně jako elektrické zásuvky). Pro vybavení například celé budovy technikou Bluetooth je zapotřebí několika BAP, spojených obvykle sítí Ethernet.



Obr. 2.1: Propojení několika pikosítí - scatternet

2.1.2 Profily Bluetooth

Přístroje standardu Bluetooth jsou vybaveny softwarovými profily pro různé způsoby využití. Je standardizováno přes dvacet profilů. Patří mezi ně např. pro budování sítí, pro přenos dat, zvuku či obrazu, pro tisk, pro přístup na kartu SIM mobilního telefonu nebo pro služby faxu. Komunikující přístroje musí mít stejný profil. V počátcích zavádění standardu Bluetooth chyběla standardizace právě těchto profilů, takže si každý z výrobců vytvořil vlastní profil a tím docházelo k neslučitelnosti přístrojů a tím i k nedůvěře zákazníků v tuto novou bezdrátovou techniku.

2.1.3 Výhody a nevýhody

Mezi výhody standardu Bluetooth patří nízká spotřeba energie, malý výkon vysílačů, velká odolnost proti poruchám, poměrně levné a miniaturní moduly, celosvětová přístupnost standardu a kompatibilita profilů. Předností může být i malý dosah, který znesnadňuje odposlech nebo zneužívání (oproti WLAN).

Standard Bluetooth má i své nevýhody. Jmenujme malou šíři pásma (tj. malá přenosová rychlost), omezenou vzdálenost a také omezený počet účastníků.

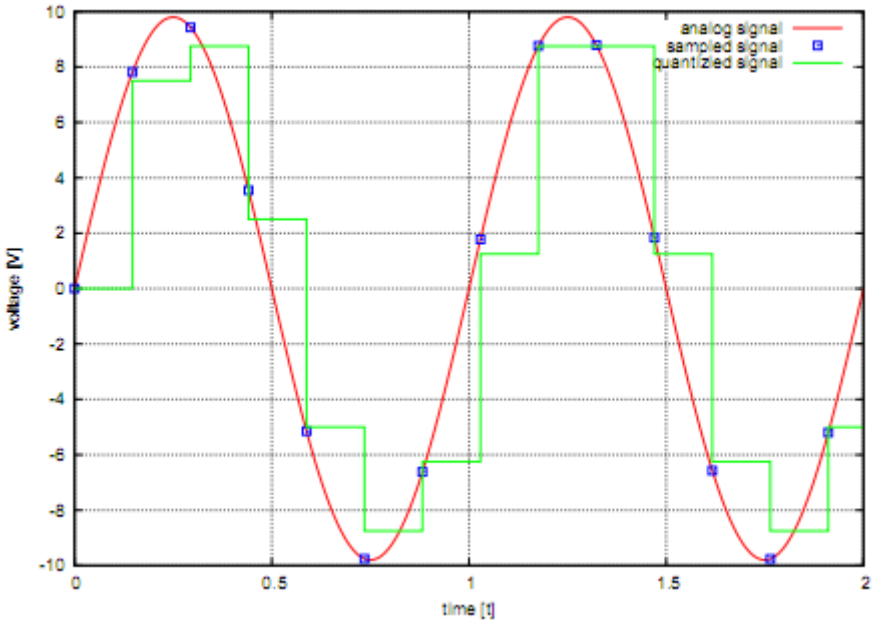
V této kapitole zmíním základní pojmy související se zpracováním digitalizovaného signálu, nebudu uvádět přílišné podrobnosti ani složitá odvození. Případné zájemce raději odkáži na odbornou literaturu, která se tímto tématem zabývá. Rovněž nebudu rozebírat vliv filtrů a obvodů, jež naměřený signál dále zpracovávají.

2.2 Vzorkování a kvantizace

Při měření téměř všech fyzikálních veličin se v současnosti používají číslicové přístroje. To s sebou nese řadu výhod - lze poměrně jednoduše uchovat a hlavně automaticky zpracovat ohromné množství dat. Je však přitom zřejmé, že číslicovými přístroji nelze spojitou veličinu zaznamenat. Data tedy získáváme vzorkováním (sampling), tzn. zaznamenáním okamžitých hodnot fyzikálních veličin v konkrétním čase. Nejčastěji se používá konstantní vzorkovací interval, tzv. vzorkovací perioda **T** [Hz] (sampling period). Převrácenou hodnotu vzorkovací periody je pak vzorkovací frekvence(sampling frequency)

$$f_s = \frac{1}{\Delta T} \qquad [Hz]$$

Spojitý a vzorkovaný signál jsou zakresleny na obrázku 2.2. Na první pohled je patrné, že kvalita vzorkovaného signálu bude záviset na vzorkovací frekvenci. Vzorkovaný signál bude tedy diskrétní v čase. Ovšem ani amplituda takového signálu nebude vinou konečného rozlišení číslicových systémů spojitá vlivem tzv. kvantizace (quantization).



Obr. 2.2: Zobrazeni analogového, vzorkovaného a kvantizovaného sig.

Uvažujme např. 8bitovou měřicí kartu schopnou měřit v rozsahu UR = ±10 [V]. Taková karta je schopna rozlišit 2^8 , tedy 256 úrovní signálu. Nejmenší měřitelný rozdíl napětí tedy bude

$$\Delta T = \frac{U_R}{2^N} = 0{,}078 \quad [V]$$

Měřící karta by tak nebyla schopna rozlišit dvě úrovně lišící se o méně než 78mV. Rovněž je z obrázku patrno, že navzorkovaný signál se signálu původnímu téměř vůbec nepodobá. Vzorkování a kvantizace nejsou tedy správně nastaveny bude zapotřebí doladit jejich parametry. Základní způsoby jak vylepšit kvalitu získaného signálu jsou následující.

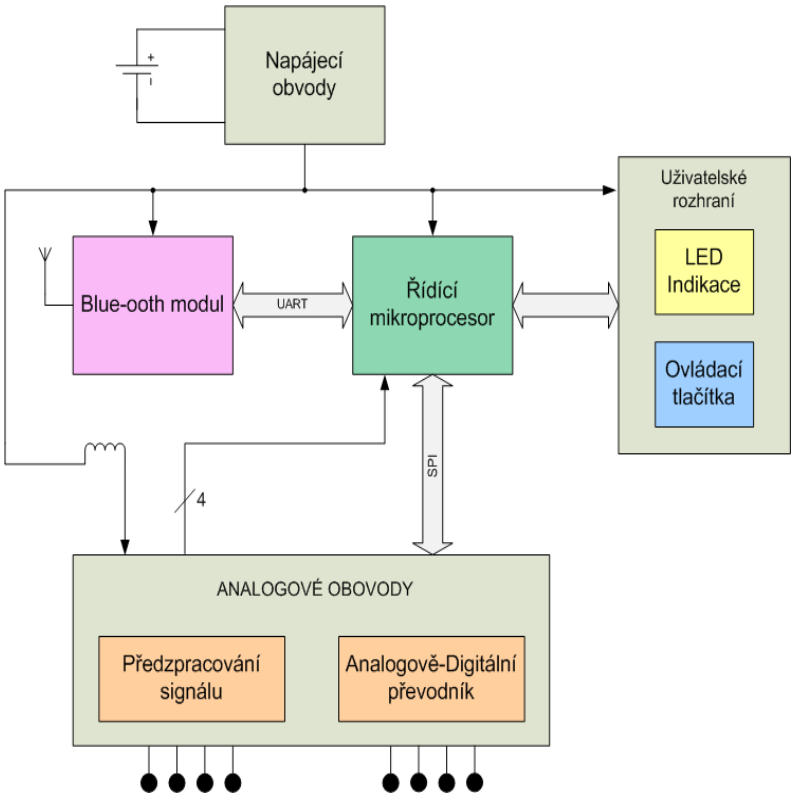
Zvýšení vzorkovací frekvence

Zvýšením úrovně kvantizace

Zvýšením počtu měření a následujícím průměrováním

Na závěr bych rád jen podotkl, že pro využití v naší aplikaci, která slouží spíše jen jako ukázkový vzorek, není zapotřebí využívat žádné pokročilé metody zpracování signálu. Volba vzorkovací frekvence bývá v obecných případech poměrně složitou záležitostí. Pro naše použití však nebyl tento parametr nikterak kritický. Rovněž na velikost kvantizační úrovně a tedy počet bitů převodníku nebyly kladeny žádné specifické požadavky. Vlastní návrh se tak omezil pouze na volbu vhodného převodní a vytvoření vstupních obvodů pro úpravu signálu.

3 HARDWAROVÉ ŘEŠENÍ, VÝBĚR KOMPONENT

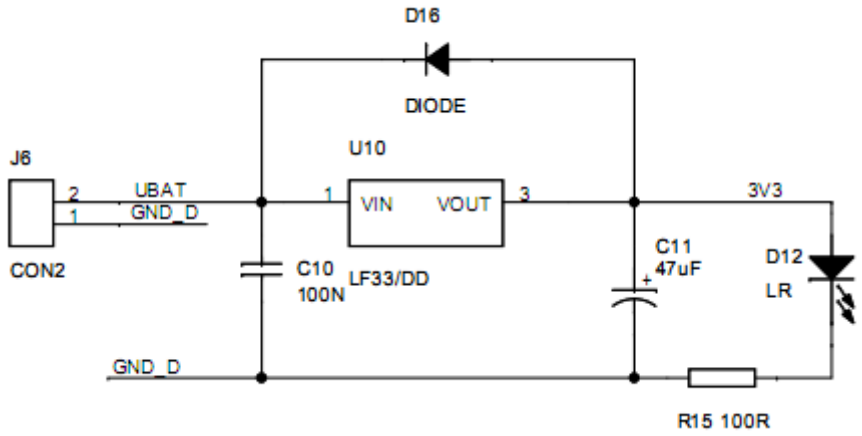


Obr. 3.1 : Blokové zapojení AD převodníku

V následující kapitole přiblížím konkrétní implementaci jednotlivých částí celého zapojení, jejich funkci a vzájemnou komunikaci. Celé zařízení lze rozdělit do několika nezávislých bloků. Jejich vzájemné propojení zobrazuje následující blokové schéma.

3.1 Napájení

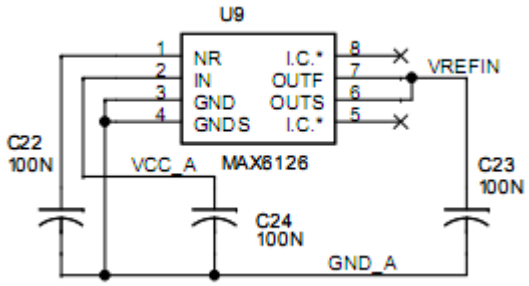
Napájení celého zařízení zajišťují čtyři tužkové baterie. Při použití standardních článků tak máme k dispozici napětí o velikosti 4 * 1.5 V. Tedy 6V. BlueTooth modul pro svou činnost vyžaduje rozsah napájecího napětí mezi 2.7V až 3.6V. Těmto úrovním bylo přizpůsobeno i napájecí napětí řídící logiky a veškerých analogových obvodů. Pro stabilizaci napájení byl zvolen lineární integrovaný obvod LE33CZ. Ten na svém výstupu poskytuje potřebnou hodnotu napětí a nabízí proudové zatížení až do 0.15A. Jelikož odběr našeho zařízení nepřekračuje 100mA, máme dostatečnou rezervu. Obvod se nikterak nezahřívá a není zapotřebí jej chladit. K přednostem tohoto regulátoru patří nejen jednoduchost zapojení, minimum externích součástek, ale rovněž velmi příznivá cena a snadná dostupnost. Zároveň se LE33CZ řadí do kategorie označované jako LOW-DROP. Pro správnou činnost tedy na vstupu vyžaduje napětí jen o 0.45V vyšší, než je požadované napětí výstupní. Zapojení regulátoru je uvedeno na obrázku č. 3.2, vychází z katalogových údajů výrobce a je pouze doplněno signalizační LE Diodu D12. Pro napájení analogových částí je do cesty vložena filtrační tlumivka, která má za úkol ořezávat případné špičky způsobené digitálními obvody.



Obr. 3.2: Zapojení stabilizace napájecího napětí

3.2 Napěťová reference

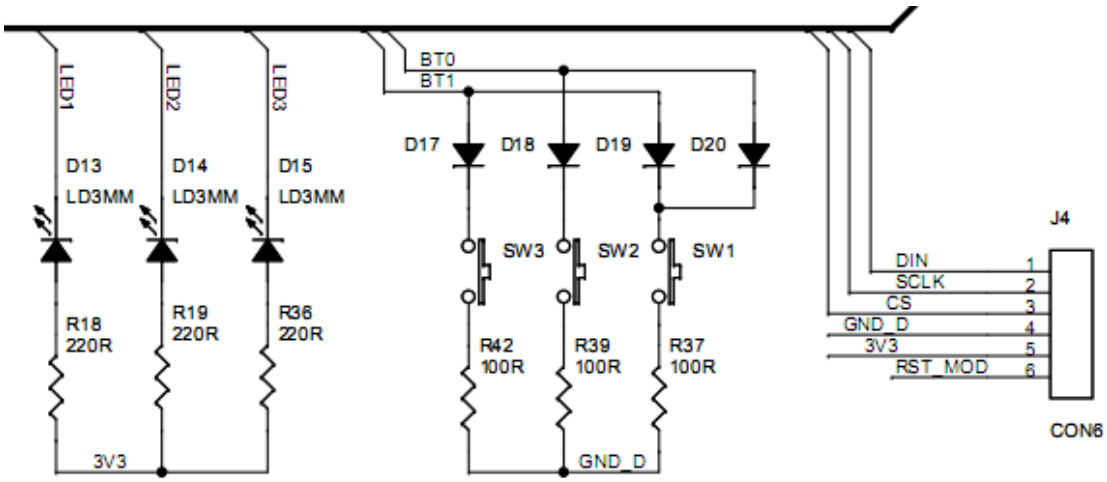
Tato část obvodu poskytuje referenční hodnotu napětí pro čtyři vstupní kanály měřené externím AD převodníkem. Rovněž může být využita pro vytvoření referenční úrovně AD převodníku umístěného na čipu řídícího mikrokontroléru a tedy i pro zbývající čtyři kanály. Tato možnost se volí konkrétním firmware. Z velké škály integrovaných referenčních obvodů, které jsou v současné době vyráběny jsem vybral MAX6126. Důvodem pro tuto volbu byla především dobrá zkušenost s produkty firmy Maxim, od které obvod pochází. Na svém výstupu poskytuje velmi přesné napětí a hodnotě 2.048V. Tato hodnota je tedy i horní hranicí pro měřené vstupní signály. Maximální proudové zatížení je 10mA. Zapojení napěťové reference vidíme na Obr.č.3.3 . I v tomto případě vychází z doporučení výrobce.



Obr. 3.3: Zapojení napěťové reference

3.3 Uživatelské rozhraní

Uživatelským rozhraním rozumím prvky sloužící k indikaci, ovládání a k vlastnímu programování mikroprocesoru. Jako základní prvky pro signalizaci jednotlivých stavů zařízení slouží čtyři barevné diody LED. Jejich přesnější význam bude popsán v jedné z následujících kapitol, zabývajících se ovládáním zařízení. K nastavení základních parametrů, jako je vzorkovací perioda, nebo navázání spojení s PC slouží tři mikrospínací tlačítka. Zapojení celého rozhraní je uvedeno na následujícím obrázku č. 3.4.



Obr. 3.4: Zapojení uživatelského rozhraní

LED diody jsou k výstupnímu portu připojeny přímo katodami. V sérii s napájením mají předřazené ochranné rezistory. Ty zabraňují proražení koncového tranzistoru portu mikrokontroléru nadměrným proudem.

Procesor je v provedení SMD, k zápisu nového firmware je tedy zapotřebí využít metody ISP (In System Programming). Tato metoda umožňuje změnu firmware bez nutnosti demontáže obvodu. K vlastnímu programování jsou využity následující čtyři piny procesoru.

- **RESET**
- **SCK**
- **MISO**
- **MOSI**

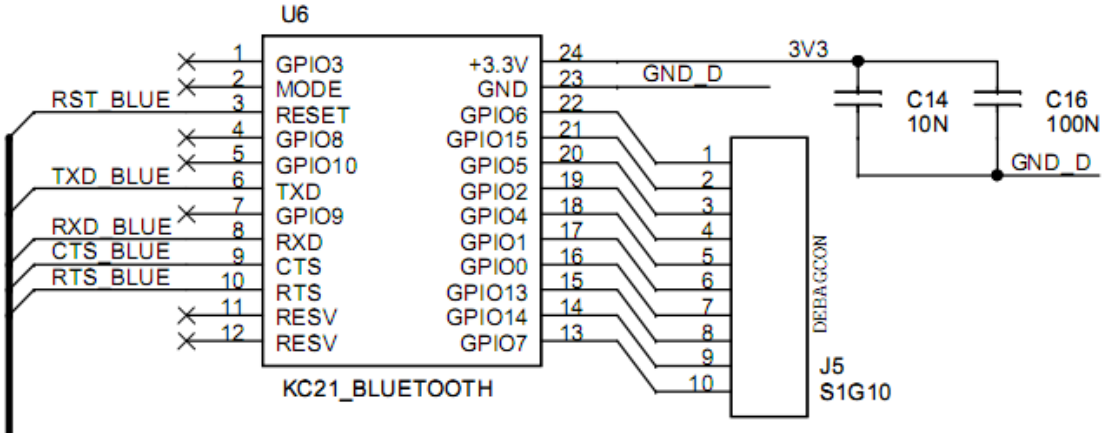
Uvedené vývody mikrokontroléru jsou vyvedeny společně s napájením na šestipinový konektor CON6.

K ovládání celého zařízení slouží tlačítka SW1, SW2 a SW3. Jelikož však veškeré I/O porty obsazují ostatní periferie a obvody, bylo zapotřebí použít matice diod D17 – D20. Díky těmto diodám je možno na dvou bitech mít připojena až čtyři tlačítka. Při stisku některého ze spínačů se na portu procesoru objeví přímo binární kód daného spínače. Tak nejen, že ušetříme vývody procesoru, ale zároveň obsluha stisklého tlačítka bude mnohem jednodušší.

3.4 Vysokofrekvenční obvody

Veškeré vysokofrekvenční obvody jsou reprezentovány zařízením KC-21. Nejedná se o obvyklý integrovaný obvod, ale o kompletní vysokofrekvenční modul, jenž je osazen na samostatném plošném spoji. Modul je navíc doplněn mikroprocesorem ARM. Ten obstarává veškerou logiku pro řízení bezdrátového přenosu v rámci sítě BlueTooth. Z pohledu uživatele se tak modul chová jako sériové rozhraní UART. Komunikace s KC-21 probíhá prostřednictvím AT příkazů s přenosovou rychlostí 115200 KBd. Základní příkazy lze rozdělit do dvou skupin, a sice OPERAČNÍ a KONFIGURAČNÍ. Výpisy těchto kategorií uvádějí tabulky č.3.1 a č.3.2.

Pro svou funkci modul nepotřebuje žádné externí součástky. Do větve napájení jsou vloženy dva blokovací kondenzátory odstraňující případné proudové rušivé špičky. Konektor J5 slouží především k ladicím účelům. Na jeho piny jsou vyvedeny vstupně/výstupní brány procesoru ARM. Je tedy možno jej využít i pro případné pozdější rozšíření celé aplikace. Vlastní přenos příkazů a dat probíhá po vodičích označených **RXD** a **TXD.** Signály **RTS** a **CTS** jsou určeny pro hardwarové řízení toku dat. V naší aplikaci jich není využíváno. Posledním ze zapojených vývodů je řídící pin s označením RST. Vystavením tohoto signálu do úrovně logické nuly máme možnost vyvolat hardwarový reset BlueTooth modulu.



Obr. 3.5 : Zapojení BlueTooth modulu

Název příkazu	**Funkce příkazu**
AT+KC Bond	Navázání spojení s konkrétním zařízením
AT+KC Bypass	Návrat do *bypass* módu. Možno využít pro změnu nastavení parametrů připojení, pokud již bylo navázána komunikace s požadovaným zařízením.(BaudRate)
AT+KC DisableBond	Zakázání požadavků na spojení (Nelze provést v průběhu již navázaného spojení)
AT+KC Discovery	Příkaz pro vyhledání dostupných zařízení
AT+KC DUNConnect	Navázání spojení s konkrétním zařízením
AT+KC DUNDisconnect	Ukončení spojení s konkrétním zařízením
AT+KC EnableBond	Povolení navázání spojení s okolními zařízeními
AT+KC ExitPark	Přepnutí zařízení z *parkovacího* do *aktivního* módu
AT+KC ExitSniff	Přepnutí zařízení ze *sniff* módu do *aktivního* módu
AT+KC GPIORead	Přečtení logických úrovní na všech IO pinech modulu
AT+KC GPIOWrite	Nastavení logických úrovní na výstupních pinech modulu
AT+KC Hold	Přepnutí zařízení z *aktivního* módu do *hold* módu
AT+KC Park	Přepnutí zařízení z *aktivního* do *parkovacího* módu
AT+KC RemoteCommand	Povolení/zakázání módu vzdálených příkazů
AT+KC RemoteCmdDisconnect	Ukončení spojení módu vzdálených příkazů
AT+KC Reset	Softwarový reset zařízení
AT+KC Sniff	Přepnutí do *sniff* módu
AT+KC SPPConnect	Navázání spojení s konkrétním zařízením
AT+KC SPPDisconnect	Ukončení spojení s konkrétním zařízením

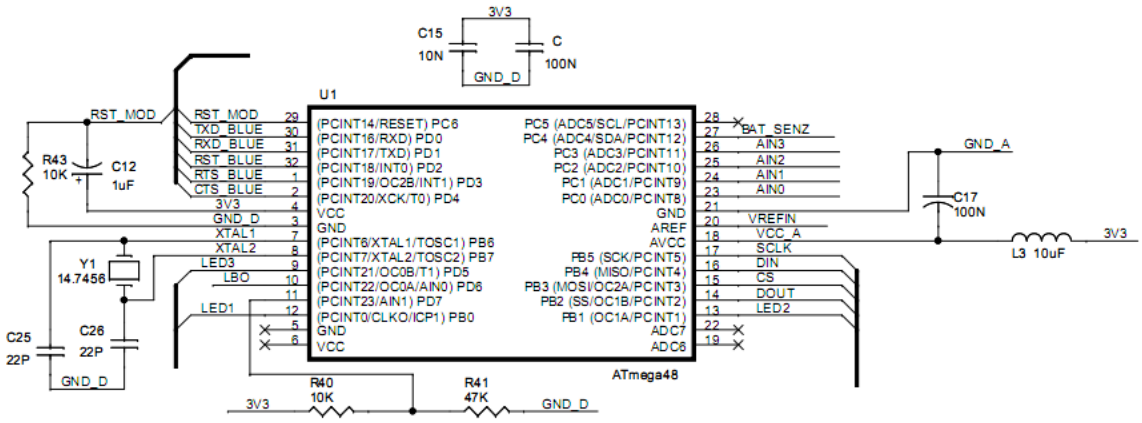
Tabulka 3.1: Seznam AT příkazů

Název příkazu	**Funkce příkazu**
AT+KC ChangeBaud	Změna přenosové rychlosti
AT+KC ChangeDefaultBaud	Změna defaultní přenosové rychlosti (na tuto hodnotu se nastaví rychlost komunikace při zapnutí modulu)
AT+KC DefaultLocalName	Nastavení názvu, pod nímž se naše zařízení bude hlásit
AT+KC DeleteSmartCable	Odstranění nastavení *Smart Cable*
AT+KC EraseBondTable	Zrušení všech navázaných připojení
AT+KC GPIOConfig	Nastavení režimu vstupně výstupních bran
AT+KC HostEvent	Povolení/zakázání ohlašovacích řetězců
AT+KC LocalName	Nastavení názvu, pod nímž se zařízení hlásí. (Pozbývá platnosti po HW resetu či odpojení napájení)
AT+KC Security	Nastavení úrovně zabezpečení
AT+KC SmartCableSetup	Příkaz pro povolení a nastavení *Smart Cable* zařízení
AT+KC StreamingSerial	Povolení/zakázání *StreamingSerial*
AT+KC UpdateInquiryScan	Nastavení vyhledatelnosti pro ostatní zařízení
AT+KC UpdatePageScan	Obnovení parametrů zobrazených ostatním zařízením při vyhledávání
AT+KC Version	Získání aktuální verze použitého modulu

Tabulka 3.2: Seznam AT příkazů

3.5 Řídící mikroprocesor

Řídící mikrokontrolér je nejdůležitější částí celého zapojení. Jeho firmware obstarává veškerou činnost spojenou s odměřením dat, jejich zabezpečením a následným odesláním do PC. Pro tuto aplikaci jsem zvolil obvod Atmega48. Mikrokontrolér ATmega48L patří do rodiny AVR osmibitových procesorů RISC od firmy ATMEL.AVR architektura vychází s koncepce rychle přístupného registrového pole, které obsahuje 32 obecně použitelných registrů délky 8 bitů. Přístup do registrového pole je proveden v jediném strojovém cyklu. To znamená, že během jednoho strojového cyklu lze vykonávat jednu aritmeticko-logickou operaci. Oba operandy aritmeticko-logické instrukce jsou načteny z registrového pole, operace je provedena a výsledek směřuje opět do registrového pole. To vše v jediném strojovém cyklu. Díky této technice nabízejí mikrokontroléry AVR velký výpočetní výkon. Na svém čipu mají AVR kontroléry integrovánu spoustu periferních obvodů. Mezi nimi například sériové rozhraní USART, Analogově-Digitální převodníky, Analogové komparátory a rovněž podpůrné obvody pro hlídání napájecího napětí a správného běhu programu. Právě díky těmto vlastnostem se jeví Atmega48 jako vhodná volba.



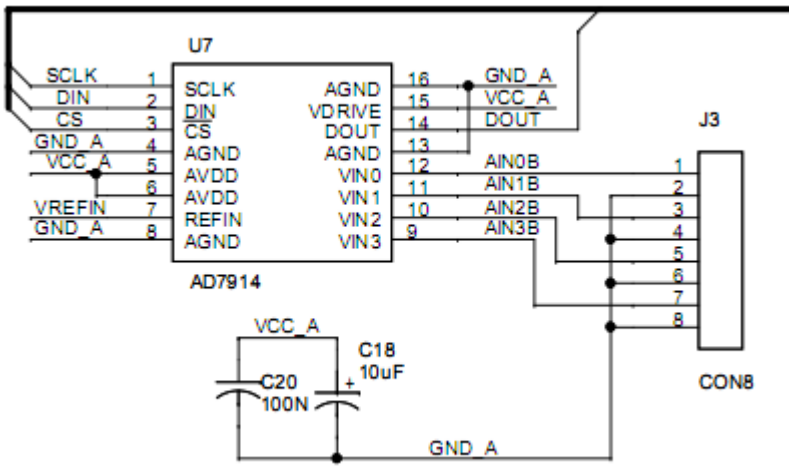
Obr. 3.6 : Zapojení řídícího mikrokontroléru

Ke své funkčnosti potřebuje procesor minimální konfiguraci. Vyžaduje pouze připojení napájecího napětí v rozsahu 2.7V – 5V. Již z dříve uvedených důvodů, je námi zvolená hodnota napájení 3.3V. Co nejblíže napájecím vývodům jsou připojeny blokovací kondenzátory odstraňující případné napěťové špičky. RC kombinace tvořena spojením

rezistoru R12 a kapacitoru C13 ošetřuje resetovací obvody procesoru. Po přivedení napájení vytvoří RC článek krátký impuls, čímž dojde k resetování celého zařízení.. Abychom mohli komunikovat s BlueTooth modulem, bylo zapotřebí doplnit zapojení o externí krystal Y1 s kmitočtem oscilací 14.7456MHz. Nelze využít interní oscilátor, jenž je integrován na čipu procesoru. Ten kmitá s frekvencí 8MHz. Z této hodnoty by však nebylo možné s dostatečnou přesností odvodit přenosovou rychlost pro sériovou linku. Hodnoty kapacitorů C25 a C26 jsou voleny tak, aby bylo zaručeno nasazení oscilačních kmitů a vycházejí z doporučení uvedeného v katalogových listech výrobce. Odporový dělič, tvořený kombinací R40 a R41 je zapojen pro případ, že by bylo potřebné měřit a odesílat do řídící aplikace i skutečnou hodnotu napájecího napětí. Abychom využili desetibitovou přesnost analogové digitálních převodníků, co možná nejefektivněji, je zapotřebí oddělit jejich napájení od napájení zbývající ryze digitální části. Tuto úlohu plní kombinace tlumivky L3 a kapacitoru C17. Jejich hodnoty byly opět zvoleny na základě doporučení výrobce.

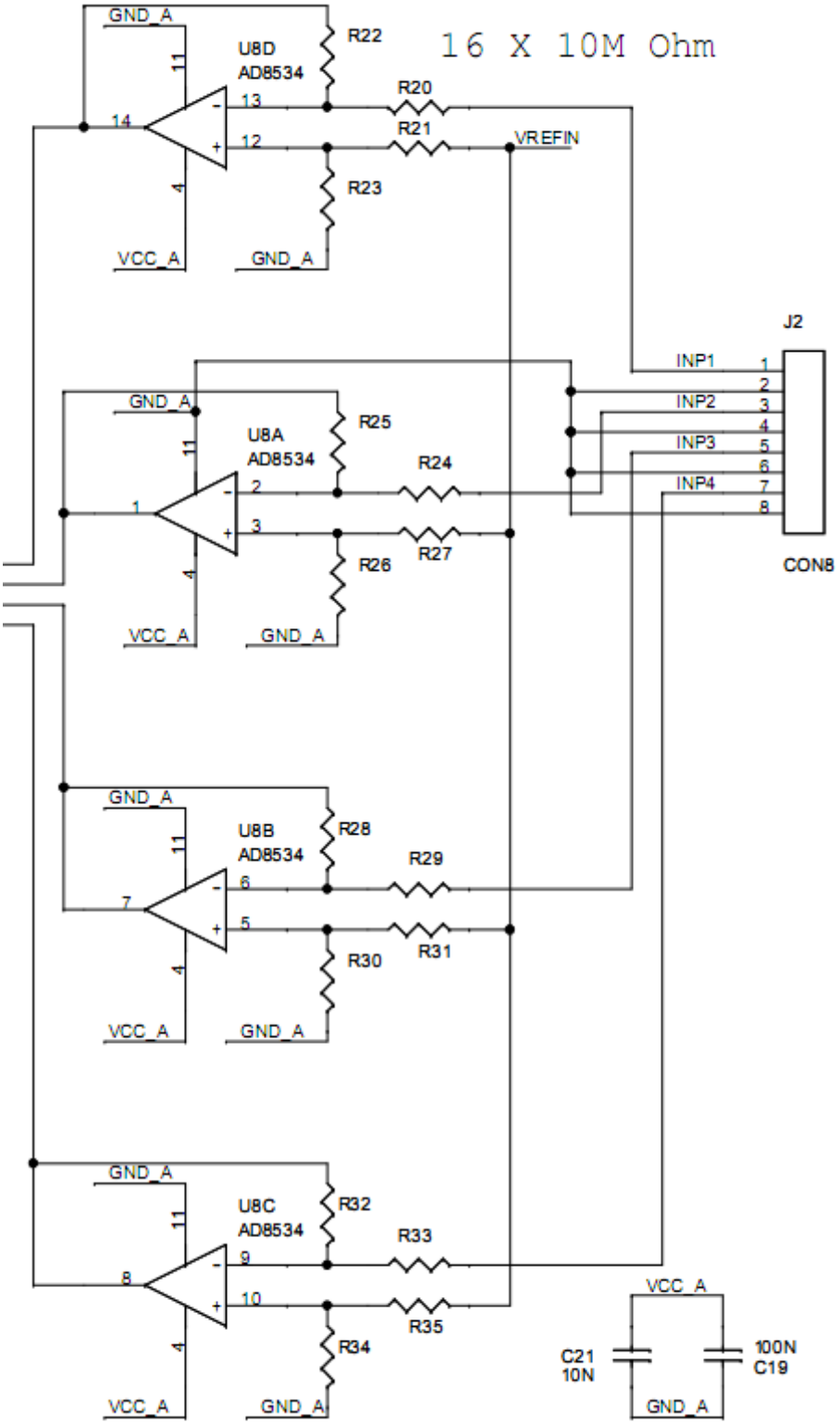
3.6 Anologové obvody

Navržené zařízení poskytuje uživateli osm měřících kanálů. Polovina kanálů je tvořena přímo vstupy analogově digitálního převodníku. Ty jsou vyvedeny na vstupní konektor. Rozsah napěťových úrovní, přiváděných k těmto vstupům se smí pohybovat v rozmezí 0V - Uref. Jelikož se jedná pouze o ukázkovou aplikaci, nejsou vstupy převodníku chráněny žádnými doplňujícími obvody a je zapotřebí dbát zvýšené opatrnosti při jejich používání. Jako obvod obstarávající AD převod byl zvolen AD7914. Ten patří do široké rodiny převodníků, které nabízí firma Analog Devices. S nadřazenými řídícími obvody komunikuje prostřednictvím čtyř-vodičové sériové sběrnice. Schématické zapojení převodníku ukazuje obrázek č. 3.7.



Obr. 3.7 : Zapojení převodníku AD7914

Zbývající čtyři kanály našeho zařízení jsou realizovány pomocí Analogově digitálního převodníku, integrovaného přímo na čipu řídícího mikrokontroléru. Vstupní signál přiveden na konektor J2 je nejprve zpracován pomocí jednoduchých rozdílových zesilovačů. Jejich vstupní úrovně jsou vztaženy vůči referenčnímu napětí. Díky takto uměle vytvořené zemi získáváme symetrický rozsah vstupů pro **– Uref /2** až **Uref / 2.** Konkrétní realizaci vstupních kanálů vidíme na obrázku č. 3.8.



Obr. 3.8: Provedení vstupních obvodů převodníku

Stejně jako již dříve zmíněný AD převodník, pochází i použitý operační zesilovač z produkce firmy Analog Devices. Omezujícím kriteriem při jeho výběru byl především rozsah napájecích napětí. Šumové parametry, rychlosti přeběhu ani Real-To-Real výstup nebyly v případě naší aplikace tolik podstatné. Zesílení celé soustavy je nastaveno na úroveň 1. Úkolem OZ tedy není předzesílení slabých signálů, ale poskytnutí vyššího vstupního odporu a umožnění symetrizace vstupů.

4 VÝVOJ SOFTWARE PRO MIKROKONTROLER

4.1 Programovací jazyky pro mikrokontrolery

V současné době se pro programování mikrokontrolerů stále více uplatňuje použití vyšších programovacích jazyků. Je to způsobeno především tím, že řada výrobců vyvíjí procesory, jejichž jádro je těmto programovacím metodám přizpůsobeno. Rovněž došlo k poměrně masivnímu vývoji kompilátorů atak v současné době asi nenajdeme platformu, pro níž by již nebyl vyvinut překladač z jazyka C, DELPHI, PASCAL či jim podobných. Pomalu tak začíná být vytlačován **ASSEMBLER**, tedy jazyk symbolických adres, který se dříve pro tvorbu firmware využíval.

I přes stále vyšší využití již uvedených strukturovaných jazyků však nelze opomenout ani fakt, že assembler nabízí i některé výhody. Hlavní výhoda jazyka symbolických adres spočívá v možnosti absolutní kontroly nad výsledným kódem. Programátor může velmi efektivně optimalizovat zdrojový kód na velikost nebo na rychlost.

Naopak značnou nevýhodou programování aplikací v jazyce assembler představuje skutečnost, že každý výrobce využívá zcela odlišnou instrukční sadu. Rovněž architektury jádra procesoru jsou často velmi odlišné. To znesnadňuje přechod mezi jednotlivými typy procesorů. Na programátora klade použití jazyka symbolických adres poněkud vyšší nároky. Nejen, že musí být dobře obeznámen s architekturou a konkrétními registry použitého obvodu, ale musí si umět rozložit daný algoritmus na elementární operace odpovídající příslušným instrukcím.

Další výraznou nevýhodou je také skutečnost, že pro rozsáhlejší a složitější kódy ztrácí assembler na přehlednosti a mnohem hůře se v něm programátor orientuje.

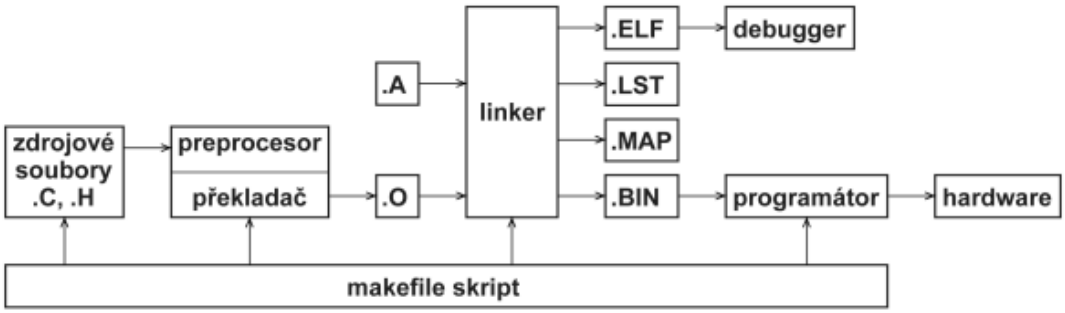
Z výše uvedených důvodů jsem při volbě vhodného programovacího jazyka sáhl po dobře osvědčeném C. Překladačů lze pro tento jazyk sehnat celou řadu, mezi jinými jmenujme například **Code VisionAVR, C Compiler**, **IAR's C Compiler**, **ICCAVR**, **WinAVR.** Poslední jmenovaný lze snadno a zcela zdarma stáhnout z internetu. Jelikož jej využívám již delší dobu, rozhodl jsem se právě pro něj i tentokrát.

4.2 VinAVR

Jedná se o programový balík, která obsahuje celou řadu potřebných a užitečných nástrojů. Ty programátorovi při tvorbě kódu značně usnadňují práci. Standardně balíček zahrnuje následující komponenty

- **GCC –** překladač ANSI C
- **GDB –** debugger
- **Simul AVR –** podpora simulace pro debugger GDB
- **AVR Dude –** programátor mikrořadičů AVR
- **Programmer's notepad –** textový editor pro psaní zdrojového kódu
- **AVR-libc Manual –** dokumentace funkcí jazyka C pro AVR
- **Dokumentace PDF i HTML**

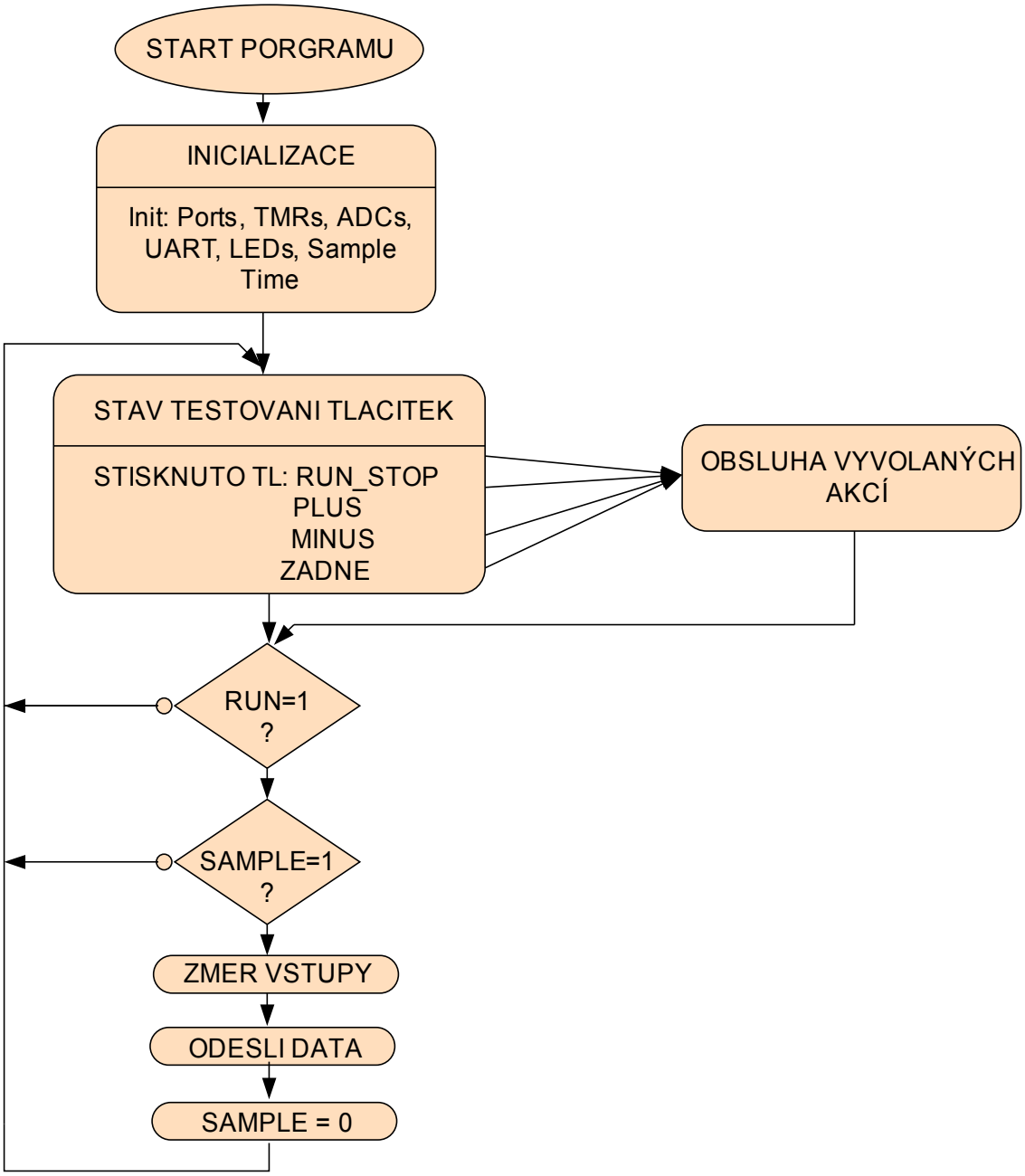
V následujícím odstavci přiblížím způsob, jakým překladač pracuje. Nejprve jsou zpracovány zdrojové soubory (*.c), Tuto část obstarává preprocesor. Ten načte veškeré hlavičkové soubory, poté rozbalí všechna makra a vzniklý kód předá překladači, který jej přeloží do tzv. Relativního kódu. Pod pojmem relativní kód mějme na mysli strojový s relativními adresami identifikátorů. Vytvořený relativní kód předá překladač linkeru, programu, který sestaví výsledný zdrojový soubor v hexadecimálním, či binárním formátu. Výstupem linkovacího programu není jen samotný zdrojový kód, ale také řada dalších pomocných souborů. Mezi ty nejvýznamnější patří soubory ***.lst**, ***.map**, ***.elf.** Soubor s příponou „lst" obsahuje zdrojový kód, prokládaný řádky v assembleru. Máme tedy možnost prohlédnout si způsob, jakým naše funkce překladač skutečně realizuje. V souboru označeném příponou map, nalezneme adresy identifikátorů přidělené překladačem. Posledním souborem je soubor označený jako *.elf. Ten lze načíst do debuggeru GDB anebo do AVR studia a využít jej tak při ladění a simulacích našeho kódu. Obrázek 4.1 graficky znázorňuje proces překladu zdrojových kódů.



Obr. 4.1 : Proces překladu zdrojových kódů

4.3 Firmware

V této podkapitole popíši základní programové vybavení mikrokontroléru a pokusím se přiblížit funkci nejvýznamnějších částí kódu. Jak již bylo dříve uvedeno, byl firmware napsán v jazyce C a následně přeložen pomocí kompilátoru WinAVR. Vývojový diagram, znázorňující běh programu je na obrázku č. 4.2.

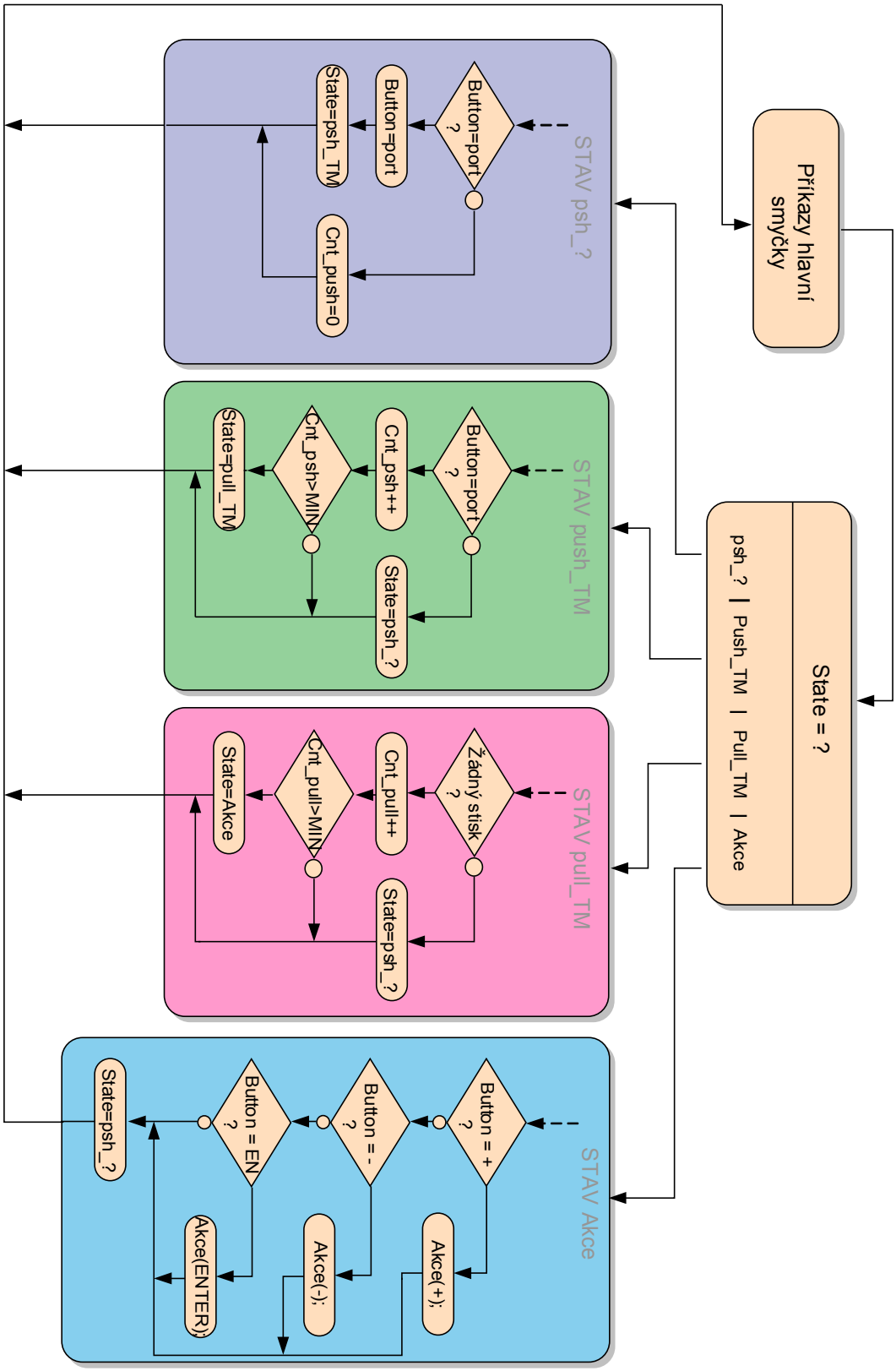


Obr. 4.2: Vývojový diagram řídícího programu mikrokontroléru

Celý řídící program je koncipován do podoby strojového automatu. Ten periodicky v nekonečné smyčce testuje své vstupy a v závislosti na jejich stavech vykonává příslušné operace. Veškeré vzorkování vstupů a testování podmínek je řízeno přesnými hodinovými pulzy, odvozenými od vnitřních časovačů. Nyní blíže popíši, případně podrobněji rozkreslím jednotlivé fáze, jež zevrubně znázorňuje uvedený vývojový diagram. Zdrojové kódy jsou ke shlédnutí v příloze a zde je uvádět nebudu.

Po té co přivedeme k procesoru napájecí napětí, provede mikrokontrolér inicializaci veškerých svých periferií. Nejprve nastaví správné alternativní funkce portů, určí které piny budou vstupní, které výstupní. Tak dojde k rozsvícení indikačních LED diod. Následně povolí funkci analogově-digitálního převodníku a nastaví jeho parametry. Spustí vnitřní časovač generující hodinové pulzy s periodou 1ms, zapne sériovou linku a jako poslední krok povolí příslušné přerušovací vektory.

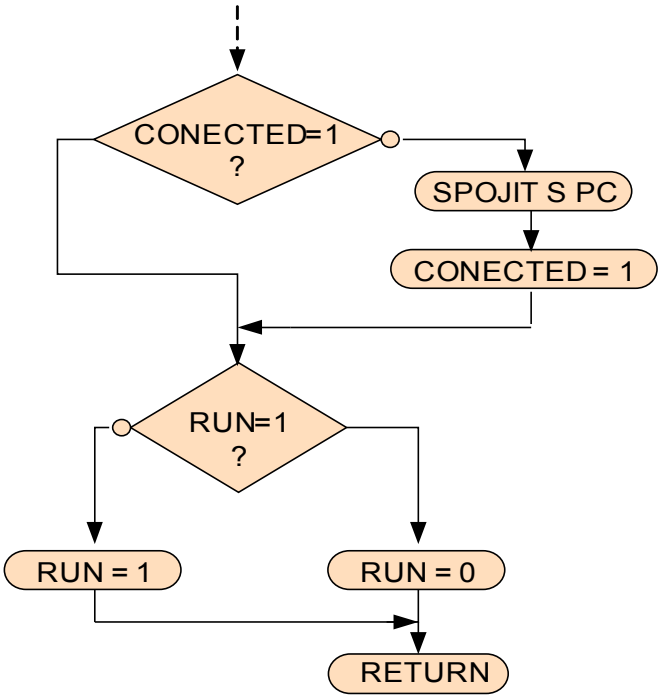
Po ukončení inicializačního procesu přechází procesor do běhu v nekonečné smyčce. V jejím průběhu cyklicky testuje stavy tlačítek a bitů, které řídí větvení celého programu. Aby bylo dosaženo maximální efektivity, rychlosti a spolehlivosti obsluhy tlačítek, jsou veškeré funkce s tlačítky spojené realizovány formou vnořeného stavového automatu. Hlavní program tak nečeká v žádné smyčce, kde by ošetřoval zakmitávání spínačů, nebo měřil délku stisku. Veškeré vyhodnocení stavů ovládacích prvků, ošetření zákmitů a vykonání požadované funkce trvá vždy konstantní dobu, bez ohledu na mechanické poruchy tlačítek, nebo délky jejich stisku. Vývojový diagram tohoto vnořeného automatu je uveden na obrázku 4.3.



Obr. 4.3: Vývojový diagram vnořeného automatu obsluhy tlačítek

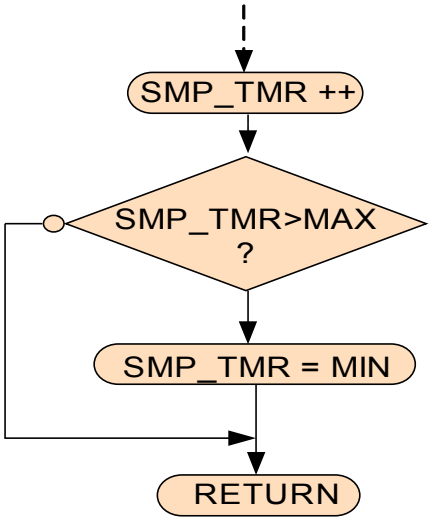
Princip takto provedeného automatu je velice jednoduchý a jeho hlavní výhoda spočívá především ve velmi dobré přehlednosti celého kódu. Zpracování programu probíhá následovně. Celý automat volá hlavní programová smyčka jako datovou strukturu. Tuto strukturu bychom mohli označit pomocí zkratky FSM, neboli Frame State Mashine. Složitější programy mohou obsahovat podobných struktur celou řadu, a proto k volání dochází pomocí pointeru jenž ukazuje na objekt se kterým chceme pracovat. Po té co hlavní smyčka předá řízení zvolené struktuře, dojde ke zjištění stavu, který má být vykonáván.

V našem konkrétním případě vše funguje následovně. Při prvním zavolání automat vstoupí do stavu ***push_?***. Zde pouze otestuje, zda-li je stisknuto některé z tlačítek. Pokud není, pouze předá řízení do hlavní programové smyčky. Naopak bylo-li vyhodnoceno některé tlačítko jako stisknuté dojde k uložení jeho „kódu“ (binární podoby na portu procesoru) a do ukazatele stavu je přiřazena hodnota stavu ***push_TM***. V průběhu následujícího volání stavového automatu tedy začne mikrokontrolér porovnávat uložený kód tlačítka s aktuální hodnotou portu. Souhlasí-li inkrementuje si do pomocného čítače počet úspěšných testů. Obsluha stavu ***push_TM*** bude volána dokud délka stisku nedosáhne hodnoty MIN, dosažením této konstanty se ukazatel stavu přepne na ***pull_TM.*** Větev ***pull_TM*** pracuje stejným způsobem, jako již popsaná ***push_TM***. Jediný rozdíl spočívá v tom, že zde měříme dobu, po kterou je dříve stisknuté tlačítko puštěno. Načítáním hodnoty MIN bude naplněn ukazatel stavu na poslední možnost, tedy stav ***Action***. V této větvi mikrokontrolér vyhodnotí kód stisknutého tlačítka a na základě výsledku zavolá příslušnou obslužnou rutinu. Pomocí takto vytvořeného algoritmu se velmi spolehlivě ošetří veškeré zákmity, způsobené mechanickou nedokonalostí spínačů. Lze také snadno rozlišovat mezi stisknutím a delším podržením tlačítka, atak jednomu ovládacímu prvku přiřadit i více funkcí. Implementace jednotlivých akcí, ukazují opět vývojové diagramy na následujících obrázcích. (Obrázky č.4.4, č.4.5, č.4.6).

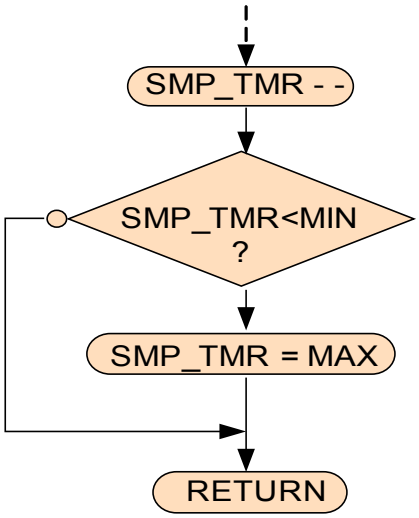


Obr. 4.4: Vývojový diagram funkce Run_Stop()

Obrázek 15 popisuje funkci, která bude volána při stisku tlačítka označeného RUN/STOP. Pokud naše aplikace zatím nekomunikuje s obsluhujícím počítačem, vyhledá procesor dostupné zařízení a naváže s ním spojení. Pokud již komunikace byla navázána, přepíná tento spínač mezi stavem, kdy je činnost zařízení pozastavena, nebo kdy dochází k pravidelnému vzorkování.



Obr. 4.5: Vývojový diagram funkce Plus()



Obr. 4.6: Vývojový diagram funkce Minus()

Kód procedur *Plus()* a *Minus()* vykonává obdobnou činnost. Jediným úkolem těchto funkcí je zvyšování, respektive snižování periody vzorkování.

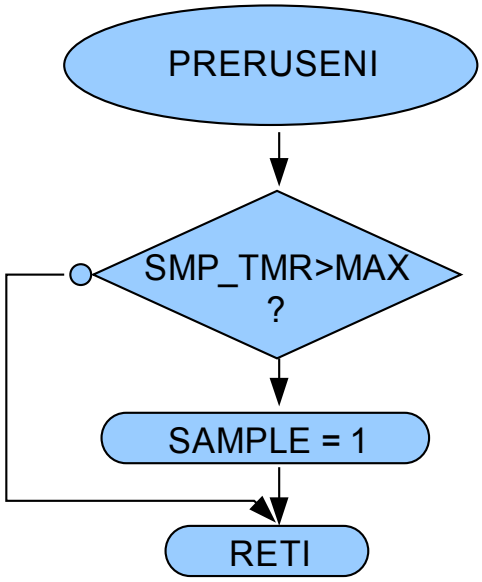
Po vykonání příslušného stavu obsluhy tlačítek se řízení předá zpět hlavnímu automatu. Provede test příznakového bitu RUN. Na základě jeho hodnoty pozná mikrokontrolér v jakém se právě nachází stavu. Je-li bit nulován, vrací se celý program na začátek hlavní smyčky a vše se opakuje. V opačném případě, tedy tehdy bude-li RUN nastaven do logické 1, provede procesor test flagového bitu SAMPLE. Ten může být nastavován pouze v rutině přerušení a kmitočet kterým se mění udává rychlost vzorkování. Je-li i hodnota tohoto nastavena na log. 1, pak dojde k volání procedury ***measure()***, ovzorkují se všechny vstupy a změřené hodnoty budou odeslány do připojeného zařízení. Nebude-li bit SAMPLE nastaven dojde opět k programovému skoku na začátek nekonečné smyčky.

4.4 Přerušení

Jak již bylo zmíněno v předchozí kapitole běh hlavní smyčky programu je synchronizován pomocí hodinových pulzů. Tato synchronizace je nutná především pro potřebu přesné periody vzorkování analogového obvodu. Pro dosažení co největší přesnosti se jako nejvhodnější řešení nabízí použití některého z integrovaných čítačů-časovačů ve spojení s přerušovacím systémem procesoru. Výhody asynchronních přerušení, jsou jistě zřejmé a nebudu je zde uvádět. V následujících řádcích pouze přiblížím princip jakým pracuje přerušovací systém procesorů AVR. Celý systém z pohledu uživatele působí velmi jednoduše. Dojde-li k vyvolání přerušení, program skočí na pevně danou adresu v paměti. Tato adresa se nazývá **přerušovací vektor (**též **vektor přerušení)** a je pevně přiřazena každému typu přerušení. Obslužná rutina psaná v jazyce C pak může vypadat následovně.

```
ISR(TIMER0_COMP_vect)
{
/* Obslužný kód */
}
```

Překladač z tohoto jednoduchého zápisu pozná, že pro přerušení vyvolané časovačem TMR0 má vykonávat *obslužný kód*. Tato rutina by měla být vždy co nejkratší, tak aby se všechny příkazy stihly vykonat před příchodem další žádosti o přerušení. Zároveň by nemělo docházet k blokování ostatních přerušení. Vývojový diagram naší obslužné rutiny znázorňuje obrázek 18.



Obr. 4.7 : Obsluha přerušení

Obsluha je omezena pouze na počítání uplynulých milisekund a porovnání s hodnotou, která udává periodu vzorkování. Dosáhne-li načítaná hodnota přednastaveného intervalu, dojde k nastavení příznakového bitu, jenž povoluje odebrání dalších vzorků. Vlastní změření nových hodnot neprobíhá v rámci přerušení, ale v hlavní programové smyčce.

5 POPIS ZPŮSOBU KOMUNIKACE S PC

Jak již je vymezeno v zadání této práce, propojení s osobním počítačem musí být bezdrátové, realizované pomocí standardu BlueToth. K navázání komunikace a nastavení přenosových parametrů slouží AT příkazy, jenž byly popsány v podkapitole **X.4 Vysokofrekvenční obvody** zabývající se popisem použitého BT modulu. Dále zde tyto příkazy nebudu rozvádět a spíše se pokusím přiblížit jakým způsobem probíhá komunikace po té, co již bylo fyzicky navázáno spojení. Přenosový kanál BlueTooth přejde do režimu, ve kterém se chová stejně jako sériové rozhraní UART.

Jedná se tedy o standardní sériovou linku, po níž jsou naměřená data posílána v podobě krátkých paketů. Každý takovýto paket má pevně stanovenou délku. Skládá celkem z 8 datových byte a třech byte, které jsou pomocné. Tak jak je v současné době zařízení koncipováno, není využito oboustranného přenosu. Veškerá komunikace probíhá pouze směrem od měřící aplikace do počítače. Význam datových byte je zcela jistě patrný. V těchto slovech jsou obsažena naměřená data jednotlivých kanálů. Bližší popis bych rád věnoval pomocným bytům, jež doplňují celkový přenášený paket. Názvy těchto tří byte jsou následující:

HEAD

STATUS

CRC

výsledný paket který vznikne jejich spojením uvádí následující obrázek s číslem 5.1.

Obr. 5.1: Tvar odesílaného paketu

HEAD - tento byte je vždy odesílán na první pozici paketu. Obsahuje jeden znak, který udává význam zbývajících slov paketu. Porovnáním tohoto znaku tedy zjistí obslužný software jakým způsobem zbytek přijatých dat zpracovat. Jedná-li se o paket, nesoucí naměřená data, potom bude hodnota *HEAD = 0x51*. Řídící program *t*akto označený paket uloží do paměti, z datových byte vypočte aktuální hodnoty a ty zobrazí na příslušných displejích. Krom hodnoty 0x51 může hlavička nabývat ještě hodnot ASCII znaků pro písmena S a P. Pokud bude v prvním byte obsažen jeden z těchto znaků, pak zbývající odeslaná data obsahují řetězec START, nebo PAUSE. Program, jenž přijatá data zobrazuje a vyhodnocuje, tak získává informaci o tom, že obsluha aplikace pozastavila, případně spustila měření dat.

STATUS – Následuje vždy ihned po odeslání hlavičkového byte. V současnosti je využíván pouze z části. Většina jeho bitů je zarezervována pro možné budoucí použití při rozšiřování funkcí naší aplikace. Následující obrázek znázorňuje rozložení jednotlivých bitů. Jejich význam bude popsán níže.

Bit0	Bit1	Bit2	Bit3	Bit4	Bit5	Bit6	Bit7
SMP2	SMP1	SMP0	Res.	F8/16	Res.	Res.	LST

Obr. 5.2: Rozložení bitů stavového slova

F8/16 - tento bit určuje formát zasílaných dat. Je-li nulován, mají data 8 bitovou délku, dochází k zarovnání vlevo a nejnižší dva bity jsou odříznuty. Pokud je bit na staven do logické 1, odesílá se plná informace, tedy šestnáct bitů, z nichž význam mají pouze bity 0–9.

LST – *Last.* Nastavením tohoto bitu do logické 1 má aplikace možnost oznámit počítači, že se jedná o poslední zasílaný paket.

SMP0, SMP1, SMP2 – Pro správné grafické znázornění odměřených dat je zapotřebí uchovávat informaci o časovém měřítku. Následující tabulka ukazuje způsob kódování vzorkovací frekvence do horních třech bitů stavového slova.

SMP2	SMP1	SMP3	Vzorkovací frekvence
0	0	0	0,1 Hz
0	0	1	0,5 Hz
0	1	0	1 Hz
0	1	1	5 Hz
1	0	0	10 Hz
1	0	1	50 Hz
1	1	0	100 Hz
1	1	1	Rezervováno

Tabulka 5.1: Kódování vzorkovací frekvence

CRC – *Control Redundanty Code. CRC* byte je odesílán na konci každého paketu. Jedná se o zabezpečovací data, která napomáhají odhalit chybu, nebo chyby, vzniklé v rámci přenosu. Program si po přijetí nových dat spočte jejich CRC, pokud se shodne hodnotou CRC, která byla přijata, jsou data uznána jako platná a předána k následnému zpracování. V opačném případě bude přijatý paket zahozen. Určení konkrétních pozic chybných bitů a jejich oprava se neprovádí. Způsobů výpočtu CRC existuje celá řada. Pro naše použití se jako zcela dostačující jeví použití funkce exclusive-or mezi jednotlivými byty. Výpočet pak vypadá následovně:

$$CRC = (CRC)\ XOR\ (i\text{-}tý\ byte\ dat)$$

6 VÝVOJ SOFTWARE PRO WINDOWS

6.1 Přiblížení problematiky

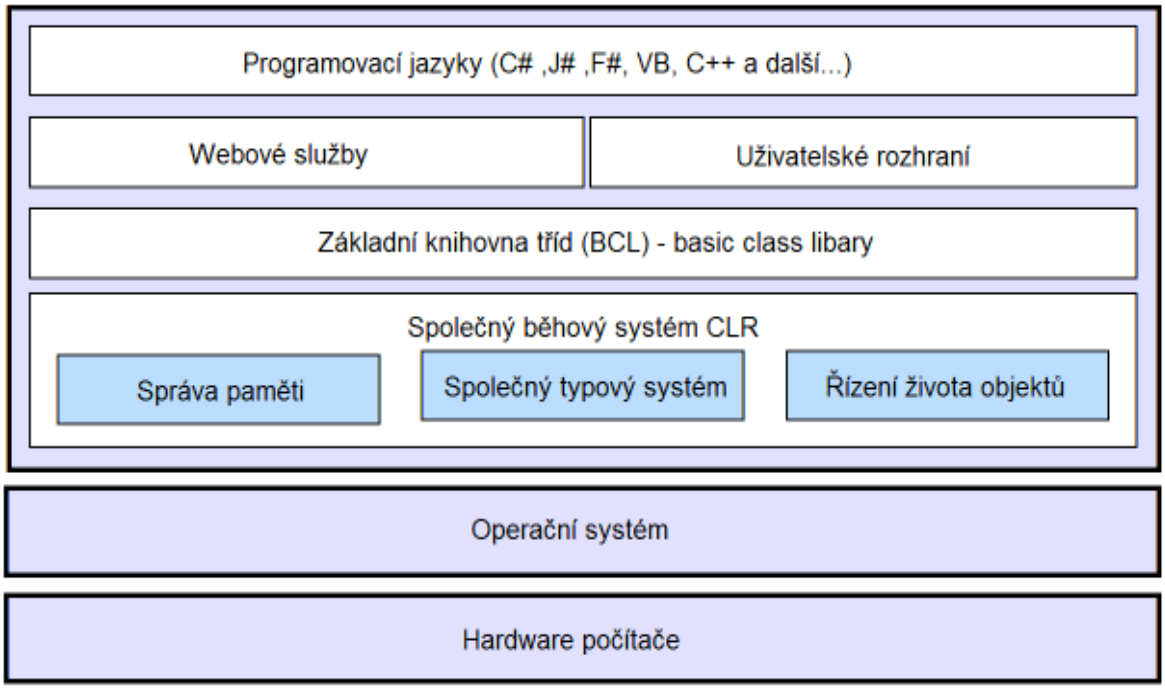
Ze specifikace zadání této bakalářské práce jednoznačně vyplývá rovněž druhá část návrhu. Tou je základní softwarové vybavení pro osobní počítače, s jehož pomocí bude možno alespoň jednoduché zobrazení naměřených dat. Na program neklade zadání žádné specifické požadavky, není určena ani platforma, pod níž by měla aplikace být spouštěna, ani programovací jazyk, ve kterém bude kód napsán. Mým vlastním požadavkem bylo, aby výsledný program měl podobu klasické „okénkové“ aplikace. Aby jeho ovládání bylo pokud možno velmi snadné a co nejvíce intuitivní. V základní podobě půjde o jednoúčelový software, jenž umí přijmout a zobrazit data. Později by však mělo jít snadno tento program rozšířit o řadu užitečných funkcí, jako například ukládání dat do souboru, vykreslování a tištění grafů, případně různé matematické operace. Jako programovací jazyk jsem zvolil C#. Jedná se o vysokoúrovňový objektově orientovaný programovací jazyk vyvinutý společností Microsoft zároveň s platformou .NET. Vychází z jazyků C++ a Java. Syntaxi čerpá z jazyka C. Zmíněná platforma .NET nabízí řadu výhod z nichž za nejvýznamnější je považována podpora více programovacích jazyků a vysoká úroveň integrace mezi nimi.

6.2 Seznámení s platformou .NET Framework

Hovoříme-li o platformě .NET Framework, máme na mysli pouze jeden ze systémů infrastruktury .NET. Tato platforma je velmi rozsáhlá a složitá. Její infrastruktura se skládá z následujících systémů:

- **.NET Framework**
- **Microsoft Visual Studio.NET**
- **.NET Enterprise Servers**
- **Microsoft Windows.NET**

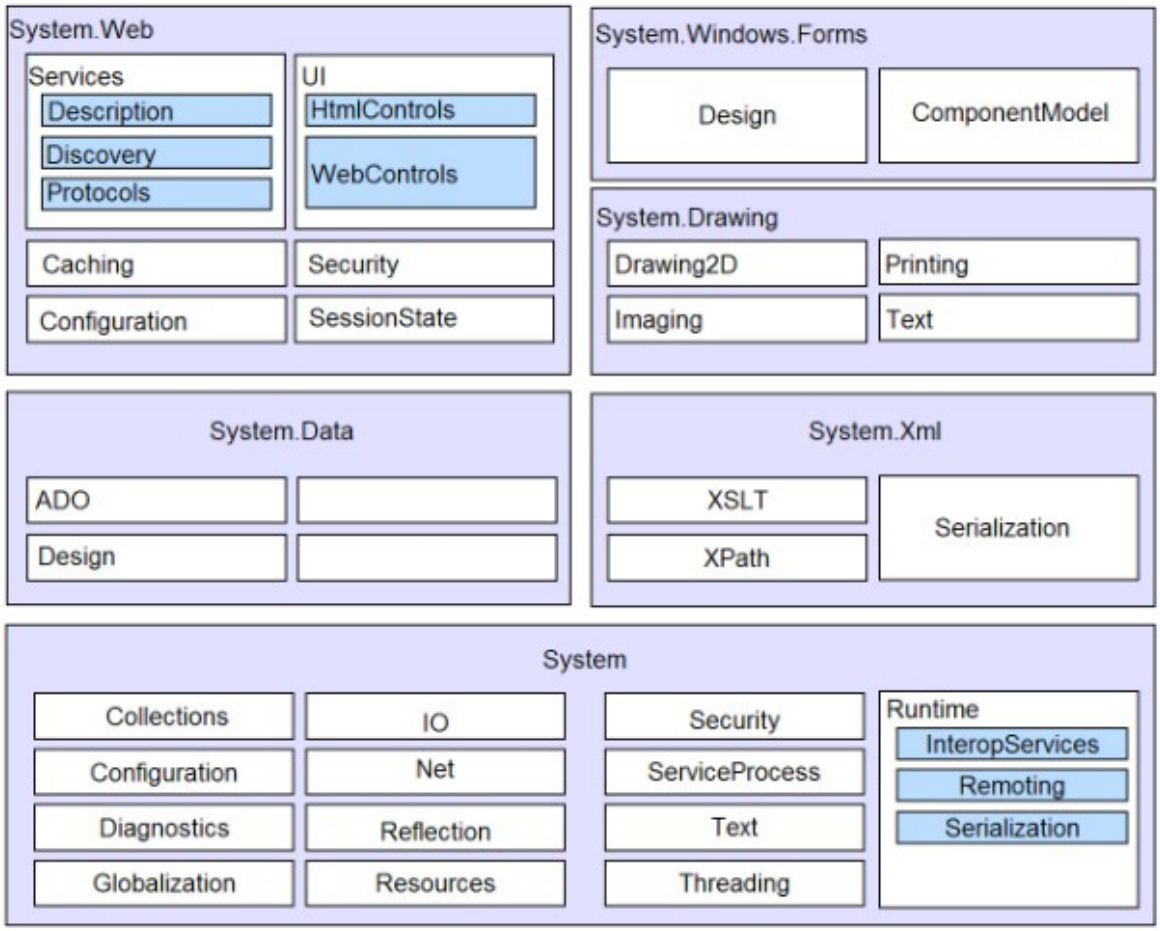
V této kapitole bych rád objasnil pojmy, které přímo souvisí se systémem .NET Framework. Jedná se především o CLR, CLS, BCL, MSIL a JIT. Jde o základní terminologii, jejíž význam bychom měli znát.



Obr. 6.1: Schéma struktury .NET Framework

6.3 Knihovny třídy systému .NET Framework

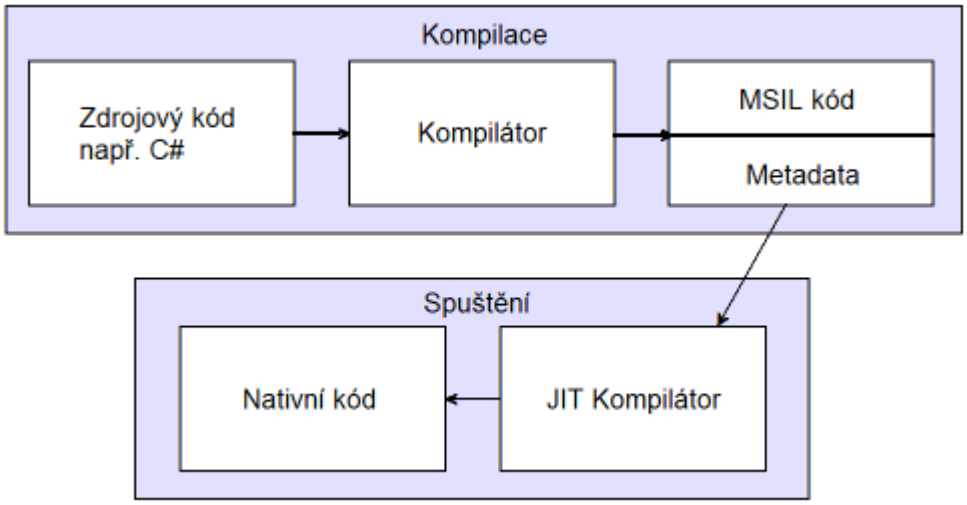
Společná knihovna v praxi znamená, že všechny programovací jazyky, které ji používají, mají teoreticky naprosto stejné možnosti. Opět vyvstává otázka, zda je výhoda, že všechny jazyky umí stejné věci. Dle mého názoru se jedná o dobrou věc, člověk si nemusí pamatovat syntaxi dalších programovacích jazyků a vystačí si pouze s jedním.



Obr. 6.2: Knihovny systému .NET Framework

6.4 MSIL (Microsoft Intermediate Language)

Aby mohli tvůrci programovacích jazyků vytvářet další jazyky pro platformu .NET, vyvinul Microsoft jazyk MSIL, který je podobný assembleru. Při kompilaci programu pro systém .NET bere překladač jako vstup zdrojový kód příslušného jazyka a výstupem je právě aplikace v MSIL (ještě se nejedná o strojový kód!). CLR zajistí, že při prvním spuštění aplikace se program z MSIL přeloží pomocí JIT (just-in-time) kompileru do strojového kódu. Je možné vyvíjet programy i v jazyce MSIL, ale je to velmi nepraktické. Jazyk MSIL je především zajímavý pro tvůrce programovacích jazyků a kompilátorů.



Obr. 6.3: Princip kompilace v prostředí .NET Framework

6.5 CLR (Common Language Runtime)

Jedná se o běhový systém. Ten si můžeme představit jako virtuální stroj, ve kterém pracují aplikační funkce platformy .NET. Všechny jazyky v něm mají k dispozici knihovny tříd systému. Jednoduše řečeno, na vašem počítači běží další systém, který spravuje aplikace pro .NET. Pro tento systém existuje obdoba strojového kódu, nazývaná MSIL. Mnohokrát se setkáte v souvislosti s platformou .NET s otázkou výkonnosti, protože aplikace napsané pro .NET jsou omezeny výkonností prostředí, ve kterém běží, tedy CLR. Tato skutečnost je způsobena tím, že kód není kompilován přímo do strojového kódu, ale do MSIL. Tento nedostatek vynahrazují mnohé výhody, jako řízený kód, jednotný typový systém a správa paměti (garbage collector).

Další výhodou je, že pro platformu .NET můžete vyvíjet v mnoha jazycích. Tato vlastnost se nazývá *"cross-language interoperability"* - možnost spolupráce mezi jazyky. Je jí dosaženo tím, že každý kompilátor daného jazyka musí splnit CLS (Commnon Language Specification). Jde o určitou sadu pravidel, která musí kompilátor dodržovat, aby aplikace po překompilování fungovaly v .NET.

6.6 JIT (Just In Time)

1.Generování kódu při instalaci

Při tomuto způsobu vytváříte strojový kód přímo při instalaci programu a již se nemusíte bát pomalého prvního spouštění.

2. Metoda JIT

Jde o běžný způsob kompilace. Metody jsou kompilovány při jejich prvním volání.

3. EconoJIT

Jde o způsob kompilace, který je v hodný pro zařízení s omezenými zdroji a nedostatkem paměti. Tímto způsobem zajistíme, že EconoJIT pozastaví kompilaci, pokud má systém nedostatek paměti.

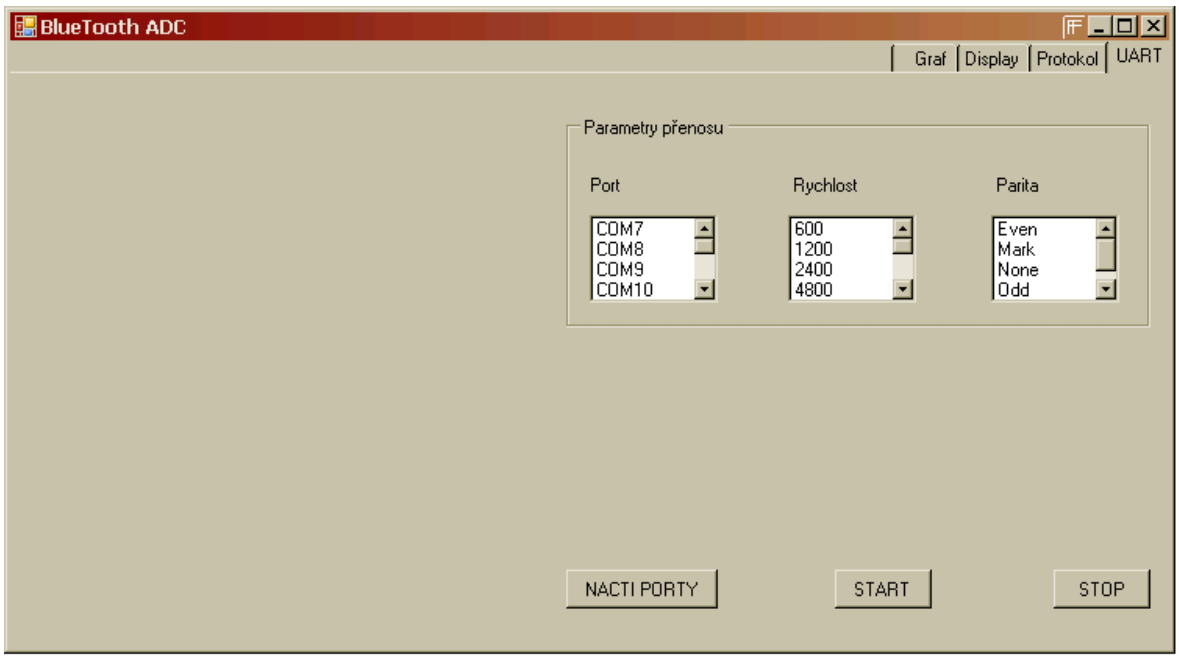
7 POPIS ŘÍDÍCÍ APLIKACE A JEJÍ OVLÁDÁNÍ

Zvolenou platformu i programovací jazyk jenž byl pro napsání této aplikace zvolen jsem přiblížil v minulé kapitole. Nyní bych rád popsal vlastní provedení tohoto programu. Mou hlavní snahou byla především jednoduchost, snadná ovladatelnost a co největší přehlednost. Vzhledem ke složitosti objektového programování, spojené s obsluhou asynchronních událostí od jednotlivých prvků a s ošetřováním případných výjimek, nebudu zde na rozdíl od kapitoly popisující funkci firmware uvádět vývojové diagramy. Zaměřím popis na základní přiblížení jednotlivých ovládacích prvků a popis programu jako celku.

Po spuštění souboru ***BT_ADC.exe*** se uživateli otevře klasická windowsovská aplikace. Hlavní okno nabízí čtyři záložky, z nichž jako první je otevřena karta s nastavením parametrů sériového připojení. Uspořádání jednotlivých prvků ukazuje obrázek č. 21. Uživatel má možnost nastavit základní parametry přenosu jako jsou

- ***Rychlost***
- ***Parita***
- ***Port***

Výběr poskytuje standardní vlastnosti. Jediný parametr, který musíme zvolit povinně představuje jméno přístupového portu. Stiskneme-li tlačítko *START*, bez vybraného portu, budeme upozorněni varovným hlášením. Rychlost ani parita být zvolena nemusí, program pak sám vybere defaultně přednastavenou hodnotu. Význam tlačítek by měl být zřejmý již z jejich názvu. Kliknutím na tlačítko *START* dojde k pokusu o připojení k vybranému sériovému portu. *STOP* komunikaci ukončuje a používaný port uvolní. Tlačítko *NACTI PORTY* slouží pro načtení nově nalezených portů.



Obr. 7.1: Záložka nastavení sériového portu

.

Druhá záložka zobrazuje uživateli aktuální nastavení přenosového protokolu. Nabízí možnost výběru jednotlivých kanálů, které mají být měřeny, výběr formátu dat. V současné verzi programu není možno ovládací prvky využít k nastavení jednotlivých parametrů, neboť nedochází k oboustranné komunikaci mezi softwarem a naší aplikaci.

Obr. 7.2: Rozložení prvků na záložce Protokol

Třetí karta v pořadí je pro obsluhu programu pravděpodobně tou nejdůležitější. Nese značení *Display* a nacházejí se na ní zobrazovací prvky, jejichž prostřednictvím jsou uživateli zprostředkována naměřená data. Kartu lze rozdělit na dvě poloviny,. Na jedné z nich se jsou umístěny zmíněné zobrazovače. Pod nimi se nachází nápis, který informuje obsluhu a aktuálním stavu měřícího zařízení. Druhá polovina této záložky je tvořena textovým oknem, do kterého program provádí veškeré informační výpisy.

Obr. 7.3: Rozložení prvků na záložce Protokol

Poslední záložka s názvem *Graf*, je podobně jako již jmenovaná karta Protokol, připravena pouze pro budoucí použití a v současné verzi programu není aktivní.

8 ZÁVĚR

Úkolem mé práce bylo navrhnout a realizovat elektronické zařízení, které bude schopno na několika vstupech změřit analogový signál a odeslat obdržené hodnoty do počítače prostřednictvím BlueTooth standardu. Nikým nebyly stanoveny konkrétní požadavky na počet kanálů ani na bitovou přesnost navrhovaného zařízení. V první fázi návrhu jsem si tedy nejprve určil jaké parametry bych od této aplikace požadoval já osobně a následně byla vybrána konkrétní součástková základna. Hlavním faktorem při volbě použitých elektronických prvků byl především jejich rozsah napájecích napětí. Dále také možnost snadné dostupnosti v podobě zdarma poskytnutých vzorků. Vlastní hardwarový návrh byl v průběhu práce postupně několikrát upravován až do uvedené současné podoby. I přes snahu co nejlépe zvolit použité komponenty se nakonec ukázal použitý externí převodník jako nevhodný. Komunikace s ním vyžaduje jistá specifika, která není řídící procesor schopen splnit. Proto v současné verzi nejsou horní čtyři kanály využívány.

Druhou poluvinou práce bylo vytvoření obslužné aplikace pro osobní počítač. Jako vhodný jazyk jsem si zvolil C# s platformou .NET. K této volbě mě vedlo především to , že se jedná o jazyk jehož syntaxe vychází z jazyka C. Oproti C++ nabízí C# řadu vylepšení a je mu připisována slibná budoucnost. Jelikož nemám s objektovým programováním v podstatě žádné předchozí zkušenosti bylo zapotřebí do toho jazyka alespoň částečně proniknout, pochopit základní principy obsluhy událostí a následně vytvořit požadovanou aplikaci.

Vytvoření grafické podoby rozhraní nebylo nikterak složité. Především zásluhou velmi dobře propracovaného vývojového prostředí Visual studio od firmy Microsoft jde pouze o výběr a umístění jednotlivých grafických prvků.

V průběhu psaní a ladění kódu aplikace jsem bohužel narazil na řadu problémů, jenž celou práci značně zkomplikovaly. Hlavním z nich bylo zpracování asynchronní události od sériové linky. Má aplikace nestíhala na zvolené přenosové rychlosti reagovat na přijímaná data a docházelo tak ke ztrátě některých byte. Jako druhý a rovněž velmi závažný problém se s časem ukázala komunikace se zvolenou BlueTooth klíčenkou. Jelikož nebyly k dispozici přímé ovladače, které by zpracovávaly přijatá data z USB, je nutno využit aplikaci výrobce vybrané klíčenky a k vlastnímu rozhraní přistupovat jako k sériovému portu. To bohužel vede ke skutečnosti, že s některými typy klíčenek si má aplikace rozumí lépe, s některými má naopak problémy.

Řešení zmíněných problémů s objektovým programováním a rovněž problémů na straně hardware si bohužel vyžádalo velké množství času, který bych potřeboval pro dokončení a zajištění spolehlivé komunikace mezi PC a mou aplikací.

9 POUŽITÁ LITERATURA A ZDROJE

[1] PUNČOCHÁŘ, J., Operační zesilovače v elektronice

[2] VÁŇA, V., Mikrokontroléry ATMEL AVR - programování v jazyce C, Nakladatelství BEN – technická literatura, Praha 2003, ISBN 80-7300-102-0.

[3] NAGEL, Ch., EVJEN, B., GLYNN, J., WATSON, K., SKINNER, M., JONES, A., C# 2005 Programujeme profesionálně, Nakladatelství Computer Press, a.s. 2006, ISBN 80-251-1181-4

[4] AVR Libc 1.4.6 Online Users Manual [online]. [2002] [cit. 2007-05-15]. Dostupný z WWW: <http://www.nongnu.org/avr-libc/user-manual/index.html>.

[5] WinAVR : AVR-GCC for Windows [online]. [2005] [cit. 2007-08-20]. Dostupný z WWW: <http://www.winavr.sourceforge.net/index.html>.

[6] Prostředí .NET Framework, Mirosoft. Dostupný z WWW: <http://www.microsoft.com/NET/>.

[7] Katalogový list integrovaného obvodu ATmega48 [online] Dostupný z WWW: http://www.atmel.com/dyn/products/devices.asp

[8] Katalogový list integrovaného obvodu AD8534 [online] Dostupný z WWW: http://www.analog.com/en/audiovideo-products

[9] Katalogový list integrovaného obvodu AD7914[online] Dostupný z WWW: http://www.analog.com/en/audiovideo-products

[10] Katalogový list integrovaného obvodu AD7914[online] Dostupný z WWW: http://www.maxim-ic.com/products.cfm

[11] Katalogový list BlueTooth modulu KC21, viz. Příloha na CD

[12] DOSTÁL, J., USB modul s mikrokontrolérem ATmega, Bakalářská práce ČVUT v Praze, 2007.

[13] KOVAČÍK, P., Robotičtí fotbaloví hráči, Diplomová práce ČVUT v Praze, 2006.

[13] KUNC, T., Ovládání hračky Furby přes BlueTooth, Bakalářská práce ČVUT v Praze 2007 .

10 POUŽITÝ SOFTWARE

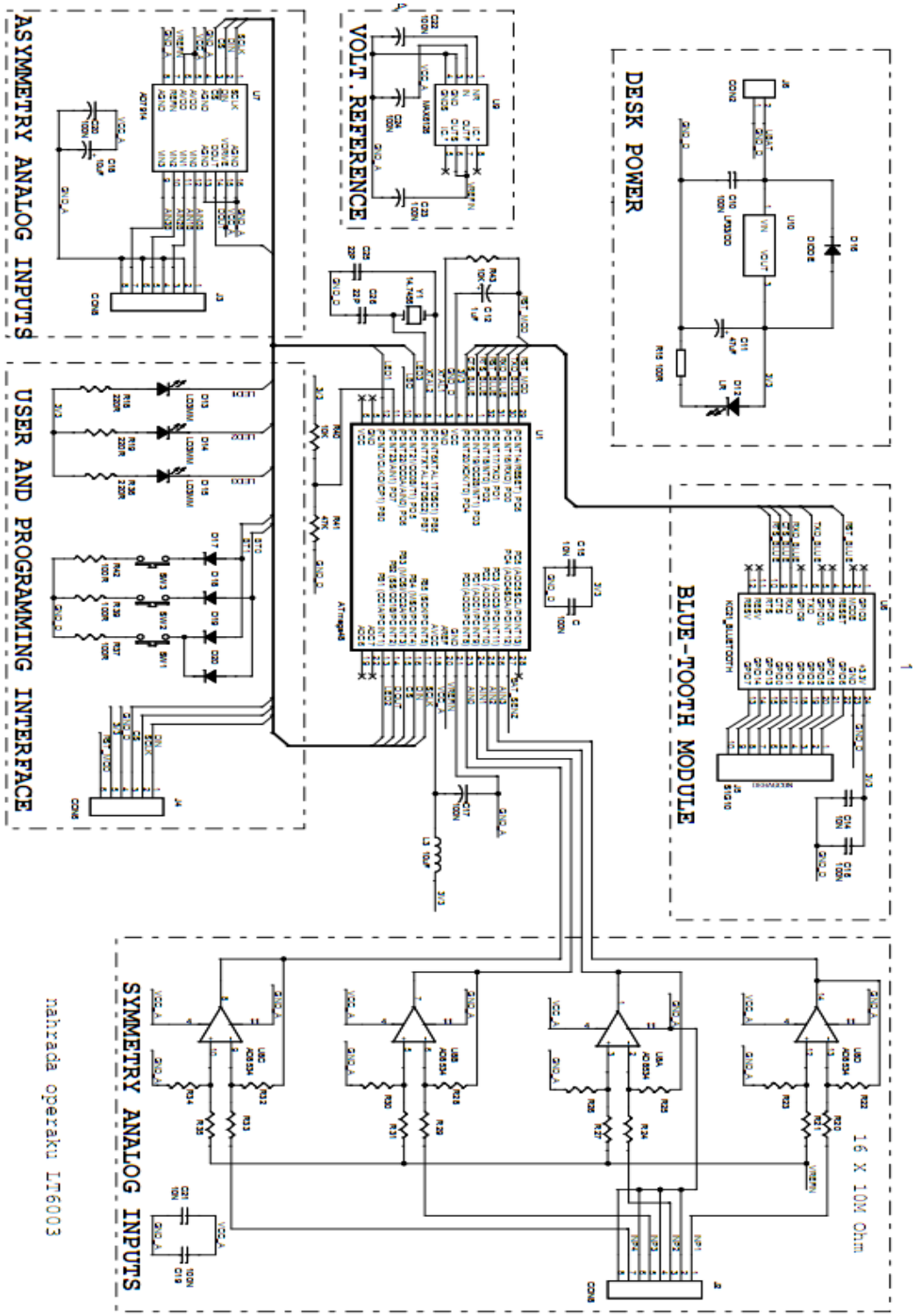
- PSPad editor
- AVR-GCC Compiler
- Microsoft Visual Studio 2008
- ORCAD Schematic Editor
- ORCAD Layout Editor
- ASIX UP
- Hercules.exe

11 OBSAH PŘILOŽENÉHO CD

- elektronická verze této bakalářské práce ve formátu PDF
- schématické zapojení převodníku ve formátu systému ORCAD
- obrazec plošného spoje převodníku ve formátu systému ORCAD
- zdrojový kód pro mikrokontrolér v jazyce C
- komplektní projekt řídící aplikace pro PC se zdrojovým kódem v jazyce C#
- spustitelný soubor ADC_BT.exe
- katalogový list integrovaného obvodu Atmega48
- katalogový list integrovaného obvodu AD7914
- katalogový list integrovaného obvodu AD8534
- katalogový list integrovaného obvodu MAX6126
- katalogový list modulu KC21
- popis příkazů pro KC21

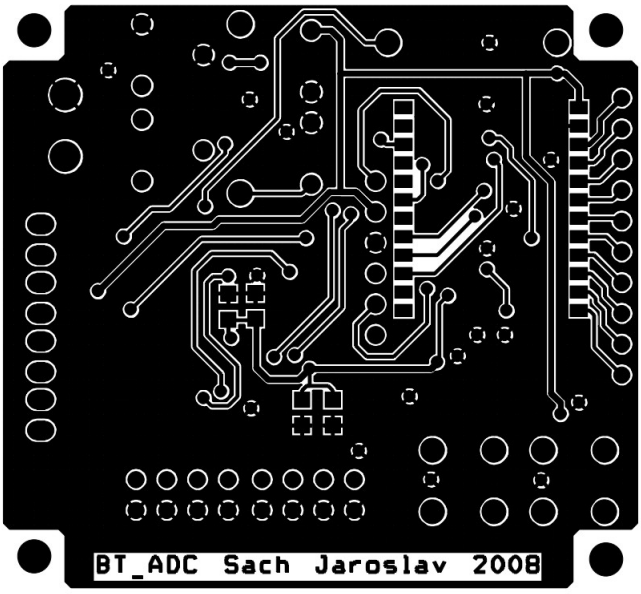
12 PŘÍLOHY

12.1 Schéma zapojení

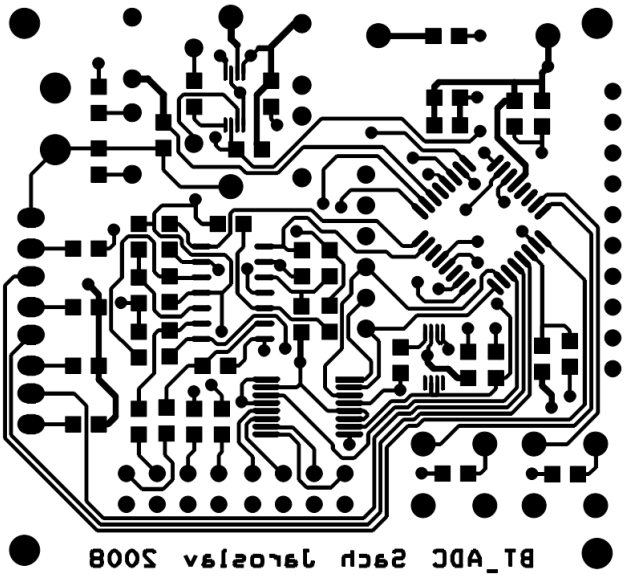


Obr. 12.1: Schéma zapojení převodníku

12.2 Obrazce plošných spojů

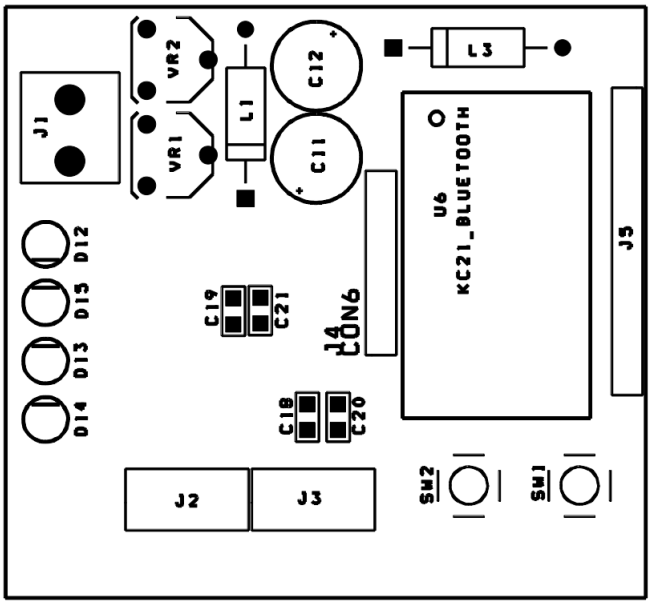


Obr. 12.2: Obrazec plošného spoje vrstvy TOP

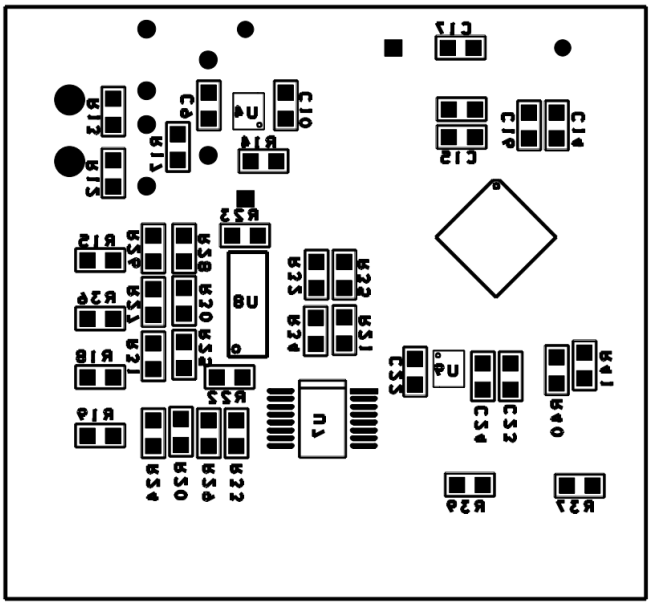


Obr. 12.3: Obrazec plošného spoje vrstvy BOT

12.3 Osazovací plány plošného spoje



Obr. 12.4: Osazovací plán vrstvy TOP



Obr. 12.5: Osazovací plán vrstvy BOT