Stredná priemyselná škola elektrotechnická

Námestie SNP č.8, 921 01 Piešťany

STREDOŠKOLSKÁ ODBORNÁ ČINNOSŤ

č. odboru: 12 – Mikroprocesorová, výpočtová, telekomunikačná technika

D.C.S. – Digital Clock System

Riešitelia
Jerguš Lysý
Ročník štúdia: **štvrtý**

2010
Piešťany

Stredná priemyselná škola elektrotechnická

Námestie SNP č.8, 921 01 Piešťany

STREDOŠKOLSKÁ ODBORNÁ ČINNOSŤ

č. odboru: 12 – Mikroprocesorová, výpočtová, telekomunikačná technika

D.C.S. – Digital Clock System

Riešitelia
Jerguš Lysý
Ročník štúdia: **štvrtý**
Konzultant
Ing. Iveta Babičová

2010
Piešťany

Čestné prehlásenie

Čestne prehlasujem, že prácu som vypracoval sám bez cudzej pomoci len na základe vlastných poznatkov a literatúry, ktorá je uvedená na konci práce.

Piešťany, 2. 3. 2010

.....
vlastnoručný podpis

Obsah

1 Úvod

Pred dvoma rokmi, v čase keď som sa učil pracovať s mikroprocesormi, kúpil som si obvod ktorého funkciou bolo posielanie teploty v digitálnom tvare. Práca s týmto obvodom bola však pre mňa zložitá, preto som s tým na čas prestal. Asi o pol roka neskôr ma kamarát požiadal, aby som mu pomohol s vyhotovením automatickej klimatizácie do auta. Žiadal ma pretože vedel, že mám už nejaké skúsenosti s prácou s mikroprocesormi a chcel teda aby som mu jeden na takéto niečo naprogramoval. Našiel som môj starý senzor a začal som podrobnejšie skúmať ako funguje. Po nejakom čase som na systém prišiel a začal som pracovať na obvode, ktorý bol zložený len z mikroprocesora, teplotného senzoru a LED displeja. A práve tu je začiatok mojej súčasnej práce. Vtedy som zhotovil len obvod, ktorý vedel zobrazovať teplotu. Neskôr som však tento obvod rozšíril, tak aby pri určitej teplote zopol relátko. Ovládané to bolo dvomi tlačítkami. Po čase som si uvedomil, že by som túto prácu mohol použiť na praktickú časť maturity z elektrotechniky. Vedel som však, že to nebude stačiť, preto som to rozšíril o pár funkcií. Neskôr mi prišiel do rúk obvod pre real-time zobrazovanie času. Práve tu bol celý zlomový bod, kde som vymenil LED displej za LCD displej a pridal funkciu pre zobrazovanie času. Zariadenie v tom čase vedelo zobrazovať čas, teplotu a to bolo všetko. Čoskoro som sa však inšpiroval jednou prácou, ktorú som našiel na internete a to bolo, zavedenie sériovej komunikácie s PC, kde by som cez Hyperterminál nastavoval čas. Lenže pripadalo mi to trochu obmedzujúco a preto som začal pracovať na konfigurácii času na samotnom zariadení. Neskôr som konfiguráciu rozšíril na rozsiahlejšie menu a keďže hyperterminál a celý konsolový systém mi pripadal tak trochu nudný, rozhodol som sa vytvoriť GUI rozhranie, ktorý mi neskôr poslúžil ako užitočný debugger a starý kábel pre sériovú linku som zamenil za bezdrôtový režim.

V súčasnosti je zariadenie v podobe, kde pre rôzne využitie je treba spraviť rôzne úpravy ako napr. vyvedenie senzoru, softwarové aj mechanické upravenie pinu použitého pri alarme... Výhodou môže byť, že zariadenie môže byť nastavené aj bez fyzického kontaktu so zariadením, napr. pracovníci, ktorí spravujú určité zariadenia a v určitých intervaloch alebo pri určitej teplote, pričom sa interval alebo čas mení, je nutné zariadiť určitú funkciu bez fyzického prístupu. V aktuálnom stave slúži alarm viac-menej ako budík alebo oznamovač.

2 Technický popis

2.1 Základné časti

2.1.1 Riadiaci mikroprocesor

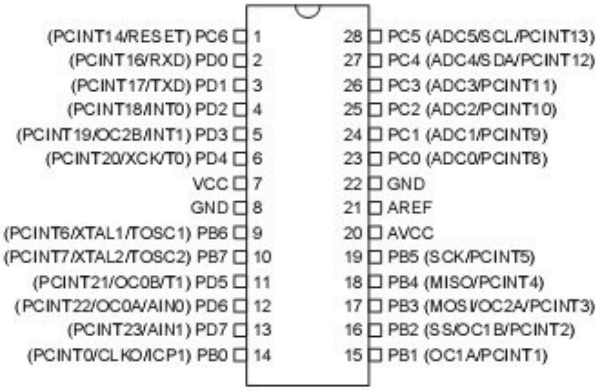
Mikroprocesor – je druh procesora, ktorý je ako celok integrovaný do jediného integrovaného obvodu. Procesor býva súčasťou viacerých elektronických zariadení. Vzhľadom na to, že softvérom (postupnosťou inštrukcií, ktoré procesor spracováva) je možné jednoduchšie realizovať aj veľmi zložité elektronické obvody, a vzhľadom na neustále klesajúce ceny a rastúce možnosti mikroprocesorov sa používajú takmer v každom zložitejšom elektronickom zariadení (rádiá, počítače, tlačiarne, práčky, chladničky, televízory a pod.).

8-bit mikroprocesor – všetky procesory na ktorých som pracoval sú označené ako 8-bit mikroprocesor, čo znamená že pracujú s registrami s veľkosťou 2^8B.

Pôvodne som používal mikroprocesor tiny2313 od firmy Atmel. Tento procesor mal jedinú úlohu a to vypisovať nameranú teplotu na LED displej. Tento procesor je lacný a dobre použiteľný tam, kde sa požaduje nízky odber. Avšak pri rozširovaní som ho musel vyradiť, pretože nemal dostatočnú kapacitu pre program (Flash pamäť) a taktiež je vybavený pomerne malým množstvom pinov. Zvolil som si teda ATmega8, ktorý už mal dostatok pinov a mal ďaleko vyššiu programovú pamäť. Po dlhšom čase sa však ukázalo, že pamäť tohto mikroprocesora nie je natoľko veľká, aby som mohol procesor rozširovať do vyššej miery. Preto som si musel nájsť mikroprocesor, ktorý je kompatibilný s mikroprocesorom ATmega8 a zároveň jeho pamäte sú vyššie. Nakoniec som našiel ideálny mikroprocesor pre mňa – bol to ATmega168. Je pinovo kompatibilný s ATmega8 a jeho programová pamäť je dvojnásobne vyššia.

Mikroprocesor ATmega168

Obr. 1 Puzdro ATmega168



Obr. 2 Popis pinov ATmega168

Základné údaje

Flash pamäť	16kB	8-bit časovač	2
EEPROM dátová pamäť	512B	16-bit časovač	1
SRAM dátová pamäť	512B	Podpora programovania cez SPI port	
Taktovacia frekvencia	0 – 20 MHz	Podpora pre vysoko napäťové	
Napájacie napätie	2,7 – 5,5 V	programovanie (12V)	
I/O piny	23	Full duplex USART	1

Tento mikroprocesor je jadrom celého obvodu a bez neho by nebolo možné, resp. veľmi náročné zabezpečovať správnu funkčnosť zapojenia, preto v ďalšej kapitole bude jeho práca podrobne rozpísaná.

2.1.2 LCD znakový displej

Hlavnou výhodou oproti LED displeju je, že nie som obmedzovaný počtom znakov a nie je treba dodatočné integrované obvody pre obsluhu displeja. Pri výbere LCD displeja som sa zameral hlavne na počet znakov, ktoré možno vypísať. Vybral som si LCD displej BC1602AYPLEH. Pre mňa bolo podstatné, že je dvojriadkový a môže byť podsvietený.

Obr. 3 LCD displej BC1602AYPLEH

Výhodou pre mňa bolo tiež, že má prevedenie, ktoré je prispôsobené pre uchytenie na krabičke. Taktiež má v sebe zabudovaný radič KS0066, ktorý uľahčuje prácu pre komunikáciu s týmto displejom a je kompatibilný s radičom HD44780 – ktorý je dlhodobým štandartom.

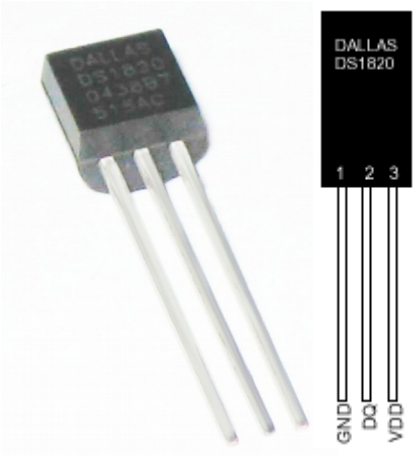
Úloha pinov

Tabuľka 1 úloha pinov LCD

Pin	Symbol	Funkcia
1	Vss	GND (zem)
2	Vdd	+5V
3	Vo	Prispôsobenie kontrastu
4	RS	H/L register select signál
5	R/W	H/L read/write signál
6	E	H -> L enable signál
7 - 14	DB 0~7	H/L dátová zbernica
15	A/Vee	Napájanie pre podsvietenie (+)
16	K	Napájanie pre podsvietenie (-)

2.1.3 Teplotný senzor DS1820

Patrí tiež medzi najdôležitejšie časti, pretože sníma teplotu, ktorú posiela do mikroprocesora v digitálnej podobe a následne je táto teplota zobrazená na LCD displeji. Tento senzor som si vybral hlavne kvôli tomu, že nie je drahý, výstup je v digitálnej podobe a využíva pre komunikáciu **1-Wire** systém (firma Dallas), čo znamená, že pre komunikáciu z hľadiska mikroprocesoru je použitý len jeden pin. Taktiež pri tomto spôsobe komunikácie môže byť prenosový vodič dlhý až 100m bez akýchkoľvek výrazných strát.



GND – zapojenie napájania (-)

Vdd – zapojenie napájania (+)

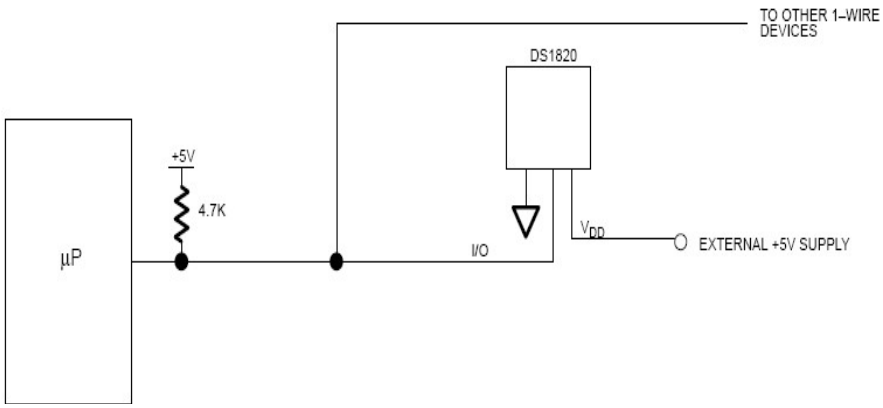
DQ – dátový pin I/O

Obr. 4 DS1820

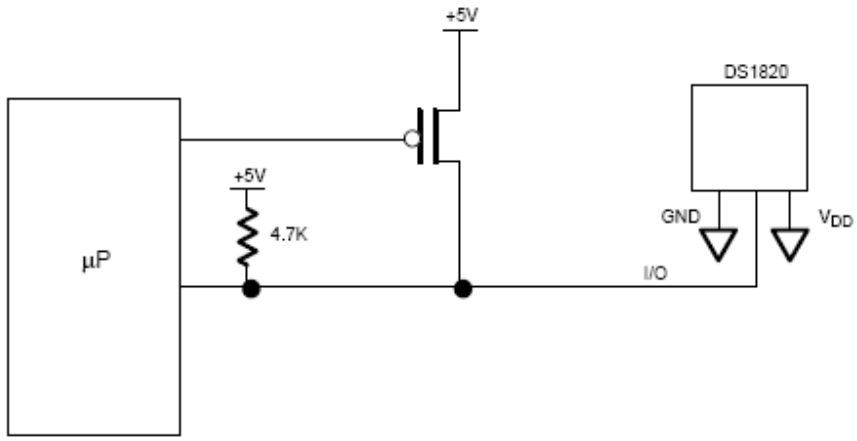
Vlastnosti senzoru

- 1-Wire komunikácia (použitý jeden pin pre komunikáciu)
- Nepožaduje žiadne externé komponenty
- Parazitný mód (napájanie cez dátový pin)
- Teplota je 9-bit hodnota (9. bit označuje kladnú/zápornú teplotu)
- Teplota je konvertovaná do 200 ms
- Ďalšie zariadenia využívajúce 1-Wire komunikáciu môžu byť navzájom prepojené cez jeden dátový pin a cez tento pin komunikovať s mikroprocesorm

Režim zapojenia som použil klasický (obr. 5), tzn. žiadne parazitné zapojenie (obr. 6), ale na pin Vdd je privedených +5V a na pin GND je privedená zem (-).



Obr. 5 *Klasický režim zapojenia DS1820*

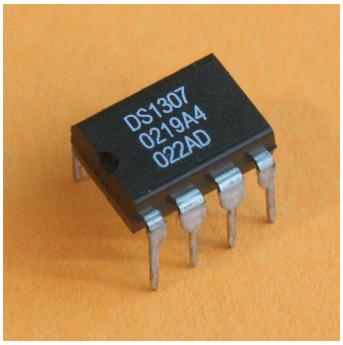


Obr. 6 *Parazitný režim zapojenia DS1820*

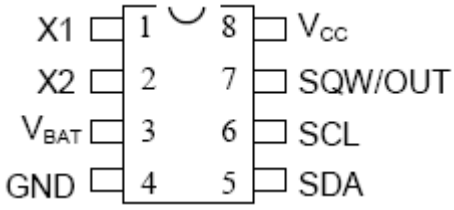
Senzor dokáže merať v od -55°C až do +125°C. Odpor znázornený na obr. 5 a obr. 6 je tzv. pullup odpor, ktorý je dôležitou súčasťou 1-Wire systému. Princíp merania a 1-Wire komunikácia bude podrobnejšie rozpísaný v ďalšej kapitole.

2.1.4 Real-Time Hodiny DS1307

Tieto Real-Time hodiny som si vybral hlavne kvôli tomu, že je od rovnakého výrobcu (Dallas) a komunikácia, ktorú využíva 2-Wire, je zhodná s I^2C. Taktiež uľahčuje prácu mikroprocesoru. Pokiaľ by som tento obvod nepoužil, musel by som v mikroprocesore dodatočne naprogramovať funkcie v časovači, ktoré by nahrádzali tento obvod. Ďalšou výhodou týchto hodín je tiež, že obvod pracuje aj po odpojení napájacieho napätia – tento stav zabezpečuje záložný zdroj.



Obr. 7 Puzdro DS1307



Obr. 8 Popis pinov DS1307

Pre prácu týchto hodín stačí napájacie napätie +5V, 2 - 3,5V záložný zdroj (batéria) a kryštál s frekvenciou 32,768kHz.

Vlastnosti hodín

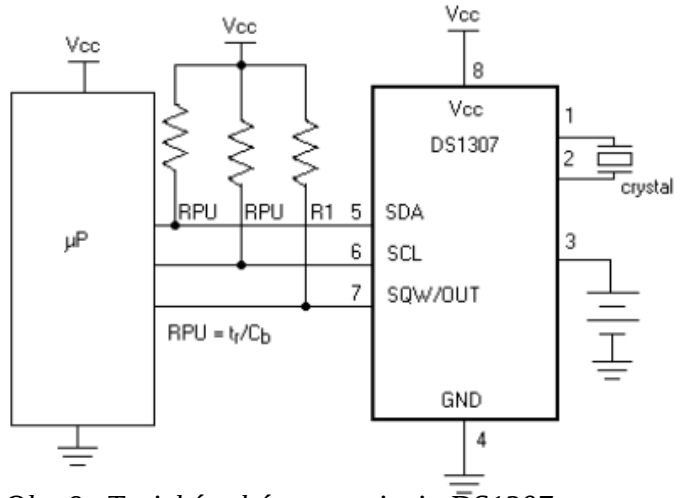
- Dokáže pracovať so sekundami, minútami, hodinami, dátumom, dňom v týždni a prestupnými rokmi
- Dostupná 56B NVRAM pre zálohu dát
- 2-Wire komunikácia (použité 2 piny pre komunikáciu)
- Veľmi nízky odber záložného zdroja (500nA)
- Programovateľný SQW

Úloha pinov

Tabuľka 2 úloha pinov RTC

Pin	Symbol	Funkcia
1,2	X1,X2	Vstup pre kryštál
3	Vbat	Záložne napájanie (+)
4	GND	Napájanie (-)
5	SDA	I/O pin pre 2-Wire
6	SCL	Synchronizácia pre 2-Wire
7	SQW/OUT	SQWE = 1 -> výstup nastavení
8	Vcc	Napájanie (-)

Na obr. 9 je vidieť, že 2-Wire zapojenie je podobné ako v prípade 1-Wire. Princíp komunikácie 2-Wire (I^2C) bude podrobnejšie rozpísaný v ďalšej kapitole.



Obr. 9 Typická schéma zapojenia DS1307

2.1.5 Bluetooth modul BTM-112

Bluetooth - je bezdrôtová komunikačná technológia pracujúca v ISM pásme 2,4 GHz (rovnakom ako u Wi-Fi). Slúži na nadviazanie spojenia medzi dvoma zariadeniami, napríklad mobilným telefónom a osobným počítačom.

Obr. 10 BTM-122

Technológia Bluetooth bola tiež vynájdená ako náhrada sériovej linky. A práve pre tento účel to využívam aj ja. V čase keď som pre komunikáciu s PC používal RS232, nebol som si istý, či pri výmene káblov za BT modul, bude komunikácia prebiehať rovnako bez zmeny. Na moje počudovanie to fungovalo presne rovnako – čo svedčí o tom že BT modul splnil funkciu – nahradil káble za vzduch.

Tento modul som si vybral pretože je lacný (v súčasnosti cca 9€), vysiela aj bez pripojenia externej anténky a funguje okamžite po zapojení bez akýchkoľvek konfigurácii.

Vlastnosti modulu

- Maximálny výkon 4 dBm
- Pracuje s napätím 3.0 – 3.6V
- Podporuje rozhrania USB, UART a PCM
- SPP firmware s AT príkazovou sadou
- Rozmery 25 x 14.5 x 2.2 mm

Naozaj veľkou nevýhodou tohto modulu sú jeho rozmery. Pred použitím je treba modul aplikovať na plošný spoj, keďže samotný modul je v SMD prevedený.

2.1.6 Zoznam ostatných súčiastok

Tabuľka 3 zoznam použitých súčiastok

Názov	**Výrobca**	**Popis**
L78L05ACZ	STMICROELECTRONIC	Stabilizátor +5V
LE33CZ	STMICROELECTRONIC	Stabilizátor +3V
2 x E105	JAMICON	Kondenzátory 1uF (elektrolyt.)
2 x CDC	-	Kondenzátory 22pF (keram.)
5 x R	VITROHM	Odpory 4k7 pre tlačítka
3 x R	VITROHM	Odpory 10R pre spoje
2 x P	PIHER	Trimre 45k pre reguláciu
6 x B3F-4000	OMRON	Mikrotlačitka

2.2 Konštrukcia

Zo začiatku som mal všetky súčiastky zapojené na breadboard-e, kde som aj testoval svoje prvé programy a kde som aj postupne rozširoval schému. V takomto stave som vytvoril aj celý program. Neskôr som však potreboval celý obvod (schéma v prílohe) zaškatuľkovať a preto som začal hľadať vhodnú škatuľku. Rýchlo som si uvedomil, že najprv potrebujem mať zhotovený plošný spoj, tak aby do škatuľky zapasoval na mieru, čiže som potreboval vedieť rozmery škatuľky.

Návrh na plošný spoj som vytvoril v programe Eagle. Tento program je freeware a veľmi mi uľahčil prácu s tvorbou ako aj schémy, tak aj s návrhom. Keďže som mal návrh nakreslený v programe a vytlačený na fólii, nezostávalo mi nič iné ako kúpiť si fotocitlivý plošný spoj a celý návrh presvietiť pomocou UV svetla. Po tomto som dostal krásny obraz z fólie na plošný spoj a po vyleptaní som ho mohol začať osádzať súčiastkami.

Po vyhotovení plošného spoja som si zakúpil škatuľku s ideálnymi rozmermi. Prvý návrh pre úpravu škatuľky bol taký, že displej bude na menšej ploche, tlačitka budú nad nim a celé zariadenie bude v základnej polohe ležať na obsahovo najväčšej ploche. Neskôr som tento návrh prerobil a rozhodol som dať displej do ľavého horného kraja väčšej plochy, s tým že v základnej polohe bude položené na dlhšej hrane a na výšku. Konštrukcia prebiehala nasledovne: Pre displej som na väčšej ploche vytvoril otvor s rozmermi samotného displeja a okolo som vyvŕtal diery na uchytenie. Strana, kde bol priestor pre displej sa stala prednou stranou. Pod neho som vytvoril priestor pre tlacitka a na vrchnej ploche som spravil priestor pre zvyšné tlačitka. Všetky väčšie otvory som vyvŕtal mikrovŕtačkou a pilníkom opracoval na požadovaný tvar. Diery pre skrutky som vyvŕtal vrtákmi s priemerom 2 a 3 mm.

2.2.1 Hlavná doska

Ako som už spomenul, plošný spoj bol vyrobený fotocestou čím som ušetril priestor a mohol ho tak osadiť menšími súčiastkami. Pre integrované obvody som použil pätice a pre batérie príslušné puzdra, hoci neskôr som musel zameniť puzdro na 9V batériu za klasickú prípojku 9V batériu. Dôvodom bolo, že s puzdrom som nedokázal škatuľku zavrieť

a takisto bol problém plošný spoj vyberať. Plošný spoj som ďalej osadil vidlicami do DPS, ktoré prepájajú hlavnú dosku s ostatnými doskami. 20 pinová vidlica je použitá pre pripojenia LCD displeja a taktiež je tadiaľ vyvedený BT. 10 pinová vidlica je použitá pre ďalšie dve dosky, ktoré obsahujú tlačidlá. Na hlavnej doske sú takisto aj potenciometre, ktoré zabezpečujú reguláciu kontrastu na displeji a napätie na alarme, v tomto prípade je to hlasitosť bzučiaku. Celkové napájanie je, ako už bolo spomenuté, z 9V batérie (najprv som uvažoval s 3x 1,5V batériami, lenže potom pre ne nebolo dostatok miesta) alebo z adaptéru. Napätie je cez stabilizátory stabilizované na 3V a 5V. 3V sú použité pre BT modul (vyžaduje 3-3,6V) a ďalších 5V je pre ostatné obvody. Za jeden aj druhý stabilizátor som dal elektrolytický kondenzátor s kapacitou 1uF. Taktiež pred obidva integrované obvody som umiestnil keramický kondenzátor s kapacitou 22pF. Všetky 2k2 odpory som použil pri tlačidlách, 4k7 odpory sú použité pri komunikácii 1-Wire a 2-Wire a zvyšné 10R odpory v podstate slúžia iba ako spoje – chcel som zabrániť voči ťahaniu akýchkoľvek káblikov po plošnom spoji. Na hlavnej doske je taktiež vyvedený bzučiak.
Obrázok hlavnej dosky je v prílohe.

2.2.2 Ostatné dosky

Okrem hlavnej dosky je v škatuľke taktiež doska s BT modulom a dve dosky s tlačidlami. Doska s BT modulom je k hlavnej doske pripevnená skrutkami. Ako som spomenul vyššie BTM-112 je v SMD prevedení, čiže bola nutnosť ho najprv napájkovať a potom aplikovať – preto je na samostatnej doske. Z mikroprocesoru sú vyvedené Tx a Rx piny na kolíky, ktoré sú pripojené do príslušných pinov na moduli. Rovnakým spôsobom je pripojené napájanie.

Nevedel som si predstaviť žiadny spôsob akým by som tlačidlá prichytil na škatuľke a pri tom ich nepridržal na samostatnom plošnom spoji, preto som všetky tlačidlá pripevnil na plošné spoje a skrutkami pripevnil do otvorov na škatuľke. Tlačidiel je dohromady šesť, dve na jednej doske a štyri na druhej doske. Obidve dosky sú navzájom prepojené. S menšej dosky sú káble ťahané do väčšej a z tej ďalej do vidlice na hlavnej doske.
Ďalšie dosky sú v prílohe

2.2.3 Zdroj

Ako napájanie som si zvoli 9V batériu, ktorá je stabilizovaná na 5V. Avšak vyskytol sa problém pri výdrži tejto batérii. Keďže odber je príliš vysoký a batéria veľa nevydrží, dorobil som prepínač s ktorým môžem prepínať medzi stavmi: napájanie – batéria, vypnutý a napájanie – adaptér. Zapojenie je nasledujúce: na prepínač je pripojená zem (-) s batérie aj s napájacieho konektoru. Tento pin na prepínači je teda spoločný pri všetkých stavoch a zároveň je vyvedený do – na hlavnej doske. Ďalej je s batérie aj s konektoru vyvedené + napájanie a ten je zapojený do prepínača v závislosti od toho, na ktorej polohe má byť napájanie – adaptér a napájanie – batéria.

3 Software/Hardware

3.1 ATmega168

Na obr. 2 vidíme popis pinov, teraz popíšem funkciu využívaných pinov:
PC6 – Reset, pôvodne som chcel tento pin využiť pre výstup na LCD displej ako podsvietenie no ukázalo sa že to nebol dobrý nápad, keďže je tento pin používaný pri programovaní a po odstavení by som tým mikroprocesor odstavil.
PD0 – RxD, využívaný pri komunikácii s PC (čítanie)
PD1 – TxD, využívaný pri komunikácii s PC (zápis)
PD2 – je výstup na podsvietenie LCD
PD4 – je vstup pre PD2 (tzn. svieť/nesvieť)
PB0 – PB7 – využívaný na LCD display, 3 piny pre riadenie a 4 piny pre prenos dát
PD5, PD6 – komunikácia s RTC
PD7 – komunikácia s teplotným senzorom
PC3 – výstup pre bzučiak
PC0,1,2,4,5 – vstup pre tlačidla

3.2 ISP programovanie

ISP (in system programming) rozhranie je jedno z najpoužívanejších a najjednoduchších spôsobov programovania mikroprocesorov. Preto aj výrobcovia jedno čipových mikroprocesorov dnes čoraz častejšie integrujú do svojich produktov rozhranie ISP. Ide

o rozhranie umožňujúce programovať interné pamäte mikrokontroléru priamo v cieľovej aplikácii. Tento prístup výrazne zrýchľuje proces vývoja, či oživovania aplikácií, pretože obchádza nutnosť vyberať mikroprocesory z aplikácie a vkladať ich do špecializovaného programátora. Takisto sa znižuje riziko poškodenia obvodu vplyvom nesprávnej manipulácie, či mechanického poškodenia aplikácie.

ISP rozhranie MCU je postavené na komunikácii prostredníctvom zbernice SPI (Serial Peripheral Interface).Štruktúra tejto zbernice obsahuje 4 vodiče, z ktorých sa pri ISP uplatňujú iba 3. Ďalej je potrebné ovládať signál RESET mikrokontroléru, teda na kompletné programovacie rozhranie sú potrebné 4 vodiče. Pre tento účel môže ako programovacie rozhranie MCU slúžiť paralelný port PC.

Jednoduchý ISP „programátor“ (píšem úvodzovky, pretože sa nejedná o skutočný programátor, ale iba akýsi interface medzi PC a mikroprocesorom) som si kedysi zložil aj ja. Tento programátor mi slúži doteraz a napaľoval som s ním všetky doteraz vyrobené programy. Pre napaľovanie používam program PonyProg, ktorý je voľne dostupný, prehľadný a podporuje pomerne veľa typov mikroprocesorov. Zdrojový kód píšem, kompilujem v programe AVRstudio v jazyku C. Tento program je taktiež freeware a je od firmy Atmel, čiže priamo určený pre programovanie AVR mikroprocesorov.

3.3 Hlavný súbor - program

Ako som už spomínal jadrom celého obvodu je mikroprocesor, resp. program ktorý je v tomto mikroprocesore napálený. Samotný program obsahuje niekoľko súborov, ktoré sú rozdelené na súbory s obsahom funkcií a hlavičkové funkcie ktoré prepájajú tieto funkcie s hlavným súborom. V hlavnom súbore sa nachádza funkcia *main*, ktorá je základnou funkciou a ktorá je vykonávaná hneď po napájaní mikroprocesora. Pre túto funkciu je typická slučka (cyklus), ktorá zabezpečuje nepretržitý chod programu. Zdrojový kód hlavného súboru je takisto kompilovaný do súboru *.hex* (súbor s obsahom, ktorému mikroprocesor rozumie), čo znamená, že tento súbor by mal obsahovať includy (pričlenenia) jednotlivých súborov, ktorých funkcie majú byť volané (používané). Hlavný súbor by mal obsahovať hneď na začiatku :

include <avr\io.h>

čo znamená, že kompilátor vie, kde má hľadať funkcie či konštanty o ktoré ho žiadame k danému mikroprocesoru.

V mojom prípade sú k hlavnému súboru pričlenené hlavičkové súbory pre obsluhu jednotlivých obvodov (senzor, RTC,..), pre samotný mikroprocesor a pre funkcie určené výhradne pre moje zariadenie (napr. menu). Hlavný súbor tiež obsahuje funkcie časovačov a prerušení. Po spustení hlavnej funkcie nastane inicializácia jednotlivých zariadení, konkrétne displeja a sériovej linky a taktiež nastavenie časovačov. Celkom využívam dva časovače, ktorých úlohou je zabezpečiť paralelný chod programu, teda snímanie teploty počas konfigurácie, taktiež samotná komunikácia cez BT počas akéhokoľvek stavu. Taktiež obsahuje funkciu vykonanú počas prerušenia USART-om, tj. prichádzajúci bajt od PC. Inicializácia vyzerá nasledovne:

```
nastav_casovac0(0, 4, 0, 1);
nastav_casovac2(0, 7, 0, 1);
USART_Init();
LCDinit();
LCDcursorOFF();
CHbit_nuluj();
DDRD &= ~0x10;
DDRD |= 4;
DDRC = 0x8;
PORTC &= ~0x8;
```

Počas inicializácie taktiež načítavam údaje z E^2PROM, kde sú uložené maximá a minimá nameranej teploty a k tomu príslušný čas. Po inicializácii nasleduje príkaz:

```
asm("sei");
```

ktorý povoľuje prerušenia. Bez toho príkazu by sa nevykonávali funkcie časovačov. Po inicializácii nasleduje hlavná slučka, ktorá má za úlohu vypisovať počiatočný stav zariadenia, t.j. čas, dátum, deň, teplotu. V tejto slučke takisto testujem, či je stlačené tlačidlo. Ak je stlačené, zavolá sa príslušná funkcia, ktorá bude vykonaná, pričom táto funkcia môže obsahovať ďalšiu slučku, ktorá bude znovu čakať na „povel zvonka“. Hlavná slučka vyzerá takto:

```
while (1) {
   display1();
        if (bit_is_set(PINC, 0))
```

```
        konfig_hodiny();
      if (bit_is_set(PINC, 1)) {
        if (udaje.mon_temp == 1)
            zobraz_mon_temp();
      }
}
```

Tu môžeme vidieť podmienky, ktoré kontrolujú stav pinov, na ktorých je zapojené príslušné tlačidlo. Z tohto je jasné, že mikroprocesor sám o sebe robí len to, že čaká na to, kým nastane nejaká zmena na pine a kým nastane prerušenie.

3.4 Obsluha RTC

Pre komunikáciu s obvodom som si mohol vybrať medzi vytvorením vlastných funkcií, alebo použitím tých, ktoré ma v sebe implementované mikroprocesor. No keďže ma pred vstavanými funkciami varovali (občasné výpadky,...), napísal som si vlastné funkcie. Ako som spomínal obvod DS1307 využíva pre komunikáciu 2-Wire, no v skutočnosti je to totožné s I^2C. Zariadenia pri komunikácii vystupujú buď ako master, alebo slave. V mojom prípade je mikroprocesor master a DS1307 je slave.

Pri zápise master na slave sú dodržané tieto kroky:

- Master generuje START signál na zbernici
- Master odošle byte s adresou slave
- Master odošle byte na slave
- Master generuje STOP signál na zbernici

Pri čítani master zo slave su dodržané tieto kroky:

- Master generuje START signál na zbernici
- Master odošle byte s adresou slave
- Master generuje START signál na zbernici
- Master odošle byte s adresou slave a nastavením nultého bitu (príznak čítania)
- Master číta byte zo slave
- Master generuje STOP signál na zbernici

Princíp START signálu

START signál môže generovať iba master, preto je úlohou mikroprocesoru tento signál generovať. Tento signál je potrebný pred zahájením prenosu dát medzi master a slave. Je generovaný nasledujúcim spôsobom: master uvoľní zbernicu (SDA aj SCL), čím spôsobí, že pullup odpory zdvihnú log. uroveň z 0 na 1.

i2c_DDR &= ~(1 << SCL);

i2c_DDR &= ~(1 << SDA);

Po tomto nastavíme na zbernici SDA log. 0, zatiaľ čo SCL ponecháme na log. 1. Toto pripraví slave na komunikáciu.

i2c_DDR |= (1 << SDA);

i2c_PORT &= ~(1 << SDA);

Princíp STOP signálu

STOP signál je generovaný pred ukončením komunikácie a je takisto generovaný master. Generuje sa nasledovne: master nastaví na zbernici (SDA a SCL) log. 0.

i2c_DDR |= (1 << SDA);

i2c_PORT &= ~(1 << SDA);

i2c_DDR |= (1 << SCL);

i2c_PORT &= ~(1 << SCL);

Po tomto uvoľníme zbernicu SCL, čim dostaneme log. 1. Krátko potom uvoľníme aj zbernicu SDA, čím dostaneme takisto log. 1.

i2c_DDR &= ~(1 << SCL);

i2c_DDR &= ~(1 << SDA);

Princíp zápisu

Pri zápise sú dáta prenášané tak, že najvýznamnejší bit je prenesený prvý. Pre prenos jedného bajtu je potrebné vytvoriť cyklus, ktorý je opakovaný 8x – pre každý bit. Princíp zápisu je nasledovný: ak prenášaný bit je 1, nastavíme zbernicu SDA na log. 0, ak bit je 0 zbernicu uvoľníme, tj. log 1.

```
if ((data & 0x80) == 0) {
  i2c_DDR |= (1 << SDA);
```

```
  i2c_PORT &= ~(1 << SDA);
}else
  i2c_DDR &= ~(1 << SDA);
```

Ďalej uvoľníme zbernicu SCL. Po chvíli znovu nastavíme zbernicu SCL na log. 0.

```
i2c_DDR &= ~(1 << SCL);
i2c_DDR |= (1 << SCL);
i2c_PORT &= ~(1 << SCL);
```

Tento cyklus opakujeme 8x pre odoslanie 1 bajtu. Nakoniec čítame ack bit. Ack bit čítame tak, že na konci slučky uvoľníme zbernicu SDA aj SCL, počkáme a čítame stav na zbernici SDA, ak je na zbernici log 0 master prijal ack a prenos bol korektný v prípade že zbernica ostane na log. 1 master vie, že sa vyskytla chyba pri komunikácii.

Princíp čítania

Pri čítaní sú dáta taktiež tak, že najvýznamnejší bit je čítaný prvý. Pred čítaním najprv posielame požiadavku na čítanie. Princíp čítania je: uvoľníme zbernicu SCL, po chvíli zisťujeme stav na zbernici SDA, ak je log. 0, dostávame od slave bit 0 v druhom prípade bit 1. Po tomto nastavíme SCL na log. 0.

```
i2c_DDR &= ~(1 << SCL);
  while (bit_is_clear(i2c_PIN, SCL))
    ;
_delay_us(5);
if (bit_is_clear(i2c_PIN, SDA))
  data |= 0;
  else data |= 1;
  _delay_us(10);
  i2c_DDR |= (1 << SCL);
  i2c_PORT &= ~(1 << SCL);
  _delay_us(10);
```

Nakoniec vyšleme ack bit. Ack pošleme tak, že uvolníme zbernicu SDA, uvolníme zbernicu SCL, počkáme, nastavíme zbernicu SCL na log. 0.

Komplexný príklad je v prílohe – súbor *ds1307.c* medzi zdrojovými kódmi.

3.5 Obsluha senzoru DS1820

DS1820 využíva na komunikáciu 1-Wire princíp s použitím jedného pullup odporu. Tento štýl komunikácie je veľmi výhodný, pretože nám stačí iba jeden pin na mikroprocesore – prenos je sériový.

Komunikácia medzi mikroprocesorom (master) a pripojeným zariadením (slave) pracuje prebieha v režime polovičného duplexu. Dáta na zbernici sú prenášané v časových slotoch.

Pre načítanie teploty je postup:

- Master vyšle RESET impulz
- Master odošle príkaz na slave pre konvertovanie teploty
- Master čaká kým slave skonvertuje teplotu
- Master vyšle RESET impulz
- Master odošle na slave požiadavku pre čítanie teploty
- Master načíta 9-bit, pričom 9-bit je príznak znamienka

RESET impulz

Komunikácia začína vždy, keď master vyšle RESET impulz. Master nastaví na zbernici log. 0 a drží to tak minimálne 280us. Po tomto čase uvoľní zbernicu do log. 1 pullup odpor. Ak po čase 40us je na zbernici log 0, znamená to, že slave sa našiel a tento stav vydrží 60 až 240us.

```
DDR |= DQ;
PORT &= ~DQ;
_delay_us(500);
DDR &= ~DQ;
_delay_us(40);
r = (DDR & DQ) ? 0 : 1;
_delay_us(240);
return r;
```

Po tomto je slave schopný komunikovať s master.

Princíp zápisu

Pri zápise je prvý odoslaný bit zapísaný ako nultý bit. Princíp: master stiahne zbernicu na log. 0 a pokiaľ je odosielaný bit 0 master uvoľní zbernicu po 50us a počká 10us, pokiaľ je odosielaný bit 1 master uvoľní zbernicu po 10us a počká 50us. Toto všetko je v slučke tak, aby bolo odoslaných 8 bitov.

```
DDR |= DQ;
PORT &= ~DQ;
if ((data & 1) == 0) {
  _delay_us(50);
  DDR &= ~DQ;
  _delay_us(10);
}
else {
  _delay_us(10);
  DDR &= ~DQ;
  _delay_us(50);
}
```

Princíp čítania

Pri čítaní je prvý načítaný bit zároveň nultým bitom. Princíp: master stiahne zbernicu na log. 0, po 10us zbernicu uvoľní. Pokiaľ je na zbernici log. 1 master načíta bit 1, v druhom prípade načíta bit 0 a čaká 60us. Toto všetko je v slučke tak, aby bolo odoslaných 8 bitov.

```
DDR |= DQ;
PORT &= ~DQ;
_delay_us(10);
DDR &= ~DQ;
_delay_us(10);
data |= (PIN & DQ) ? 0b10000000 : 0;
_delay_us(60);
```

Pre odoslanie požiadavky na konvertovanie teploty zapíšeme na slave hodnoty CCh a 44h a pre požiadavky na samotnú teplotu príkazy CCh a Beh

Komplexný príklad je v prílohe – súbor *ds1820.c* medzi zdrojovými kódmi.

3.6 Spracovanie stavu z tlačidiel

Všetky tlačidlá, ktoré sú zakomponované v zariadení sú zapojené klasickou metódou. Tlačidlo je zapojené na príslušný pin mikroprocesora, na napájanie +5V a na odpor, ktorý je zapojený na zem. Pri nestlačenom tlačidle je na pine pripojená zem (GND) a v momente stlačenia tlačidla je na príslušnom pine log. 1. Tlačidlá sú načítavané podľa aktuálnej potreby zariadenia, tzn. pokiaľ sme v hlavnom zobrazovacom panely tlačidlo slúžiace na prepínanie v menu nemá žiadny vplyv, pretože nie je načítavané, naopak pri stlačení tlačidla pre zobrazenie menu, je menu okamžite zobrazené. To znamená, že tam, kde sa vyžaduje určité tlačidlo, je vždy zahrnutá podmienka pre stisk tlačidla, t.j. či je na príslušnom pine log 1. Napr.

```
if (bit_is_set(PINC, 1)) {
  cakaj(1);
```

znamená, pokiaľ je prvý bit (tlačidlo pre prepínanie v menu) nastavený, počkaj kým log. 1 na príslušnom pine je zmenená na log. 0. Pri vstupoch z tlačidiel nevyužívam žiadne prerušenia.

3.7 Obsluha LCD displeja

Pri displeji som si mohol vybrať, či budem displej obsluhovať siedmimi dátovými vodičmi alebo štyrmi. Pokiaľ by som obsluhoval siedmimi, musel by som využiť desať pinov, pričom by som zabral dva porty, na druhú stranu by som nepotreboval posielať dáta na 2x. Keďže šetrenie pinov mi je prvoradé, zvolil som si možnosť použiť len štyri dáta pre prenos. Vyšlo mi to presne, sedem dostupných pinov na mikroprocesore a na to tri riadiace vodiče a štyri dátové vodiče. Pre obsluhu som použil zdrojový kód zo stránky Scienceprog.com, ktorý je dostupný pod licenciou GNU Public License.

Inicializácia

Pred samotnou komunikáciou s displejom je potrebná najprv inicializácia, ktorá zároveň zabezpečuje mód prenosu (4 a 8 - bit) a taktiež rozlišuje, koľko riadkový bude displej.

Inicializácia pre môj displej prebieha nasledovne: všetky dátové piny nastavíme na log. 0 a nastavíme log. 1 na E, pričom RW a RS ponecháme na log. 0. Po 1ms nastavíme E na log. 0. Po tomto môžeme začať komunikovať s LCD displejom.

```
LDP=0<<LCD_D7|0<<LCD_D6|1<<LCD_D5|1<<LCD_D4; //4 bit mode
LCP|=1<<LCD_E|0<<LCD_RW|0<<LCD_RS;
_delay_ms(1);
LCP&=~(1<<LCD_E);
_delay_ms(1);
```

Zápis do DD RAM LCD

Zápis prebieha v dvoch fázach: najprv sa odošlú horné štyri bity a hneď potom dolné štyri bity. Zapisuje sa takto: do dátových vodičov vložíme prvé štyri bity nášho prenášaného bajtu. Po tomto nastavíme RS a E na log. 1 a po 1ms ich znovu vynulujeme. Neskôr po 1ms nastavíme na dátových vodičoch posledné štyri bity prenášaného bajtu. Po tomto urobíme rovnaké kroky ako pri prvých štyroch bitoch a to nastavíme RS a E a po 1ms ich znovu vynulujeme.

Zápis príkazu

Bajt sa zapisuje takisto v dvoch fázach, avšak s rozdielom, že nastavujeme iba E. Časové oneskorenia sú takisto rovnaké. Zápis:

```
asm("cli");
LDP=(cmd&0b11110000);
LCP|=1<<LCD_E;
_delay_ms(1);
LCP&=~(1<<LCD_E);
_delay_ms(1);
LDP=((cmd&0b00001111)<<4);
LCP|=1<<LCD_E;
_delay_ms(1);
LCP&=~(1<<LCD_E);
_delay_ms(1);
asm("sei");
```

Pri jednotlivých funkciách displeja, ako napr. vypnúť kurzor, vyčistiť displej, alebo posunúť kurzor, nám teda stačí odoslať príkaz – príkaz v podobe príslušného bajtu.

4 Ladiaci program pre PC

4.1 Výber prostredia

Pri výbere prostredia, resp. programu, v ktorom budem program vytvárať, boli dve možnosti: stiahnuť jednoduchý program, bez nejakého komplexného prostredia a všetky komponenty a objekty naprogramovať sám vo Windows API – tento postup bol však zdĺhavý a po prezretí WinAPI ma definitívne prešla chuť. Druhá možnosť bola zohnať si program s komplexným prostredím a pred vytvorenými objektmi, resp. komponentmi, ktoré možno osádzať na hlavnú formu. Vybral som si teda druhý spôsob, ktorý mi veľmi uľahčil celkovú prácu.

4.2 Návrh programu

Celý program sa skladá z vizuálnej a programovej časti.

Vizuálna stránka

Program je rozdelený na hlavné okno a na jednotlivé komponenty, ktoré sú umiestnené na tomto okne. Hlavné menu je pomerne jednoducho vytvorené z menu položiek. Každá položka má za sebou jeden alebo viac boxov, ktoré hromadia jednotlivé komponenty. Pre výpis údajov som používal Label komponentu, pre vstup som používal Edit komponentu a pre samotné ovládanie som použil klasický Button.

Programová stránka

Programová časť je úzko spätá s vizuálnou časťou. Všetky komponenty reagujú na jednotlivé podnety – funkcie zvané metódy. Napr. pri stlačení tlačidla pre príjem a vyplnenie času je volaná metóda OnClickButton, v ktorej sú ďalej vykonávane moje príkazy, v tomto prípade je to zobrazenie hodnôt do Edit komponentov. A takýmto štýlom pracuje celý program – využíva jednotlivé metódy pri obsluhe tohto programu. Tento štýl taktiež vytvára program viac - vláknovým (multi-threaded), čo znamená paralelný chod

Komunikácia so zariadením

Pre komunikáciu som použil knižnicu CPortLib, ktorá je dostupná v prílohe. Táto knižnica je takisto multi-threaded, čo mi situáciu skôr skomplikovalo. Zo začiatku, teda bez tejto knižnice som plánoval, aby komunikácia prebiehala tak, že prijímam len vtedy, keď si to vyžiadam. Lenže multi-threaded princíp je paralelný a tým pádom je tu metóda ktorá je spustená pri príjme. Z toho dôvodu som v programe použil časovač, ktorý v intervale 300ms posiela požiadavku na mikroprocesor a ten následne vyvolá prerušenie, ktoré odpovedá o svojom stave a hodnotami premenných.

4.3 Úloha programu

Hyperterminál a celkovo komunikácia s PC sa overila ako veľmi dobrá pomôcka a mal som skutočne prehľad o to čo sa práve deje v mikroprocesore. Preto som vyrobil grafický program, ktorý mi to väčšmi sprehľadnil a taktiež som nebol obmedzený len čítaním vstupu, ale zároveň som dokázal na zariadenie vypisovať v reálnom čase.

4.4 Nedostatky programu

Hlavným nedostatkom je, že komunikácia nie je úplne zabezpečená voči chybným údajom vzniknutým posunutím postupnosti bajtov (bajty neprídu v poradí ako program očakáva), alebo chybným údajom vzniknutým nepresnosťou kryštálu.

5. Návod na obsluhu

5.1 Zariadenie

Po prechode do hlavného menu stlačíme tlačidlo enter (obr. 10 tl. 2). Medzi jednotlivými voľbami prepíname tlačidlami – šípka doprava, šípka doľava (tl. 1, 3). Na vstup do nejakej položky v hlavnom menu stlačíme tlačidlo enter. Jednotlivými šípkami potom zväčšujeme alebo zmenšujeme hodnoty, ktoré práve nastavujeme. Tlačidlo 4 slúži na preskočenie položky, napr. pokiaľ sme nastavili nejakú hodnotu a túto hodnotu nakoniec nechceme meniť, stlačíme tlačidlo 4, ktoré túto hodnotu ponechá a skočí do ďalšej položky.

Obr. 11Tlačitka (predný panel)

5.2 Program (Debugger)

Práca s debugger-om je veľmi jednoduchá. Po spustení si vyberieme port, na ktorom je zariadenie pripojené, potom klikneme na tlačidlo pripojiť. Pokiaľ port nie je v rozsahu medzi COM1 až COM10, klikneme na vyhľadať, čo zároveň vyhľadá port, na ktorom je zariadenie pripojené. Po pripojení sa nám sprístupnia ďalšie položky v menu – Nastavenie DCS, Ladenie a Reálny režim. V položke Nastavenie DCS môžeme nastavovať hodnoty, ktoré následne odošleme do DCS. Taktiež je tu možnosť automatického vyplnenia času. V položke Ladenie sú monitorované všetky premenné nachádzajúce sa na zariadení. V položke Reálny režim je zobrazený displej zariadenia.

6 Záver

Konštrukcia tohto zariadenia mi ponúkla veľa poznatkov a taktiež zlepšenie sa v programovaní mikroprocesorov. Nadobudol som aj nejaké technické zručnosti, ktoré som pred tvorbou tejto práce nemal a pomocou ktorej, som aj celé zariadenie nakoniec skonštruoval. Zoznámil som sa taktiež s rôznymi integrovanými obvodmi a súčiastkami. Taktiež som sa naučil pracovať s jednovodičovou (1-Wire) a dvojvodičovou (2-Wire) komunikáciou, ktorú v súčasnosti využíva veľa obvodov.

Použitá literatúra

Prace s mikrokontrolery Atmel AVR AT90S, David Matousek, BEN
Učíme se programovat v Borland C++ Builder, Václav Kadlec, Computer Press

Internetové zdroje

http://sk.wikipedia.org/wiki/Bluetooth
http://www.avrfreaks.net/

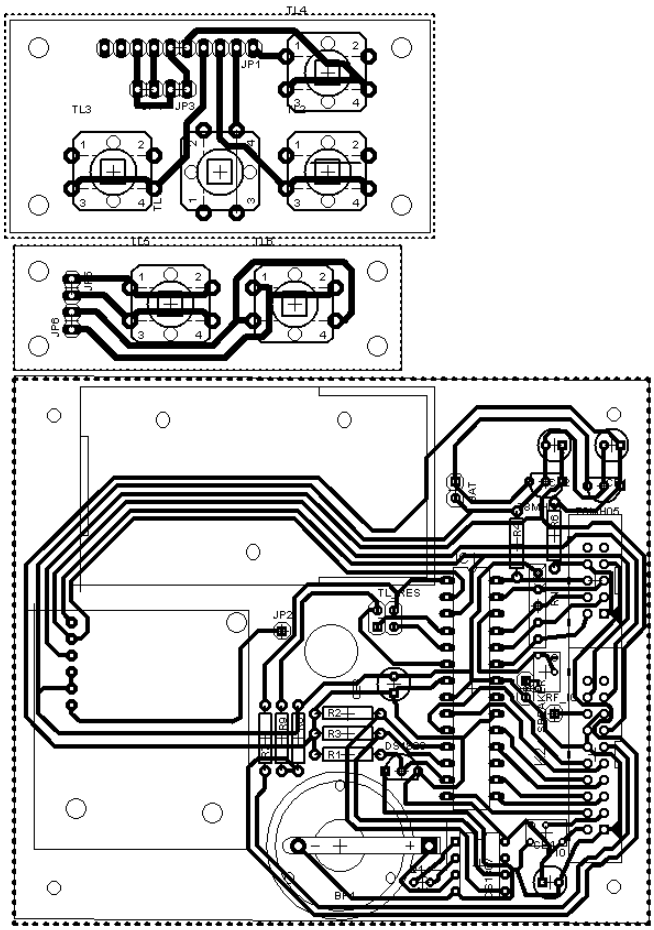
7 Prílohy

Datasheet k jednotlivým integrovaným obvodom (BTM-112, DS1307, DS1820, ATmega168, BC1602a).

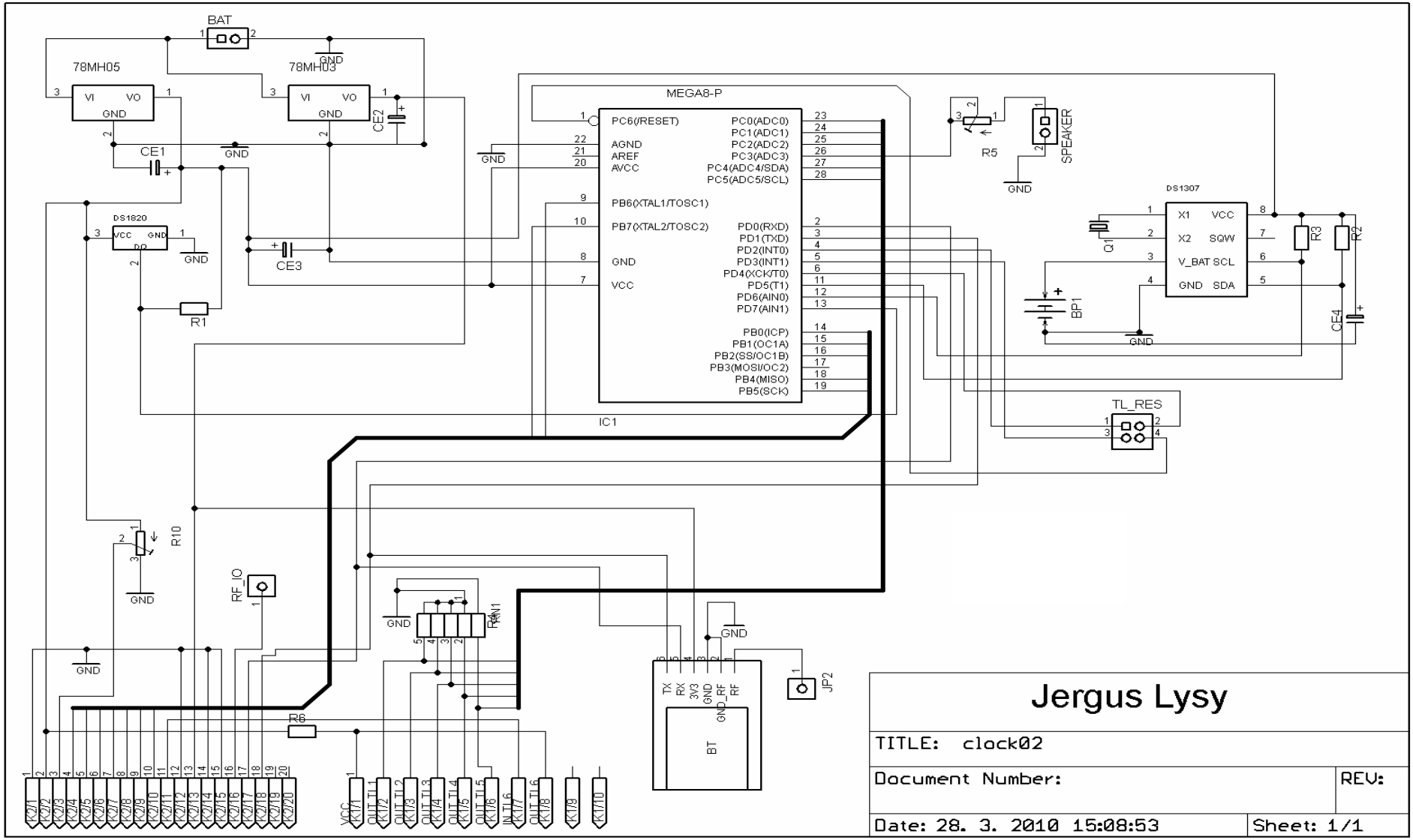
Ladiaci program pre správu DCS.

Schéma zapojenia a návrhy DPS.

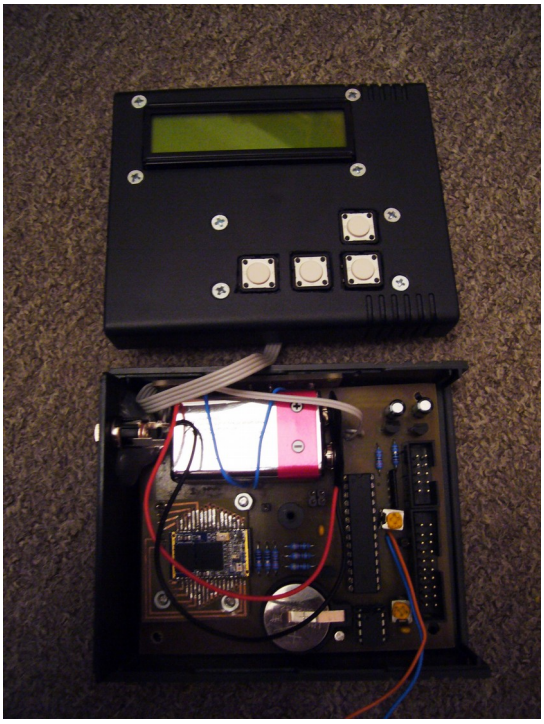
Program pre mikroprocesor a všetky potrebné knižnice.



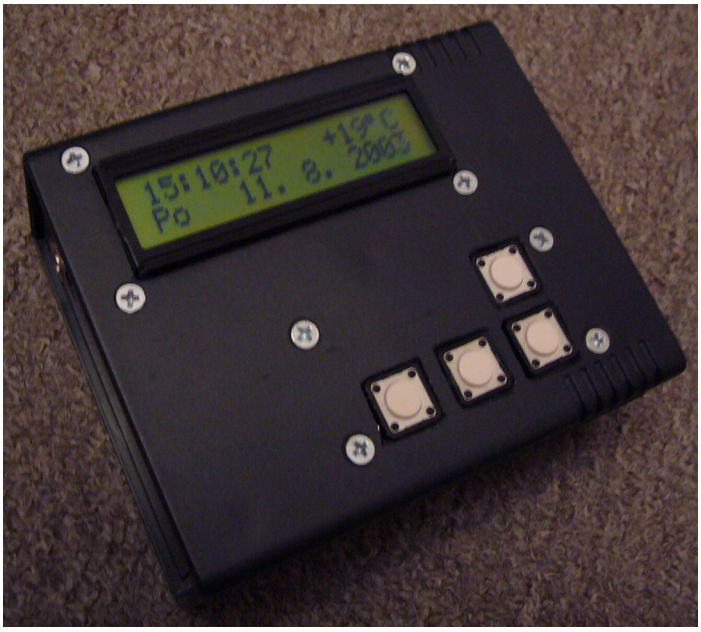
Obr. 12 Návrh plošných spojov



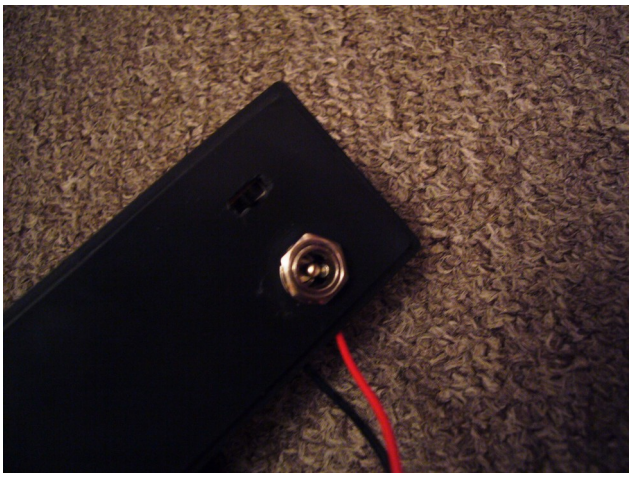
Obr. 13 *Schéma zapojenia*



Obr. 14 Vnútorné prevedenie



Obr. 15 Stav po spustení



Obr. 16 Napájanie