Zařízení pro měření času a dráhy doběhu

Michal Vopršal

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra řídící techniky
Technická 2
166 27 Praha 6
E-mail: mvoprsal@seznam.cz

25. května, 2007

Anotace

Tato diplomová práce se zabývá návrhem zařízení pro měření času a dráhy doběhu. První část detailně popisuje návrh algoritmu měření a zapojení celého zařízení.

Další části se zabývají konstrukcí dvou zhotovených verzí zařízení a následně vytváření softwarů pro procesor a počítač. Je zde popsáno oživování, testování jednotlivých částí a nalezené chyby. V závěru jsou zmíněny dosažené výsledky.

Zařízení měří velmi přesně čas doběhu. Výsledky měření dráhy doběhu velmi záleží na rychlosti pohybu IRC senzoru, a nejsou tedy ve většině případů uspokojivé. Tento problém je dán nevhodným návrhem plošného spoje.

Největším neúspěchem této práce je nefunkčnost programu obsluhující klávesnici, která velmi omezuje obsluhu celého zařízení a manipulaci s daty.

Závěr pak obsahuje výčet všech dosažených výsledků a návrhy řešení na odstranění chyb pro realizaci další verze.

Vedoucí práce: Ing. Martin Hlinovský

Annotation

These diploma thesis presents the design of an equipment for stop time and stop distance measurement. The first section describes an algorithm-design for measuring and wiring of the whole equipment.

The next sections describe the construction of two bulit-up versions and subsequently the creation of firmware for the processor and software for computer. Debugging, testing of particular parts and ascertained errors are described there as well.

The stop time is measured very exactly by this device. The results of stop distance measuring depends on velocity of the IRC sensor and hence they are mostly not satisfactory. This problem is caused by an unsuitable design of the printed circuit.

The greatest failure of this thesis is malfunction of the program controling the keyboard which limits servicing of the device and data handling.

The conclusion contains list of all reached results and solutions for elimination of the mistakes in next versions.

Obsah:

1. Úvod - 4 -
1.1 Požadavky na zařízení - 5 -
1.2 Inkrementální senzor polohy - 6 -

2. Návrh elektrického schématu - 7 -
2.1 Blokové schéma celého zařízení - 7 -
2.2 Algoritmus měření - 8 -
2.2.1 Chyba měření - 9 -
2.3 Výběr součástek - 9 -
2.4 Procesor - 10 -
2.5 Paměť SRAM - 12 -
2.5.1 Obvod pro zálohování paměti - 15 -
2.6 Paměť EEPROM - 15 -
2.7 Obvod reálného času - 16 -
2.8 Maticová klávesnice - 17 -
2.9 LCD display - 19 -
2.10 Komunikační rozhraní USB - 20 -
2.11 Komunikační rozhraní RS232 - 22 -
2.12 Napájení zařízení - 23 -
2.12.1 Nabíjecí akumulátory - 23 -
2.12.2 Externí napájení - 23 -
2.13 Nabíječka akumulátorů - 24 -
2.14 Stabilizátor napětí - 27 -
2.14.1 Spotřeba zařízení - 28 -
2.15 Napěťový watchdog - 28 -
2.16 Vyvedené konektory - 29 -
2.17 Obvody pro podporu IRC senzoru - 30 -
2.17.1 Obvod pro bezpotenciálové spínání měření - 30 -
2.17.2 Obvod pro napájení IRC senzoru - 31 -
2.17.3 Vzorkovací obvod - 31 -
2.17.4 Konektor senzoru - 32 -

3. Plošný spoj - 34 -
3.1 První verze - 34 -
3.2 Druhá verze - 38 -

4. Montáž - 42 -

5. Firmware pro procesor - 45 -
5.1 Inicializace aplikace - 45 -
5.2 Watchdog - 46 -
5.3 Obsluha paměti SRAM - 46 -
5.4 Procedura měření - 49 -
5.5 Přenos dat - 51 -
5.6 Ovládání LCD displeje a UART - 53 -
5.7 Obsluha klávesnice - 55 -
5.8 Programování procesoru - 57 -

6. Software pro zpracování dat v počítači - 59 -
6.1 Program pro zpracování dat - 59 -
6.2 Programátor konfiguračních EEPROM sběrnice USB - 60 -
6.3 Ovladač pro obvod FTDI - 62 -

7. Měření - 64 -

8. Závěr - 66 -

9. Seznam příloh - 69 -

10. Vybrané přílohy - 71 -

1. Úvod

Téma mé diplomové práce je konstrukce zařízení pro měření doběhu lisu. Pro lepší představu, toto zařízení měří čas a dráhu doběhu lisu po jeho vypnutí. Tedy jakou dráhu a za jakou dobu ji lis urazí díky své setrvačnosti.

Na začátku je nutné zmínit důvod, proč jsem toto zařízení vyvíjel. Hlavním důvodem je potřeba měření doběhového času lisu, který je důležitý při určování bezpečné vzdálenosti **S** pracovníka od stroje. Tato vzdálenost se vypočítá podle ČSN EN 999 z následujícího vzorce:

$$S = (K \times T) + C \qquad (1),$$

kde **K** je parametr rychlosti přiblížení člověka nebo částí jeho těla ke stroji, **T** je čas doběhu stroje a **C** je přídavná konstanta, která se volí podle použitého bezpečnostního senzoru.

Podobná sériově vyráběná zařízení jsou velmi drahá. Navíc toto zařízení lze použít k měření kdekoli v průmyslu, kde jsou využívány lisy. Jedním z hlavních cílů tedy byla minimalizace ceny zařízení.

Zařízení využívá pro měření dráhy inkrementální lankový snímač polohy od firmy SICK. Toto zařízení by mělo být schopno změřit dráhu a čas doběhu, uložit změřená data z několika měření a ty následně přesunout do počítače, kde bude možno z těchto dat vytvořit protokol o provedených měření.

Obrázek 1 zobrazuje celé zařízení sestavené dohromady.

Obr.1: Sestavené celé zařízení

Následující text uvádí průběh celého vývoje a veškerých prácí na tomto zařízení. A to od prvního zadání a specifikace požadavků, výběru součástek, samotného návrhu zapojení jednotlivých částí a celého zařízení. Dále vývoj plošného spoje a jeho osazení. A také následně vytváření firmwaru pro toto zařízení.

1.1 Požadavky na zařízení

Prvním krokem při vývoji tohoto zařízení je specifikace, toho co toto zařízení bude muset umět. Jaké požadavky musí být splněny. Tato část popisuje, které komponenty budou muset být součástí zařízení, aby toto zařízení splňovalo požadavky na měření, manipulaci s daty a pohodlnou obsluhu celého zařízení.

Požadavky a jejich řešení:

1. Rychlost měření:
   Zařízení musí být schopno přijímat údaje z IRC senzoru a zpracovat je v reálném čase. Což v tomto případě znamená přijímání dat s vzorkovací frekvencí asi 1kHz.
   Pro tyto účely by měl vystačit nějaký 8-bitový mikrokontrolér s dostatečným počtem periférií.

2. Ukládání dat:
   Zařízení musí zvládnout ukládat data během měření, ale i uchovat je dokud nebudou smazána, nebo přenesena do počítače. Požadavkem bylo, aby se dala uchovat nejen data z posledního měření, ale i z několika předchozích měření.
   K tomu vedl požadavek na několik měření na jednom stroji bez nutnosti přenášet každé měření samostatně do počítače a z toho plynoucí odpojování zařízení.
   Tento úkol jsem vyřešil použitím dvou druhů pamětí. Primární paměť je typu SRAM z důvodu potřeby ukládání dat v reálném čase během měření a jejich uchování až do vypnutí zařízení. Druhý typ použité paměti je EEPROM. Tato paměť slouží k uchování dat na delší dobu i při vypnutém zařízení a v situacích, kdy nemůžeme data přenést do počítače hned.

3. Přenos dat:
   Dalším požadavkem byla specifikace komunikace s počítačem. Celou konstrukci jsem vybavil dvěma komunikačními rozhraními. Základní myšlenka spočívala hlavně v jednoduchosti připojení, které by nevyžadovalo žádný další speciální hardware. Jako primární komunikaci jsem tedy zvolil USB rozhraní, kterým je dnes vybaven každý počítač a jeho zapojení není složité. Druhé rozhraní je sériová linka RS 232.

4. Ovládání:
   Provoz a ovládání takovéhoto zařízení by mělo být minimálně náročné na údržbu a uživatelsky maximálně příjemné. Celé ovládání by mělo probíhat přes klávesnici umístěnou na přední straně zařízení. Informace pro uživatele jsou zobrazovány pomocí LCD displeje umístěného nad klávesnicí.

5. Baterie a jejich nabíjení
   Zařízení také musí obsahovat baterie, aby ho bylo možné použít kdekoli bez ohledu na externí napájení. To je vyřešeno použitím nabíjecích akumulátorů, které by měly být schopny vydržet i několik hodin provozu. A to i s napájením IRC senzoru. Nabíjení těchto akumulátorů byl další požadavek. Byla možnost vytvořit externí nabíječku, nebo ji integrovat jako součást zařízení. Jelikož jsem objevil velmi jednoduché a praktické zapojení, tak je i nabíječka součástí zařízení.

6. Velikost zařízení:
   Jedním z nejdůležitějších požadavků bylo umístění celého zařízení do předem dané krabičky. Tou byla krabička firmy BOPLA, typ BOPLA ARTEB 655 viz příloha [1]. Rozměr plošného spoje tedy musí odpovídat vnitřnímu rozměru krabičky a je uveden v příloze [36].

1.2 Inkrementální senzor polohy

K měření doběhu je použit inkrementální lankový senzor polohy firmy SICK. Senzor je typu PRF 08, což je velmi přesný senzor s rozlišením 0,025 mm a délkou měření až 2 m. Hlavní důvod pro použití tohoto senzoru je jeho TTL rozhraní, které využívá i navržené měřící zařízení. Obrázek 2 vyobrazuje senzor a technická specifikace tohoto senzoru je uvedena v příloze [2].

Obr. 2: Encodér PRF 08

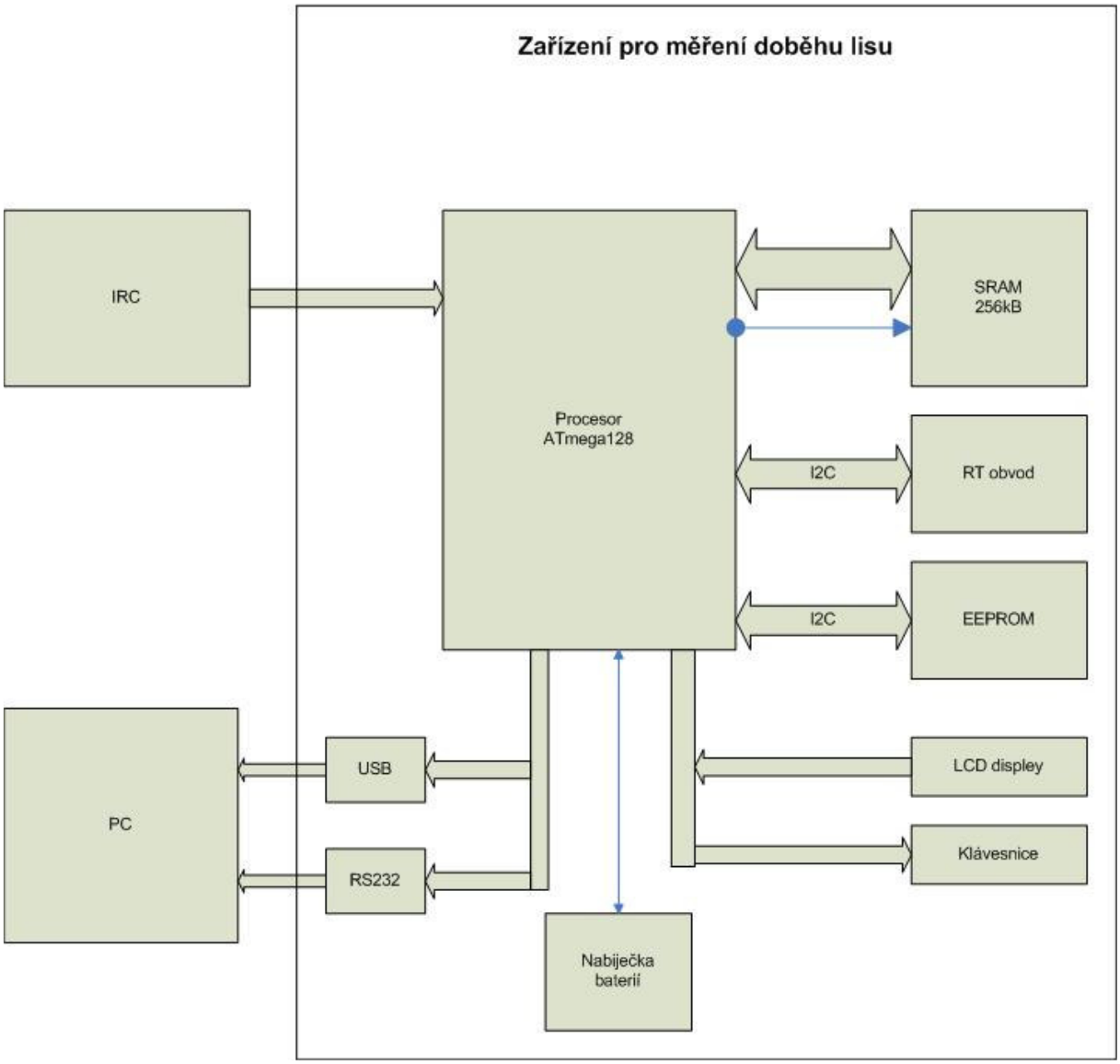
2. Návrh elektrického schématu

V této sekci postupně vysvětlím jednotlivé části celého zařízení. Bude vysvětleno:

1. jak jednotlivé části pracují.
2. proč byl návrh proveden právě takto.
3. proč byly použity dané součástky.
4. jak jednotlivé součásti spolupracují.

Blokové schéma zapojení je na obrázku 3.

2.1 Blokové schéma celého zařízení



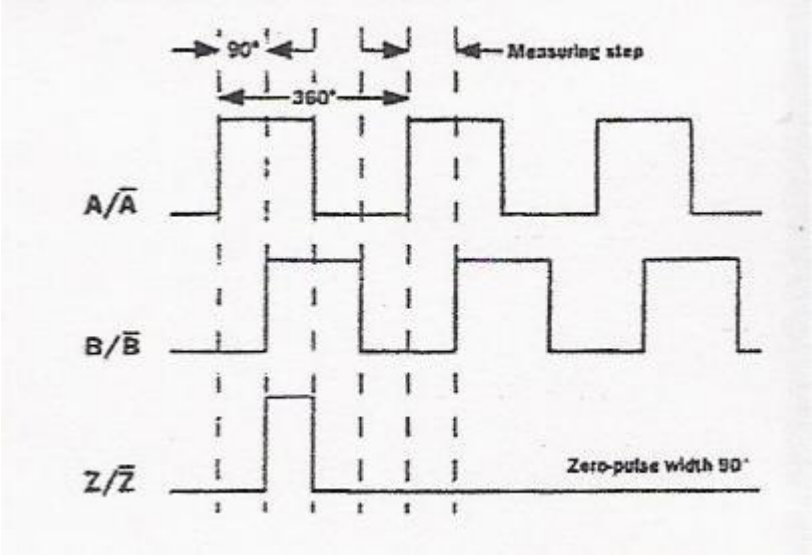
Obr. 3: Blokové schéma

Z tohoto zapojení je vidět, že základem celého přístroje je procesor. Tento procesor zajišťuje veškeré funkce zařízení. Procesor komunikuje přes datovou sběrnici s pamětí SRAM, která slouží pro uchovávání měřeních dat. Dále je zde

paměť EEPROM pro dlouhodobé ukládání dat a obvod reálného času pro uložení času a data měření. Obě tyto periferie komunikují s procesorem přes sběrnici I2C. Pro usnadnění ovládání celého přístroje je zde LCD displej o velikosti čtyř řádků, každý z nich šestnácti znacích. A maticová klávesnice s dvaceti jedna tlačítky. Komunikace s počítačem zajišťují USB a RS 232 rozhraní. Tyto rozhraní využívají dvou sériových komunikační rozhraní procesoru. Poslední částí je nabíječka akumulátorů, která je nezávislá na procesoru. Baterie mohou být nabíjeny jak při zapnutém tak i při vypnutém zařízení.

2.2 Algoritmus měření

Po sestavení struktury celého zařízení bylo nutné vytvořit algoritmus samotného měření. Pro další návrh jednotlivých částí bylo nutné promyslet, jak připojit IRC senzor k procesoru, a především jak zpracovávat změřené údaje. Objem dat musí být dostatečně malý, aby se do paměti vešlo více měření najednou. A jejich zpracování nesmí být výpočetně příliš náročné, aby se data dala ukládat v reálném čase. Průběh výstupních signálů IRC senzoru je na obrázku 4.



Obr. 4: Výstup IRC senzoru

Po analýze těchto signálů jsem navrhl algoritmus, který je velmi výkonný, protože používá malý počet matematických operací ke zpracování měřených dat a velikost ukládaných dat je minimální.

Základem algoritmu je detekce všech náběžných i spádových hran signálů A a B, která vyvolají přerušení v procesoru. Detekce všech hran výstupních signálů umožňuje změřit každý krok měření (Measuring step). Ten indikuje natočení senzoru o 90 stupňů, které reprezentují vzdálenost 0,025 mm. Z toho plyne, že čas mezi dvěmi přerušeními reprezentuje posunutí senzoru o jeden měřící krok. Jelikož vzdálenost mezi jednotlivými přerušeními je tedy stále stejná, není zapotřebí ji ukládat. Jediné co tedy musíme ukládat, je čas jednotlivých měřících kroků. Za předpokladu, že přesnost měření bude dostačující v milisekundách a měření nebude delší než několik sekund, pak pro uložení jednoho časového údaje do paměti budou stačit 2 byty paměti. Jediným požadavkem na procesor tedy je, aby měl

minimálně dva vstupy pro externí přerušení schopné detekovat přerušení jak při náběžných tak i sestupných hranách.

Měření ještě vyžaduje jedno přerušení indikující začátek měření. V tomto okamžiku začne měření času. Toto přerušení ale vyžaduje pouze detekci náběžné hrany.

Ukončení měření je poslední částí algoritmu. Měření se ukončí pokud po nějakou nastavenou dobu nepřijde žádný další měřící signál, tedy po tuto dobu nenastane žádné posunutí senzoru.

Signál Z není pro měření využit.

Nevýhodou tohoto algoritmu je, že nemusí dojít k detekci změny směru pohybu stroje při zákmitech .

2.2.1 Chyba měření

Součástí návrhu algoritmu měření bylo také stanovit maximální chybu měření. Jelikož měření vzdálenosti spočívá pouze v přičítání jednotlivých kroků měření při každém přerušení procesoru, pak chyba určení vzdálenosti může vzniknout pouze na začátku a na konci měření. Tato chyba je dána přičtením jednoho celého kroku měření, i když počáteční a koncová poloha mu přesně nemusí odpovídat. **Maximální celková chyba měření vzdálenosti je tedy 50µm**. Měření vzdálenosti je velmi závislé na detekci všech náběžných i spádových hran. Pokud by některé hrany měřeného signálu nebyly detekovány, pak chyba měření může velmi narůst a změřená hodnota nemusí odpovídat skutečnosti.

Naopak měření celkového času je na tomto faktu nezávislé, protože je měřen vždy čas mezi dvěmi přerušeními a ten je pak přičítán. Je tedy jedno kolik přerušení nastane. Přesnost tedy bude záležet pouze na použitém výpočtu a předdělícím poměru čítače procesoru. **Maximální chyba měření v tomto případě by neměla přesáhnout 1ms**. Tato chyba bude způsobena zaokrouhlování času na milisekundy.

2.3 Výběr součástek

Důvody výběru všech hlavních součástek jsou následující:

Nejdůležitější otázkou byl výběr vhodného procesoru. První myšlenky směřovaly k použití nějakého procesoru firmy Microchip. Ale tyto procesory nedisponují dostatečnou programovou pamětí a hlavně pro ně zatím neexistuje dobrý překladač jazyka C.

Především z důvodu existence velmi dobrých vývojových prostředí jsem se na konec rozhodl pro použití mikrokontroléru typu AVR firmy Atmel. Pro tuto aplikaci se mě jako velmi vhodný jevil jeden z nejvýkonnějších procesorů řady Atmega. Procesor **Atmega128** je 8-bitový mikrokontrolér s architekturou RISC. Procesor je dostatečně malý a obsahuje potřebné množství periférií pro řešení zadaného problému. A především obsahuje čtyři vstupy využitelné jako zdroje externích přerušení schopné detekovat náběžnou i sestupnou hranu signálu.

Výběr paměti pro ukládání naměřených dat se skládal ze dvou částí.

V první části šlo o zvolení vhodného typu paměti. Aby procesor stíhal ukládat data v reálném čase a nezabralo mu to celý výpočetní čas, bylo zapotřebí využít paměti typu RAM, která je dostatečně rychlá. Na výběr jsem měl buď statickou nebo dynamickou paměť. Zvolil jsem statickou RAM, jelikož dynamická RAM vyžaduje obnovování dat, což v tomto případě, s použitím běžné logiky, je skoro nemožné.

V druhé části šlo o volbu velikosti paměti. Za výše zmíněných předpokladů (počet vzorků z jednoho měření asi 2000, počet měření větší než 5, velikost jednoho vzorku v paměti 1W = 2B) můžeme spočítat požadovanou velikost:

$$C_{RAM} = p.n.B = 2000.5.2 = 20kB \qquad (2).$$

Protože požadovaná velikost paměti mi vyšla 20 kB, musíme použít paměť větší. V nabídce výrobců je nejbližší větší paměť s kapacitou 32 kB, což samozřejmě vyhovuje požadavku. Typ paměti jsem tedy zvolil SRAM **HM62256.**

Paměť EEPROM byla volena stejným způsobem. Využil jsem paměti o kapacitě také 32 kB od firmy Atmel **AT24C256**. Hlavním důvodem byla velikost celého obvodu a především možnost komunikace s touto pamětí přes rozhraní I2C, které obsahuje i procesor.

Výběr obvodu reálného měl úplně stejný důvod. Vybral jsem obvod firmy Philips **PCF8583**, který komunikuje s procesorem také po rozhraní I2C, pouze s jinou adresou.

Navíc pro rozhraní I2C i pro tento obvod jsou již napsané zdrojové kódy v jazyce C, takže jejich začlenění do návrhu a pak i do programu by mělo být poměrně jednoduché.

Jako primární komunikaci s PC byla zvolena sběrnice USB. Převod ze sériové komunikace s procesorem na rozhraní USB zajišťuje obvod **FT232MB** firmy FTDI. Toto zapojení jsem vybral pro jeho jednoduchost a minimální velikost obvodu. USB zapojení jsem ještě doplnil o EEPROM **93LC46B.** Obvod FT232MB totiž umožňuje nahrání deskriptoru zařízení do této paměti. To usnadňuje identifikaci zařízení na sběrnici USB a umožňuje připojení více zařízení najednou.

Druhou možností komunikace s počítačem je využití sériové linky pomocí RS 232. Převod na tuto komunikaci obstarává obvod **MAX232**.

Bylo také důležité bylo najít vhodný obvod pro nabíječku akumulátorů. Jako velmi vhodný se jevil obvod firmy MAXIM **MAX713.** Zapojení s tímto obvodem vyžaduje jen minimum externích součástek a umožňuje nastavení počtu nabíjených článků a doby nabíjení.

Výběr většiny dalších součástek již nebyl kritický a bude popsán v následujících kapitolách.

2.4 Procesor

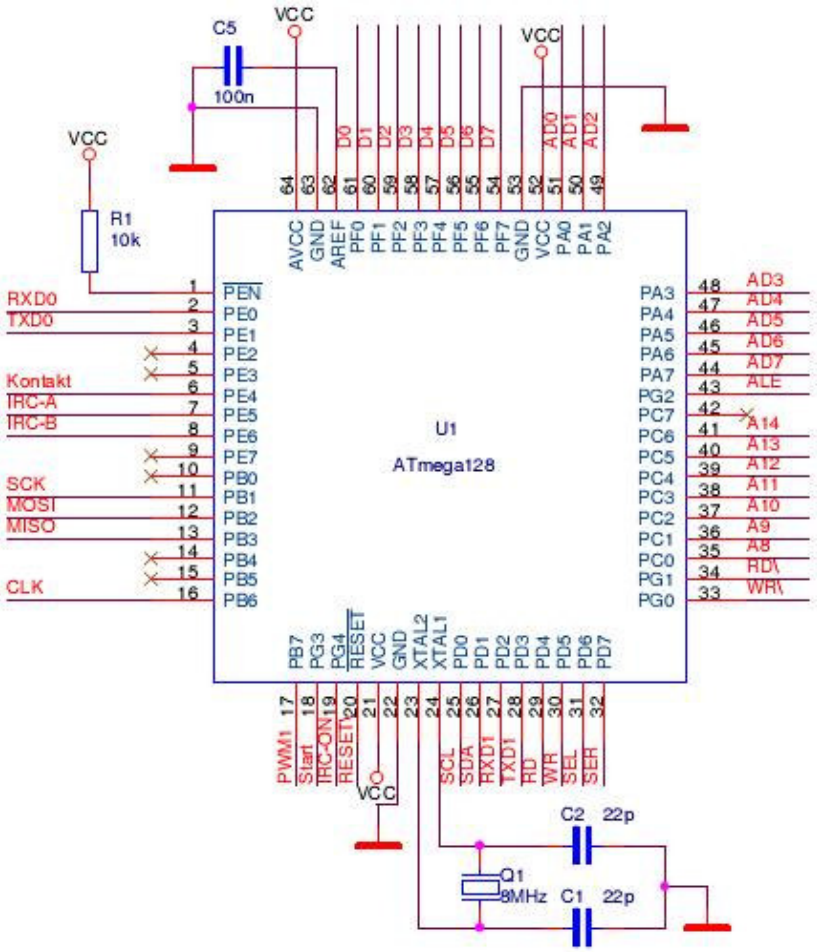
V této kapitole je podrobně popsáno zapojení celého procesoru **U1**, zapojení jednotlivých portů a periferií. Podrobná specifikace procesoru je uvedena v příloze [3]. Zapojení je znázorněno na obrázku 5.

<u>Zapojení jednotlivých portů:</u>

PORT A je celý využit pro komunikaci s pamětí RAM. Port je využíván obousměrně. Jednak pro generování dolní části adresy v paměti, jednak pro odesílání a přijímání dat z paměti.

PORT B má vícero využití. Pin **PB0** je nezapojen. Pin **PB1** (SCK) slouží jako hodinový signál pro rozhraní SPI. Piny **PB2** (MOSI), **PB3** (MISO) jsou pak využity jako datová komunikace pro toto rozhraní. Další dva piny jsou opět nezapojeny. Na výstup pinu **PB6** (CLK) je možno přivést hodinový signál, a tak slouží jako generátor

vzorkovací frekvence pro měřící obvod **U6**. Na výstup pinu **PB7** (PWM1) je možno přivést jeden z generátorů PWM a je využit ke generování řídícího signálu regulující intenzitu podsvícení LCD displeje.



Obr. 5: Zapojení procesoru

PORT C je opět celý využit pro generování adresy paměti SRAM. Zde je generována horní část adresy. Pin **PC7** je nezapojen, protože adresování 32 kB paměti vyžaduje pouze patnáct bitů.

PORT D je využit několika způsoby. Pin **PD0** (SCL) je využit jako zdroj hodinového signálu pro sběrnici I2C. Pin **PD1** (SDA) je pak využit pro datovou komunikaci po této sběrnici. Piny **PD2** (RXD1) a **PD3** (TXD1) jsou zapojeny jako sériová komunikace pro rozhraní RS 232 a využívají UART1 procesoru viz příloha [3]. Piny **PD4** (RD), **PD5** (WR), **PD6** (SEL), **PD7** (SER) jsou použity jako řídící signály pro ovládání klávesnice a LCD displeje. Viz kapitola 2.9.

PORT E má dvě využití. Piny **PE0** (RXD0), **PE1** (TXD0) slouží jako komunikace pro USB rozhraní a využívají UART0 procesoru viz [3]. Pin **PE4** (Kontakt) slouží k externímu přerušení a indikuje začátek měření, tedy vypnutí chodu lisu. Piny **PE5** (IRC-A) a **PE6** (IRC-B) jsou také využity jako zdroje externích přerušení a slouží k samotnému měření. Tyto vstupy pro měření jsem vybral, protože umožňují detekovat přerušení při náběžné i spádové hraně vstupního signálu. Více o

přerušení bude uvedeno v sekci o firmwaru**.** Ostatní piny tohoto portu jsou nezapojeny.

Další port je **PORT F.** Ten slouží buď jako 8-bitový vstup z tlačítek klávesnice, nebo jako výstup dat pro LCD displej. Jelikož ten je ale zapojen ve 4-bitovém módu, slouží k tomuto účelu pouze první čtyři piny **PF0-PF3.** Piny **PF4-PF7** jsou dále využity pro **JTAG** (Joint Test Action Group). Ten může být použit k diagnostice zapojení.

Poslední je **PORT G,** který je 5-bitový. Piny **PG0** (WR\), **PG1** (RD\), **PG2** (ALE) pak generuje řídící signál pro SRAM a s ní spojenou LATCH. Pin **PG3** (Start) spíná zdroj napájení celého zařízení a musí být tedy po celou dobu provozu zařízení ve stavu log 1, až do jeho vypnutí. **PG4** (IRC-ON) pak souží pro zapnutí napájení pro IRC senzor.

Napájení procesoru:

Procesor má dvě vstupní napájení. Piny 21 a 52 jsou připojeny na napájecí napětí. Piny 22, 53 a 63 pak na zem.

Zdroj hodinového signálu:

Jako zdroj hodinového signálu jsem použil externí krystalový oscilátor o frekvenci 8MHz. Ten je připojen na piny 23 (XTAL1), 24 (XTAL2). Stabilitu oscilátoru zaručují kondenzátory **C1**,**C2** o velikosti 22 pF. Tato hodnota je doporučena výrobcem procesoru pro tuto pracovní frekvenci.

Externí reset:

Externí reset procesoru je přiveden na pin 20 (RESET\). Ten je aktivován přivedením log 0 a je využíván buď při programování přes SPI rozhraní nebo napěťovým watchdogem **U16**.

Ostatní piny:

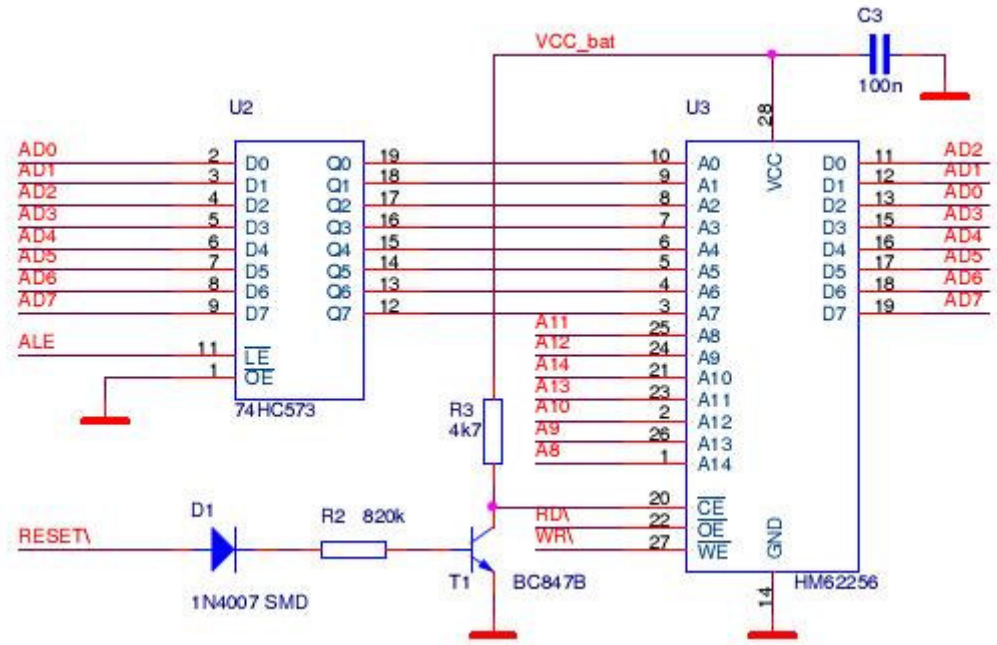
Pin 64 (AVCC) je zdroj napětí pro A/D převodník na portu F. Jelikož převodník v této aplikaci není využit, tak výrobce doporučuje připojit tento vstup na napájecí napětí.

Pin 62 (AREF) je analogová reference pro A/D převodník. Jelikož převodník v této aplikaci není využit, tak tento vstup musí být zablokován přes kondenzátor **C5** o velikosti 100 nF.

2.5 Paměť SRAM

Tato část se zabývá připojením statické RAM **U3** a jejích podpůrných obvodů k datové sběrnici procesoru. Paměť jsem volil podle několika kriterií. Prvním kritériem je dostatečná rychlost přístupu k datům, druhým minimální spotřeba, třetím TTL rozhraní a posledním kritériem je bateriové zálohování paměti. V konstrukci jsem použil paměť **HM62256**, která splňuje všechny požadavky (doba přístupu: 100 ns,

spotřeba při provozu: 40 mW) a je i cenově dostupná. Pro bližší informace odkazuji na přílohu [4]. Zapojení je znázorněno na obrázku 6.



Obr. 6: Zapojení paměti SRAM

Paměť vyžaduje připojení na 8-bitovou datovou sběrnici a zároveň na 15-bitovou adresní sběrnici. Tolik bitů ovšem není k dispozici. Jak bylo popsáno v kapitole o procesoru máme k dispozici všech patnáct bitů adresní sběrnice, ale už nezbývá místo pro data. Tento problém je nutno vyřešit jinak. Port A procesoru je využit pro generování adresy i pro přenos dat. Princip spočívá v tom, že na portu A se vygeneruje dolní část adresy. Ta se pomocí paměti *LATCH* **U2** uchová dále. Na portu C je vygenerována horní část adresy. V tomto okamžiku máme na adresní vstupy paměti přivedenou adresu a přitom jsme si uvolnili port A, jelikož jeho část adresy je nyní udržována na výstupech paměti *LATCH*. V tomto okamžiku mohou být data buď uložena na danou adresu do paměti, nebo z této adresy přečtena do procesoru. Čtení a zápis jsou ovládány řídícími signály paměti RD\ a WR\ viz [4].

Jako paměť *LATCH* jsem vybral obvod TTL **74HC573,** příloha [5]. Typ obvodu HC jsem použil z důvodu nároků na rychlost paměti. Tento obvod je ovládán pomocí signálu ALE (adress latch enable). Pokud je tento signál ve stavu log 1, je adresa na vstupu převedena na výstup. Po změně řídícího signálu ALE na log 0 je pak tato adresa držena . Tím je problém s ukládáním dat a k přístupu k nim vyřešen.

Dalším problémem bylo, jak vyřešit konzistenci dat a jejich náhodné přepisování při poklesu napájecího napětí. Při poklesu napájecího napětí pod nastavenou hodnotu je vstup RESET\ přiveden na log 0. Tím se tranzistor **T1** stane nevodivým a přes rezistor **R3** se na vstupu CE\ objeví kladné napětí, které způsobí, že paměť je v režimu *NOT SELECTED* a změna dat není možná. Otázkou zůstává jak toto vyřešit v normálním stavu. Na signálu RESET\ je kladné napětí, protože *watchdog* **U16** je ve stavu vysoké impedance. Tím pádem je **T1** sepnut a na vstupu CE\ je log 0, potřebná pro provoz paměti. Je zapotřebí tedy vypočítat odpor kolektoru **R3** a odpor báze **R2**. Základní předpoklad byl ten, že odpor **R3** musí být dostatečně velký, aby při uzavřeném tranzistoru **T1** nedošlo k poškození řídícího vstupu paměti a

při tom v sepnutém stavu neprotékal příliš veliký proud, který by zatěžoval celé zařízení.

Maximální proud v sepnutém stavu jsem si zvolil $I_{C\max} = 1mA$. Z toho vychází následující výpočet:

Kolektorový odpor **R3**:

$$Rc = \frac{U_{CC} - U_{CEsat}}{I_C} = \frac{5-0,6}{0,001} = 4400\Omega \qquad (3).$$

V zapojení jsem použil odpor R3=4,7 kΩ, jelikož jde o nejbližší větší možnou hodnotu. Ta způsobí, že kolektorový proud bude ještě o něco menší, což je v tomto případě výhodou. Navíc tato velikost již splňuje i požadavek na ochranu řídícího vstupu paměti.

Odpor báze **R2**:

Z technických údajů o tranzistoru jsem převzal hodnotu $h_{fe} = 200$ viz příloha [6], což je typické zesílení tohoto tranzistoru. Z toho vypočítáme proud báze:

$$I_C = h_{fe}.I_B \Rightarrow I_B = \frac{I_C}{h_{fe}} = \frac{0,001}{200} = 5\mu A \qquad (4).$$

Nyní již můžeme vypočítat odpor **R2.** Při výpočtu nesmíme zapomenout, že v cestě je již zapojen odpor **R4**, který musíme od výsledku odečíst:

$$R_B = \frac{U_{CC} - U_B}{I_B} - R_4 = \frac{5-0,7}{0,000005} - 10000 = 82000\Omega \qquad (5).$$

Odpor 820 kΩ je standardně vyráběná hodnota a nic nebrání v jeho použití.

<u>Poznámka</u>:

$U_{CC} = 5V$ – Napájecí napětí

$U_{CESAT} = 0,6V$ - Saturační napětí tranzistoru

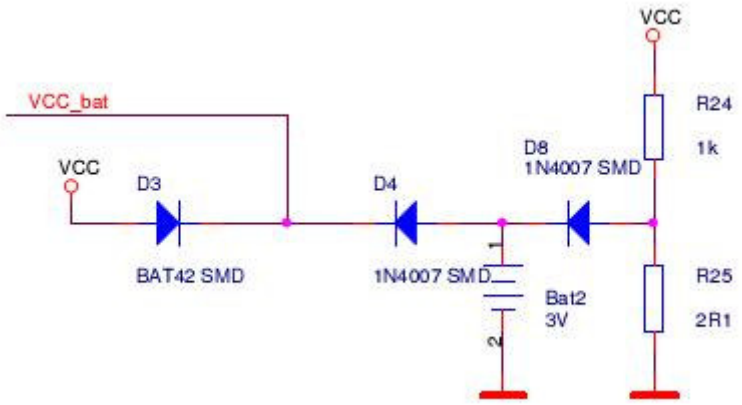
$U_D = 0,7V$ - Prahové napětí diody

Dioda **D1** je v zapojení použita z důvodu bezpečného rozepínání tranzistoru. Pro tento účel bez problému postačí univerzální dioda **1N4007**.

Zapojení je ještě doplněno o filtraci napájecího napětí, tu zajišťuje kondenzátor **C3**. Napájení paměti není připojeno přímo na napájení, ale je přivedeno přes záložní obvod, který napájí paměť při poklesu napájení pod povolenou hranici. Celé zálohování je popsáno v kapitole 2.5.1.

2.5.1 Obvod pro zálohování paměti

Zálohování paměti je velmi jednoduché. Je zajištěno záložní baterií **BAT2** o napětí 3V a kapacitě 280 mAh**.** Ta zároveň zálohuje i činnost obvodu reálného času. Zapojení celého obvodu je na obrázku 7.



Obr. 7: Obvod pro zálohování paměti

Princip činnosti zapojení je velmi jednoduchý. Výstup z tohoto obvodu je napájecí napětí pro paměť a obvod reálného času **VCC_bat.** Pokud je přístroj zapnutý a akumulátory nabité, je na napájecím napětí VCC stabilizované napětí 5V. Jelikož napětí na baterii je pouze 3V, dioda **D3** je polarizována v propustné směru, ale dioda **D4** v závěrném. Tím pádem přes **D3** prochází proud a napájí paměť normálně z hlavního zdroje, zatímco zavřená dioda **D4** brání vybíjení baterie **BAT2**. V opačném případě, kdy je primární napájení odpojeno je **D3** uzavřena, dioda **D4** se otevírá a napájení paměti je zajištěno.

Úbytek napětí na diodách **D4** a **D8** není důležitý, protože obvod reálného času dokáže pracovat při napětí nad 1V a tím pádem nedojde ke ztrátě informací o aktuálním čase a datu. Proto postačí libovolné univerzální diody např. **1N4007**. Větší problém nastává s výběrem diody **D3.** Pro správnou funkci paměti je zapotřebí napájecí napětí větší než 4,5V. Jelikož výše zmíněné křemíkové diody mají úbytek napětí větší než 0,6V, nelze je v tomto případě použít. Jako vhodná se jevila Schotkyho dioda **BAT42**, která má úbytek pouze asi 0,33V a tím pádem zaručí dostatečné napájecí napětí paměti.

Zbývající část obvodu slouží k dobíjení záložní baterie. Odpory **R24** a **R25** slouží jako napěťový dělič. A dioda D8 zabraňuje vybíjení baterie při vypnutém napájení VCC. Hodnoty rezistorů jsou doporučené firmou GM Elektronic pro toto zapojení viz příloha [7].

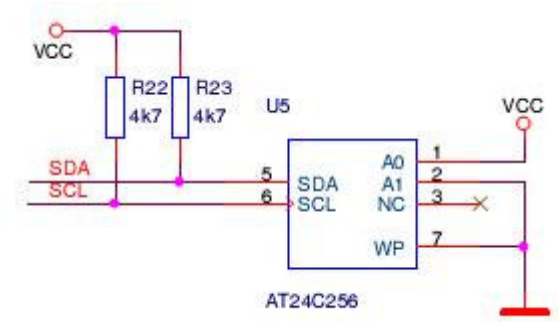
2.6 Paměť EEPROM

Tento druh paměti jsem použil pro případ, kdy bude zapotřebí uchovat data v paměti přístroje na delší dobu. Vybral jsem paměť od firmy Atmel **AT24C256**. Paměť je z jasného důvodu stejné kapacity jako primární SRAM. Hlavním důvodem výběru byla, jak jsem již zmínil výše, možnost připojení paměti k procesoru přes dvouvodičovou sběrnici I2C firmy Philips viz příloha [8].

Při napájení 5V lze dosáhnout komunikační frekvence až 1 MHz. Adresace paměti na sběrnici je provedena pomocí pinů A0 a A1. To dovoluje použití až čtyř pamětí na jedné sběrnici. Jelikož je použita jen jedna, byla volba adresy pouze na

mě. Adresu jsem zvolil jako A0 = 1 a A1 = 0. Adresu A0 = 0, která by byla asi nejlogičtější nebylo možno použít, protože tuto adresu již používá obvod reálného času, jak vyplyne z další části.

Schéma celého zapojení paměti EEPROM je na obrázku 8.



Obr. 8: Paměť EEPROM

Vstup SCL (Serial Clock) je použit jako hodinový vstup s náběžnou hranou pro vstupní data a se sestupnou hranou pro výstupní data.

Pin SDA (Serial Data) je obousměrný pro sériová data.

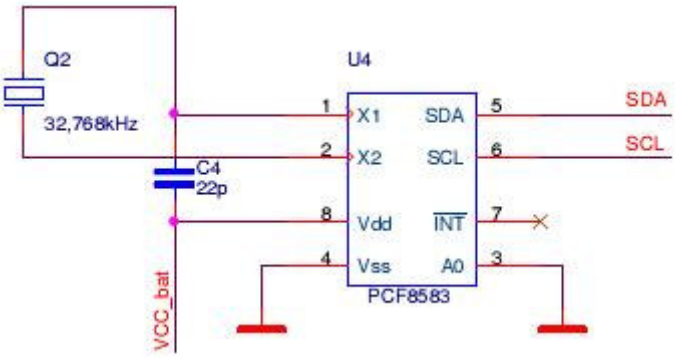
Pin WP (Write Protect) je vstup, který připojený na zem umožňuje normální zapisování do paměti, jako v tomto případě. Pokud by byl připojen na napájecí napětí, pak by zápis do paměti nebyl možný.

Bližší informace v příloze [9].

2.7 Obvod reálného času

Obvod reálného času jsem do zapojení zařadil z důvodu potřeby ukládat nejen data z měření, ale i datum a čas provedeného měření. K tomuto účelu velmi dobře posloužil obvod firmy Philips Semiconductors **PCF8583.** Tento obvod je velmi nenáročný na zapojení a komunikace s procesorem probíhá prostřednictvím sběrnice I2C. Schéma zapojení ukázáno na obrázku 9.

Obvod komunikuje s procesorem úplně stejně jako v předcházejícím případě paměť EEPROM, a to přes sběrnici I2C a vodiče SDA a SCL. Jediný rozdíl je v adresaci obou zařízení. Paměť musela mít na sběrnici adresu A0 = 1 právě proto, že A0 = 0 je využita pro tento obvod.



Obr. 9: Obvod reálného času

Výstup INT (Interrupt) není zapojen, protože toto zařízení nevyžaduje vnější přerušení nebo alarm způsobený časem nebo datem.

Obvod dále vyžaduje připojení externího krystalového oscilátoru **Q2** o frekvenci 32,768 kHz, který s vnitřní děličkou frekvence zajišťuje pevnou časovou základnu. Mezi tento oscilátor a kladné napájecí napětí musí ještě být, z důvodu stability vložen kondenzátor **C4.** Výrobce udává typickou hodnotu tohoto kondenzátoru 22 pF viz příloha [10].

Napájení v tomto případě není připojeno přímo na 5V, ale jak již bylo zmíněno, je připojeno přes záložní napájecí obvod, který dodává proud pro chod obvodu reálného času i při vypnutém zařízení a zabraňuje tak ztrátě dat o aktuálním čase.

Poznámka:

Jelikož oba vodiče sběrnice I2C jsou s otevřeným kolektorem, je nutné použít PULL-UP rezistory **R22** a **R23** k provozu celé sběrnice. Rezistory stačí dva na celou sběrnici a minimalizují možnost vzniku přeslechů. Jejich velikost není kritická a může se pohybovat od 1,8 kΩ až po 10 kΩ. Zvolil jsem hodnotu rezistorů 4,7 kΩ, která je nejčastěji používána. Rezistory mohou být umístěny k libovolnému zařízení připojenému na tuto sběrnici.

2.8 Maticová klávesnice

Jako ovládací prvek celého zařízení jsem vybral maticovou klávesnici. Hlavním důvodem byla úspora portů procesoru, protože tato klávesnice vyžaduje na obsluhu dvaceti jedna tlačítek pouze deset signálových vodičů. Další velkou výhodou této klávesnice jsou minimální rozměry a s tím související snadná montáž. vyobrazení klávesnice je na obrázku 10.

Obr. 10: Maticová klávesnice

Klávesnice má tři vstupy reprezentující sloupce a sedm výstupů prezentujících řádky. Dále ještě obsahuje jeden vstup a jeden výstup navíc. Ten reprezentuje pouze tlačítko START. Bližší informace jsou uvedeny v příloze [11].

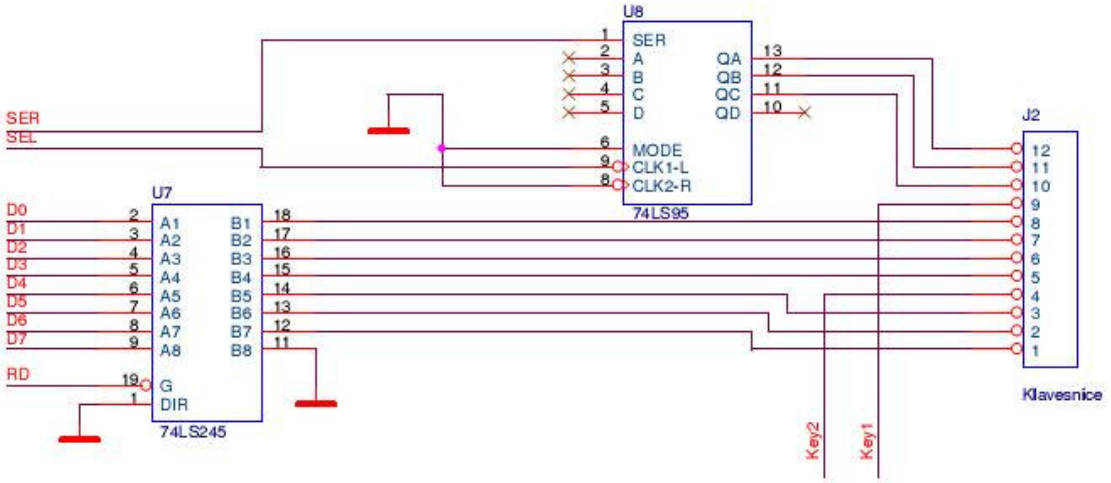
Samotné zapojení klávesnice do obvodu vyžaduje další dva podpůrné TTL obvody a je uvedeno na obrázku 11. Prvním obvodem je 4-bitový posuvný registr **U8** typu **74LS95,** příloha [12]. Druhým je pak 8-bitový výkonový budič sběrnice **U7** typu

74LS245, příloha [13]. Typy obvodů LS byly vybrány kvůli nízké spotřebě celého zařízení.

Princip celého zapojení spočívá především v činnosti posuvného registru. Ten je zapojen tak, že jeho funkce spočívá v přepínání jednotlivých sloupců klávesnice. Posuvný registr může pracovat buď v paralelním nebo sériovém režimu. Pro úsporu portů procesoru jsem vybral sériový režim. Ten je vybrán připojením vstupu **MODE** na zem, tím je zároveň vybrán posun doprava (QA->QD), který je zapotřebí ke správnému přepínání sloupečků klávesnice. Registr potom využívá sériový vstup a paralelní vstupy zůstávají nezapojeny. První tři výstupy jsou připojeny na vstupy klávesnice, čtvrtý je nezapojen. Celý posuvný registr je potom ovládán dvěma signály z procesoru **SER** a **SEL**.

Signál **SER** je připojen na sériový vstup registru. Signál **SEL** je připojen na hodinový vstup tohoto registru **CP1**. Hodinový vstup **CP2**, který provádí posun vlevo je připojen na zem, aby díky rušení nemohlo dojít k nežádoucímu posunu vlevo a tím k výběru nesprávného sloupce.

Výběr řádku potom probíhá následovně. Procesor na sériovém vstupu a zároveň na hodinovém vstupu nastaví úroveň log 1. Potom vrátí hodinový vstup zpět na úroveň log 0. Jelikož posuvný registr reaguje na sestupnou hranu hodinového signálu, bude na výstup QA převedena log 1 ze sériového vstupu. V dalším kroku je zapotřebí sepnout výstup QB, ale zároveň vypnout vstup QA, aby byl vybrán stále jen jeden sloupec klávesnice. To se provede jednoduše. Nyní procesor na sériovém vstupu registru nastaví úroveň log 0 a vytvoří impulz na hodinovém signálu, s jehož sestupnou hranou dojde k posunutí celého registru o jedno místo a tedy na QB bude log 1, ale na QA už log 0. Tím je vybrán další sloupec. Výběr třetího sloupce bude probíhat zcela analogicky s předcházejícím.



Obr. 11: Zapojení klávesnice

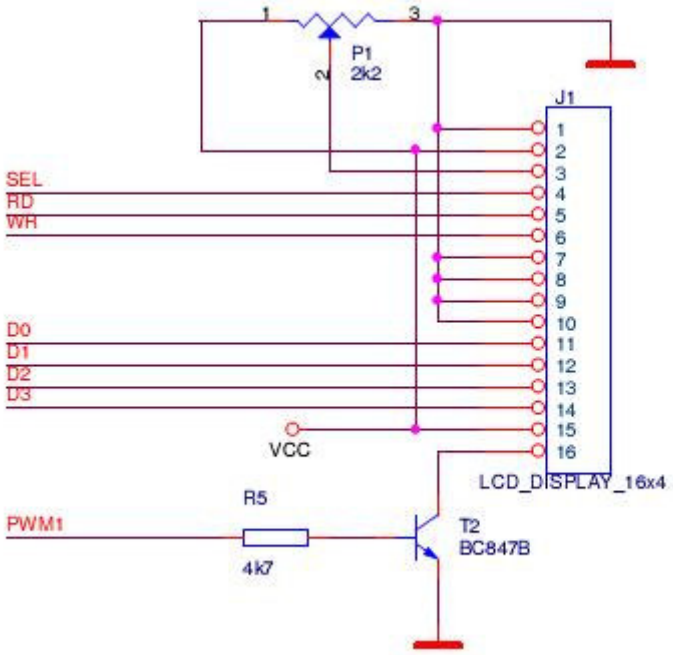
Při opakovaném čtení z klávesnice, kdy je opět vybrán první sloupec, bude předchozí výběr odsunut na výstup QD, kde nás nezajímá.

Čtení dat z klávesnice je následně prováděno výkonovým budičem sběrnice. Ten je nastaven pomocí vstupu **DIR** pro přenos ze sběrnice B na sběrnici A. K přenesení dat na sběrnici A tedy na vstupy procesoru dojde, je-li řídící signál **RD** na úrovni log 0. Ke čtení tedy může docházet po celou dobu přepínání registru, až do sepnutí nějakého tlačítka, ale to už je softwarová záležitost.

Vstup B8 je připojen na zem, protože z klávesnice nám stačí číst pouze sedm hodnot.
Na schématu je ještě vidět jeden vstup **Key2** a výstup **Key1.** Ty reprezentují již zmíněné tlačítko START. Jelikož toto tlačítko je použito k zapínání celého zařízení, nemůže být čteno procesorem, ale je vyvedeno přímo ke spínacímu tranzistoru **T3.** Zapínáním celého zařízení se zabývá kapitola 2.14.

2.9 LCD display

Zařízení používá ke komunikaci s uživatelem klasický LCD displej založený na standardu HD44780 a rozměru 4x16 znaků. Připojení displeje k procesoru je zobrazeno na obrázku 12. Displej je připojen na stejnou datovou sběrnici jako klávesnice, ale protože je provozován ve 4-bitovém módu využívá pouze první čtyři její vodiče D0 až D3.



Obr. 12: Zapojení LCD displeje

Řídící signály displeje jsou potom připojeny na port D. **SEL** je signál sloužící k výběru registru displeje. K zápisu dat musí být ve stavu log 1, k zápisu řídících příkazů je třeba, aby byl ve stavu log 0. **RD** je signál řídící směr toku dat. Pro zápis dat do displeje musí být ve stavu log 0. Pro čtení dat musí být ve stavu log 1.

Pokud chceme pouze data zapisovat, ale ne číst, je možno tento signál připojit trvale na zem. Tím je ovšem znemožněno čtení příznakového bitu BUSY a procedura obsluhující displej musí obsahovat čekací smyčky na dokončení operací, čímž dochází k plýtvání výpočetního času procesoru.

WR je uvolňovací signál. Při zápisu dat na displeji budou data displejem přečtena se sestupnou hranou tohoto signálu.

Piny 1 a 2 jsou určeny k napájení displeje. Pin 3, který je připojen přes odporový trimr **P1** na napájecí napětí, potom slouží k nastavení intenzity svitu.

Vstupy 7 až 10 jsou datové, ale slouží pouze v případě, že je displej používán v 8-bitovém módů. V tomto případě zůstávají nevyužity a jsou připojeny na zem.

Vstupy 15 a 16 slouží k napájení LED podsvícení displeje. Maximální odběr je uváděn až 220 mA, to je pro celkový odběr zařízení nepřijatelné. Intenzita podsvícení je proto řízena spínáním přes tranzistor **T2**. Řídící signál PWM1 je generován jedním z PWM výstupů procesoru a je přiveden na bázi tohoto tranzistoru. Hodnotu odporu báze **R5** uvádí následující výpočet:

Odpor báze **R5:**

$$I_C = 220mA$$

$$I_C = h_{fe}.I_B \rightarrow I_B = \frac{I_C}{h_{fe}} = \frac{220}{200} = 1{,}1mA \qquad (6)$$

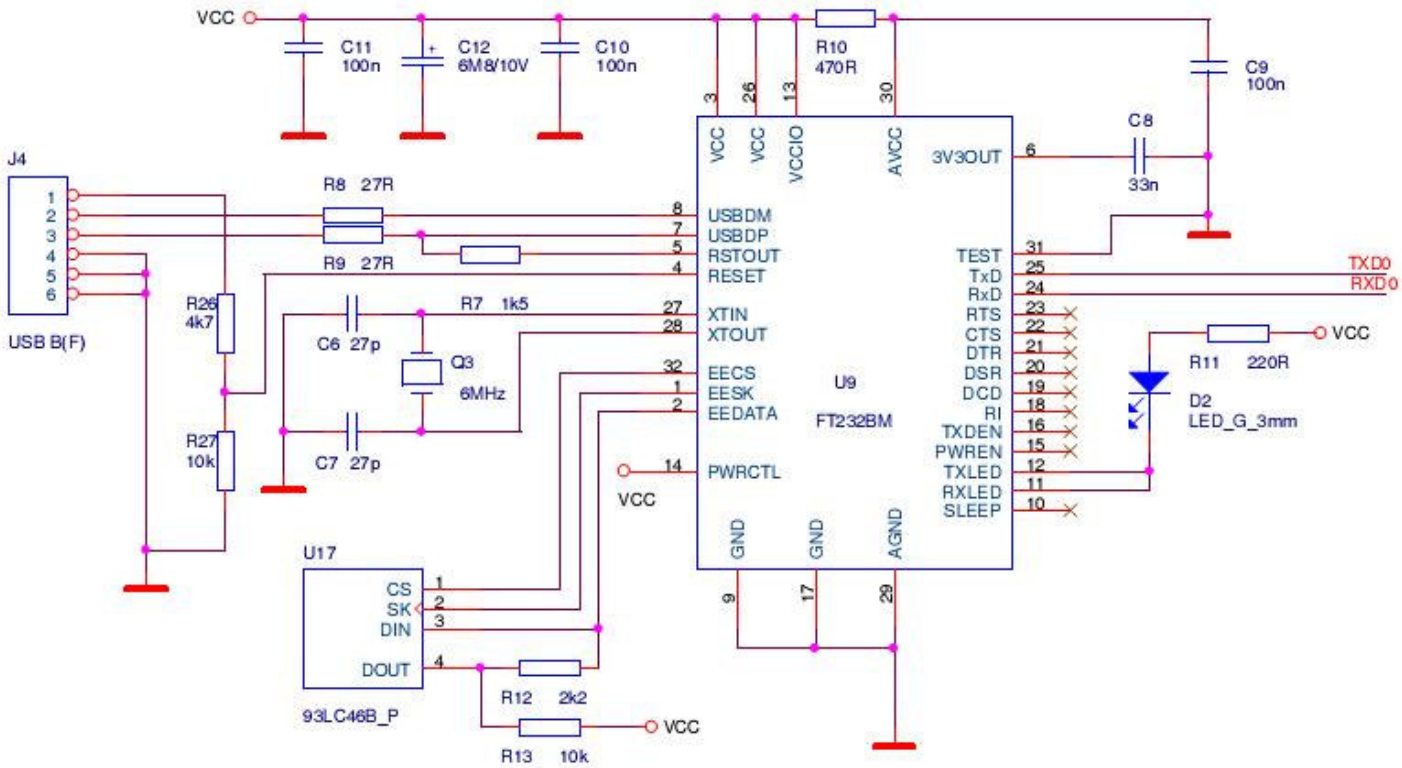
$$R_B = \frac{U_{CC}}{I_B} \doteq 4550\Omega \qquad (7).$$

Nejbližší větší vyráběný odpor je 4,7 kΩ, což je plně dostačující hodnota. Intenzita osvětlení potom bude záležet pouze na střídě generovaného PWM signálu.

2.10 Komunikační rozhraní USB

Jako primární komunikační rozhraní tohoto zařízení s počítačem jsem vybral USB komunikaci, především z důvodu rozšířenosti tohoto protokolu.

Pro převod na USB komunikaci jsem využil konvertoru **FT232BM** firmy FTDI Chips. Jedná se o jednočipový převodník $USB \Leftrightarrow UART$. Celé zapojení je uvedeno na obrázku 13.



Obr. 13: USB konvertor

Základem celého zapojení je integrovaný obvod FT232BM **U9,** příloha [14]. Druhou jeho velmi podstatnou součástí je sériová paměť EEPROM **U17** o velikosti 1kB, která slouží k uložení VID, PID, sériového čísla, řetězce popisu výrobku a hodnoty odebíraného proudu. Paměť je typu **93LC46B** firmy Microchip, příloha [15], protože musí být v 16-bitové šíři a musí být schopná pracovat na rychlosti 1Mb/s. Paměť EEPROM je programovatelná přímo na desce pomocí speciálního programu, příloha [43], nebo funkcemi uživatelského rozhraní.

UART rozhraní:

UART rozhraní obsahuje datové linky UART a řídící signály modemu. Vstup RXD a výstup TXD jsou připojeny na UART0 rozhraní procesoru. Řídící signály modemu nejsou zapotřebí a jsou nezapojeny.

USB rozhraní:

USB rozhraní obsahuje datové signály pro připojení k USB. USB signál D+ (USBDP) vyžaduje odpor R7 = 1,5 kΩ, jako pull-up směrem k RSTOUT. USB rozhraní je ještě doplněno zakončovacími odpory R8 a R9 o velikosti 27 Ω.

EEPROM rozhraní:

EEPROM rozhraní obsahuje signály pro připojení konfigurační EEPROM. Všechny tyto vývody jsou po resetu ve třetím stavu.

Pin **EECS** je chip select paměti a je přímo připojen na vstup **CS** paměti. Výstup **EESK** je hodinový signál přímo připojený na signál **SK** paměti. Pin **EEDATA** je vstup/výstup a je přímo připojen na datový vstup **DIN**. Na tento vývod je připojen i datový výstup paměti **DOUT**. Aby se zabránilo konfliktům je připojení provedeno přes odpor **R12** o doporučené hodnotě 2,2 kΩ.

Data EEPROM jsou použita pro definici deskriptoru USB. Paměť potvrzuje požadavek stažením signálu DOUT do log 0. Pro test této podmínky je potřeba pull-up **R13** obvyklé hodnoty 10 kΩ.

Připojení oscilátoru:

Jako zdroj vnějšího hodinového signálu je použit krystal **Q3** o kmitočtu 6 MHz. V tomto případě je nutné použít vnější zatěžovací kondenzátory **C6**, **C7** o kapacitě 27 pF.

Rozhraní pro připojení indikačních LED:

FT232BM má dva výstupy pro připojení LED diod indikujících provoz na sběrnici. Jedná se o výstupy s otevřeným kolektorem, takže při indikaci provozu přejdou ze stavu vysoké impedance do stavu log 0. Jelikož nám stačí indikovat aktivitu sběrnice obecně, můžeme oba výstupy spojit a ovládat jimi jednu led **D2**. Pro omezení proudu diodou je k němu do série zapojen rezistor **R11** o velikosti 220 Ω.

Řízení spotřeby:

Signály pro řízení spotřeby jsou zapojeny následovně. Vstup **PWRCTL** je připojen na napájecí napětí, aby se obvodu sdělilo, že je použito napájení z vlastního zdroje. Výstup **PWREN** se používá pro napájení vnější logiky se spotřebou větší než 100 mA z USB sběrnice. To v tomto zapojení není využito, a proto je tento vývod nezapojen. Výstup **SLEEP** se využívá v režimu USB SUSPENDED, a jelikož ani toto zapojení zde není využito, je také nezapojen.

Pomocné signály:

Vstup **RESET** slouží pro resetování FT232BM vnějším obvodem. Napájení získané ze sběrnice USB je použito k řízení vývodu **RESET**. Když je USB hostitel nebo HUB vypnut, je **RESET** ve stavu log 0 přes odpor **R27** a obvod zůstane v resetu.

Když je **RESET** v log 0, je také **RSTOUT** v log 0, takže USBDP neodebírá z pull-upu **R7** žádný proud. Když je FT232BM v resetu, jsou vývody UART ve třetím stavu.

Když je napájení z USB odpojeno, je FT232BM držen v resetu. Stále ale běží 6MHz oscilátor a zařízení má tedy určitý odběr.

Napájecí vývody:

Napájecí vývody slouží pro připojení napájecího napětí obvodu, definici napěťových úrovní rozhraní UART a přivedení napájecího napětí pro násobičku kmitočtu.

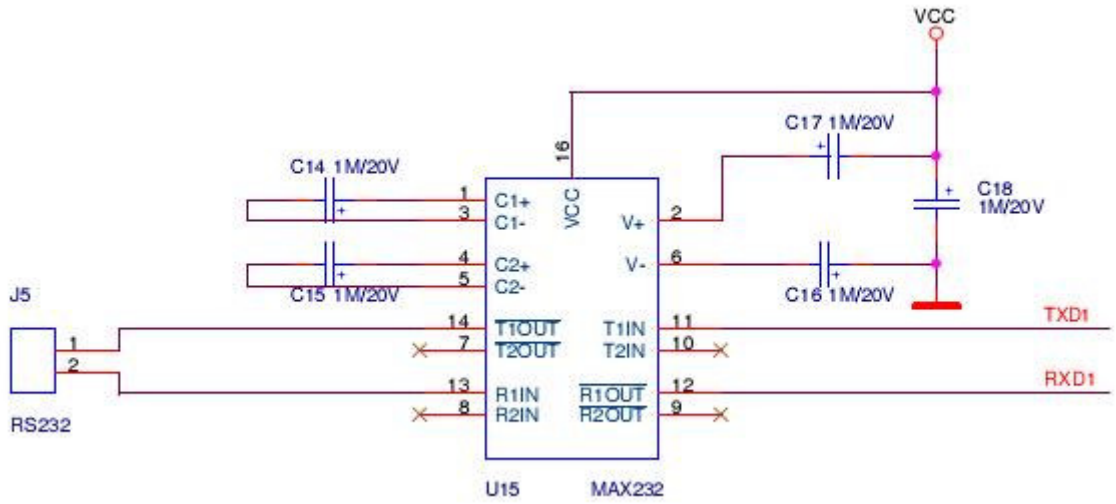
Výstup **3V3OUT** je výstup zabudovaného regulátoru. Ten musí být zablokován kondensátorem **C8** o velikosti 33 nF. **VCCIO** slouží k napájení UART a je připojeno na napájecí napětí. **AVCC** a **AGND** jsou napájením pro zabudovanou násobičku hodin.

Napájení celého obvodu je ještě doplněno blokovacími kondensátory **C9**, **C10**, **C11**, **C12** pro vyhlazení napájecího napětí.

2.11 Komunikační rozhraní RS232

Pro vytvoření sériové komunikace RS 232 jsem použil velmi frekventovaný obvod firmy MAXIM **MAX232,** příloha [16], jelikož jeho použití je velmi jednoduché.

Jedná se o duální převodník TTL na RS 232. Obsahuje dvě dvojice oddělovačů konvertujících napěťové úrovně. V zapojení je využit pouze první komunikační kanál. Napětí pro RS 232 se získává pomocí nábojové pumpy. Výstupní napětí proto značně závisí na kvalitě použitých kondenzátorů. Vzhledem k tomu, že napětí u elektrolytických kondenzátorů časem značně klesá, použil jsem kvalitnější tantalové kondezátory, jejichž kapacita 1μF je požadována výrobcem pro správnou funkci obvodu. Schéma zapojení je na obrázku 14.



Obr. 14: RS 232

2.12 Napájení zařízení

Celé zařízení má dva možné způsoby napájení. Prvním je napájení z nabíjecích akumulátorů, druhým je pak napájení pomocí externího napájení.

2.12.1 Nabíjecí akumulátory

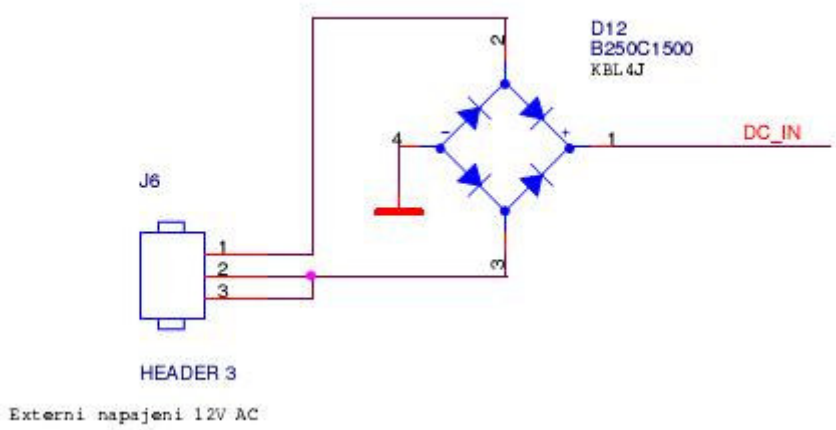
Akumulátory jsou hlavním zdrojem energie pro napájení celého zařízení. Jelikož jsou součástí zařízení, umožňují provoz kdekoliv bez ohledu na externí napájení.

Použil jsem šest kusů Ni-MH baterií o napětí 1,2 V a energii 1800 mAh. Tyto baterie zapojeny do série dávají napětí 7,2 V, které je dostatečné pro stabilizátor napětí 5V. Kapacita baterií by měla být postačující pro několikahodinový provoz.

2.12.2 Externí napájení

Externí napájení je použito především pro nabíjení akumulátorů. Ale je možno ho použít i pro provoz celého zařízení v době, kdy jsou akumulátory vybity, nebo dokonce i vyjmuty ven ze zařízení.

Schéma zapojení pro externí napájení je jednoduché a je ukázáno na obrázku 15.



Obr. 15: Externí napájení

Vstupem do tohoto obvodu je externí napájecí napětí o velikosti 12 V AC. Toto napětí je usměrněno na diodovém můstku **D12**. Tento můstek je dimenzován tak, aby vydržel dodávat proud až 1,5 A. Výstup je pak použit k napájení samotného zařízení a především nabíječky akumulátorů.

2.13 Nabíječka akumulátorů

Nabíjení akumulátorů je jedním z velmi důležitých problémů. Při řešení tohoto úkolu se nabízelo několik variant. Nejjednodušší řešení by bylo používat externí nabíječku. To by ovšem znamenalo, že akumulátory by se musely vyndávat ze zařízení, což by ovšem stěžovalo obsluhu zařízení. Navíc by zařízení bylo vypnuté a po dobu nabíjení by s ním nebylo možno pracovat.

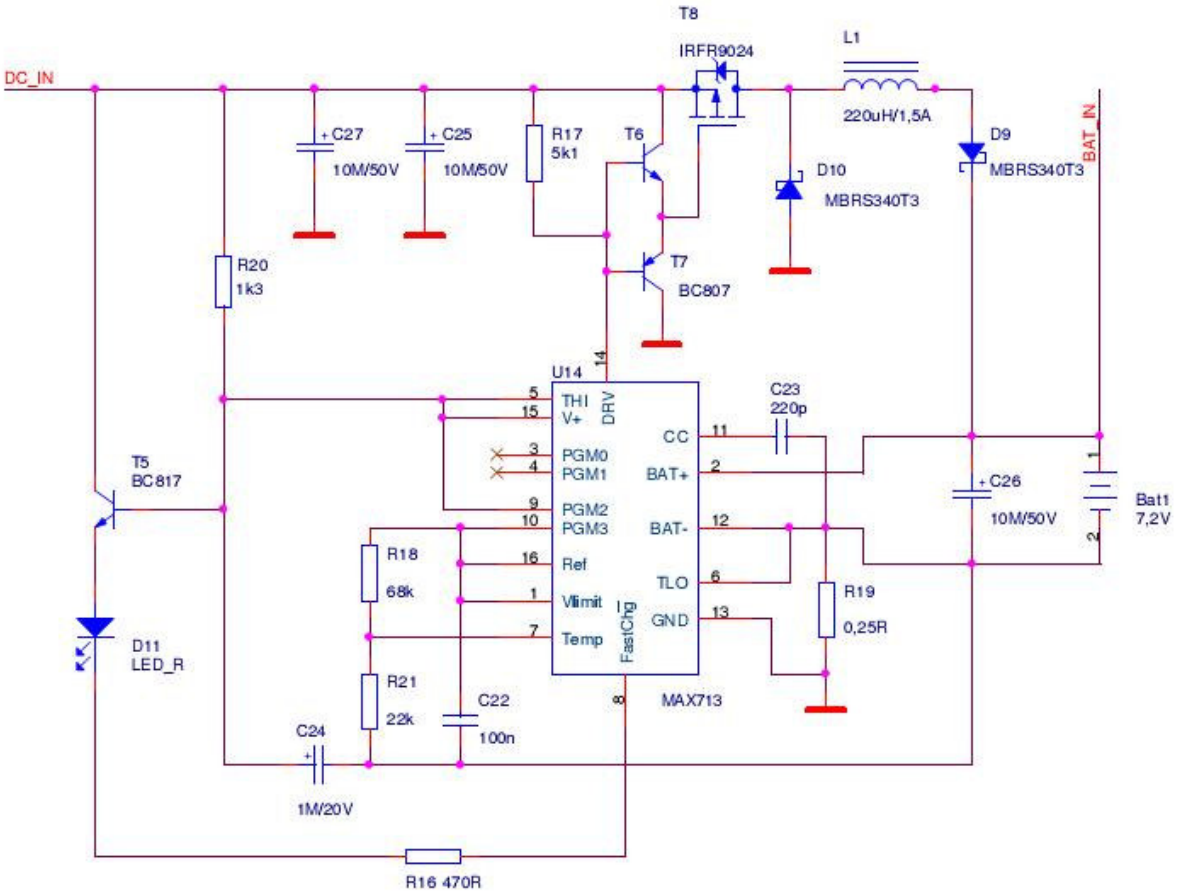
Druhou variantou bylo postavit externí nabíječku, která by akumulátory nabila pouhým připojením k zařízení. Toto řešení ovšem také neumožňuje použití zařízení při vybitých akumulátorech .

Nejlepší řešení tedy bylo integrovat celou nabíječku přímo do zařízení. Bylo zapotřebí tedy navrhnout takové zapojení, které je dostatečně malé, aby se vešlo na desku s plošnými spoji s ostatními obvody a aby dokázalo nabít akumulátory v dostatečně krátkém čase.

Po usilovném hledání jsem objevil obvod firmy MAXIM **MAX713,** který vyhovoval všem podmínkám. Zapojení nabíječky s tímto obvodem je velmi jednoduché s minimem externích součástek, ale umožňuje široké spektrum možností nastavení. Jde o obvod pro nabíjení Ni-MH a NiCd akumulátorů, kterému stačí napájecí napětí pouze o 1,5 V než je napětí akumulátorů. Umožňuje nabíjení až šestnácti článků a nastavení doby nabíjení od 22 do 264 minut. Velkou výhodou je také, že tento obvod umožňuje nabíjení akumulátorů i při připojené zátěži, a tedy nabíjení baterií při provozu celého zařízení.

Obvod **U14** MAX713 detekuje zápornou derivaci přírustku napětí v čase pro ukončení nabíjení. V tomto okamžiku se již akumulátory nenabíjí, pouze se zahřívají. Tím dochází k zmenšování vnitřního odporu a tím i k poklesu napětí. K tomuto účelu by bylo možno použít i obvod MAX712, který detekuje nulovou derivaci, tento obvod se k nám ale nedováží.

Pro minimalizaci ztrát jsem tento obvod použil ve spínaném režimu. Více informací o tomto obvodu je uvedeno v příloze [17]. Schéma celého zapojení je na obrázku 16.



Obr .16: Nabíječka akumulátorů

Pro napájení je použit externí zdroj napětí. Toto napětí musí být ve spínaném režimu alespoň o 2V větší než je napětí baterií. Externí napájení je 12 V, což je dostatečné.

Prvním krokem bylo určit si nabíjecí čas a z toho nabíjecí proud. Dobu nabíjení jsem si zvolil 132 minut viz příloha [17], aby nabíjecí proud nebyl příliš veliký a zároveň doba nabíjení příliš dlouhá. Nabíjecí proud se pak spočítá následovně:

$$I_{FAST} = \frac{(Capacity_of_battery_in_mAh)}{(Charge_time_in_hours)} = \frac{1800}{2,2} \doteq 820mA \qquad (8).$$

Počet článků akumulátoru určuje zapojení vstupů **PGM0** a **PGM1**. Dobu nabíjení potom zapojení vstupů **PGM2** a **PGM3**.

Pro napájení obvodu je zapotřebí zaručit proud do vstupu **V+** v rozmezí 5 až 20 mA. Jelikož je napájecí napětí stabilní postačí k tomuto účelu odpor **R20**. Jeho velikost udává následující výpočet:

$$R = (DC_IN - 5V)/5mA = \frac{12-5}{0,005} = \frac{7}{0,005} = 1400\Omega \qquad (9).$$

Nejbližší menší vyráběný odpor je 1,3 kΩ.

Dioda **D9** zabraňuje přenosu proudu na výstup **DRV**. Odpor **R19** nastavuje proud do baterie pro rychlé nabíjení. Rozdíl napětí mezi zemí a **BAT-** je regulován na 250 mV. Výstup **DRV** zvyšuje proud tranzistoru **T8**, pokud rozdíl napětí spadne pod 250 mV, nebo snižuje proud, pokud rozdíl napětí přesáhne 250 mV.

$$R_{SENSE} = \frac{0{,}25V}{I_{FAST}} \doteq 0{,}30\Omega \qquad (10)$$

Jelikož teplota akumulátorů není měřena, musí být vstupy **TLO** a **TEMP** zapojeny podle následujícího pravidla. Aby nabíjení vůbec začalo, musí být napětí na **TEMP** větší než na **TLO**. Z toho důvodu je **TLO** připojen na zem a **TEMP** je připojen přes napěťový dělič **R18** a **R21** na referenční výstup **REF** o napětí 2V.

Vstup **Vlimit** nastavuje maximální napětí článků. Zajišťuje, že napětí na baterii nepřesáhne **Vlimit * Počet článků**. Napětí na tomto vstupu nesmí přesáhnout 2,5 V. Z tohoto důvodu je připojeno na referenční výstup **REF**.

Kondenzátor **C23** snižuje frekvenci spínání (nominální je 30 kHz). V technických údajích o tranzistoru **T8,** příloha [18], je uvedeno, že frekvenční pásmo pro spínání musí být menší než $f_B = 10kHz$. Z rovnice 10) vypočítáme potřebnou kapacitu pro omezení šířky pásma:

$$BW_{CRL} = \frac{gm}{C_{23}} = \frac{0{,}0018}{220pF} = 8181Hz \qquad (11).$$

Podmínka stability proudové smyčky $BW_{CRL} < f_B$ je tedy splněna.

Stav nabíjení je indikován LED diodou **D11**. Její katoda je připojena na tranzistor **T5**, který je sepnutý po přivedení napájecího napětí. Anoda je potom připojená na výstup **FASTCHG**, který je s otevřeným kolektorem. Po dobu nabíjení dioda nepřetržitě svítí. Pokud je akumulátor nabit dioda bliká, v tomto stavu je nabíjecí proud baterie omezen asi na 50 mA, aby nedocházelo k vybíjení. Tento proud je dán zapojením vstupu **PGM3**, jelikož ten je připojen na výstup **REF.** Proud v tomto stavu se vypočítá následovně:

$$I_{TRICKLE} = \frac{I_{FAST}}{16} = \frac{820}{16} \doteq 51mA \qquad (12).$$

Pokud není připojeno externí napájení nebo pokud je akumulátor odpojen od zařízení, dioda nesvítí. Proud diody je omezen odporem **R16**.

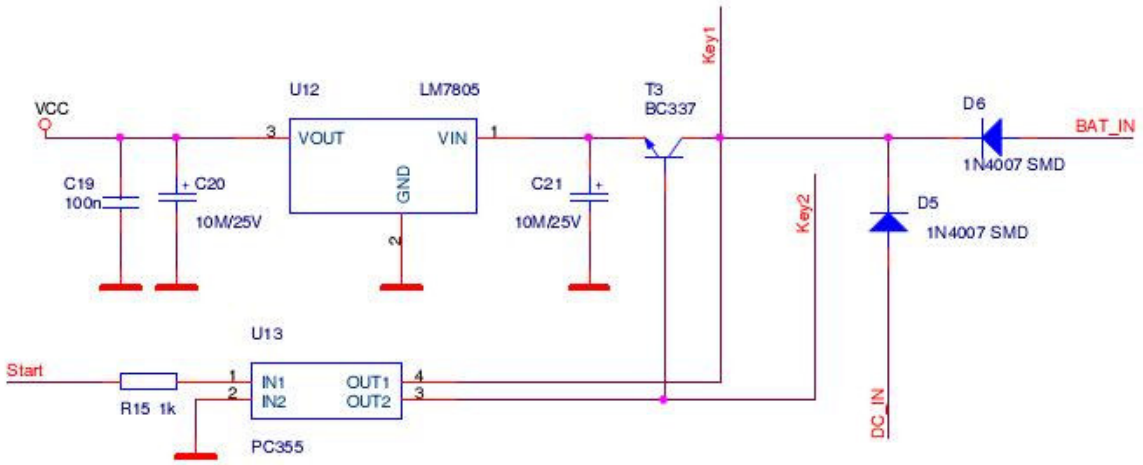
Spínání výkonového tranzistoru **T8** zajišťuje dvojice komplementárních tranzistorů **T6** a **T7**. Tranzistory by podle výrobce měly být typu CMPTA06 resp. CMPTA56. Tyto tranzistory se ale u nás nedají sehnat, proto jsem použil BC807 a BC817, které mají velmi podobné charakteristiky. Nabíjecí obvod ještě obsahuje tlumivku **L1**. Ta zabraňuje vzniku proudových špiček při spínání. Dioda **D10** filtruje záporné napěťové špičky, které vznikají při vypínání výkonového tranzistoru T8.

Napájení obvodu nabíječky jsem ještě doplnil o kondenzátory **C25** a **C27** o velikosti 10 µF, které mají za úkol dodávat proud do obvodu v případě krátkodobých poklesů napájecího napětí.

Jedinou nevýhodou této nabíječky je, že procesor nemá možnost sledovat stav nabití akumulátorů a tedy informovat uživatele o stavu baterií.

2.14 Stabilizátor napětí

Zdroj zaručuje stabilizaci napájecího napětí na hodnotu 5 V. Původní myšlenkou bylo použít nějaký stabilizátor typu LDO, tedy stabilizátor s minimálním vlastním odběrem. Jako nejlepší typ jsem objevil stabilizátor MCP1700 od firmy Microchip, který má vlastní spotřebu udávanou výrobcem 1 µA. Bohužel tento typ stabilizátorů nevydrží vstupní napájecí napětí větší než 8 V. Jelikož se ale při napájení z externího zdroje dostane na jeho vstup napětí okolo 12 V, je použití těchto stabilizátorů vyloučeno.



Obr. 17: Stabilizátor napětí.

Nakonec jsem se rozhodl pro použití obyčejného stabilizátoru **LM7805**, příloha [19], který vyhovuje všem požadavkům. Zvládne dodávat proud až 1 A a vydrží napájecí napětí až 35 V. Jeho vlastní spotřeba je maximálně 8 mA, což je v rámci spotřeby celého zařízení zanedbatelné. Stabilizátor jsem ještě doplnil chladičem, aby nedocházelo k jeho zahřívání během provozu s větším odběrem. Schéma celého zapojení stabilizátoru je na obrázku 17.

Stabilizátor je stabilní s externími kondenzátory **C20** a **C21** o velikosti 10 µF. Výstup je ještě doplněn filtračním kondenzátorem **C19** o velikosti 100 nF.

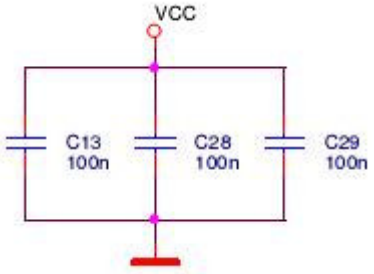
Pokud je k přístroji připojeno externí napájení, dioda **D6** je uzavřena a tím je baterie odpojena od zátěže. Dioda **D5** zabraňuje napájení obvodu nabíječky při napájení z baterie.

Velmi podstatnou součástí tohoto zapojení je spínací tranzistor **T3**. Ten je využit k zapínání celého zařízení. Je zařazen mezi akumulátor a stabilizátor **U12**. Zvolil jsem typ **BC337**, protože jeho kolektorový proud může být až 500 mA. To je z hlediska spotřeby dostatečné.

Tranzistor je možno spínat dvěma způsoby. První možnost je stiskem tlačítka START na klávesnici, kterým se celé zařízení zapne. To je připojeno na výstup napájení Key1. Po jeho sepnutí se na vstup Key2, který je připojen přímo na bázi tranzistoru, dostane napájecí napětí, a tím dojde k jeho sepnutí a oživení celého zařízení. Jelikož je zapotřebí dostat tranzistor do saturace a tlačítko má vnitřní odpor 25 Ω, není zapotřebí další omezující odpor.

Tlačítko je zapotřebí držet dostatečně dlouho, aby došlo k restartu procesoru. Ten pak musí vygenerovat řídící signál Start. Tím dojde k sepnutí optočlenu **U13**, který přivede opět napájecí napětí na bázi, a tím udrží tranzistor sepnutý. Signál Start pak musí být sepnut po celou dobu provozu zařízení, protože procesor tímto způsobem udržuje napájení celého zařízení. Spínací proud optočlenu je omezen rezistorem **R15** o velikosti 1 kΩ.

Napájení obvodů TTL je ještě zapotřebí doplnit blokovacími kondenzátory o velikosti 100 nF. Tyto kondenzátory by měli být na desce plošných spojů umístěny co nejblíže k napájecím vývodům obvodů TTL. Jejich zapojení okazuje obrázek 18.



Obr. 18: Blokovací kondezátory

2.14.1 Spotřeba zařízení

Nyní je třeba se zastavit u spotřeby celého zařízení. Rozvaha o spotřebě je důležitá ze dvou důvodů.

Maximální spotřeba celého zařízení byla důležitá při výběru spínacího tranzistoru **T3**. A dále pak určuje minimální dobu provozu při napájení z baterií.

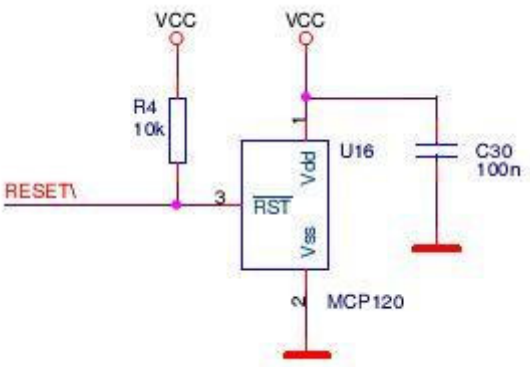
Po sečtení maximálních napájecích proudů všech obvodů vyšla celková spotřeba zařízení asi 350 mA. V tom není ale započítáno napájení pro IRC senzor, které může být až 120 mA. V součtu není překročeno povolených 500 mA. Problém se objeví při přičtení dalších až 200 mA, které je zapotřebí pro napájení podsvícení LCD displeje. Z toho vyplývá, že pokud bude IRC senzor napájen přímo z akumulátorů, bude zapotřebí vypnout podsvícení displeje po dobu měření. Navíc podsvícení bude muset být řízeno pomocí PWM signálu, který sníží spotřebu, protože po přičtení maximálního odběru se dostaneme přes povolených 500 mA.

Jelikož plně nabitý akumulátor má kapacitu 1800 mAh, tak zařízení by mělo být schopno provozu minimálně tři hodiny.

Při nabíjení baterií je zapotřebí proud o velikosti asi 900 mA. Pokud bude při nabíjení celé zařízení zapnuto, může tedy celkový odběr vzrůst až na 1400 mA. Na tento odběr je ale usměrňovací můstek D12 dimenzován, a provoz zařízení proto nebude omezen.

2.15 Napěťový watchdog

Pro hlídání úrovně napájecího napětí jsem v zapojení použil napěťový watchdog **U16** firmy Microchip **MCP120**, příloha [20]. Ten udržuje procesor v resetu a paměť SRAM ve stavu NOT SELECTED, aby nedošlo ke ztrátě dat, pokud napětí poklesne pod stanovenou hranici. Zapojení ukazuje obrázek 19.



Obr. 19: Napěťový watchdog

Pokud je napájecí napětí v požadovaných mezích je výstup watchdog ve stavu vysoké impedance a signál RESET je přes pull-up rezistor R4 držen ve stavu log 1. Pokud napětí klesne pod stanovenou hranici, je výstup držen ve stavu log 0.

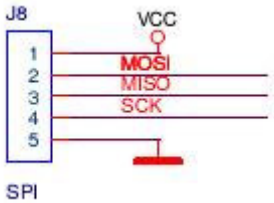
Kondenzátor **C30** o velikosti 100 nF zajišťuje přídavnou přechodovou odolnost vůči rušivým vlivům. Kondenzátor by měl být při montáži umístěn co nejblíže k napájecímu vývodu.

2.16 Vyvedené konektory

Tato kapitola popisuje, které další konektory jsou v zapojení ještě vyvedeny, i když třeba nejsou využity**.**

Konektor sběrnice SPI

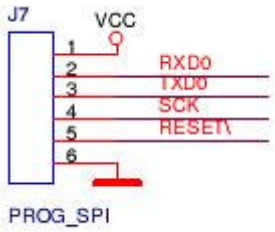
Prvním vyvedeným konektorem je sběrnice SPI (Serial Peripheral Interface Bus). Ta využívá SPI rozhraní procesoru. Tato sběrnice není využita, ale mohla by v budoucnu sloužit k dalšímu rozšíření periférií. Zapojení je na obrázku 20.



Obr. 20: Konektor SPI

Konektor programovacího rozhraní SPI

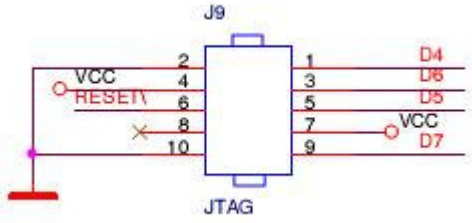
Dalším konektorem je programovací rozhraní SPI. Toto rozhraní slouží k nahrávání softwaru a zpětnému čtení obsahu pamětí procesoru. Zapojení je na obrázku 21.



Obr. 21: Programovací rozhraní SPI

Konektor JTAG

Poslední konektor je JTAG (Joint Test Action Group), tento standard představuje efektivní metodu testování užívající metody skenování rozhranní – boundary – scan technique. Ten umožňuje dálkové nahrávání dat, testování a řízení zařízení. Zapojení je na obrázku 22.



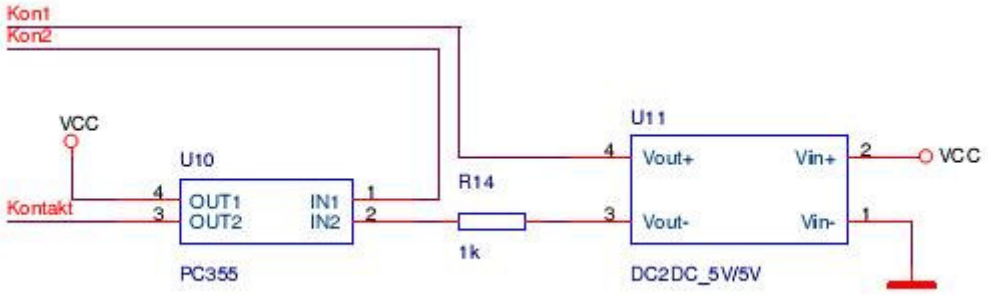
Obr. 22: Konektor JTAG

2.17 Obvody pro podporu IRC senzoru

Poslední kapitola této části popisuje zapojení obvodů, které podporují samotné měření pomocí IRC senzoru, ať už jde o jeho napájení nebo detekování vypnutí lisu.

2.17.1 Obvod pro bezpotenciálové spínání měření

Tento obvod zajišťuje detekci vypnutí lisu a tím zahájení měření. Obsahuje DC/DC měnič **U11**, ten je použit z důvodu impedančního oddělení spínacího napětí. Celý obvod je na obrázku 23.



Obr. 23: Obvod pro bezpotenciálové spínání měření

K samotnému spínání začátku měření slouží optočlen **U10**. Ten je sepnut v okamžiku spojení vstupů Kon1 a Kon2. Jelikož měnič **U11** dodává napětí pro sepnutí optočlenu, může být k spínání použit libovolný koncový spínač nebo tlačítko, bez nutnosti přítomnosti externího napětí. Po sepnutí je signál Kontakt přiveden na vstup vnějšího přerušení procesoru a tím je indikován začátek měření.

Pokud by spínání bylo řízenou nějakým vnějším signálem, bylo by zapotřebí toto zapojení upravit a ke spínání využít např. relé.

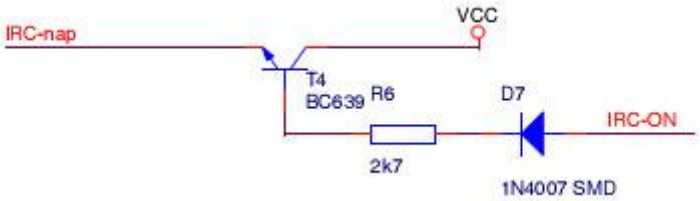
2.17.2 Obvod pro napájení IRC senzoru

V praxi by se mohlo stát, že IRC senzor použitý pro měření, bude zapotřebí napájet z akumulátorů z důvodu nepřítomnosti dalšího napájení. Obvod je v podstatě tvořen pouze spínacím tranzistorem **T4**. Ten po sepnutí přivede napájecí napětí na konektor senzoru a umožní tak jeho napájení. Tranzistor byl zvolen tak, aby vydržel napájecí proud 120 mA, který udává výrobce senzoru. Dioda **D7** je v zapojení použita z důvodu bezpečného rozepínání tranzistoru. Pro tento účel bez problému postačí univerzální dioda **1N4007**. Velikost odporu **R6** je vypočítána následovně:

$$I_B = \frac{I_C}{h_{fe}} = \frac{120}{90} = 1{,}3mA \qquad (13)$$

$$R_B \geq \frac{U_{CC} - U_D - U_{BE}}{I_B} = \frac{5 - 1 - 0{,}7}{0{,}0013} = \frac{3{,}3}{0{,}0013} = 2540\Omega \qquad (14).$$

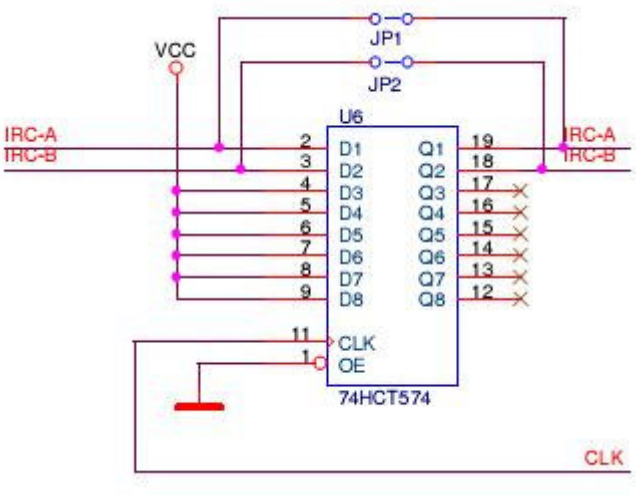
Nejbližší větší vyráběný odpor je o velikosti 2,7 kΩ. Použité číselné údaje jsou převzaty z dokumentace k tranzistoru **T4 BC639**, příloha [21]. Schéma zapojení je na obrázku 24.



Obr. 24: Obvod pro napájení IRC senzoru

2.17.3 Vzorkovací obvod

Vzorkovací obvod **U6** má za úkol odebírat vzorky z IRC senzoru s danou vzorkovací frekvencí. Jeho celé zapojení je na obrázku 25.



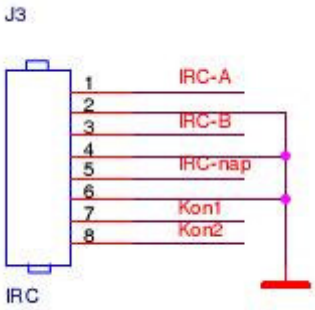
Obr. 25: Vzorkovací obvod

Základ zapojení tvoří obvod **74HCT584,** příloha [22]. Jde o osminásobný klopný obvod typu D. Výstupy ze senzoru jsou přivedeny na první dva vstupy. Ty jsou s náběžnou hranou hodinového signálu **CLK** převedeny na výstupy, a tím i na vstupy procesoru, kde budou dále zpracovány. Vstup **OE** (Output Enable) musí být připojen na zem, jinak by přenos ze vstupu na výstup byl blokován. Nevyužité vstupy dalších klopných obvodů doporučuje výrobce připojit na napájecí napětí. Jejich výstupy zůstávají nezapojeny.

Celé zapojení je doplněno dvěma zkratovacími propojkami **JP1** a **JP2**, které umožňují přivést signály ze senzoru přímo na vstupy vnějších přerušení procesoru. Jelikož uvažovaná vzorkovací frekvence měla být asi 1 kHz a procesor by měl zvládat zpracovávat data na základě přerušení mnohem rychleji, další konstrukce počítá s využitím přerušení pro zpracování měření a tento obvod nebude osazen. V návrhu ale zůstává možnost jeho budoucího využití.

2.17.4 Konektor senzoru

Poslední částí, o které jsem se ještě nezmínil, je zapojení konektoru samotného IRC senzoru. Toto zapojení je na obrázku 26.



Obr. 26: Konektor IRC senzoru

Konektor obsahuje výstupy dvou signálů senzoru IRC-A a IRC-B, dále napájení senzoru a vývody pro kontakt bezpotenciálového spínání.

Shrnutí

V této části byl popsán návrh celého zařízení a detailní zapojení všech součástek. Schéma celého zařízeni je uvedeno v příloze [23]. Schéma bylo vytvořeno pomocí programu OrCad Capture CIS, verze 9.2. Celý projekt je přiložen v příloze [24].

3. Plošný spoj

Po dokončení návrhu tohoto zařízení bylo zapotřebí vytvořit plošný spoj. V rámci této práce jsem vytvořil dvě verze zařízení. V této kapitole budou obě dvě popsány.

Nejdříve je popsána první verze. První verze obsahuje především výčet všech nalezených chyb, ať už v samotném návrhu celého zařízení či v návrhu plošného spoje. Dále je v této části zmíněno, co se v této verzi povedlo oživit a co funguje.

Druhá verze je již funkční a v další části je popsáno její oživování a testování.

3.1 První verze

První část této kapitoly zahrnuje výčet všech nalezených chyb v první postavené verzi celého zařízení.

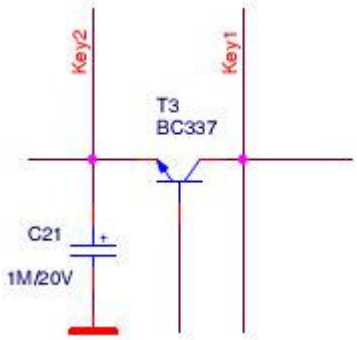
Chyby v návrhu zapojení

- Sběrnice I2C neobsahuje pull-up rezistory **R22** a **R23**. Jelikož oba vodiče sběrnice jsou s otevřeným kolektorem, tak připojená zařízení jsou sice schopna stáhnout logickou úroveň na těchto vodičích na log 0, ale už nejsou schopna je vytáhnout na log 1. Z toho plyne, že bez těchto odporů nejsou schopna zařízení připojená na I2C mezi sebou komunikovat a v této první verzi tedy nefungují.

- Výkonový budič datové sběrnice z klávesnice **U7** nemá připojený vstup **DIR** na žádnou definovanou logickou úroveň a tedy není určen směr přenosu dat přes tento obvod. Z tohoto důvodu by nebylo možné čtení z klávesnice. Tento problém jsem vyřešil drátovou propojkou.

- Obvod komunikačního rozhraní USB je zapojen s předpokladem napájení celého rozhraní z této sběrnice. To ale při výše uvedené spotřebě zařízení není možné, protože toto zapojení je schopno dodávat ze sběrnice USB maximálně 100mA.

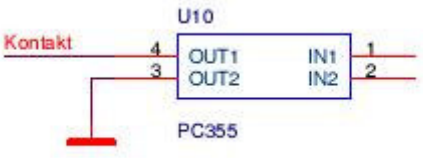
- Spínání celého obvodu pomocí tranzistoru **T3** je ukázáno na obrázku 27:



Obr. 27: Původní zapojení spínaní zařízení

Původní úvaha spočívala v tom, že přes tlačítko START dojde k přímému připojení napájecího napětí až na stabilizátor a tím k oživení celého zařízení. Bohužel toto tlačítko má moc veliký vnitřní odpor a díky velkému úbytku napětí na něm nelze dosáhnout požadovaného vstupního napětí pro správnou funkci stabilizátoru.

- Výstup optočlenu **U10** pro spínání signálu **Kontakt** je připojen proti zemi a ne proti napájecímu napětí, jak ukazuje obrázek 28:



Obr. 28: Původní zapojení optočlenu signálu Kontakt

V tomto zapojení není výstup pro signál Kontakt spínán proti žádnému potenciálu, a nedojde tedy k sepnutí. Nelze tedy indikovat začátek měření.

<u>Změny schématu, které první verze neobsahuje</u>

- Nabíječka akumulátorů v první verzi je zapojena s nabíjecím časem 90 minut. Z tohoto důvodu obvod obsahuje velkou tlumivku na proud 3A.

- První verze neobsahuje konektor JTAG

- První verze neobsahuje filtrační kondenzátory **C13**, **C28**, **C29** a **C30** pro zvýšení odolnosti obvodů TTL proti rušení.

<u>Chyby v návrhu plošného spoje</u>

- Footprint pro obvod **U6** je úzký, ale tento obvod není osazen.

- Footprinty pro obvody **U2** a **U7** jsou úzké, při osazování těchto obvodů bylo zapotřebí přihnout všechny jejich vývody.

- Footprinty pro optočleny **U10** a **U13** jsou malé, při osazování těchto obvodů bylo zapotřebí přihnout všechny jejich vývody.

- Footprinty pro diody **D9** a **D10** jsou malé, při osazování těchto součástek bylo zapotřebí jeden jejich konec nastavit drátkem a ohnout ho pod diody.

- Footprint pro napájecí konektor je špatný, pro osazení bylo zapotřebí jeden vývod nastavit drátkem.

- Footprint pro stabilizátor napětí **U12** není určen pro obvod **LM7805**, ale pro MCP1700. Z tohoto důvodu bylo zapotřebí ohnout vývody tohoto obvodu a přehodit jejich pořadí.

- Footprint pro konektor USB má špatné pořadí vývodů, a proto není připojení USB sběrnice možné.

- Konektor pro připojení LCD displeje je osazen konektorovými kolíčky. Tento konektor neumožňuje kvůli své velikosti montáž displeje do krabičky.

Nyní se dostávám k té části, která popisuje co všechno se v první verzi povedlo a co v ní funguje.

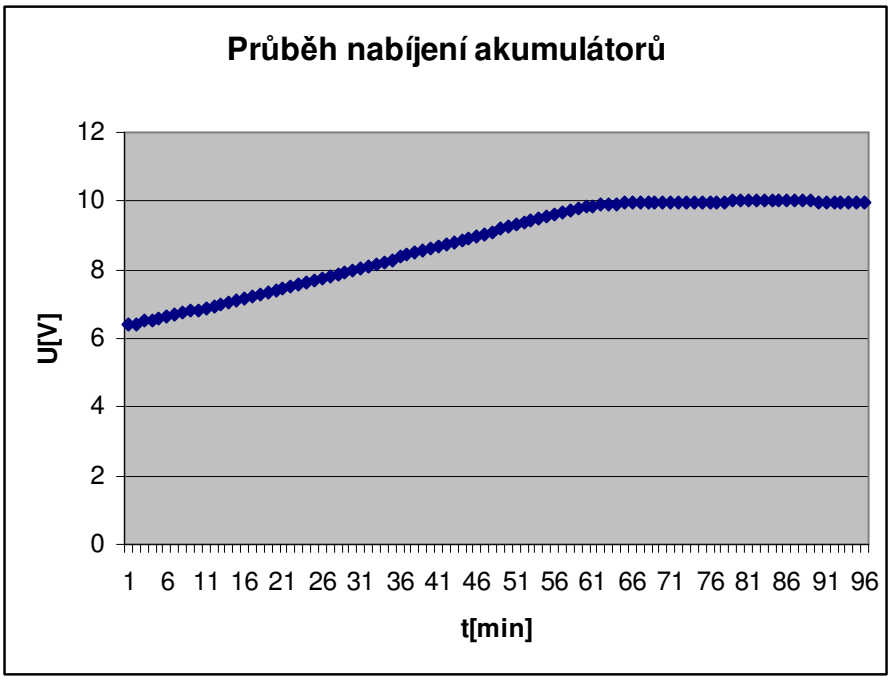
Nabíječka akumulátorů

Nabíječka akumulátorů je nezávislá na všech ostatních částech obvodu. Po osazení a přivedení externího napájecího napětí nabíječka fungovala bez dalšího nastavování. V této verzi je nastaven nabíjecí čas na 90 minut. V tabulce 1 jsou naměřené hodnoty napětí na akumulátorech.

t[min]	U[V]	t[min]	U[V]	t[min]	U[V]
0	6,41	32	8,18	64	9,94
1	6,43	33	8,23	65	9,95
2	6,5	34	8,3	66	9,95
3	6,55	35	8,37	67	9,96
4	6,57	36	8,44	68	9,96
5	6,63	37	8,49	69	9,97
6	6,71	38	8,55	70	9,97
7	6,76	39	8,61	71	9,98
8	6,8	40	8,66	72	9,98
9	6,84	41	8,72	73	9,98
10	6,89	42	8,79	74	9,99
11	6,93	43	8,85	75	9,99
12	6,99	44	8,9	76	9,99
13	7,05	45	8,97	77	9,99
14	7,11	46	9,04	78	10
15	7,17	47	9,11	79	10
16	7,22	48	9,19	80	10
17	7,26	49	9,25	81	10
18	7,32	50	9,31	82	10,01
19	7,37	51	9,37	83	10,01
20	7,43	52	9,42	84	10,01
21	7,51	53	9,48	85	10,01
22	7,55	54	9,54	86	10
23	7,61	55	9,6	87	10
24	7,68	56	9,66	88	10
25	7,73	57	9,71	89	9,99
26	7,8	58	9,79	90	9,99
27	7,86	59	9,82	91	9,99
28	7,91	60	9,86	92	9,99
29	7,99	61	9,89	93	9,98
30	8,06	62	9,91	94	9,98
31	8,12	63	9,93	95	9,98

Tab. 1: Změřené napětí při nabíjení akumulátorů

Akumulátory nebyly zcela vybity, a proto nabíjení neprobíhalo celých 90 minut. Během nabíjení svítí nepřetržitě červená LED dioda **D11**. V 86. minutě došlo k detekci záporné derivace časového průběhu napětí na akumulátorech a nabíjení bylo ukončeno. Od tohoto okamžiku udržuje nabíječka proud baterií asi 50mA viz. (12) a LED dioda bliká. Časový průběh napětí je znázorněn na grafu 1.



Graf 1: Časový průběh napětí při nabíjení akumulátorů

Další obvody

Podařilo se mi oživit některé další obvody. Obvod pro obsluhu klávesnice pracuje přesně podle požadavků. LCD displej funguje včetně ovládání intenzity podsvícení. Dle požadavků pracuje i napěťový watchdog. U ostatních zkoušených obvodů bylo na napájecích vývodech přítomno napájecí napětí.

Při oživování bohužel došlo ke zkratu a procesorem protekl poměrně veliký proud o velikosti několika ampér. Tím byl procesor nejspíše poškozen, protože se mi ho nepodařilo uvést do provozu. Tím bylo znemožněno i testování dalších součástí zařízení, které jsou procesorem ovládány.

Fotografie osazené desky plošného spoje jsou uvedeny v příloze [25].

3.2 Druhá verze

Po nalezení a odstranění všech chyb byla vytvořena druhá verze plošného spoje. Ani v této verzi se mě nepodařilo se zcela vyhnout chybám v návrhu zapojení a návrhu plošného spoje.

V této verzi je footprint konfigurační paměti **U17** malý, ohnutím nožiček obvodu více k sobě se mi ale naštěstí podařilo i tento problém vyřešit a tento obvod osadit.

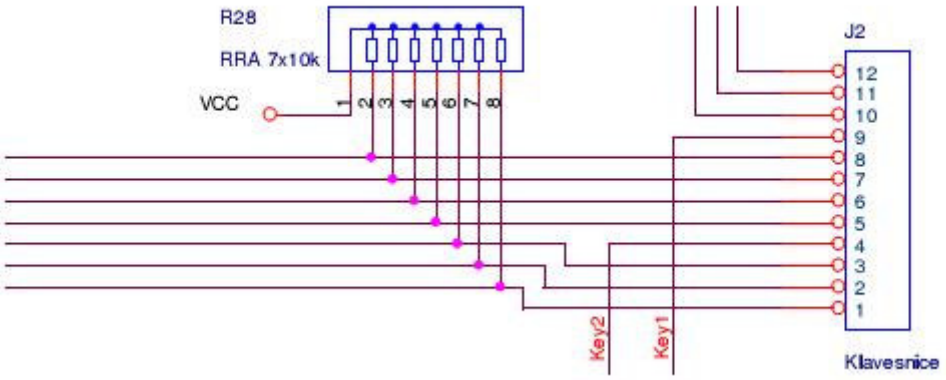
Po připájení všech součástek jsem ještě připevnil stabilizátor napájecího napětí **U12** k chladiči a k desce s plošným spojem pomocí šroubu o průměru 3 mm.

Pak jsem již začal oživovat celé zařízení. To bylo velmi jednoduché, protože celé zapojení obsahuje pouze jeden nastavovací prvek. Tím byl odporový trimr **P1**, kterým jsem nastavil intenzitu svitu LCD displeje. Ostatní obvody nastavování nepotřebují a po přivedení napájecího napětí ihned fungují.

Při oživování jsem přišel i na některé další chyby v samotném návrhu zapojení celého zařízení.

<u>Chyby v návrhu zapojení</u>

- Při čtení dat z klávesnice pomocí výkonového budiče je na jeho vstupech nedefinované napětí 1,4 až 1,6 V, což je velmi blízko rozhodovací úrovně obvodu. Z tohoto důvodu nejsou přesně definována čtená data a provoz klávesnice není možný. Řešením této situace bylo přidání pull-up rezistorů o velikosti 10 kΩ, které jednoznačně určily logickou úroveň všech tlačítek, tak jak to ukazuje obrázek 29.



Obr. 29: Pull-up rezistory ke klávesnici

Odporovou síť jsem umístil z druhé strany konektoru klávesnice. Algoritmus pro obsluhu klávesnice je potom potřeba upravit pro čtení opačných hodnot než je popsáno v kapitole 2.8.

- Připojení vodičů UART0 procesoru k převodníku USB sběrnice je obrácené. Vstup RXD0 procesoru má být připojen k výstupu TXD0 převodníku a naopak. Správné zapojení je obrázku 30.

Obr. 30: Správné zapojení USB

Tento problém jsem vyřešil přerušením vodičů na desce plošného spoje a jejich nahrazením drátovými propojkami.

Nabíječka akumulátorů

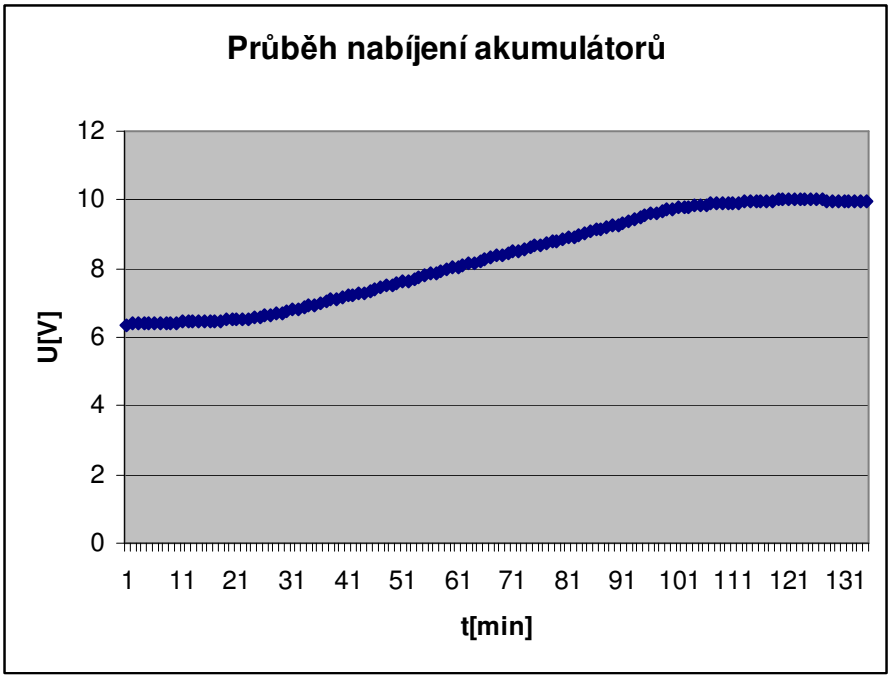
Nabíječka akumulátorů v druhé verzi opět funguje bez problémů a bez dalšího nastavování ihned po osazení. Jelikož tlumivka L1 pro proud 1A je mnohem menší než pro proud 3A, kterou jsem použil v první verzi, bylo ji možno umístit přímo na plošný spoj bez potřeby dalšího přichycení. Právě z důvodu velikosti této tlumivky jsem nastavil v této verzi menší nabíjecí proud prodloužením nabíjecí doby na 132 minut. V tabulce 2 jsou naměřené hodnoty napětí na akumulátorech během nabíjení.

t[min]	U[V]	t[min]	U[V]	t[min]	U[V]
1	6,37	46	7,39	91	9,34
2	6,38	47	7,44	92	9,4
3	6,39	48	7,5	93	9,45
4	6,39	49	7,53	94	9,5
5	6,4	50	7,58	95	9,54
6	6,4	51	7,62	96	9,59
7	6,41	52	7,66	97	9,63
8	6,41	53	7,71	98	9,68
9	6,42	54	7,76	99	9,71
10	6,42	55	7,8	100	9,74
11	6,44	56	7,84	101	9,77
12	6,44	57	7,89	102	9,79
13	6,45	58	7,92	103	9,81
14	6,46	59	7,97	104	9,83
15	6,47	60	8,02	105	9,85
16	6,48	61	8,06	106	9,85
17	6,48	62	8,09	107	9,88
18	6,49	63	8,13	108	9,88
19	6,51	64	8,17	109	9,9
20	6,51	65	8,22	110	9,9
21	6,52	66	8,26	111	9,92
22	6,53	67	8,31	112	9,92
23	6,55	68	8,36	113	9,95
24	6,57	69	8,4	114	9,95
25	6,6	70	8,44	115	9,96
26	6,62	71	8,49	116	9,97
27	6,65	72	8,53	117	9,98
28	6,69	73	8,58	118	9,98
29	6,72	74	8,62	119	10
30	6,76	75	8,66	120	10
31	6,79	76	8,7	121	10,01
32	6,83	77	8,74	122	10,01
33	6,88	78	8,78	123	10,01
34	6,91	79	8,81	124	10

35	6,96	80	8,85	125	10
36	7,01	81	8,89	126	10
37	7,05	82	8,94	127	10
38	7,09	83	8,99	128	9,99
39	7,13	84	9,03	129	9,99
40	7,16	85	9,08	130	9,99
41	7,2	86	9,12	131	9,99
42	7,24	87	9,16	132	9,98
43	7,27	88	9,21	133	9,98
44	7,31	89	9,25	134	9,98
45	7,35	90	9,29	135	9,98

Tab. 2: Změřené napětí při nabíjení akumulátorů

Ani v tomto případě nebyly akumulátory zcela vybity, a tak bylo nabíjení ukončeno dříve než po nastavených 132 minutách. Časový průběh napětí je znázorněn na grafu 2.



Graf 2: Časový průběh napětí při nabíjení akumulátorů

Další obvody

Všechny další obvody jsou po přivedení napájecího napětí funkční a jejich testování probíhalo paralelně s vytvářením firmwaru pro procesor. Celé zařízení funguje i při napájení pouze z baterií.

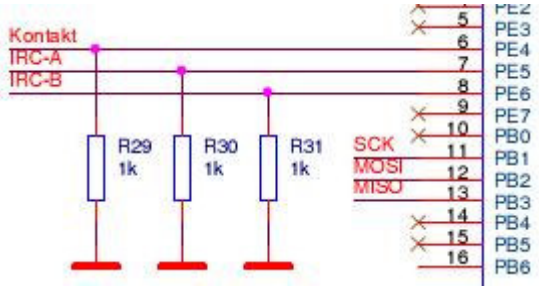
IRC senzor

Po oživení celého plošného spoje jsem připojil IRC senzor a začal testovat celé zařízení dohromady. Bohužel odběr senzoru je příliš veliký a při napájení pouze

z baterií nefunguje. Oživit se mi ho podařilo až po připojení externího napájení. Měření je tedy možné pouze při zapnutém externím napájení nabíječky.

Při testování samotného senzoru se vyskytly další problémy, a to s detekcí jednotlivých přerušení. Nejdříve nebyly přerušení detekovány vůbec. Po analýze problému a měření na desce plošného spoje, jsem zjistil, že plošný spoj obsahuje parazitní kapacity. Ty způsobují , že na vstupech přerušení je napětí asi 2,5V, které zapříčiňuje, že přerušení nejsou detekována.

Tento problém jsem vyřešil připojením odporů o velikosti 1kΩ, které tyto kapacity vybijí. Zapojení odporů je na obrázku 31.



Obr. 31: Zapojení vybíjecích odporů

Po připojení těchto odporů již procesor bez problémů přerušení detekuje. Odpor **R29** se mě díky prokovům podařilo umístit přímo na desku s plošným spojem. Odpory **230** a **R31** jsem potom umístil do krytu konektoru IRC senzoru.

Bohužel tyto parazitní kapacity můžou způsobit, že pokud budou přerušení následovat rychle po sobě, nemusí být detekovány všechny, jelikož se tyto kapacity nestihnou vybít.

Na druhou stranu, pokud budou časy mezi přerušeními dlouhé, může dojít k detekci více přerušení než je jejich skutečný počet, z důvodů přeslechů způsobených právě těmito kapacitami.

Tyto parazitní kapacity vznikly nejspíše nezkušeností při návrhu plošného spoje a rozmísťování jednotlivých součástek.

Tento problém by asi nejlépe vyřešilo zařazení předděliček kmitočtu před vstupy přerušení.

Problém s velkým odběrem senzoru navíc může způsobit, pokles napájecího napětí při rychlém pohybu senzoru. Napěťový watchdog potom restartuje procesor a měření neproběhne.

Navíc je IRC senzor velmi náchylný na vnější rušení, proto musí být on i přívodní kabel odstíněn a uzemněn.

Shrnutí

V této části bylo popsáno oživení a testování obou vytvořených verzí vyvíjeného zařízení. Návrh desky plošného spoje druhé funkční verze je uveden v příloze [26], osazení součástek je potom v příloze [27]. Návrh byl vytvořen pomocí programu OrCad Layout plus, verze 9.2. a je přiložen v příloze [28].

Fotografie osazené desky plošného spoje jsou uvedeny v příloze [29]. Seznam použitých součástek a materiálu je uveden v příloze [30].

4. Montáž

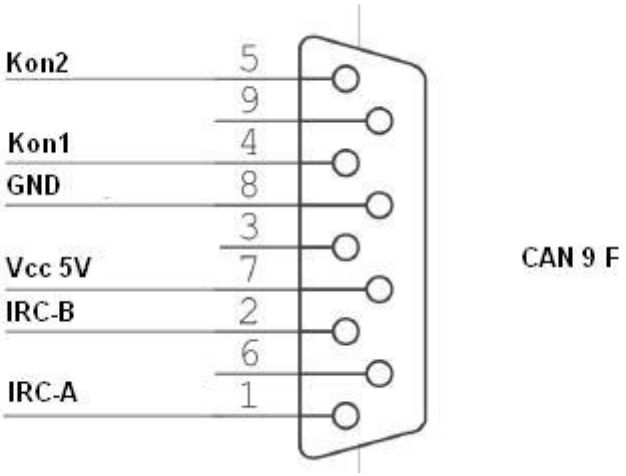
Montáž celého zařízení je díky složení krabičky velmi jednoduchá. Nejprve bylo zapotřebí zkompletovat všechny konektory. Jejich fotografie jsou v příloze [31].

Konektor pro napájení z baterií je tvořen dvěmi vodiči. Dle normy IEC 446 pro značení vodičů značí červený vodič + a modrý -. Pro připojení pouzdra na články je použit napájecí konektor pro 9V baterie 006-PT.

LCD displej je připojen pomocí plochého šestnácti vodičového kabelu, který je zakončen dvouřadými konektorovými kolíčky zahnutými do úhlu 90 stupňů.

Jelikož klávesnice má vlastní přívodní kabel, stačilo ho zakončit dutinkovou lištou o délce 12 vývodů zahnutou do úhlu 90 stupňů.

Poslední je konektor pro připojení IRC senzoru. Ten je proveden pomocí šesti vodičového kabelu na jedné straně zakončeného konektorovými kolíčky zahnutými do úhlu 90 stupňů a na druhé straně konektorem CAN 9 F. Tento konektor jsem zvolil z důvodu jeho spolehlivosti a snadné montáže na bok krabičky pomocí dvou šroubů o průměru 3mm. Popis zapojení tohoto konektoru je na obrázku 32.



Obr. 32: Zapojení IRC konektoru

Všechny konektory jsem ještě doplnil teplem smršťovatelnou bužírkou, aby nemohlo dojít k poškození konektorů při manipulaci.

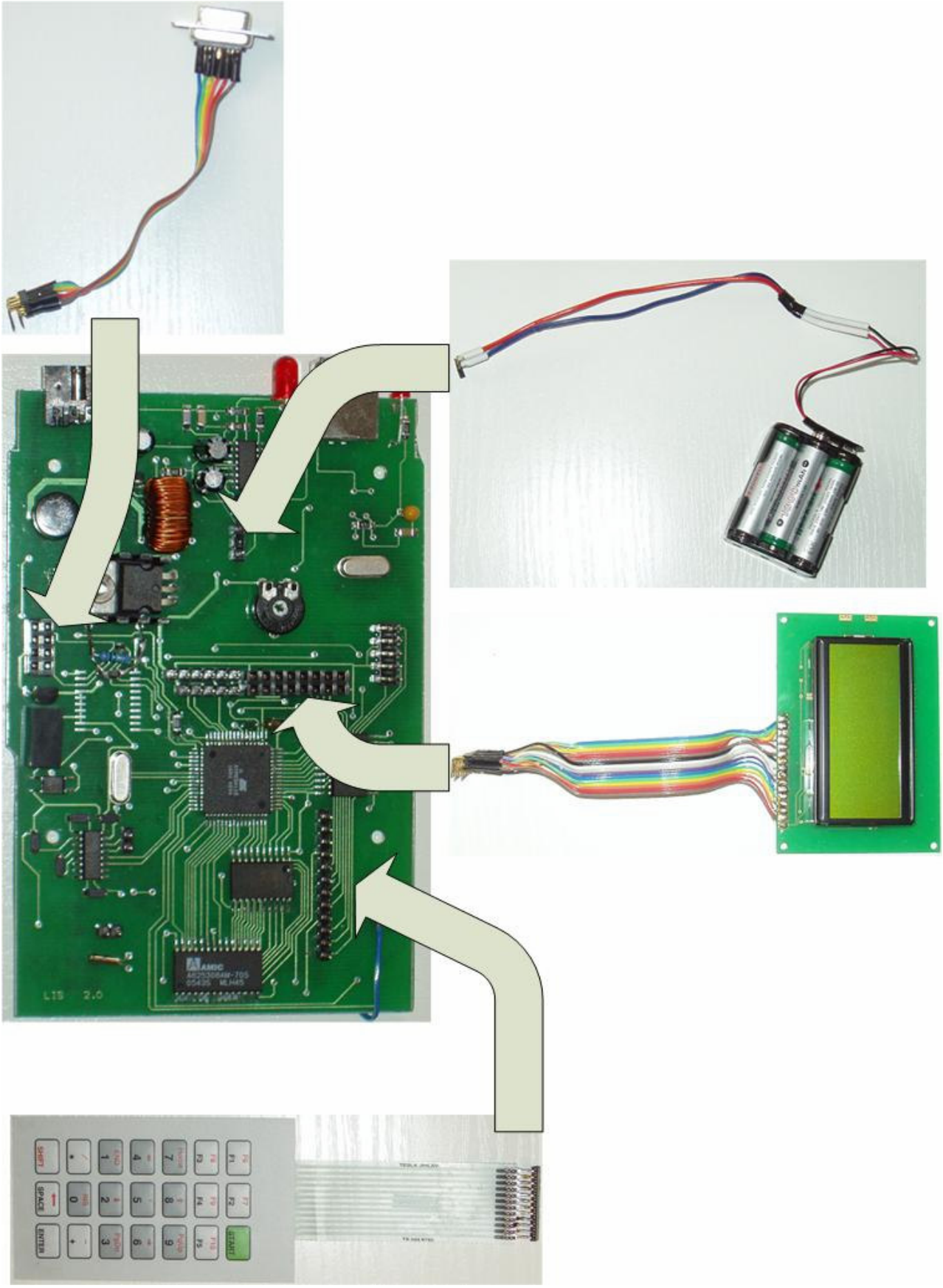
Po kompletaci konektorů bylo zapotřebí udělat několik mechanických úprav na samotné krabičce. Z důvodu oboustranné montáže součástek jsem musel ze spodní části krabičky odvrtat všechny nepotřebné distanční sloupky, aby nepřekážely montáži plošného spoje.

Do přední části krabičky jsem vyfrézoval otvor pro kabel klávesnice. Tvar a rozměry tohoto otvoru jsou uvedeny v příloze [32].

Největší úpravy bylo potřeba udělat na boční části krabičky. Zde bylo potřeba vyvrtat díry na signalizační LED diody a vyfrézovat otvory pro všechny vyvedené konektory. Rozměry všech otvorů jsou uvedeny v příloze [33]. Přední stranu jsem ještě přelepil šablonou s otvory a popisem jednotlivých konektorů. Její vzor je v příloze [34].

Deska s plošným spojem je ke krabičce připevněna pomocí samořezných vrutů o průměru 2,5 mm a délce 12 mm.

Stejnými vruty je připevněn LCD displej k horní části. Na tu je ještě nalepena klávesnice.



Obr. 33: Připojení konektorů

Připojení jednotlivých konektorů je vyobrazeno na obrázku 33. Návod pro skládání krabičky je potom uveden v příloze [1].

Poslední mechanickou částí bylo vytvořit propojovací kabel mezi zařízením a IRC senzorem. Senzor má vlastní konektor, jehož zapojení je uvedeno v příloze [2]. Na druhém konci je konektor CAN 9 M, který je protikusem ke konektoru vyvedeném ze zařízení. Propojení těchto konektorů je pomocí čtyřvodičového kabelu CMSM 4 x 0,75.

Shrnutí

V této kapitole je popsáno kompletování všech konektorů, mechanická úprava všech částí a konečná montáž celého zařízení. Fotodokumentace této montáže je uvedena v příloze [35]. Výkresy mechanických součástí byly vytvořeny v programu Autodesk AutoCAD 2007.

5. Firmware pro procesor

V této kapitole je popsán firmware vytvořený pro procesor. Tento program nahraný do procesoru obsluhuje činnost celého zařízení. Jelikož se mi bohužel nepodařilo uvést do provozu všechny periférie, je program zjednodušený tak, aby se na něm dalo demonstrovat základní funkce celého zařízení. Největším neúspěchem je nefunkčnost klávesnice, kterou se mě přes veškerou snahu nepodařilo rozchodit. Po instalaci odporové sítě jsou sice čtená data definována, ale posuvný registr nefunguje vůbec dle předpokladu a při zmačknutí klávesy dojde k poklesu napájecího napětí a restartu procesoru, tomuto problému se věnuje kapitola 5.7. Bez klávesnice je bohužel znemožněna jakákoliv uživatelská obsluha zařízení a program se tedy zmenšil na samotné měření.

Program funguje tak, že po zapnutí přístroje procesor čeká na příchod signálu Kontakt. Po něm začne samotné měření a po jeho ukončení je na displeji zobrazen výsledek. Na závěr se zařízení samo vypne.

Dílčí úspěch je bezproblémový provoz paměti SRAM. Ta funguje i při nižším napájecím napětí než je uvedeno v příloze [4]. Pokud je připojena záložní baterie BAT2, zůstávají data v paměti uchována i po vypnutí přístroje.

V následujících kapitolách jsou nejprve popsány funkce, které jsou bez problémů fungují a jsou součástí programu procesoru. Dále jsou pak popsány věci, které se mi buď nepodařilo otestovat přímo na procesoru, nebo nejsou použity, protože to obsluha bez klávesnice neumožňuje.

5.1 Inicializace aplikace

Po zapnutí celého zařízení dojde k restartování procesoru. Po zapnutí je zapotřebí provést nastavení některých registrů pro správnou funkci zařízení. To je provedeno ve funkci **AppInit().** Funkce Appinit() je zavolána hned na začátku funkce **Main(),** kterou začíná celý program.

```
void AppInit(void)
{
        DDRG=0xFF;                      /* Nastaveni PORTG jako vystupni */
        SET_BIT(PORTG, 3);              /* Sepni Start */
        lcd_init(LCD_DISP_ON);          /* Inicializace LCD*/
        lcd_clrscr();
        lcd_gotoxy(2, 1);
        lcd_puts("Waiting");
        TCCR2=0x65;                     /* Nastaveni timer2 jako PWM */
        ASSR=0x00;
        TCNT2=0x00;
        OCR2=0x01;
        TIMSK=0x07;                     /* Nastaveni timer1 */
        TCNT1H=0x00;
        TCNT1L=0x00;
        TCCR1A=0x00;
        TCCR1B=0x05;                    /* Start timer1 */

        outp((1<<INT4), EIMSK);         /* Povoleni přerušeni INT4 */
```

```
        outp((1<<ISC40)|(1<<ISC41), EICRB); /* Pouze nabezna hrana */
        sei();                              /* globální povoleni preruseni */

        SET_BIT(PORTG, 4);                  /* Zapni napajeni senzoru */

        return;
}
```

V této funkci je nejdříve nastaven PORT G jako výstupní a hned potom je na jeho třetí výstup nastavena log 1. Tím je zaručeno, že po uvolnění tlačítka START se zařízení samo nevypne. Dále je zinicializován LCD displej. Následně je nastaven TIMER2 jako generátor PWM signálu pro podsvícení displeje a TIMER1 jako čítač. Ten v tomto případě slouží jako časový watchdog. Pokud je zařízení v klidu po dobu asi 40 sekund, dojde k jeho vypnutí, aby se zbytečně nevybíjely baterie. Tato funkce je popsána v následující kapitole.

Na závěr této funkce je povoleno externí přerušení INT4 (Kontakt), aby mohlo dojít ke spuštění měření. A je zapnuto napájení IRC senzoru.

Funkce nemá žádnou návratovou hodnotu ani vstupní parametry.

5.2 Watchdog

Tato funkce slouží k vypnutí zařízení pokud je určitou dobu v klidu. Její princip spočívá přerušení, které nastane po přetečení čítače TIMER1. Vstupní parametr funkce je vektor přerušení od tohoto čítače. Funkce potom počítá počet nastalých přerušení. Po pátém přetečení čítače, které odpovídá času asi 40 sekund dojde, k vypnutí.

```
SIGNAL(TIMER1_OVF_vect)
{
        pom++;                          /* Pricti preteceni */
        if (pom == 5)                   /* Pokud je to pate preteveni */
                CLR_BIT(PORTG,3);       /* pak zarizeni vypni */
        else if (pom < 5)               /* Pokud ne tak */
        {
                TCNT1H=0x00;            /* Vynuluj citac */
                TCNT1L=0x00;
        }
}
```

5.3 Obsluha paměti SRAM

K obsluze paměti sram, a tedy k obsluze ukládání dat z měření jsem si musel vytvořit vlastní funkce. Procesor AtMega128 sice dovede pracovat s externí pamětí o velikosti až 64 kB, ale tato funkce vzhledem k přístupu a k použití paměti není úplně vhodná. Zdrojový kód všech potřebných funkcí je ukázán níže. Doba přístupu k paměti je sice o něco pomalejší, pro použití v běžných aplikacích postačí.

Makra definující připojení paměti:

```
// Define how the ram is wired
#define SRAM_DATA_DDR        DDRA
#define SRAM_DATA_OUT        PORTA
#define SRAM_DATA_IN         PINA
#define SRAM_RD_PORT         PORTG
#define SRAM_RD_PIN          1
#define SRAM_WR_PORT         PORTG
#define SRAM_WR_PIN          0
#define SRAM_ALE_PORT        PORTG
#define SRAM_ALE_PIN         2
#define ADDR_LOW_PORT        DDRA
#define ADDR_HIGH_PORT       DDRC
#define ADDR_LOW             PORTA
#define ADDR_HIGH            PORTC
```

Makra definující základní nastavení řídících signálů (použity pro zpřehlednění kódu):

```
// Macros for SRAM operations
#define SRAM_SET_WRITE() {\
        SET_BIT(SRAM_RD_PORT, SRAM_RD_PIN); \
        CLR_BIT(SRAM_WR_PORT, SRAM_WR_PIN); \
        SRAM_DATA_DDR = 0xFF; }
#define SRAM_COMMIT_WRITE() { \
        SET_BIT(SRAM_WR_PORT, SRAM_WR_PIN); \
        CLR_BIT(SRAM_WR_PORT, SRAM_WR_PIN); }
#define SRAM_SET_READ() { \
        CLR_BIT(SRAM_RD_PORT, SRAM_RD_PIN); \
        SET_BIT(SRAM_WR_PORT, SRAM_WR_PIN); \
        SRAM_DATA_DDR = 0x00; }
#define SRAM_DESELECT() { \
        SET_BIT(SRAM_RD_PORT, SRAM_RD_PIN); \
        SET_BIT(SRAM_WR_PORT, SRAM_WR_PIN); }
#define ALE_LOW() { \
        CLR_BIT(SRAM_ALE_PORT, SRAM_ALE_PIN); }
#define ALE_HIGH() { \
        SET_BIT(SRAM_ALE_PORT, SRAM_ALE_PIN); }
```

Nastavení adresy:

```
void setAddr(tRamAddr addr_low, tRamAddr addr_high)
{
        ALE_HIGH();                 /* latch je transparentni */
        ADDR_LOW_PORT = 0xFF;       /* port pro dolni část adresy jako vystup */
        ADDR_HIGH_PORT = 0xFF;      /* port pro horni část adresy jako vystup */
        ADDR_LOW = addr_low;        /* nastav obe casti adresy */
        ADDR_HIGH = addr_high;
        ALE_LOW();                  /* uchovani dolni casti adresy v latch */
        return; }
```

Funkce pro nastaveni adresy pouze obdrží jako vstupní parametry horní a dolní část adresy. Adresu nastaví na výstup a po nastavení uloží její spodní část do *LATCH*. Funkce nemá žádnou návratovou hodnotu.

Zapis do paměti:

```
void ramWrite(tRamAddr addr, unsigned char data)
{
        float temp1 = fmod(addr, 256);      /* vypocet dolni casti adresy */
        float temp2 = 0;                    /* vypocet horni casti adresy */
        if (addr>256)
                temp2 = floor(addr/256);
        setAddr(temp1, temp2);              /* nastav adresu */
        SRAM_SET_WRITE();                   /* nastav ridici bity pro zapis */
        SRAM_DATA_OUT = data;               /* nastav data na vystup */
        SRAM_COMMIT_WRITE();                /* potvrd zapis */
        SRAM_DESELECT();                    /* uvolni pamet */
        return;
}
```

Tato funkce řeší celý cyklus zápisu jednoho znaku do paměti na určené místo. Funkce má dva vstupní parametry. Prvním je adresa zadané pozice v paměti, na kterou se mají data uložit. Druhý jsou pak samotná data. Data představují jeden byte.

V prvním kroku je ze zadané adresy vypočtena její horní a dolní část. Ty jsou potom nastaveny na výstup. Následuje nastavení řídících bitů pro zápis do paměti a dále samotná data. Na závěr je sběrnice uvolněna. Funkce nemá žádnou návratovou hodnotu.

Čtení z paměti:

```
unsigned char ramRead(tRamAddr addr)
{
        float temp1 = fmod(addr, 256);      /* vypocet dolni casti adresy */
        float temp2 = 0;                    /* vypocet horni casti adresy */
        if (addr>256)
                temp2 = floor(addr/256);
        setAddr(temp1, temp2);              /* nastav adresu */
        SRAM_SET_READ();                    /* nastav ridici bity pro cteni */
        return(SRAM_DATA_IN);               /* přečti data jeko navratovou hodnotu */
        SRAM_DESELECT();                    /* uvolni pamet */
}
```

Tato funkce řeší celý cyklus čtení jednoho znaku z určeného místa v paměti. Funkce má jeden vstupní parametr, a to adresu zadané pozice v paměti.

V prvním kroku je ze zadané adresy vypočtena její horní a dolní část. Tyto části jsou potom nastaveny na výstup. Následuje nastavení řídících bitů pro čtení z paměti. A následně jsou data přečtena a vrácena jako návratová hodnota celé funkce. Funkce vrací vždy jeden byte přečtený z dané dresy. Na závěr je sběrnice uvolněna.

5.4 Procedura měření

Celý kód pro zpracování měřeních dat se skládá z několika funkcí obsluhující přerušení procesoru. Nastavení celého procesoru provede následující funkce:

```
SIGNAL(INT4_vect)
{
        TCCR2=0x00;
        lcd_clrscr();
        lcd_puts("Measurement in\n");
        lcd_puts("progress");
        cli();                          /* Zakaz všechny preruseni */
        outp((0<<INT4), EIMSK);         /* Zakaz int4 */
        outp((1<<INT5)|(1<<INT6), EIMSK);  /* Povol externi preruseni int5,int6 */
        outp((1<<ISC50)|(1<<ISC60), EICRB); /* Nabezna i spadova hrana */

        TIMSK=0x13;                     /* Prepni typ preruseni pro TIMER1  */

        TCNT1H=0x00;                    /* Vynuluj TIMER1 */
        TCNT1L=0x00;
        OCR1AH=0x40;                    /* Nastav porovnavaci hodnotu  */
        OCR1AL=0x00;
        TCCR1B=0x05;                    /* Start TIMER1 */
        sei();                          /* Povol všechny preruseni */
}
```

Tato funkce zinicializuje celé měření. Jako vstupní parametr má vektor od vnějšího přerušení **int4.** Nejprve je vypnut PWM generátor a tím i podsvícení LCD displeje nutné k minimalizaci odběru celého zařízení během měření.

Následně je zakázáno povolení INT4, aby nedocházelo k chybám při opakovaném stisku nebo při zákmitech spínače, a jsou povoleny přerušení **int5** a **int6** od signálů senzoru. Tato přerušení jsou nastavena tak, aby reagovala na náběžnou i spádovou hranu z důvodů popsaných v algoritmu měření.

Potom dojde k nastavení časovače **timer1.** Ten je v tomto okamžiku vynulován. Důležitou součástí je i nastavení povolení přerušení při shodě s porovnávanou hodnotou v registru **OCR1A.**

Na konec je časovač **timer1** zpuštěn**,** a to s předděličkou v poměru 1:1024 vůči hodinovému signálu procesoru. Z toho vyplývá, že procesor bude schopen měřit čas s přesností na 128 µs. To je více než dostačující k požadavku měřit se vzorkovací frekvencí 1kHz. Pokud by se ukázala tato rychlost nedostačující, je možno dělící poměr zmenšit nastavením v registru **TCCR1B.** Ale maximálně tak, aby procesor stihnul ještě ukládat naměřená data do paměti.

Samotné měření a zpracování dat vykonávají funkce obsluhující externí přerušení **int5** a **int6.** Ty zachytávají signály **IRC-A** a **IRC-B** od senzoru.

```
SIGNAL(INT5_vect)
{
        unsigned char time1[2] = {0,0};     /* buffer pro data */
        cli();                              /* Zakaz všechny preruseni */
```

```
        time = (TCNT1/7.8125)+1;        /*  prevod casu na ms /*
        tTime = tTime+time;             /* celkovy cas */
        tDist++;                        /* celkova vzdálenost */
        time1[0] = fmod(time, 256);     /* prevod dat pro ram */
        if (time > 256)
                time1[1] = floor(time/256);
        TCNT1H=0x00;                    /* vynulovani timeru1 */
        TCNT1L=0x00;
  for (int i = 0; i<2; i++)             /* zapsani dat do pameti */
  {
        IncrementAddress();
        ramWrite(addr, time1[i]);
        }
        sei();                          /* Povol všechny preruseni */
}
```

```
SIGNAL(INT6_vect)
{
        /* uplne stejny kod jako SIGNAL(INT5_vect) */
}
```

Při příchodu měřeného signálu, dojde k jednomu z přerušení. Obě přerušení mají identický kód, protože data se zpracovávají stejně, bez ohledu na to, jde-li o přerušení **int5** nebo **int6.** V tomto případě jsem se nevyhnul duplicitě kódu, jelikož těmto funkcím nejde předat jiný parametr než vektor přerušení.

Funkce nejdříve přečte hodnotu z časovače **timer1.** Čas přičte k již změřenému a inkrementuje hodnotu vzdálenosti. Následně je hodnota čítače přepočítána na milisekundy a rozložena na dvoubytovou reprezentaci. Tu už pak jenom jednoduše uloží do paměti. Data jsou do paměti ukládána postupně za sebe se vzrůstající adresou. Adresa musí být globální proměnná, aby zůstala uložená i po ukončení měření a bylo ji možno použít při dalším měření. Jinak by docházelo k přepisování starších měření namísto přidávání za ně.

Proměnná tTime reprezentuje celkový čas měření, a jelikož čas je v milisekundách a uvažujeme měření dlouhé pouze několik milisekund, stačí aby tato proměnná byla typu **INT**.

Proměnná tDist reprezentuje celkový počet přerušení, kterých může být až 80000, musí tato proměnná být typu **LONG INT**.

Poslední funkce spojená s měřením zajišťuje ukončení měření. Jde o vnitřní přerušení procesoru indikující, že **timer1** dosáhl nastavené hodnoty. Toto přerušení značí, že po určitou nastavenou dobu nepřišel již žádný signál od senzoru. Předpokládá se, že po uplynutí této doby je již lis zastaven a měření je ukončeno. S ohledem na maximální dobu měření jsem tuto dobu zvolil asi. 2s.

```
SIGNAL(TIMER1_COMPA_vect)
{
        cli();                          /* Zakaz všechny preruseni */
        outp((0<<INT5)|(0<<INT6), EIMSK); /* zakaz přerušeni in5 a int6 */
        TCCR1B = 0x00;                  /* vypni timer1 */
        for (int i = 0; i<2; i++)       /* zapis poslední vzorek */
```

```
 {
        IncrementAddress();
        ramWrite(addr, 0xFF);
        }
        lcd_clrscr();
        lcd_puts("Measurement\n");
        lcd_puts("complete\n");
        CLR_BIT(PORTG, 4);              /* vypni napajeni senzoru */

        TCCR2=0x65;                     /* spust PWM pro displej */
        _delay_ms(2000);
        char bufferD[10];               /* zobraz vysledek */
        char bufferT[10];
        tDist = tDist*25;
        ltoa(tDist, bufferD, 10);
        lcd_puts(bufferD);
        lcd_puts(" um\n");
        itoa(tTime, bufferT, 10);
        lcd_puts(bufferT);
        lcd_puts(" ms\n");
        for (int i = 0; i<500; i++)
                _delay_ms(200);
        CLR_BIT(PORTG,3);               /* po urcite době se vypni */
}
```

V této funkci dojde k zakázání všech přerušení a zastavení časovače. Do paměti je uložena na konec měření dvojice bytu 0xFF. Tato dvojice značí poslední vzorek měření, jelikož tato hodnota se tam jinak nemůže objevit, protože **timer1** této hodnoty nikdy nedosáhne.

Dále je vypnuto napájení senzoru. V další části převedou výsledné hodnoty měření na řetězec a jsou zobrazeny na displeji. Hodnota proměnné tDist je ještě vynásobena hodnotou 25, protože jedek krok měření reprezentuje právě 25µm.

Na závěr následuje čekací smyčka o délce asi 10 vteřin a pak je zařízení vypnuto.

Tímto jsem popsal celý firmware pro procesor. Tomuto programu schází samozřejmě obsluha periferií. Jelikož se mě nepodařilo zprovoznit klávesnici, tak pro obsluhu paměti EEPROM a pro obvod reálného času komunikující přes sběrnici I2C jsem software nevytvářel. V příloze [37] ale přikládám knihovnu **I2Cmaster,** která tuto sběrnici plně obslouží.

Dále se mě ještě nepodařilo zkomunikovat procesor s počítačem pomocí sběrnice USB. Při testování této komunikace pomocí simulátoru procesoru tato komunikace bez problémů funguje. Její nefunkčnost tedy bude způsobena programem v počítači, proto ji zde uvádím pro další využití.

5.5 Přenos dat

Pro přenos uložených dat do počítače, bylo zapotřebí vytvořit funkce obsluhující obě **uart** komunikační rozhraní procesoru. Rozhraní **uart0** komunikuje přes USB, **uart1** pak přes RS 232. Funkce jsou vytvořeny tak, že přenesou celý

obsah paměti ram do počítače po vybraném rozhraní. Paměť je vždy zkopírována celá, a to především z důvodu jednoduchosti kódu. Zdrojový kód pro přenos dat přes USB je uveden níže:

```
void PrenosUSB(void)
{
        int position = 0;                       /* pocatecni pozice progres baru */
        sei();                                  /* povoleni preruseni, protoze uart knihovna  */
                                                /* je rizena prerusenimi */
        unsigned char data = 0;                 /* data z ram */
        char buffer[3];                         /* buffer pro uart */
        lcd_clrscr();                           /* zobraz zpravu o prenosu */
        lcd_gotoxy(1,1);
        lcd_puts("Probiha prenos");
        lcd_gotoxy(1,2);
        lcd_puts("pomoci USB");
        lcd_gotoxy(position,3);                 /* pocatecni pozice progres baru */
        lcd_putc('O');

        /* inicializace uart0 na danou rychlost */
        uart_init(UART_BAUD_SELECT(UART_BAUD_RATE,F_CPU) );
        for (int i=1; i<SRAM_MAX_ADDR; i++)  /* cyklus přes všechny adresy */
        {                                       /* v pameti */
                if (floor(i/2048) > position)   /* vypocet další pozice pro */
                {                               /* pro progres bar */
                        position++;
                        lcd_gotoxy(position,3);
                        lcd_putc('O');
                }
                data = ramRead(i);              /* přečteni dat z pameti */
                itoa( data, buffer, 10);        /* prevod dat pro prenos */
                uart_puts(buffer);              /* posli data */
                uart_putc('\n');                /* posli oddelovaci znak */
        }
        lcd_clrscr();                           /* zobraz zpravu o dokonceni */
        lcd_gotoxy(0,1);
        lcd_puts("Prenos dokoncen");
        _delay_ms(2000);
        DispMenu1();                            /* navrat na uvodni menu */
        cli();
        return;
}
```

Funkce nejdříve zinicializuje **uart0** na danou přenosovou rychlost. Já jsem použil standardní rychlost 9600 baudů. Potom proběhne cyklus přes všechny adresy v paměti. V každém cyklu jsou z aktuální adresy přečtena data. Ty jsou převedena na řetězec v desítkové reprezentaci a odeslána. Mezi data je vkládán oddělovací znak pro jednoduší zpracování v počítači.

Tuto funkci jsem ještě doplnil o progres bar v posledním řádku displeje, který informuje uživatele o vývoji přenosu.

5.6 Ovládání LCD displeje a UART

Pro obsluhu LCD displeje a sériových linek procesoru jsem v programu použil knihovny funkcí, které jsou volně dostupné na internetu. Proto zde uvádím pouze výčet funkcí, které jsou v těchto knihovnách obsaženy, společně se vstupními a výstupními parametry.

Funkce pro obsluhu LCD displeje jsou v knihovně **lcd.h,** příloha [38].

extern void lcd_init(uint8_t dispAttr);

Popis: Inicializuje displej a vybere druh kurzoru
Vstup: Příkaz pro tvar a druh kurzoru
Výstup: Žádný

extern void lcd_clrscr(void);

Popis: Vymaže displej a nastaví kurzor na začátek
Vstup: Žádný
Výstup: Žádný

extern void lcd_home(void);

Popis: Nastaví kurzor na začátek
Vstup: Žádný
Výstup: Žádný

extern void lcd_gotoxy(uint8_t x, uint8_t y);

Popis: Nastaví kurzor na zadanou pozici
Vstup: Souřadnice požadované pozice
Výstup: Žádný

extern void lcd_putc(char c);

Popis: Zobrazí znak na současnou pozici kurzoru
Vstup: Znak, který má být zobrazen
Výstup: Žádný

extern void lcd_puts(const char *s);

Popis: Zobrazí řetězec bez odřádkování
Vstup: Řetězec, který má být zobrazen
Výstup: Žádný

extern void lcd_puts_p(const char *progmem_s);

Popis: Zobrazí řetězec z programové paměti bez odřádkování
Vstup: Řetězec z programové paměti, který má být zobrazen
Výstup: Žádný

extern void lcd_command(uint8_t cmd);

Popis: Pošle příkaz z instrukční sady displeje
Vstup: Příkaz z instrukční sady displeje
Výstup: Žádný

extern void lcd_data(uint8_t data);

Popis: Pošle byte na displej
Vstup: Byte, který má být poslán
Výstup: Žádný

Funkce pro obsluhu UART0 a UART1 jsou v knihovně **UART.h,** příloha [39].

extern void uart_init(unsigned int baudrate);

Popis: Inicializuje UART0 a nastaví přenosovou rychlost
Vstup: Přenosová rychlost
Výstup: Žádný

extern unsigned int uart_getc(void);

Popis: Vezme přijatý byte z bufferu
Vstup: Žádný
Výstup: V dolním bytu je přijatý znak
V horním bytu je poslední přijatý status

extern void uart_putc(unsigned char data);

Popis: Vloží byte, který má být poslán přes UART0, do bufferu
Vstup: Datový byte, který má být poslán
Výstup: Žádný

extern void uart_puts(const char *s);

Popis: Vloží řetězec, který má být poslán přes UART0, do bufferu
Vstup: Řetězec, který má být poslán
Výstup: Žádný

extern void uart_puts_p(const char *s);

Popis: Vloží řetězec z programové paměti, který má být poslán přes UART0, do bufferu
Vstup: Řetězec z programové paměti, který má být poslán
Výstup: Žádný

Funkce pro obsluhu UART1 jsou stejné, proto uvádím jen jejich seznam.

extern void uart1_init(unsigned int baudrate);

extern unsigned int uart1_getc(void);

extern void uart1_putc(unsigned char data);

extern void uart1_puts(const char *s);

extern void uart1_puts_p(const char *s);

5.7 Obsluha klávesnice

Na závěr této kapitoly uvádím funkci, pomocí které jsem chtěl ovládat celou klávesnici. Jelikož jsem klávesnici neuvedl do provozu, zařazuji ji sem spíše jako inspiraci pro další vývoj.

Tato funkce generuje řídící signály, čte výstup z klávesnice a po jeho zpracování vrací kód zmačknuté klávesy. Algoritmus hardwarové části čtení by již popsán v kapitole 2.8. Kód této funkce je ukázán níže:

```
int keypressed(void)
{
        key = 0x00;                     /* inicializace a vynulovani promenne klaves */
        bool pressed = false;           /* flag zmacknuti klavesy */
        DDRF = 0x00;                    /* port F jako vstupni */
        PORTF = 0x00;
        SET_BIT(PORTD, KEY_SER); /* nastaveni log 1 na seriovy vstup */
        _delay_us(1);
        SET_BIT(PORTD, KEY_SEL);  /* nastaveni log 1 na hodiny */
        CLR_BIT(PORTD, KEY_RD);   /* povolit prenos pres */
                                  /* vykonovy budic sbernice */
        for (int i = 1 ; i < 4 ; i++)   /* smycka pro prepinani sloupcu */
        {
                key = 0x00;             /* opetovne vynulovani pro kazdy sloupec */
                CLR_BIT(PORTD, KEY_SEL);      /* strobe hodiny */
                _delay_ms(10);
                key = PINF;             /*  precteni poru F */
                if(key != 0x00)         /*  porovnani zmacknute klavesy */
                {                       /*  vraci kod klavesy */
                        pressed = true;  /* nastaveni flagu zmacknuti */
                        if(key == 0x01)  /* porovnani a vytvoreni kodu jednotlive klavesy */
                        {       return i*10+1;
                                break;          }
                         if(key == 0x02)
                        {       return i*10+2;
                                break;  }
                        if(key == 0x04)
                        {       return i*10+3;
                                break;  }
                        if(key == 0x08)
                        {       return i*10+4;
                                break;  }
```

```
                    if(key == 0x10)
                    {       return i*10+5;
                            break;  }
                    if(key == 0x20)
                    {       return i*10+6;
                            break;  }
                    if(key == 0x40)
                    {       return i*10+7;
                            break;  }
            }
        CLR_BIT(PORTD, KEY_SER);   /* vynulovani serioveho vstupu pro dalsi radky */
        _delay_us(1);
        SET_BIT(PORTD, KEY_SEL);   }/* strobe hodiny */
SET_BIT(PORTD, KEY_RD);            /* uvolneni zbernice */
CLR_BIT(PORTD, KEY_SEL);
CLR_BIT(PORTD, KEY_SER);
if (!pressed)                      /* vraci 0 pokud nebyla */
    return 0;   }                  /* zmacknuta klavesa */
```

Funkce nemá vstupní parametry. Výstupem je potom celé číslo reprezentující kód zmačknuté klávesy. Kódy jednotlivých kláves jsou uvedeny v tabulce 3.

Klávesa	Kód na klávesnici	Návratová hodnota (dec)	(hex)	(bin)
F1	A/4	14	0x0E	b00001110
F2	B/4	24	0x18	b00011000
F3	A/6	15	0x0F	b00001111
F4	B/6	25	0x19	b00011001
F5	C/6	35	0x23	b00100011
1	A/3	13	0x0D	b00001101
2	B/3	23	0x17	b00010111
3	C/3	33	0x23	b00100001
4	A/8	17	0x11	b00010001
5	B/8	27	0x1B	b00011011
6	C/8	37	0x25	b00100101
7	A/7	16	0x10	b00010000
8	B/7	26	0x1A	b00011010
9	C/7	36	0x24	b00100100
0	B/2	22	0x16	b00010110
*	A/2	12	0x0C	b00001100
+	C/2	32	0x20	b00100000
SHIFT	A/1	11	0x0B	b00001011
SPACE	B/1	21	0x15	b00010101
ENTER	C/1	31	0x1F	b00011111
START	D/5	Nečteno		

Tab. 3: Kódy kláves

Algoritmus nejdříve nastaví všechny řídící signály pro posuvný registr. Poté uvolní čtení přes výkonový budič sběrnice. Následuje smyčka pro přečtení a vyhodnocení. V té je nejprve přečten celý PORT F a uložen do proměnné **key.** Pokud

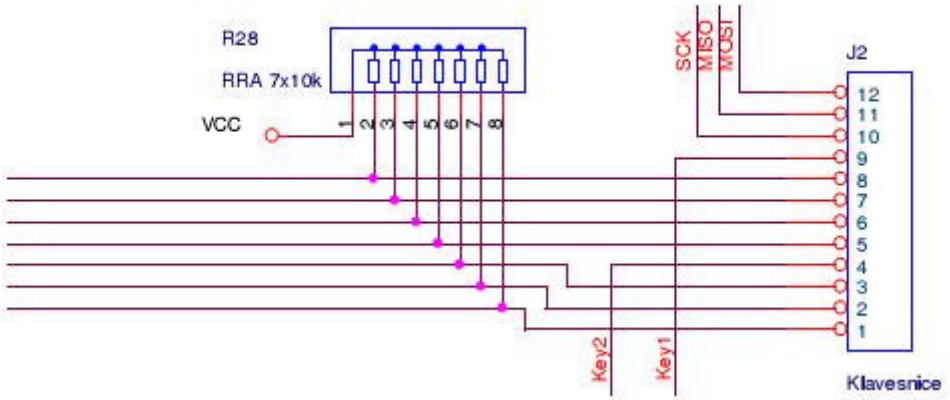
tato proměnná je nenulová, tedy došlo ke stisku některé klávesy, dojde k výpočtu její návratové hodnoty a celá procedura je ukončena. Návratová hodnota je reprezentována dvojciferným číslem. Desítky znamenají číslo sloupce, jednotky potom číslo řádku.

Pokud ke stisku klávesy nedošlo, tak je v dalším cyklu přepnut sloupec na druhý sloupec a celá procedura se opakuje. Pokud FOR cyklus proběhne celý, což znamená, že nedošlo ke zmačknutí klávesy v žádném sloupci, nedojde k nastavení příznakového bitu **pressed,** a funkce vrátí hodnotu nula.

V místech, kde dochází k přepínání řídících signálu, jsou dány zpoždění v řádu jednotek mikrosekund. Jelikož obvody TTL potřebují ke své činnosti délky signálů řádově desítek nanosekund nemělo by při provozu docházet ke kolizím.

Celá procedura čtení trvá maximálně 40 ms, a to v případě, že nedošlo ke stisku klávesy. To je mnohem kratší doba ve srovnání s periodou jednotlivých čtení, která je 200 ms.

Problém s nefunkčností posuvného registru by se dal obejít jeho vynecháním. K přepínání jednotlivých sloupečků klávesnice by bylo možné použít vyvedené výstupy konektoru ISP rozhraní a sloupečky pak přepínat přímo na procesoru. Po upravení konektoru a připojení klávesnice podle obrázku 34 se mě podařilo přepínání sloupečků uvést do provozu.



Obr. 34: Správné přepínání sloupečků klávesnice

Ani po této úpravě ale čtení dat z klávesnice správně nefunguje, protože díky signálu RD, který je společný pro klávesnici i LCD displej, jsou data na této sběrnici nekonzistentní.

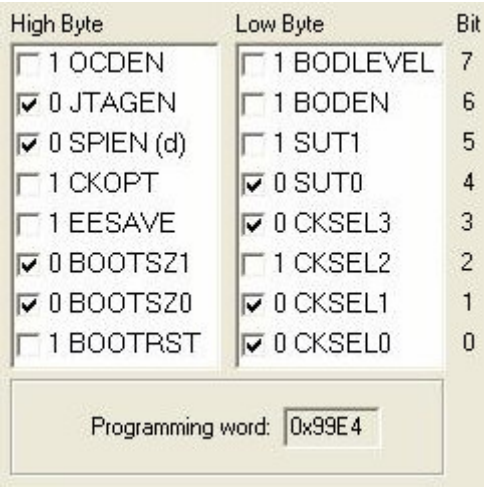
Řešením této situace je použít k ovládání výkonového budiče sběrnice uvolněný signál SER.

5.8 Programování procesoru

Celý program jsem vytvářel v simulátoru VMLab, který je vynikajícím nástrojem pro vývoj programů a simulaci procesorů AVR. Tento simulátor využívá externí překladač WINAVR a umožňuje simulování veškerých komponent procesoru včetně některých periferií. Popis tohoto vývojového prostředí je ale nad rámec této práce.

Samotné naprogramování procesoru jsem potom prováděl přes ISP rozhraní a pomocí programátoru **HWSTK500,** který je velmi rychlý a funguje přímo z VMLabu. Pro bližší informace odkazuji na informace z přílohy [49].

Při programování procesoru je zapotřebí dávat velký pozor na nastavení pojistek procesoru viz příloha [3]. Na obrázku 35 je zobrazeno nastavení pojistek procesoru v této aplikaci.



Obr. 35: Nastavení pojistek procesoru

Shrnutí

V této kapitole jsem popsal vytvořený firmware pro procesor. Jak jsem již uvedl výše, nepodařilo se mě rozchodit některé periférie. Především nefunkčnost klávesnice velmi ovlivnila vývoj tohoto programu. Na druhou stranu se spousta funkcí podařila oživit a zařízení je schopné měření a ukládaní naměřených dat. Celý zdrojový kód je uveden v příloze [40]. Celý program včetně všech knihoven a přeloženého souboru HEX je v příloze [41].

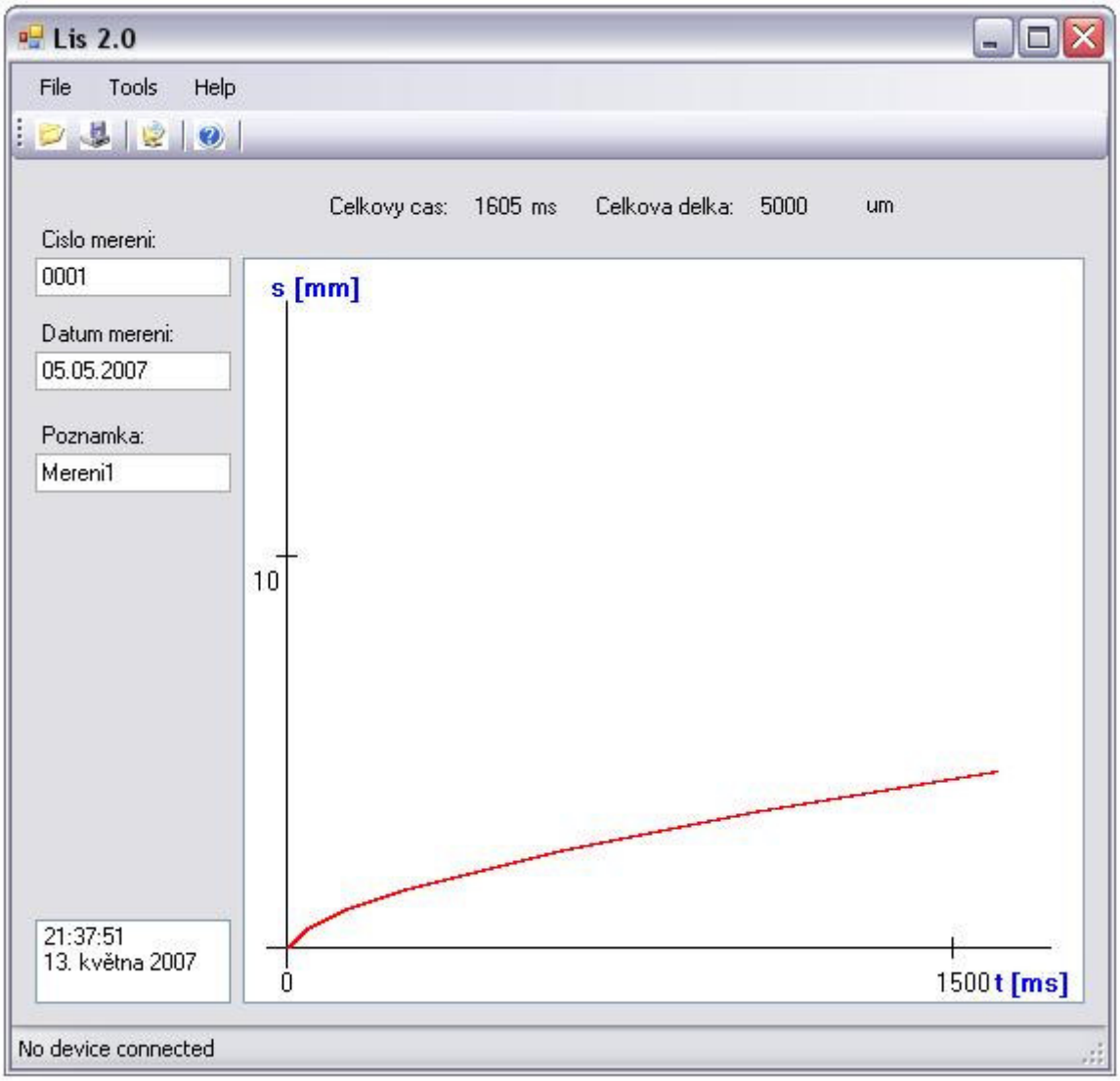
Vývojové prostředí VMLab je volně dostupné na internetu, příloha [46]. Stejně tak je dostupný i překladač jazyka C pro AVR procesory WINAVR, příloha [47].

6. Software pro zpracování dat v počítači

Poslední část vývoje tohoto zařízení obnášelo vytvoření softwaru, který bude schopný přečíst změřená data z paměti zařízení pomocí sběrnice USB a nějakým způsobem je zpracovat.

6.1 Program pro zpracování dat

Program pro zpracování dat jsem se rozhodl vytvořit ve vývojovém prostředí Microsoft Visual Studio 2005 a jazyce C#, protože s tímto jazykem mám největší zkušenosti. Program z načtených dat vypočte čas a délku doběhu lisu a vytvoří graf. Data lze samozřejmě uložit do souboru a později je z něj opět načíst. Vzhled aplikace je na obrázku 36.



Obr. 36: Program pro zpracování dat

Zdrojový kód zpracování dat této aplikace je poměrně jednoduchý a je uveden s celým programem v příloze [42]. Program nejprve načte data do pole. Dráha

doběhu je potom pouhý počet prvků pole vynásobený krokem měření (25µm). Celkový čas doběhu je součtem všech prvků pole.

Jako mnohem větší problém se ukázalo samotné vyčítání dat. Pro tento programovací jazyk bohužel zatím neexistuje odladěná knihovna pro obsluhu funkcí obvodu FTDI a import knihovny z C++ není úplně triviální. Řešením by nejspíše bylo vytvořit tuto aplikaci pomocí C++ builderu, protože tento jazyk již knihovnu funkcí vytvořenou má a její import je jednoduchý pomocí direktivy preprocesoru *#include*.

Program by potom využíval funkcí **FT_READ** a **FT_WRITE**.

Minimální zkušenosti s tímto jazykem a nedostatek času mi už ale nedovolily tuto aplikaci přepracovat do funkční verze.

6.2 Programátor konfiguračních EEPROM sběrnice USB

Jako součást softwarového vybavení počítače jsem vytvořil aplikaci, která umožní do konfigurační paměti EEPROM USB rozhraní nahrát deskriptor připojeného zařízení. To umožní počítači určit a používat správný ovladač pro obvod firmy FTDI. Pokud by deskriptor do paměti nebyl nahrán, byl by použit obecný ovladač a k počítači by bylo možno připojit pouze jedno zařízení využívající obvod FTDI.

Program umožňuje zapsat do paměti tyto údaje:

- Sériové číslo výrobku
- Popis výrobku
- Jméno výrobce
- Identifikaci výrobce
- Vendor ID
- Product ID
- USB standard
- Maximální odběr zařízení ze sběrnice
- Libovolnou poznámku

Sériové číslo umožňuje rozeznání jednotlivých zařízení od sebe v případě připojení více zařízení ve stejnou dobu. Jedná se o libovolný řetězec maximální délky osm znaků.

Popis výrobku je text, který se zobrazí při připojení zařízení k počítači, jak to ukazuje obrázek 37.

Obr. 37: Nalezení zařízení

Jméno výrobce a identifikace výrobce obsahují informace o výrobci zařízení.

VendorID a ProductID jsou hexadecimální čísla, které používá operační systém k nalezení správného ovladače.

!Tyto dvě informace by neměly být měněny, protože operační systém potom nenajde správný ovladač a USB připojení nebude fungovat!

USB standard definuje použitou verzi USB připojení.

Maximální odběr udává jaký proud odebírá zařízení ze sběrnice.

Program jsem z důvodu neúspěchu importu knihovních funkcí pro ovládání obvodu FTDI do C# napsal ve vývojovém prostředí Borland C++ Builder 5. Aplikace využívá knihovních funkcí obsluhujících obvod FTDI a jeho konfigurační paměť EEPROM. Aplikace umožňuje zapsat zadané informace do této paměti, nebo tyto informace z paměti přečíst a následně je modifikovat. Zadané informace je možné uložit pro další použití i do počítače.

K zápisu dat do paměti využívá aplikace funkci **FT_EE_PROGRAM,** jak to ukazuje následující kód.

```
ftStatus = FT_ListDevices( &numdevices, NULL, FT_LIST_NUMBER_ONLY );
.
.
.
ftStatus = FT_Open( 0, &ftHandle );
if ( ftStatus != FT_OK )
        throw Exception("Zařízení se nepodařilo otevřít");
ftStatus = FT_EE_Program( ftHandle, &data );
if ( ftStatus != FT_OK )
        throw Exception("Chyba zápisu");
FT_Close( ftHandle );
```

Aplikace nejdříve zjistí připojené obvody FTDI a vytvoří pro ně handlery. Následně podle vytvořeného handleru otevře požadované zařízení a do paměti uloží zadané informace. Na závěr je handler zařízení zrušen.

Ke čtení dat z paměti využívá aplikace funkci **FT_EE_READ**, jak to ukazuje následující kód.

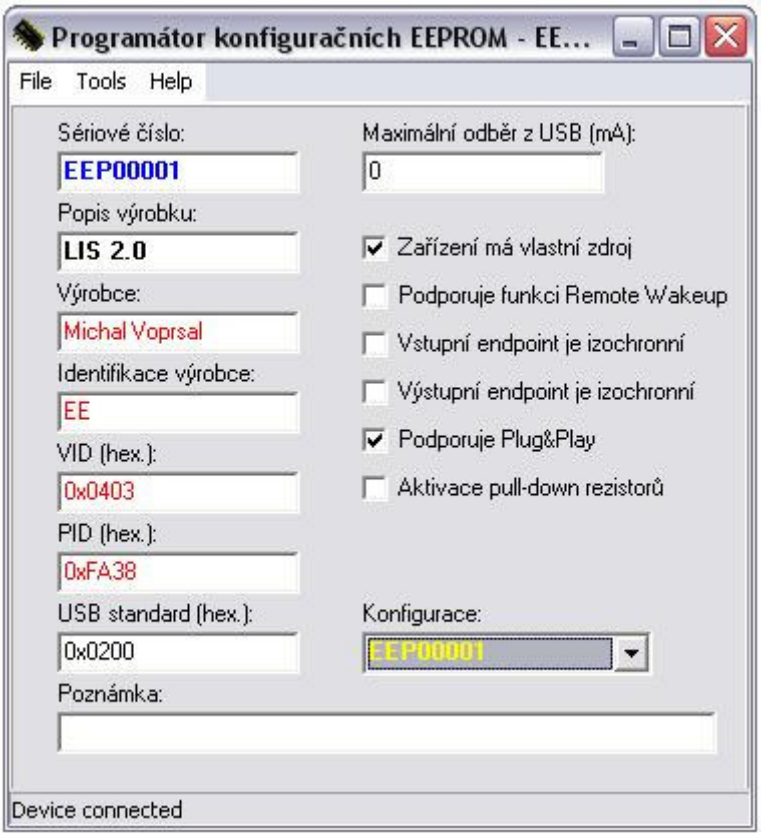
```
ftStatus = FT_ListDevices( &numdevices, NULL, FT_LIST_NUMBER_ONLY );
.
.
.
ftStatus = FT_Open( 0, &ftHandle );
if ( ftStatus != FT_OK )
        throw Exception("Zařízení se nepodařilo otevřít");
ftStatus = FT_EE_Read( ftHandle, &data );
if ( ftStatus != FT_OK )
        throw Exception("Chyba čtení");
FT_Close( ftHandle );
```

Aplikace nejdříve zjistí připojené obvody FTDI a vytvoří pro ně handlery. Následně podle vytvořeného handleru otevře požadované zařízení, z paměti přečte uložené informace a zobrazí je v na obrazovce. Na závěr je handler zařízení zrušen.

Zbytek kódu je triviální a spočívá v podstatě pouze v převodu zadaných hodnot do požadovaného formátu a ošetření možných výjimek. Celá aplikace i se zdrojovým kódem je uvedena v příloze [43].

Knihovní funkce pro obsluhu obvodu FTDI v jazyce C++ jsou v souboru **Ftd2xx.h,** který je v příloze [44].

Vzhled aplikace je ukázán na obrázku 38. Pomocí zaškrtávacích políček lze definovat některé další vlastnosti připojeného zařízení. Pro bližší informace o deskriptoru USB sběrnice odkazuji na přílohu [48].



Obr. 38: Vzhled aplikace

6.3 Ovladač pro obvod FTDI

Aktuální ovladač pro obvody FTDI jsou dostupné na oficiální stránce výrobce, příloha [48]. Použitý ovladač je v příloze [45].

Originální hodnoty pro FTDI Chip jsou VID = 0x0403 a PID = 0x0601. Pokud se do paměti nahrají jiné hodnoty, je zapotřebí upravit soubor **Ftdibus.inf.**

Do tohoto souboru je zapotřebí dopsat řádek se zadanými hodnotami.

%USB\VID_0403&PID_FA38.DeviceDesc%=FtdiBus,USB\VID_0403&PID_FA38

Shrnutí

V této kapitole jsem popsal vytvořený software pro zpracování dat v počítači. Bohužel se mě nepodařilo zprovoznit USB komunikaci a díky tomu nelze vyčítat naměřená data ze zařízení.

V další části je potom popsán software pro programování konfiguračních pamětí EEPROM sběrnice USB. Ten slouží k ukládání deskriptoru zařízení.

7. Měření

Na závěr vývoje celého zařízení bylo ještě zapotřebí otestovat, zda se mi podařilo splnit požadavky kladené na toto zařízení, a to především jestli zařízení správně měří čas a dráhu doběhu pomocí IRC senzoru.

Testování jsem rozdělil do dvou etap. V první etapě jsem zkoušel zda zařízení měří správný čas při pohybu senzoru. Toto testování dopadlo velmi pozitivně. Změřené časy velmi přesně odpovídaly skutečným časům pohybu senzoru. V tabulce 4 jsou uvedeny výsledky z těchto měření:

Číslo měření	Skutečný čas [ms]	Změřený čas [ms]
1	1600	1621
2	2500	2536
3	4000	4006
4	6800	6830
5	9200	9252
6	16400	16410

Tab. 4: Měření času

Jelikož toto zařízení měří s přesností na milisekundy a skutečný čas jsem měřil pomocí stopek, které měří s přesností na desetiny sekund, jsou změřené časy nejspíš přesnější.

Dobu doběhu tedy toto zařízení měří správně a s požadovanou přesností.

V druhé etapě přišlo na řadu testování měření správné vzdálenosti. Správnost měření jsem ověřoval na dvou fixních délkách. V tabulce 5 jsou výsledky tohoto měření:

Číslo měření	Skutečná vzdálenost [mm]	Změřená vzdálenost [mm]	Změřený čas [ms]
1	250	162,3	1726
2	250	189,1	1988
3	250	250,2	2540
4	250	250,5	3112
5	250	364,5	8754
6	1000	680,6	3955
7	1000	892,8	5422
8	1000	986,1	6887
9	1000	1001,3	7546
10	1000	1541,7	15325

Tab. 5: Měření vzdálenosti

Na těchto výsledcích je patrné, že měření vzdálenosti již tak dobře nedopadlo. Při měření vzdálenosti se naplno projevily problémy s parazitními kapacitami popsanými v kapitole 3.2. Velmi záleží na rychlosti vytahování senzoru. Z výsledků vyplývá, že pokud je pohyb příliš rychlý, nejsou detekována všechna přerušení a změřená vzdálenost je menší než skutečná. Pokud je pohyb příliš pomalý, změřená vzdálenost je větší.

Při optimální rychlosti pohybu senzoru jsou výsledky velmi přesné, jak dokazují měření 3,4 a 9.

I při tomto měření byly změřené časy správné.

Shrnutí

V této kapitole byly popsány dosažené výsledky měření času a dráhy pomocí IRC senzoru. Jak jsem popsal v kapitole 2.2.1, je měření času nezávislé na detekci všech náběžných i spádových hran signálů od senzoru, a tedy výsledky tohoto měření jsou správné. Tato závislost společně s parazitními kapacitami na desce s plošným spojem způsobuje, že měření vzdálenosti je velmi závislé na rychlosti pohybu senzoru. Toto měření je pak správné jen při určité optimální rychlosti. Jinak jsou výsledky značně nepřesné.

8. Závěr

V mé diplomové práci jsem se snažil navrhnout, zkonstruovat a oživit zařízení pro měření času a dráhy doběhu lisu. Během práce na tomto projektu jsem vytvořil dvě verze zařízení. Zatímco první verze ještě nefungovala a byla použita pro zjištění chyb v návrhu zapojení, druhá verze je již funkční a umožňuje měření pomocí tohoto zařízení. Bohužel ani v této verzi se mi nepodařilo uvést do provozu některé části tohoto zařízení.

Zhodnocení funkčních částí:

- Procesor zařízení funguje bez nejmenších problémů a je rychle a jednoduše přeprogramovatelný použitím ISP konektoru vyvedeným přímo na desce s plošným spojem.

- Paměť SRAM funguje dle požadavků. Paměť je funkční i při nižším napájecím napětí než uvádí výrobce, což je zapotřebí při provozu z baterií.

- Obvod pro zálohování paměti také pracuje spolehlivě. Pokud je k zařízení připojena záložní baterie, jsou data v paměti uchována i pro vypnutí zařízení.

- Napěťový watchdog funguje správně. Po poklesu napájecího napětí pod stanovenou úroveň dojde k restartu procesoru a odpojení paměti SRAM.

- LCD displej pracuje po nastavení intenzity jasu bez problémů, včetně řízení podsvícení.

- Obvody pro podporu IRC senzoru také pracují přesně dle požadavků. Obvod pro bezpotenciálové spínání měření správně detekuje začátek měření. Obvod pro napájení senzoru dodává dostatek proudu pro jeho provoz. Vzorkovací obvod není využit.

- Nabíječka akumulátorů pracuje správně. Po připojení externího napájení jsou baterie nabity v nastaveném čase. Jedinou nevýhodou tohoto zapojení je, že procesor nezná stav baterií a nemůže tedy uživatele o něm informovat.

- Výsledky měření doběhového času jsou správné a velmi přesné. Přesnost měření je v milisekundách, ale dala by se ještě zvětšit změnou předdělícího poměru časovače **timer1**.

Důležité také je zmínit se o tom, co se nepodařilo oživit, nebo o tom co nefunguje správně. Následující část popisuje nefunkční komponenty a uvádí některé nápady na odstranění závad:

- Volba stabilizátoru napájecího napětí není vhodná. Při napájení pouze z baterií stabilizátor není schopen dodávat požadované napětí 5V. Zařízení sice funguje, ale napětí je příliš nízké pro napájení IRC senzoru. Senzor tedy funguje pouze pokud je přítomno externí napájecí napětí.

Nejlepším řešením tohoto problému by bylo použití nějakého LDO stabilizátoru (např. MCP1700 firmy Microchip). Příliš vysoké externí napájecí napětí by potom bylo zapotřebí snížit zařazením několika diod zapojených do série.

- Sběrnice USB nekomunikuje s procesorem. Po připojení zařízení k počítači je nalezen správný ovladač pomocí dat uložených v konfigurační paměti EEPROM. Zařízení se chová jako připojené. Je vytvořen handler pro jeho obsluhu a konfigurační paměti EEPROM je přístupná. Bohužel se mi ale nepodařilo odeslat žádná data procesoru nebo je z něj přijmout.

- Měření vzdálenosti je velmi nepřesné, to přisuzuji špatné detekci všech hran signálů senzoru. Tento problém je způsoben parazitními kapacitami na desce s plošným spojem.

  Tento problém by nejspíš vyřešil lepší návrh a především zkrácení délky vodičů na desce plošného spoje.

- Poslední nefunkční částí tohoto zařízení je program pro čtení dat z klávesnice. Pří oživování této části se mě nepodařilo uvést do provozu přepínání sloupečků pomocí posuvného registru.

  Tento problém by bylo nejlepší vyřešit vynecháním posuvného registru a přepínat sloupečky přímo pomocí některých volných výstupů procesoru (např. na PORTU B), jak jsem popsal v kapitole 5.7.

  Dalším problémem je samotné čtení dat z klávesnice. Jelikož signál RD je využíván klávesnicí i displejem dochází k ovlivňování a nekonzistenci dat na této sběrnici.

  Nejlepším řešením pro čtení z klávesnice by bylo využít uvolněný signál SER a tím tyto řídící signály od sebe oddělit.

Některé části nebyly ještě testovány. To bylo způsobeno právě nefunkčností klávesnice, které zapříčinila nemožnost se pohybovat v programu pomocí vytvořených menu.

- Program pro sériovou paměť EEPROM a obvod reálného času není napsán, ale jejich začlenění by nemělo být problém, protože funkce pro jejich obsluhu jsou již napsány v knihovně **i2cmaster.h**.

- Nebyla otestována sběrnice RS 232. Její provoz by ale měl být jednoduchý pomocí funkcí pro obsluhu UART1 procesoru.

Závěrečné shrnutí

Výsledkem mé diplomové práce je vytvořené a naprogramované funkční zařízení a dále software pro zpracování dat v počítači.

- Mnoho požadavků na toto zařízení se mi podařilo splnit (např. měření času doběhu, LCD displej, paměť RAM, nabíječka baterií).

- Na druhou stranu pořád ještě zbývá mnoho věcí dodělat (např. USB komunikace, klávesnice, přesnost měření dráhy doběhu). Nefunkčnost těchto částí zařízení je dána především mojí malou zkušeností s vývojem takto rozsáhlého hardwaru.

  Při dalším vývoji softwaru a testování procesoru doporučuji použít rozhraní JTAG, které umožňuje mnohem dokonalejší diagnostiku procesoru.

Celkový komentář k projektu

- Velmi časově náročné
- Velmi zajímavé, především programování procesoru
- Programování procesoru často probíhalo metodou “pokus-omyl“
- Velmi náročné na zkušenosti při vývoji takto rozsáhlého projektu
- Mnoho získaných zkušeností

Poděkování

V závěru této práce bych chtěl velmi moc poděkovat vedoucímu práce panu Ing. Martinu Hlinovskému za všechny dodané podkladové materiály a za podnětné rady během celého vývoje tohoto zařízení.

Dále bych chtěl poděkovat panu Ing. Pavlu Němečkovi za pomoc při osazování plošného spoje a při programování procesoru, a dále pak panu Ing. Pavlu Píšovi za pomoc s ovladačem USB sběrnice.

9. Seznam příloh

[1] – Datasheet ke krabičce BOPLA ARTEB 655
[2] – Datasheet k IRC senzoru PRF08
[3] – Datasheet k procesoru ATmega128
[4] – Datasheet k paměti SRAM HM62256
[5] – Datasheet k LATCH 74HC573
[6] – Datasheet k tranzistoru BC847
[7] – Datasheet k zálohování paměti
[8] – Datasheet ke sběrnici I2C firmy Philips
[9] – Datasheet k paměti EEPROM 24C256
[10] – Datasheet k obvodu reálného času PCF8583
[11] – Datasheet k maticové klávesnici
[12] – Datasheet k posuvnému registru 74LS95
[13] – Datasheet k výkonovému budiči sběrnice 74LS245
[14] – Datasheet k USB převodníku FT232BM
[15] – Datasheet k EEPROM 93LC46B
[16] – Datasheet k převodníku sběrnice RS232 MAX232
[17] – Datasheet k nabíječce akumulátorů MAX713
[18] – Datasheet výkonovému tranzistoru IRFR9024
[19] – Datasheet ke stabilizátoru napětí LM7805
[20] – Datasheet k napěťovému watchdogu MCP120
[21] – Datasheet k tranzistoru BC639
[22] – Datasheet k obvodu 74HC574
[23] – Schéma zapojení celého zařízení
[24] – Soubory projektu pro ORCAD CIS
[25] – Fotografie první vrze zařízení
[26] – Výkresová dokumentace návrhu desky s plošným spojem
[27] – Výkresová dokumentace osazování součástek
[28] – Soubory projektu pro ORCAD Layout
[29] – Fotografie druhé verze zařízení
[30] – Seznam použitého materiálu
[31] – Fotografie všech konektorů
[32] – Výkresová dokumentace pro úpravu přední strany krabičky
[33] – Výkresová dokumentace pro úpravu boční strany krabičky
[34] – Šablona pro přelepení boční strany
[35] – Fotodokumentace montáže celého zařízení
[36] – Výkresová dokumentace pro rozměr desky plošného spoje
[37] – Knihovna funkcí pro obsluhu sběrnice I2C
[38] – Knihovna funkcí pro obsluhu LCD displeje
[39] – Knihovna funkcí pro obsluhu sériových linek procesoru UART0 a UART1
[40] – Zdrojový kód firmwaru pro procesor
[41] – Soubory firmwaru
[42] – Program pro zpracování dat v počítači
[43] – Program pro konfigurační paměti EEPROM USB
[44] – Knihovna funkcí pro obsluhu obvodu FTDI přes USB
[45] – USB ovladač pro obvody FTDI

Internetové odkazy:

[46] – http://www.amctools.com/vmlab.html - Oficiální stránky vývojového prostředí VMLab
[47] – http://winavr.sourceforge.net/ - Oficiální stránky překladače WINAVR
[48] – http://www.beyondlogic.org/usbnutshell/usb5.htm - Stránky věnované popisu deskriptoru sběrnice USB
[48] – http://www.ftdichip.com/ - Oficiální stránky firmy FTDI Chip
[49] – http://www.hw.cz/Produkty/ART974-HW-STK500-programmer.html - Stránky o programátoru HWSTK500
[49] – http://www.avrbeginners.net/ - Základní kurz programování procesorů AVR
[50] – http://www.avrfreaks.net/ - Vše o programování procesorů AVR
[51] – http://www.alldatasheet.com/ - Databáze datasheetů k součástkám
[52] – http://www.hw.cz/ - Server věnovaný elektronice
[53] – http://www.csharp-station.com/ - Server věnovaný programování v C#
[54] – http://www.atmel.com/ - Oficiální stránka výrobce procesoru firmy ATMEL

Použitá literatura:

[55] – Vobecký, J., Záhlava, V.: Elektronika – součástky a obvody, principy a příklady. GRADA Publishing 2001

[56] – Váňa, V.: Mikrokontroléry ATMEL AVR – Programování v jazyce C. Nakladatelství BEN 2003

[57] – Robinson, S.: C# - Programujeme profesionálně. Vydavatelství Computer Press 2003

[58] – Kadlec, V.: Učíme se programovat v C++ Builder a jazyce C++. Vydavatelství Computer Press 2002

[59] – Matoušek, D.: C++ Builder 4.0, 5.0 a 6.0, vývojové prostředí - 1. díl. Nakladatelství BEN 2004

[60] – Matoušek, D.: USB PRAKTICKY s obvody FTDI – 1. díl. Nakladatelství BEN 2003

[61] – Jedlička, P.: Přehled obvodů řady TTL 7400, díl 1. 7400 … 7499. Nakladatelství BEN 1997

[62] – Záhlava, V.:Metodika návrhu plošných spojů. Vydavatelství ČVUT 2002

Poznámka

Všechny uvedené přílohy jsou umístěny na přiloženém CD. Některé přílohy jsou vytištěny a přiloženy v následující kapitole. Ostatní jsou pouze na CD, protože jejich rozsah je nedovoluje vytisknout, nebo se jedná o soubory projektu.