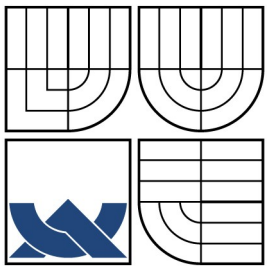
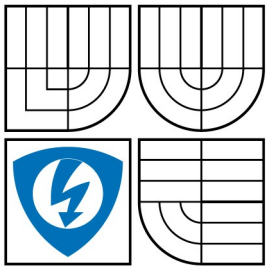


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV RADIOELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF RADIO ELECTRONICS

# MODUL PRO MONITOROVÁNÍ TEPLOTY

TEMPERATURE MONITORING UNIT

DIPLOMOVÁ PRÁCE
MASTER´S THESIS

AUTOR PRÁCE — Bc. JAROSLAV MARTÍNEK
AUTHOR

VEDOUCÍ PRÁCE — doc. Dr. Ing. ZDENĚK KOLKA
SUPERVISOR

BRNO 2008

# Licenční smlouva
## poskytovaná k výkonu práva užít školní dílo

uzavřená mezi smluvními stranami:

**1. Pan/paní**

Jméno a příjmení: Bc. Jaroslav Martínek

Bytem: č.p. 8, 78316, Dolany - Véska

Narozen/a (datum a místo): 23.3.1983, Bílovec

(dále jen „autor“)

a

**2. Vysoké učení technické v Brně**

Fakulta elektrotechniky a komunikačních technologií

se sídlem Údolní 244/53, 602 00, Brno

jejímž jménem jedná na základě písemného pověření děkanem fakulty:

prof. Dr. Ing. Zbyněk Raida

(dále jen „nabyvatel“)

### Čl. 1
### Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
   - ☐ disertační práce
   - ☒ diplomová práce
   - ☐ bakalářská práce
   - ☐ jiná práce, jejíž druh je specifikován jako .....

   (dále jen VŠKP nebo dílo)

Název VŠKP: Modul pro monitorování teploty

Vedoucí/ školitel VŠKP: doc. Dr. Ing. Zdeněk Kolka

Ústav: Ústav radioelektroniky

Datum obhajoby VŠKP: _____

VŠKP odevzdal autor nabyvateli v*:

- ☒ tištěné formě – počet exemplářů: 2
- ☒ elektronické formě – počet exemplářů: 2

* hodící se zaškrtněte

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

## Článek 2
## Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizovaní výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti
   - ☒ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

## Článek 3
## Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: …….

…….. ……
Nabyvatel Autor

## ABSTRAKT

Diplomová práce se zabývá návrhem a realizací modulu pro vícebodové monitorování teploty v chladících a mrazících zařízeních ve skladech a prodejnách potravin. Jako řídící procesor pro měřící modul byl vybrán Atmel ATMega128, čidla jsou realizována monolitickými teploměry DS18S20, připojenými pres 1-wire sběrnici. Modul umožňuje připojení až 32 teplotních čidel, obsahuje vlastní paměť pro naměřená data, hodiny reálného času, rozhraní pro přenos dat sítí Ethernet a displej s ovládacími tlačítky pro základní obsluhu. Měření probíhá periodicky v intervalu definovaném uživatelem, naměřené hodnoty jsou ukládány do paměti a zobrazovány na displeji. Data je též možno přenášet přes síť Ethernet do klientského PC, kde je na uživatelském software možno prohlížet data a zobrazit grafické průběhy naměřených teplot.



**Klíčová slova:** Atmel, AVR, programování, měření teploty, teploměr, datalogger, komunikace, 1-wire, DS18S20, $I^2C$



## ABSTRACT

Content of the Bachelor Thesis covers design and realisation of an unit for multipoint temperature measuring in cooling and freezing equipments in food stores and shops. As a controlling microcomputer is used an Atmel ATMega128, temperature sensors are realised by DS18S20 digital thermometers, connected via 1-wire bus. Unit can read temperature from 32 temperature sensors, includes memory for measured data, real time clock, Ethernet interface and display with buttons for basic user control. Temperatures are measured periodically with user defined interval, values are stored into internal memory and showed on display. Measured data can also be transferred via Ethernet to clinent PC, where can be viewed in form of table and graphs.



**Key words:** Atmel, AVR, programming, temperature measuring, thermometer, datalogger, communication, 1-wire, DS18S20 , $I^2C$

## BIBLIOGRAFICKÁ CITACE DLE ČSN ISO 690:

MARTÍNEK, J. Modul pro monitorování teploty. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008. 52 s. Vedoucí diplomové práce doc. Dr. Ing. Zdeněk Kolka.

# Prohlášení

Prohlašuji, že svou diplomovou práci na téma „Modul pro monitorování teploty“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny uvedeny v seznamu literatury na konci práce.

V Brně dne 30.05.2008

.....
podpis autora

# Poděkování

Děkuji vedoucímu diplomové práce doc. Dr. Ing. Zdeňku Kolkovi za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne 30.05.2008

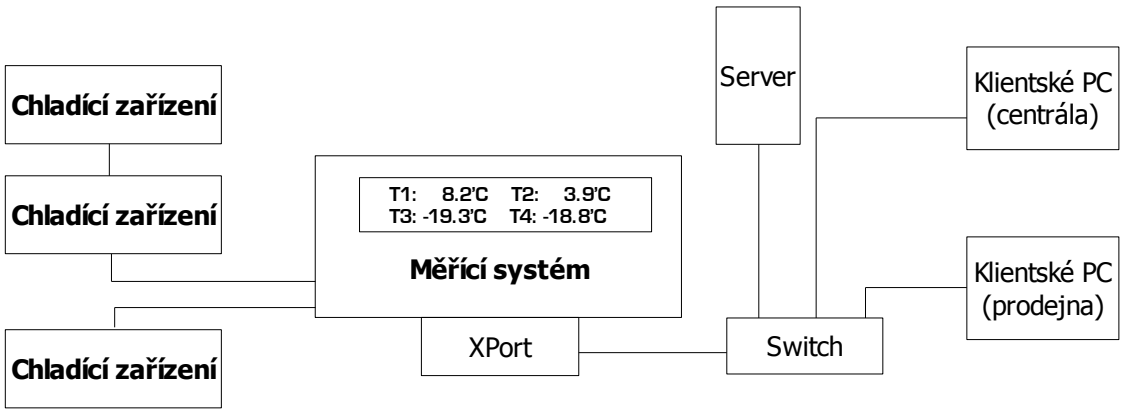
.....
podpis autora

# Obsah

# 1. Úvod

Skladování potravin v obchodních a výrobních řetězcích je věcí velmi složitou a je řízeno patřičnými zákony a předpisy. V největší míře se to týká potravin chlazených a mražených, kde je potřeba dodržet skladovací parametry, v tomto případě hlavně teplotu v zákonem přesně stanovených mezích. Jelikož provozovatel takovéhoto skladu či obchodu přebírá při převzetí tohoto zboží i odpovědnost za jeho správné skladování, je velmi důležité mít aktuální přehled o stavu a teplotách v použitých mrazících a chladících zařízeních kvůli odpovědnosti za zboží, ať už vůči koncovým zákazníkům či kontrolním orgánům, které tak mohou mít na vyžádání kompletní a přesné přehledy o průběhu skladování zboží. A také provozovatel se takto okamžitě dozví např. o poruše některého z provozovaných zařízení.

Celé zařízení je postaveno na tzv. „embedded-systému“, tedy autonomním mikropočítači, který zajistí samostatný provoz monitorovací části systému i v případě výpadku ostatních podpůrných částí a je schopen bateriového provozu pro případ výpadku elektrické energie. Informace o aktuálním stavu měření a základní nastavení je možné provádět přímo na základní jednotce pomocí alfanumerického displeje. Zařízení je dále připojeno do místní sítě typu Ethernet, přes níž probíhá komunikace s klientským PC. Zpracování naměřených dat je realizováno v klientském PC pomocí obslužného software, v němž je možno zjistit veškerá data z vnitřní paměti jednotky a dále s nimi pracovat, tj. vytvářet grafy, ukládat naměřené hodnoty, jejich export do tabulky (např. pro MS Excel). Dále je možno přes tento software provádět nastavení systému.




Obr. 1.1 – Návrh koncepce systému

# 2. 1-wire sběrnice

## 2.1 Obecné informace

1-wire je systém sběrnice pocházející z dílen společnosti Dallas Semiconductors (nyní pod Maxim), který poskytuje nízkorychlostní přenos dat, signálů a napájení po jednom vodiči (plus druhý zemnící vodič). 1-wire pracuje na stejném konceptu jako sběrnice $I^2C$, ovšem s menší datovou rychlostí a daleko nižší cenou. Používána bývá převážně pro komunikaci s malými a levnými zařízeními jako např. digitální teploměry a jiné senzory.

Jak již bylo zmíněno, jednou z nejdůležitějších vlastností této sběrnice je, že zařízení k ní připojené potřebují pouze dva vodiče – datový a zem. Toho bylo docíleno tím, že zařízení obsahují 800pF kondenzátor, který zajišťuje jejich napájení z datového vodiče.

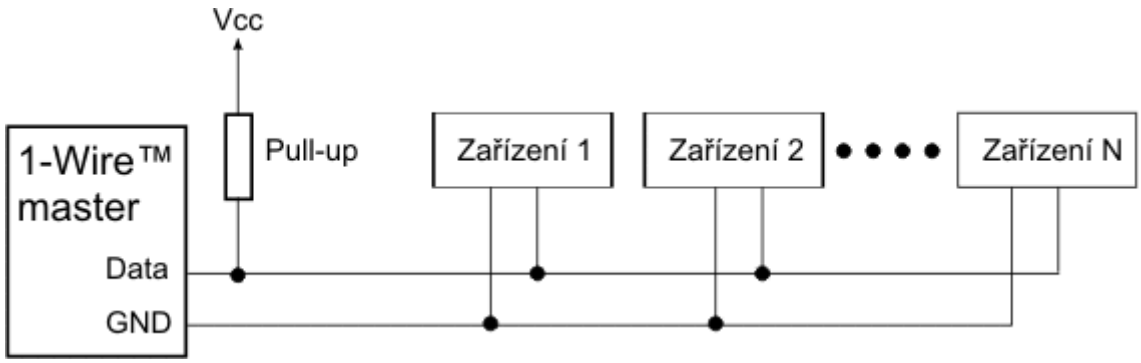
Takto může být velmi jednoduše sestaven systém senzorů, kdy každý obsahuje vlastní čidlo a logiku potřebnou pro komunikaci po sběrnici.

1-wire zařízení bývají většinou integrované obvody umístěné v běžných pouzdrech (TO92,...), připojení na sběrnici bývá realizované pomocí modulárních konektoru (RJ11, RJ45,...) a patřičné kabeláže.

iButton je druhou možností připojení 1-wire zařízení. Je to vlastně jiný standard mechanického pouzdra, kdy je určitý 1-wire komponent umístěn v kovovém „knoflíku“ (odtud iButton), vypadajícím podobně jako známe lithiové baterie např. do hodinek. Takováto zařízení se ke sběrnici připojují pomocí adaptéru.

Obr. 2.1 – Možná provedení iButton

Hardwarově je implementace protokolu řešena speciálním software v master zařízení a rezistorem. Skrze něj je na sběrnici přivedeno +5V, což zajistí napájení slave zařízení. Master zařízením může v tomto případě být téměř jakékoliv PC nebo mikropočítač. Samozřejmě jsou také k dispozici různé řadiče a převodníky, realizující protokol přímo hardwarově.




Obr. 2.2 – Zapojení zařízení na sběrnici

## 2.2 Komunikace po sběrnici

Master zařízení začíná sekvenci bitů „reset“ pulsem, který na 480 µs vyšle na sběrnici 0 V. Tímto dojde k resetu všech slave zařízení v podstatě jejich odpojením od napájení. Po tomto pulsu, tedy po opětovném zapojení napájení, o sobě dá každé přítomné slave zařízení vědět tzv. pulsem přítomnosti – přízemní sběrnici na alespoň 60 µs po jejím uvolněním masterem.

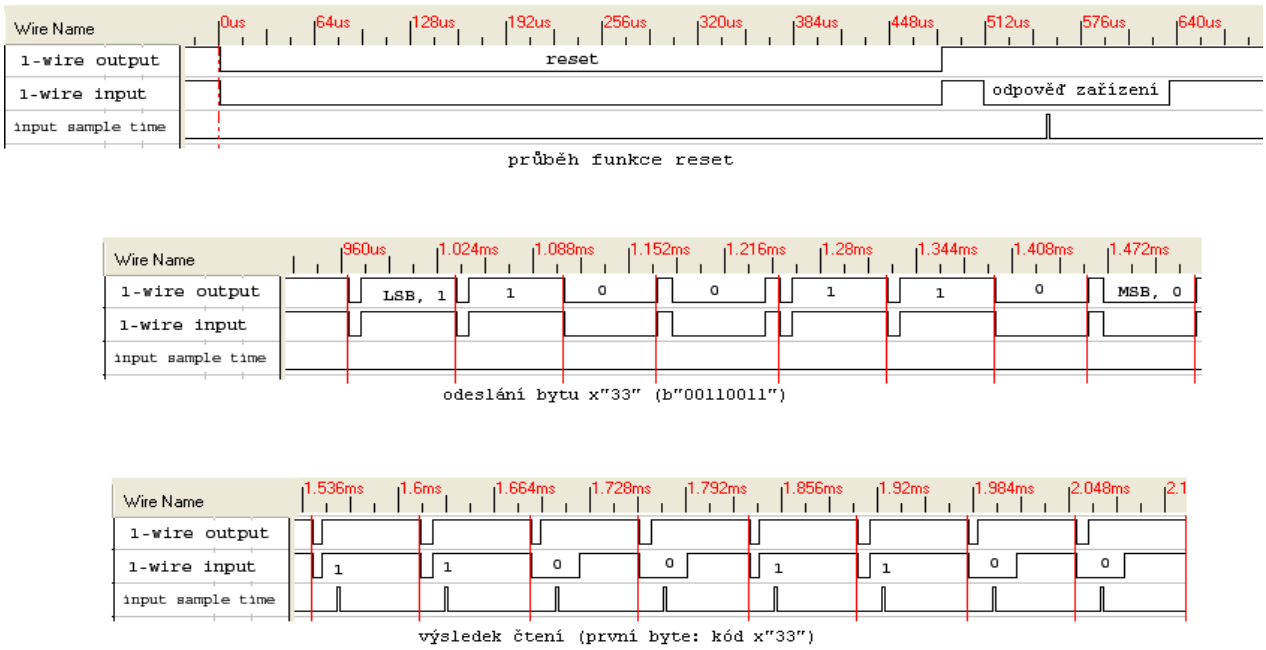
Pro vyslání logické 1 software masteru vyšle krátký 0 V puls (1-15 µs), pro vyslání logické 0 software odešle 60 µs dlouhý 0 V puls. Sestupná hrana těchto pulsů způsobí spuštění monostabilního klopného obvodu ve slave zařízení, který funguje jako časovač, čímž je zajištěno, aby slave četl data ze sběrnice po dobu přibližně 30 µs. Klopný obvod má samozřejmě své tolerance, takže časování není příliš přesné. Kvůli tomu je potřeba, aby výstupní pulsy byly dlouhé 60 µs pro logickou 0 a nebyly delší než 15 µs pro logickou 1.

Při příjmu dat vyšle master 1-15 µs dlouhý 0 V puls na startu každého bitu. Pokud chce slave zařízení vyslat log. 1, nedělá nic a na sběrnici se tak hned objeví napájecí napětí. Pokud chce slave odeslat log. 0, přidrží sběrnici na 0 V po dobu 60 µs.

Základní komunikační sekvence tedy vypadá tak, že je vyslán reset puls následovaný 8-bitovým příkazem. Potom jsou odesílána nebo přijímána data ve skupinách po osmi bitech.

Většina zařízení může sdílet jednu sběrnici. Pro tento účel má každé z nich jedinečné 64 bitové sériové číslo. Nejvyšší bit (MSB) tohoto sériového čísla je osmibitové číslo, které udává typ zařízení. Nejnižší bit je pak klasický osmibitový kontrolní součet (CRC).

V definici komunikace existuje také několik všeobecných vysílacích příkazů a příkazů adresovaných pouze pro určité zařízení. Master může odeslat příkaz pro výběr následovaný adresou příslušného zařízení, následující příkaz pak bude proveden přímo tímto zařízením.




Obr. 2.3 – Průběh komunikace po sběrnici

Příkazy SEARCH a ALARM SEARCH

Většina 1-wire zařízení implementuje příkaz SEARCH (kód 0xF0). Zařízení, která mohou být v nějakém "poplachovém stavu" implementují i příkaz ALARM SEARCH (kód 0xEC). Tento příkaz se od předchozího liší pouze tím, že na něj reagují zařízení, která jsou momentálně v "poplachovém stavu" (teplotní čidlo, u něhož je překročena nastavená hranice teploty apod.), což umožňuje rychleji zjistit, které zařízení má být obslouženo.

Zařízení, které přijme SEARCH příkaz, odpoví tím, že vyšle první (nejnižší) bit svého 64bitového kódu. Je-li na sběrnici víc oslovených zařízení, odpoví všechny naráz. Jak vyplývá ze specifikace 1-wire sběrnice (zařízení jsou připojena paralelně ke společnému vodiči výstupem s otevřeným kolektorem), výsledkem je logický součin (AND) všech bitů. Po vyslání tohoto bitu požádá master o vyslání dalšího bitu. Zařízení na tento požadavek odpoví negací prve vyslaného bitu. Z těchto dvou přijatých bitů lze odvodit situaci na sběrnici. Mohou nastat čtyři různé možnosti:

| První bit | Druhý bit | Situace |
|---|---|---|
| 0 | 0 | Na sběrnici je více zařízení. Tato zařízení mají na tomto místě ve svých kódech různé hodnoty. Došlo k neshodě. |
| 0 | 1 | Na sběrnici je jedno či více zařízení. Tato zařízení mají na tomto místě ve svých kódech bit s hodnotou 0. |
| 1 | 0 | Na sběrnici je jedno či více zařízení. Tato zařízení mají na tomto místě ve svých kódech bit s hodnotou 1. |
| 1 | 1 | Na sběrnici není žádné zařízení, které by reagovalo na příkaz SEARCH. |

Tab. 2.1 - Možné situace na sběrnici

Master nyní vyšle jeden potvrzovací bit. Dále se budou vyhledávání účastnit pouze zařízení, která mají na prvním místě bit s hodnotou stejnou jako vyslal master.

Pokud mají všechna zařízení na dané pozici stejnou hodnotu bitu, vyšle master tuto hodnotu. Pokud mají různé hodnoty (tzn. nastala první možnost - oba přečtené bity byly nulové), musí si master poznamenat, na které pozici došlo k neshodě, a vyšle buď 1 nebo 0. Zda master vyšle 0 nebo 1 určí na základě předchozích hledání, především pozice poslední nalezené neshody. Dále popsaný algoritmus vysílá nejprve nulu a při druhém průchodu vysílá jedničku.

Tento postup se opakuje, dokud není načteno všech 64 bitů identifikace. Pokud při načítání došlo k neshodě v nějakém bitu, znamená to, že je na sběrnici více zařízení a celý postup se opakuje s tím, že na pozici poslední neshody se nyní vysílá bit opačné hodnoty (prochází se "druhá větev").

Popis algoritmu

Procházení sběrnice sestává z opakovaných hledání jednotlivých zařízení. V každém cyklu je nalezeno jedno zařízení. V algoritmu je třeba si neustále udržovat informaci o kódu posledního nalezeného zařízení a o pozici neshody (zdali k nějaké došlo nebo ne anebo je nalezené zařízení na sběrnici samo).

Algoritmus pracuje s následujícími proměnnými:

**id_bit_number** - Pořadové číslo bitu, který je právě prohledáván (1 - 64).

**id_bit** - Bit přečtený jako první. Jeho hodnota je logický součin všech bitů na pozici id_bit_number ze všech zařízení, které se účastní vyhledávání.

**cmp_id_bit** - Bit přečtený jako druhý, doplněk id_bit.

**LastDiscrepancy** - Hodnota udávající pořadové číslo bitu, u něhož došlo při posledním hledání k neshodě.

**LastDeviceFlag** - Příznak posledního nalezeného zařízení.

**Last_Zero** - Hodnota udávající pořadové číslo bitu, u něhož došlo při aktuálním hledání k neshodě, ze které bylo pokračováno odpovědí "0".

**ROM_NO** - Buffer o velikosti 64 bitů (8 byte), který obsahuje aktuálně načítané číslo zařízení.

**search_direction** - Bitová proměnná, udávající směr dalšího hledání. Zařízení, která mají bit na pozici id_bit_number roven této hodnotě se budou dále účastnit prohledávání, ostatní zařízení přestanou komunikovat až do vyslání RESET pulsu.

Před prvním hledáním (funkce FIRST) je proměnná LastDiscrepancy vynulována. Po volání hledací rutiny je v ROM_NO číslo nalezeného zařízení. Pokud není žádné zařízení nalezeno, je hledání ukončeno s chybou.

Hledání dalšího zařízení (funkce NEXT) využívá hodnoty z prvního hledání, uložené v proměnných ROM_NO a LastDiscrepancy. "Hledací" rutina vrátí v ROM_NO číslo dalšího zařízení, nebo, bylo-li předcházející nalezené zařízení posledním, vrátí chybový kód.

Hledání jednotlivých zařízení začíná tím, že master vyšle RESET puls a čeká na potvrzení PRESENCE pulsem. Pokud přijde, hledání může pokračovat a jsou nastaveny proměnné: id_bit_number (pozice zpracovávaného bitu) na hodnotu 1 a last_zero (pozice poslední odpovědi "0" na zjištěnou neshodu) na hodnotu 0. Poté je vyslán příkaz SEARCH nebo ALARM SEARCH. Následuje smyčka, v níž je skládán obsah ROM_NO. Tato smyčka se opakuje 64krát, tedy pro každý bit identifikačního kódu.

Ve smyčce jsou nejprve přečteny dva bity (zařízení vysílá vždy hodnotu bitu a její doplněk), viz výše. Pokud se od sebe oba přijaté bity liší (což znamená, že všechna zařízení na dané pozici mají stejnou hodnotu bitu), je hodnota prvního zapsána do ROM_NO na pozici danou id_bit_number a jeho hodnota je zároveň zapsána do search_direction. Pokud jsou oba jedničkové, znamená to, že žádné zařízení na hledání neodpovídá a funkce končí. Pokud jsou oba nulové, znamená to, že je třeba se rozhodnout, jestli bude vyslána nula nebo jednička. Rozhoduje se na základě proměnných LastDiscrepancy a id_bit_number takto:

| **LastDiscrepancy vs. id_bit_number** | **Reakce** |
|---|---|
| id_bit_number > LastDiscrepancy | Last_zero = id_bit_number<br>search_direction = 0 |
| id_bit_number == LastDiscrepancy | search_direction = 1 |
| id_bit_number < LastDiscrepancy | search_direction = (ROM_NO & (1<< id_bit_number) > 0)<br>(search_direction je nastavena na hodnotu bitu na stejné pozici v předchozím nalezeném ROM_NO) |

Tab. 2.2 – Možnosti odeslání bitu

Následně je vyslána hodnota bitu search_direction, tento bit je zapsán do ROM_NO na odpovídající pozici a je zvýšeno počítadlo id_bit_number. Pak se celá smyčka opakuje.

Po zjištění všech bitů je do proměnné LastDiscrepancy zapsána hodnota proměnné LastZero. Pokud je tato hodnota nulová, znamená to, že je nalezené zařízení poslední.

# 3. Teplotní čidlo DS1820

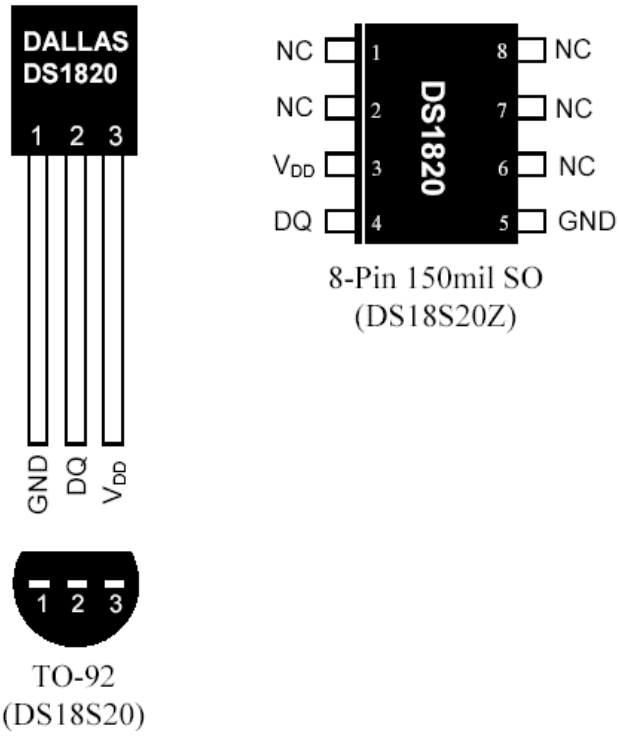
## 3.1 Základní vlastnosti

DS1820 je 1-wire teplotní senzor (v podstatě samostatný teploměr) taktéž od Dallas Semiconductors, který poskytuje 9-bitové měření teploty ve stupních Celsia. Obsahuje také funkci alarmu s uživatelsky programovatelným horním a dolním limitem. Jelikož výstup čtený ze senzoru (tedy výstupní teplota) je v digitální podobě, jeho spojení s mikroprocesorem je vcelku jednoduché.

Základní vlastnosti tohoto teploměru:

- Jelikož pracuje na 1-wire sběrnici, potřebuje jen jeden datový vodič, což značně zjednodušuje připojení
- Díky schopnosti vícebodového připojení je snadnější návrh zařízení pro měření teploty
- Nepotřebuje žádné další externí součástky
- Rozsah měřených teplot je od –55°C do +125°C (–67°F až +257°F)
- V rozsahu –10°C až +85°C je udávaná přesnost ±0.5°C
- 9-bitové rozlišení
- Napájecí napětí v rozsahu 3 V – 5,5 V
- Doba převodu teploty je 750 ms

(v současnosti se typ DS1820 díky chybě některých sérií nevyrábí a je nahrazen novějším DS18S20, což ovšem nemá pro naši aplikaci žádný význam)



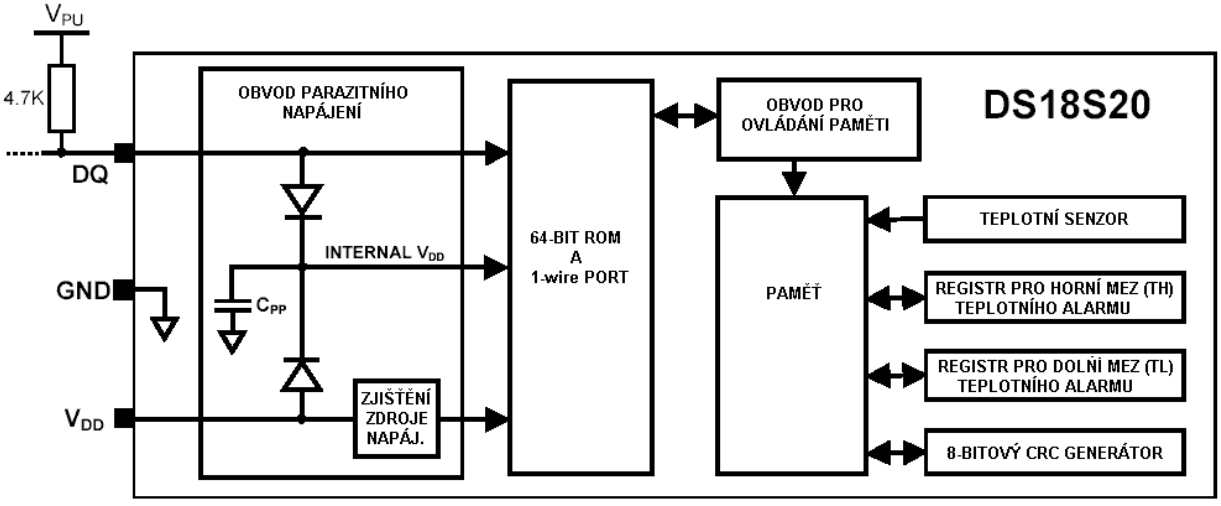

Obr. 3.1 – Možná provedení teploměru DS18S20

| POUZDRO | | SYMBOL | POPIS |
|---|---|---|---|
| 8-PIN SOIC | TO-92 | | |
| 5 | 1 | GND | Zem |
| 4 | 2 | DQ | Datový vstup/výstup. Pin pro připojení k 1-wire sběrnici. Také se používá k napájení čidla v případě použití parazitního napájení. |
| 3 | 3 | $V_{DD}$ | Volitelný napájecí pin. Musí být uzeměný pro použití parazitního napájení. |

Tab. 3.1 – Zapojení vývodů teploměru DS18S20

## 3.2 Popis funkce

Na obrázku 4 je nakreslen blokový diagram teplotního čidla DS18S20 a popis pinů je uveden v tabulce výše. V 64-bitové ROM paměti je uloženo unikátní sériové číslo čidla. Paměť obsahuje 2-bytový teplotní registr, ve kterém je uložen digitální výstup z teplotního senzoru. Dále poskytuje přístup k 1-bytovým registrům horního a dolního teplotního alarmu (TH a TL). Tyto registry jsou řešeny pamětí EEPROM, takže nastavené hodnoty zůstávají v platnosti i při vypnutém napájení.



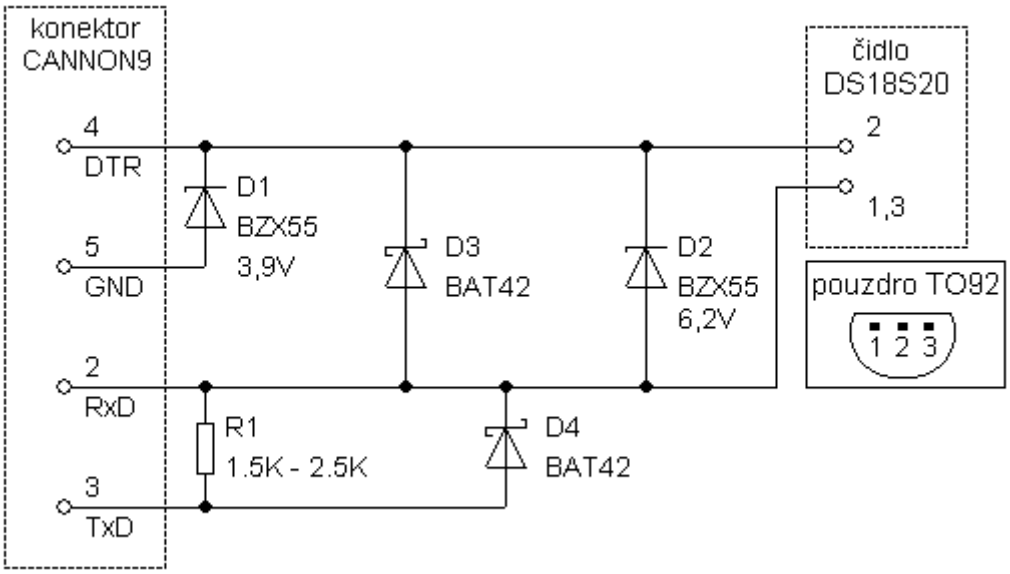

Obr. 3.2 – Blokový diagram čidla DS18S20

Další vlastností obvodu DS18S20 je možnost funkce bez externího napájení. To je v tomto případě řešeno přes 1-wire pull-up rezistor na pinu DQ když je na sběrnici napájecí napětí. To zároveň nabije vnitřní kondenzátor (Cpp), který potom napájí zařízení když je sběrnice uzemněná.

## 3.3 Ověření funkčnosti

Funkčnost teplotního čidla byla ověřena na běžném PC při použití operačního systému Linux.

Čidlo je potřeba k COM portu PC připojit přes tzv. adaptér. Lze použít aktivní adaptér DS9097U, který sice umožňuje připojení i jiných čidel fy Dallas, nicméně stojí řádově stovky Kč. Pro naše účely jednoduchého ověření měření teploty je tedy výhodnější pasivní adaptér, jehož schéma je na obrázku.




Obr 3.3 – Zapojení teploměru DS18S20 k sériovému portu PC

Jak již bylo uvedeno dříve, každé čidlo má z výroby své jedinečné číslo, což umožňuje jejich připojení paralelně k sobě. Počet čidel je teoreticky neomezený, prakticky je limit pouze v délce a kvalitě kabelů k čidlům. Samotné čidlo se k adaptéru připojí tak, že se jeho krajní vývody spojí a připojí na pin 2 sériového portu (RxD), prostřední vývod potom na pin 4 (DTR).

V tomto případě je napájení čidla zajištěno při komunikaci ze sériového portu. To má výhodu hlavně ve své jednoduchosti. Nevýhodou je omezená horní hranice čtení teploty na cca 70°C u většiny běžných PC. Také je možné narazit na nekompatibilitu s některými sériovými porty PC, v tomto případě může pomoci změna (zvětšení) odporu R1 anebo zařazení 10K trimru mezi piny TxD a GND.

Obslužný program vychází z projektu DigiTemp a je uvolněn jako Open Source pod licencí GNU. Je k dispozici pod operační systémy Unix/Linux, DOS a Windows, nicméně v této chvíli je podporována pouze verze pro Linux. Program samotný je pouze konzolová aplikace, která "jen" přečte teploty z čidel, případně je uloží do logu.

Po zkompilování zdrojových kódů je aplikaci nejprve potřeba spustit příkazem:

```
digitemp_DS9097 -s/dev/ttyS0 -i
```

Ten prohledá sběrnici na zadaném portu a nalezne všechna zařízení na ní připojené. Zjištěné informace zapíše do konfiguračního souboru .digitemprc

```
DigiTemp v3.3.2 Copyright 1996-2004 by Brian C. Lane
GNU Public License v2.0 - http://www.brianlane.com
Turning off all DS2409 Couplers
..
Searching the 1-Wire LAN
1030A8BA000800E7 : DS1820/DS18S20/DS1920 Temperature Sensor
108396BA0008009F : DS1820/DS18S20/DS1920 Temperature Sensor
ROM #0 : 1030A8BA000800E7
ROM #1 : 108396BA0008009F
Wrote .digitemprc
```

Poté už stačí spustit program s parametrem -a:

```
digitemp_DS9097 -a
```

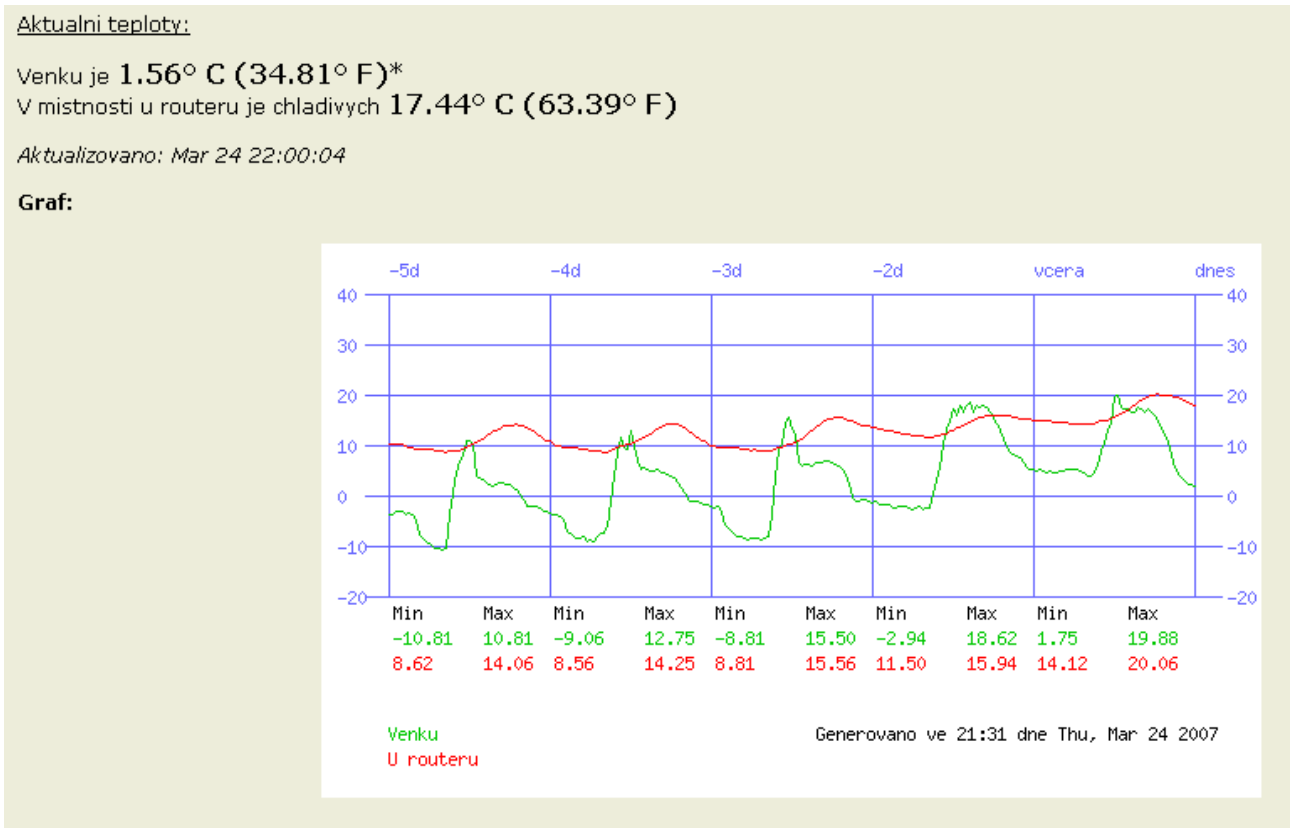
a ten již provede vlastní čtení teploty z připojených čidel:

```
DigiTemp v3.3.2 Copyright 1996-2004 by Brian C. Lane
GNU Public License v2.0 - http://www.brianlane.com
Mar 13 17:58:07 Sensor 0 C: 1.06 F: 33.91
Mar 13 17:58:09 Sensor 1 C: 12.88 F: 55.17
```

a naměřené hodnoty zároveň zapíše do log souboru v tomto tvaru:

```
Apr 30 22:01:02 Sensor 0 C: 5.12 F: 41.23
Apr 30 22:01:04 Sensor 1 C: 14.88 F: 58.77
Apr 30 22:31:02 Sensor 0 C: 5.19 F: 41.34
Apr 30 22:31:04 Sensor 1 C: 14.88 F: 58.77
Apr 30 23:01:02 Sensor 0 C: 5.06 F: 41.11
Apr 30 23:01:04 Sensor 1 C: 14.81 F: 58.66
Apr 30 23:31:02 Sensor 0 C: 5.06 F: 41.11
Apr 30 23:31:04 Sensor 1 C: 14.69 F: 58.44
```

Tyto hodnoty je pak samozřejmě možné dále zpracovávat, ukládat do databáze, tvořit grafy, přehledy atd.




Obr. 3.4 – Možný výstup naměřených dat v podobě online teploměru

Na této velice jednoduché aplikaci měření teplot s 1-wire senzory DS1820 byla ověřena funkčnost použitých senzorů a jejich souběžného zapojení na stejné sběrnici. Je vidět, že tato čidla jsou opravdu velmi jednoduchá pro zapojení i obsluhu, obdobně i jejich přesnost je pro běžná měření teploty dostačující.

# 4. RTC Dallas DS1307

## 4.1 Základní vlastnosti

Jedná se o sériové hodiny reálného času (RTC), obsahující hodiny a kalendář v BCD kódu plus 56 bytů NV SRAM. Adresy a data jsou přenášeny sériově přes obousměrnou $I^2C$ sběrnici. Hodiny a kalendář poskytují přesnou informaci o roku, měsíci, dni, hodinách, minutách a sekundách. Počet dnů v měsíci je automaticky upraven podle měsíce včetně korekce pro přestupný rok. Hodiny můžou fungovat ve 24 hodinovém nebo 12 hodinovém formátu (s indikací dopoledne nebo odpoledne). DS1307 má vestavěný kontrolní obvod napájení, který v případě, že dojde k odpojení napájení ze zdroje, přepne na připojenou baterii a tím zajistí uchování aktuálního času při napájení ze zálohy.

Hodiny reálného času, počítající sekundy, minuty, hodiny, dny, měsíce, dny v týdnu a rok s kompenzací pro přestupný rok platnou do roku 2100

56 bytů NV RAM pro uložení dat, zálohovanou baterií

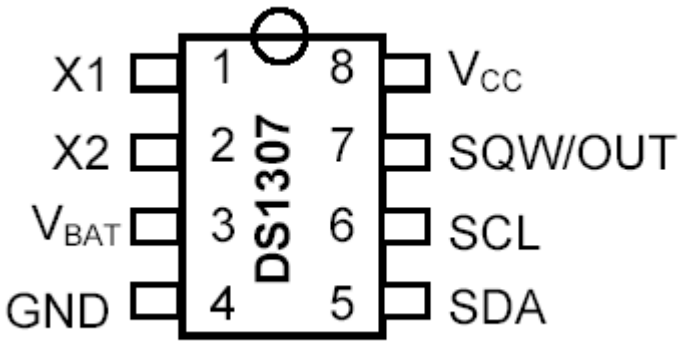
Komunikace přes sériovou $I^2C$ sběrnici

Programovatelný výstupní obdélníkový signál

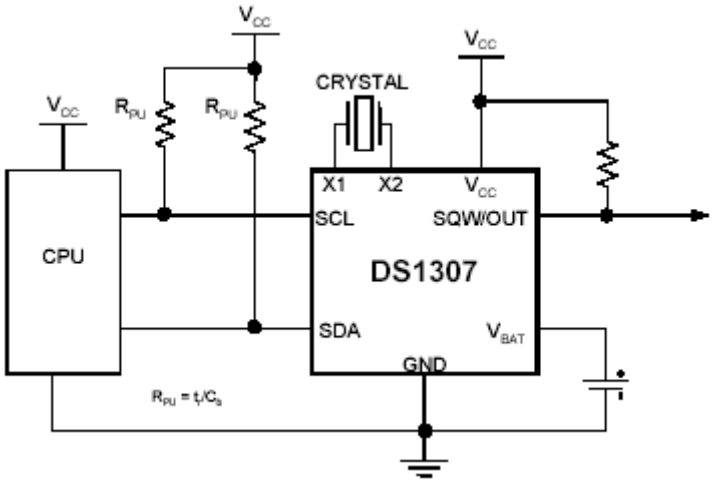
Automatická detekce výpadku napájení a přepínací obvod

V případě napájení ze záložní baterie je při běžícím oscilátoru spotřeba menší než 500nA

Teplotní rozsah -40°C až +85°C

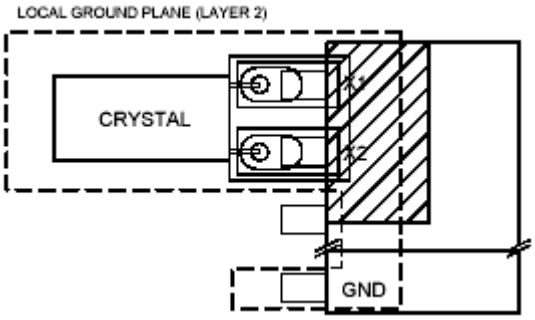



Obr. 4.1 – Rozložení vývodů obvodu RTC




Obr. 4.2 – Základní zapojení obvodu [6]

Obvod se k mikroprocesoru připojí přímo, ke sběrnici se přidají pouze pull-up rezistory. Dále se připojí napájení, zálohovací baterie a krystal. Ten by měl být umístěn co nejblíže pouzdru a signálové cesty by neměly křižovat vyšrafovanou část pouzdra, pokud mezi nimi není zem, viz obrázek:




Obr. 4.3 – Připojení krystalu [6]

# 5. Modul XPort

## 5.1 Základní informace

DSTni-XPort je nejkompaktnější integrovaný modul pro řešení zajištění síťové a WWW konektivity pro libovolné zařízení vybavené sériovým rozhraním.


Obr. 5.1 – Modul XPort

XPort byl navržen pro výrobce kteří potřebují nástroj, pomocí kterého lze snadno a rychle zajistit připojení jejich zařízení do sítí LAN/WAN a Internetu. XPort nabízí výrobcům nejvyšší dostupnou úroveň integrace v oboru terminálových serverů (převodníků RSxxx na IP). Uvnitř kompaktního krytu konektoru RJ-45 je umístěn DSTni-LX 186 procesor s řadiči, paměť, obvody rozhraní Ethernet 10/100, rychlý port RS232, diagnostické LED a 3 programovatelné I/O piny. V prostoru, který je obvykle určen pro pouhý konektor, poskytuje modul XPort kompletní síťové rozhraní a výpočetní výkon přibližně odpovídající PC XT. Pro připojení k lokální síti nebo internetu obsahuje XPort kompletní TCP/IP stack a operační systém, nad nímž lze k standardním funkcím firmware doplnit i libovolné zákaznické aplikace. XPort také obsahuje kompletní WWW server, který lze využít např. pro dálkovou konfiguraci, dohled, vizualizaci a ovládání připojeného zařízení.

Kdekoliv je zapotřebí vytvořit uživatelské rozhraní a současně použít standardní, známé a široce rozšířené nástroje, může XPort posloužit jako poskytovatel Java appletů obsahujících zákaznickou aplikaci pro web browser. Tímto se XPort může stát prostředníkem komunikace mezi Vámi a Vaším zařízením připojeným k síti LAN nebo internetu.

Pro zjednodušení instalace a nastavování je k dispozici konfigurační SW pro Windows - XPort Installer. XPort může být konfigurován lokálně po sériovém portu nebo dálkově s použitím telnetu nebo web browseru. Vestavěná paměť Flash slouží pro bezúdržbové uložení obsahu webových stránek a umožňuje bezproblémový upgrade systémových SW modulů v případě potřeby. Použitím modulů XPort jako vysoce integrované hardwarové a SW platformy lze dosáhnou významných přínosů a úspor podstatným zrychlením procesu vývoje, snížení rizik a nákladů.

## 5.2 Parametry

**XPort - Vlastnosti**

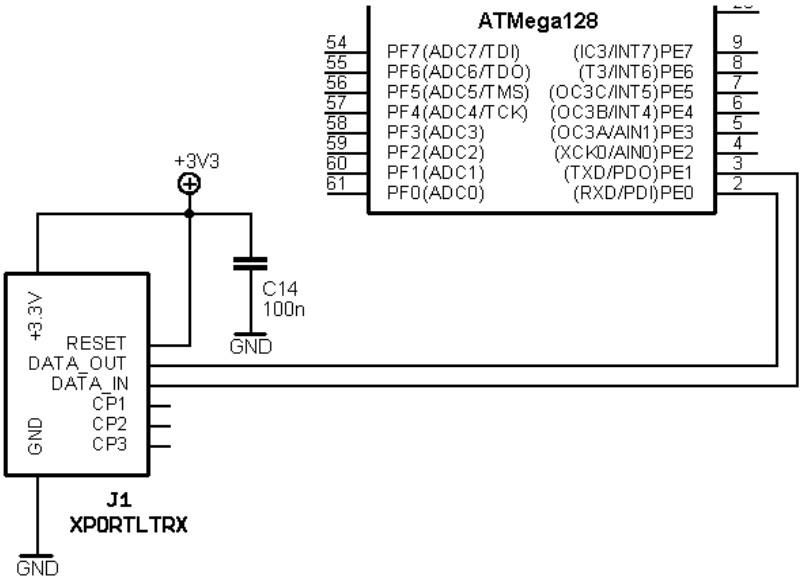
- Protokoly: TCP/IP, UDP/IP, ARP, ICMP, SNMP, TFTP, Telnet, DHCP, BOOTP, HTTP, AutoIP.
- Sériová linka: CMOS (Asynchronní, 5V Tolerant) Podpora RS422 a RS485. Rychlost 300 Bd až 921600 Bd.
- Interní WEBové stránky (384 kB).
- Odesílání e-mailů.
- Na přání k dispozici varianta (XPort SE) s šifrováním (256-bit AES Rijndael).
- Ethernet 10Base-T nebo 100Base-TX (Automatické rozpoznání).
- Provedení v konektoru RJ45.
- Výkonný processor (12 MIPS, založen na rozšířeném 16 bit DSTni-EX, 48MHz nebo 88MHz, architektura x86).
- Napájení 3,3V.
- Vývojový kit pro snadné ladění aplikací s modulem XPort.
- Certifikát RoHS a WEEE.

**XPort - Použití**

- Vývoj a výroba zařízení s Ethernetovou konektivitou.
- Snadné připojení Vašich stávajících zařízení se sériovým rozhraním do Ethernetu.
- Ovládání zařízení přes webové rozhraní.
- Průmyslová i domácí automatizace.
- Informační systémy.
- Měřící přístroje s Ethernetovým rozhraním.

## 5.3 Použití modulu s mikrokontrolérem

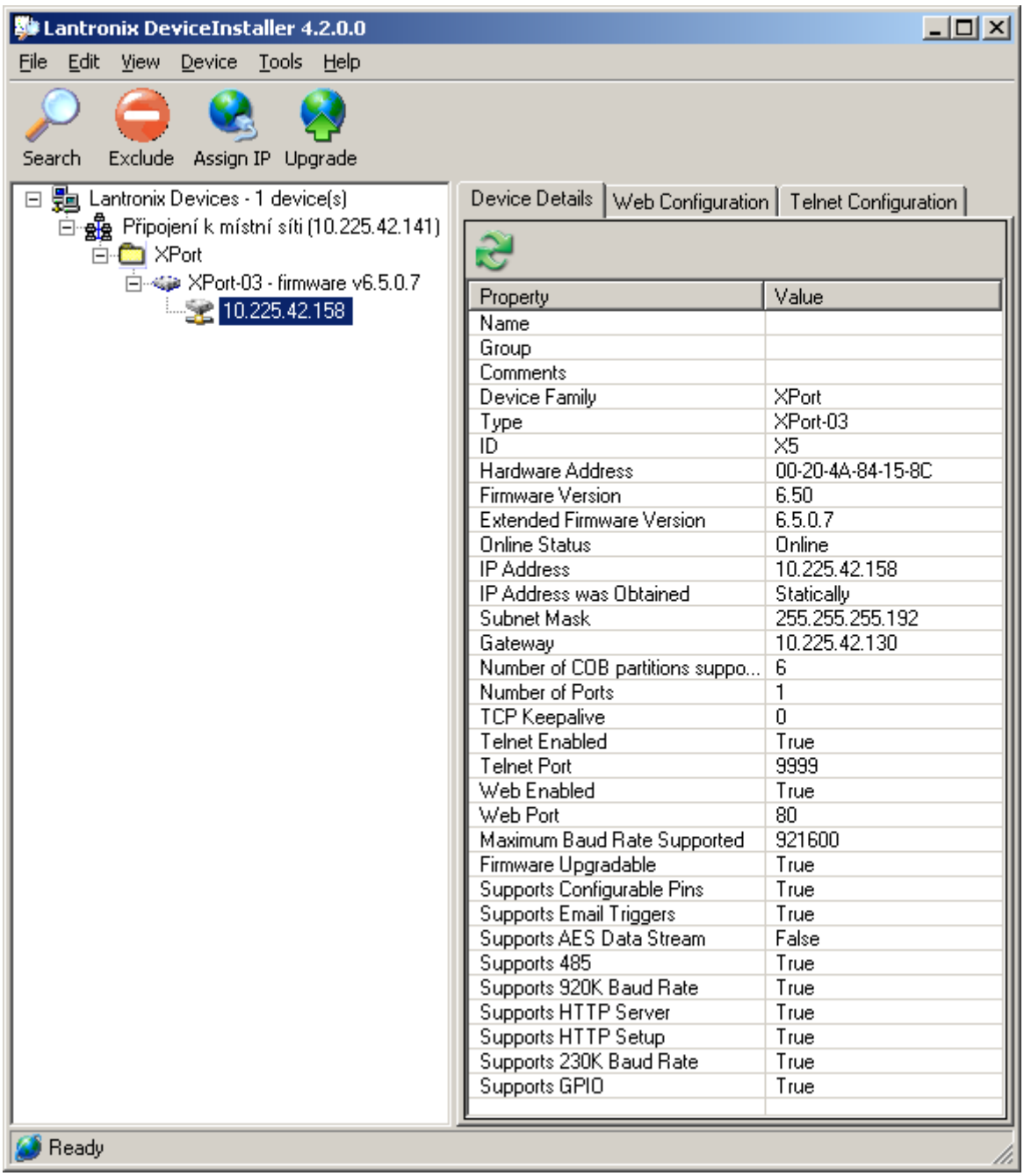
Jak již bylo řečeno, modul XPort slouží v zásadě jako převodník sériové linky na ethernet. Na straně sériové linky může komunikovat pomocí protokolů RS232, RS422 (4-wire) a RS485 (2-wire), na straně ethernetu může komunikovat pomocí TCP nebo UDP, může být trvale připojen k jinému zařízení anebo může akceptovat příchozí připojení na daném portu od dalších zařízení. V tomto projektu byl nastaven pro sériovou RS232 linku a pro akceptování příchozích připojení, kdy po přijetí připojení a určitého datového slova, toto slovo převede na sériovou linku procesoru a ten vyvolá přerušení a provede akci dle příchozího datového slova (odeslání dat, akceptace nastavení,...). Připojení modulu XPort k mikroprocesoru je znázorněno na následujícím obrázku:




Obr. 5.2 – Připojení modul k mikroprocesoru

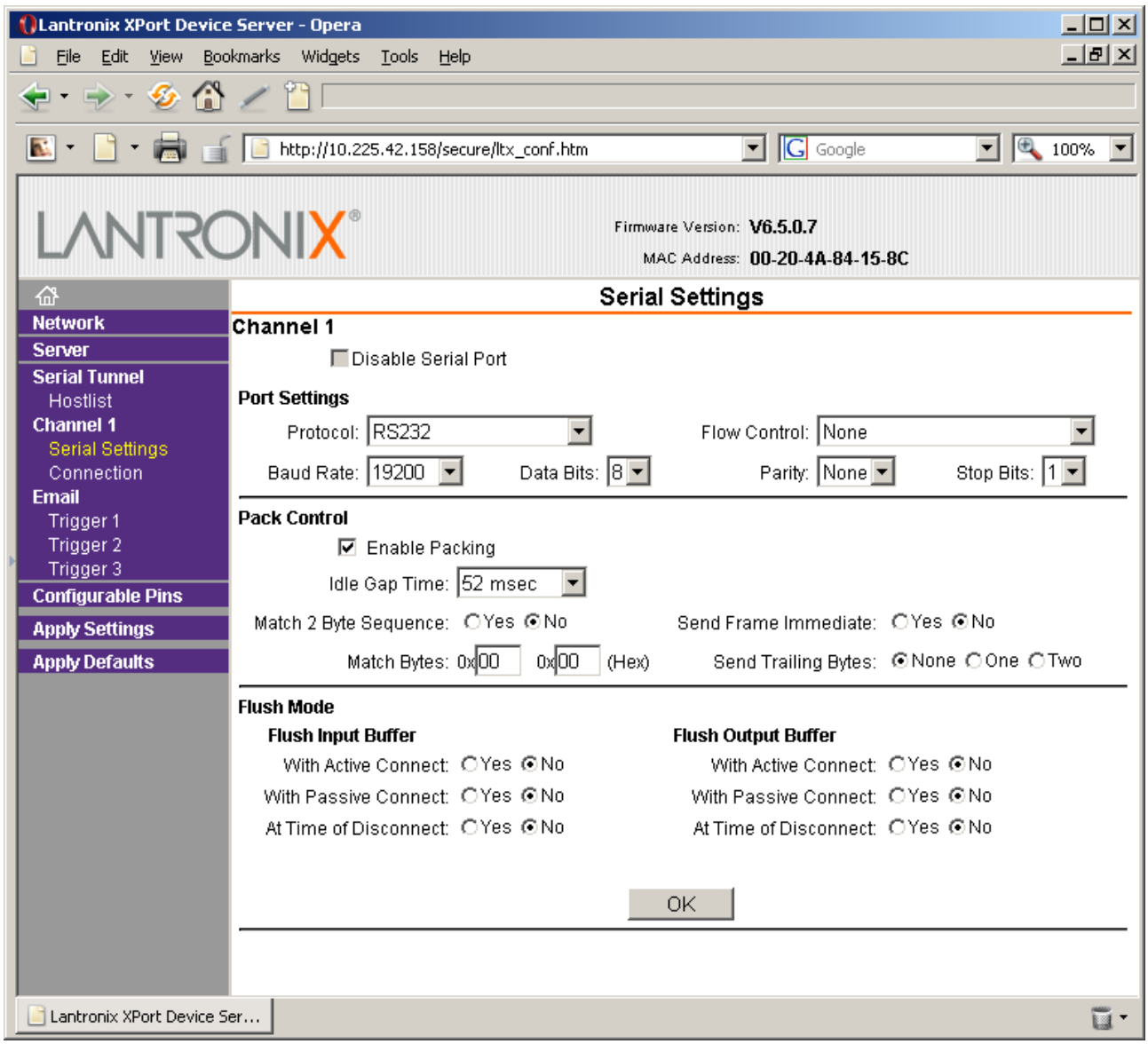
Z obrázku je patrné, že připojení je velmi jednoduché, stačí propojit patřičné datové piny XPortu s piny sériové linky (RXD/TXD) u mikroprocesoru.

Konfiguraci modulu je možné provádět několika způsoby. Nejjednodušší možností je pomocí volně stažitelného programu výrobce – firmy Lantronix, DeviceInstaller. Tento program pro operační systém Windows umožňuje zjistit veškeré parametry modulu, a co je nejdůležitější, přiřadit k němu IP adresu. DeviceInstaller totiž dokáže najít modul na ethernetové síti podle jeho HW (MAC) adresy. Další možností přiřazení IP adresy je pomocí ARP protokolu. Příkazem arp se k MAC adrese modulu přiřadí adresa IP, následně se k této IP připojí telnetem na port 1 a potom na port 9999, kde se již objeví interaktivní menu, v němž je možno mimo jiné nastavit IP adresu, masku a další síťové parametry modulu. Obdobně je možno IP adresu nastavit i přes sériovou linku, tyto dvě možnosti umožňují prvotní nastavení i pod jinými operačními systémy než MS Windows.



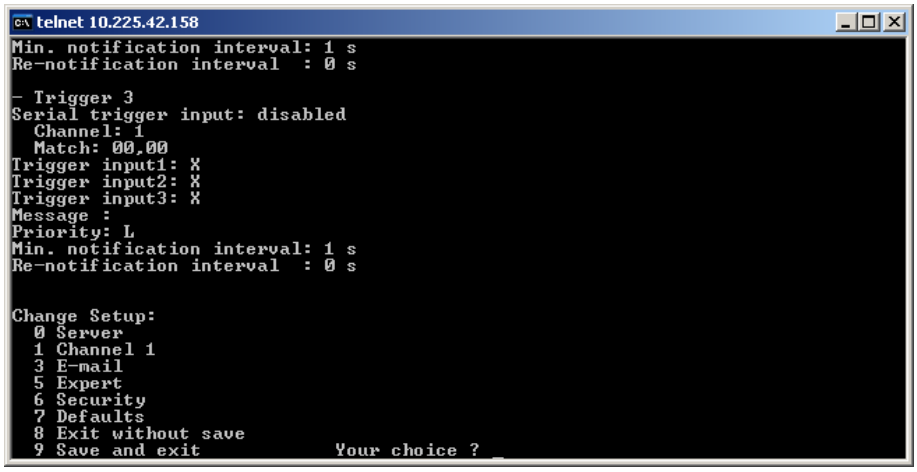

Obr. 5.3 – Ukázka zobrazení parametrů modulu v DeviceInstalleru

Jakmile je k modulu přiřazena IP adresa, můžeme pokračovat v konfiguraci přes WWW rozhraní, přístupné na definované adrese na standardním portu 80:




Obr. 5.4 – Ukázka možností nastavení v prohlížeči

Zde je již možno nastavit veškeré další možnosti a parametry modulu, stejně tak i v případě použití TELNETu:




Obr. 5.5 – Ukázka možností nastavení pomocí TELNETu

# 6. Procesor Atmel ATMega128

## 6.1 Základní vlastnosti

Jedná se o 8-bitový mikropočítač s RISC jádrem, s vysokým výkonem a nízkou spotřebou

Vysoký výkon až 16 MIPS při 16MHz (většina instrukcí se provede v 1 cyklu)

Nízká spotřeba, podpora několika power módů

Poměrně velká velikost vnitřních pamětí

Široké spektrum integrovaných periferií

Možnost programování přímo na desce přes rozhraní SPI

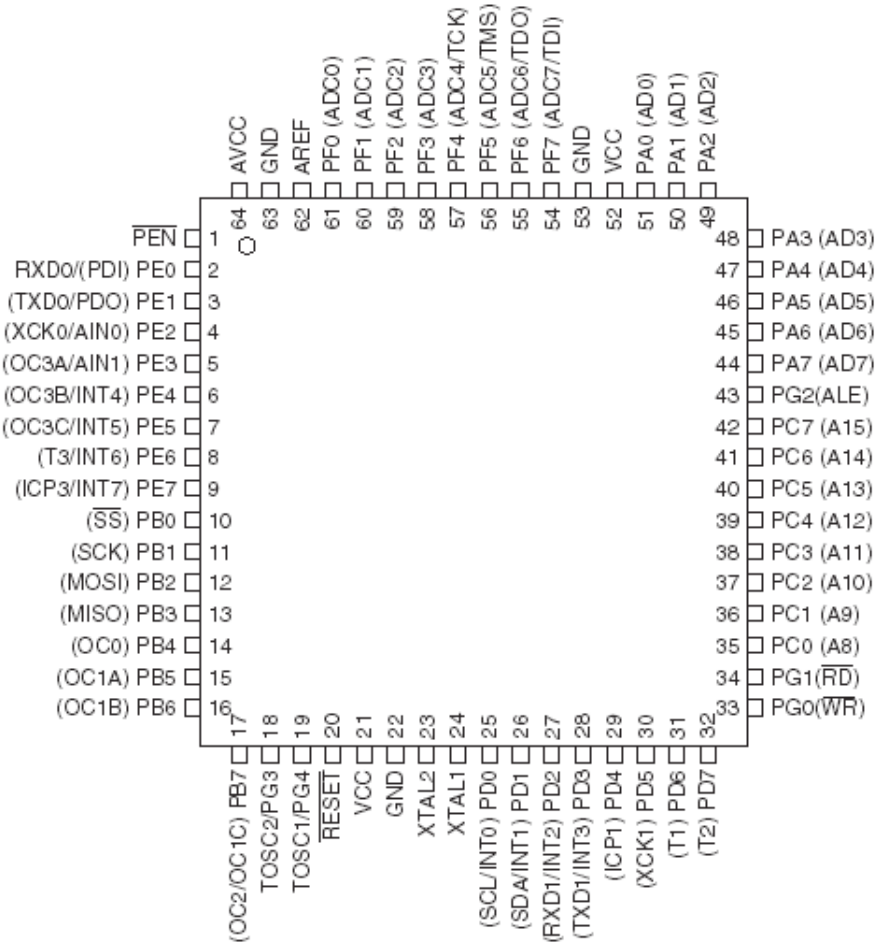
Možnost programování a ladění programu přímo v mikroprocesoru přes rozhraní JTAG

Dostupnost vývojových nástrojů zdarma, programování pomocí jazyka C

**Vlastnosti:**

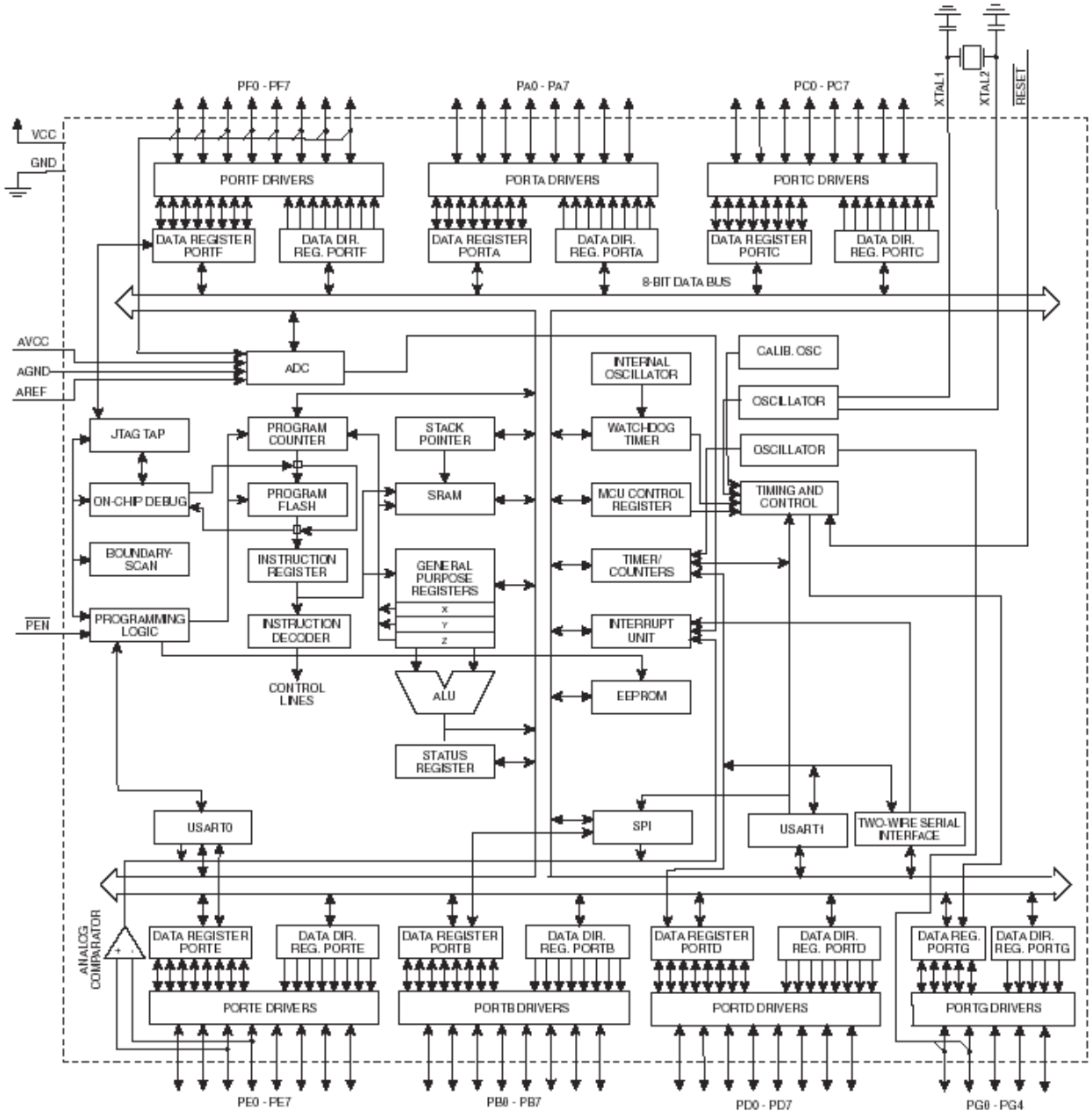
- 133 instrukcí, většina se provádí v jednom hodinovém cyklu
- 32x8 funkčních registrů + registry řízení periferií
- Dvoucyklový násobič přímo na čipu
- 128Kbytů Flash paměti pro program
- 4Kbyty paměti EEPROM
- 4Kbyty interní SRAM
- Celkem 10 000 přepisovacích cyklů pro Flash, 100 000 cyklů pro EEPROM, trvanlivost dat je udávána na 20 let při 85°C a na 100 let při 25°C
- Dalších 64KB možného rozšíření paměti
- Podpora ladění programu přímo na čipu
- programování Flash, EEPROM, pojistek a Lock bitů přes rozhraní JTAG
- Dva osmibitové čítače/časovače s oddělenými módy
- Dva rozšířené šestnáctibitové čítače/časovače s oddělenými módy
- Real Time čítač s odděleným oscilátorem
- Dva osmibitové PWM kanály
- 6 PWM kanálů s programovatelným rozlišením od 2 do 16 bitů
- 8-kanálový, 10-bitový ADC
- Two wire sériový interface

- Duální programovatelné sériové USART
- Sériový SPI interface (master/slave)
- Programovatelný Watchdog s integrovaným oscilátorem
- Integrovaný analogový komparátor
- Interní kalibrovaný RC oscilátorem
- Vnější i vnitřní zdroje přerušení
- Šest sleep módů
- Softwarově zjistitelná rychlost hodin
- Volitelná kompatibilita s ATMega103
- Globálně vypínatelný pull-up na portech
- 53 programovatelných vstupů/výstupů
- Rozsah napájecího napětí 4,5 – 5,5V pro ATMega128, 2,7 – 5,5V pro ATMega128L
- Možné rychlosti hodin 0 – 16MHz pro ATMega128, 0 – 8MHz pro ATMega128L




Obr. 6.1 – Rozložení vývodů ATMega128 [8]

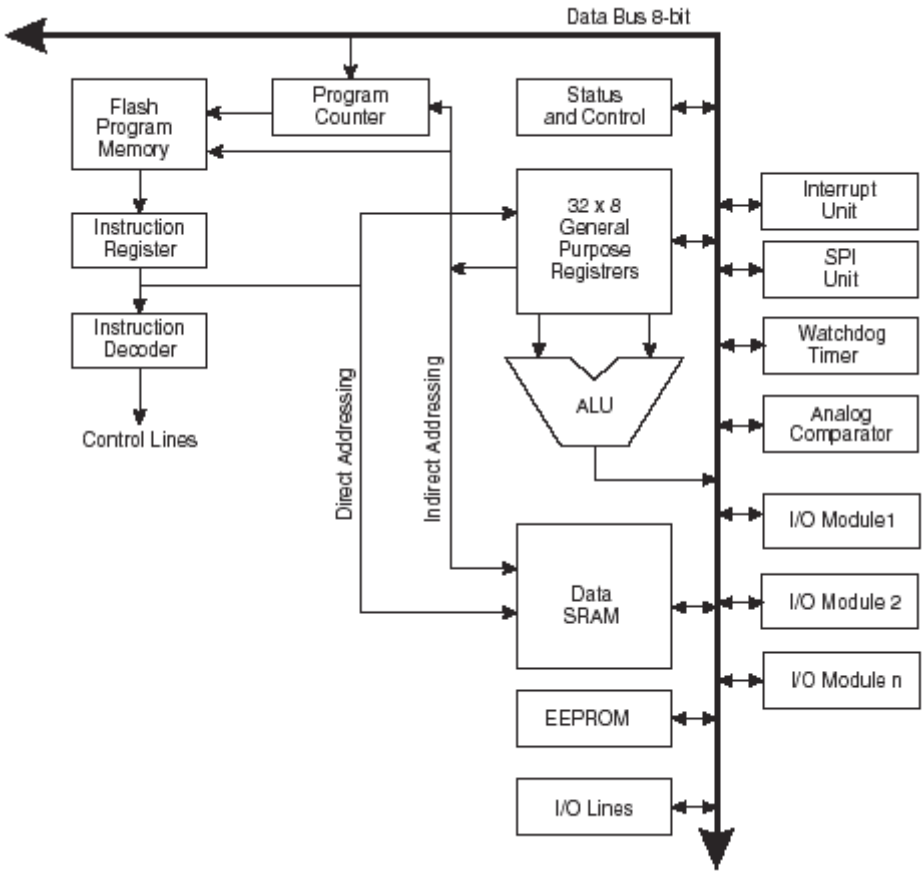
ATMega128 je tedy 8-bitový CMOS mikropočítač, založený na AVR RISC architektuře. Prováděním instrukcí pouze v jednom hodinovém cyklu dosahuje propustnosti 1 MIPS na MHZ, což umožňuje dosáhnout výhodného kompromisu mezi výkonem a spotřebou.




Obr. 6.2 – Blokové znázornění vnitřní struktury ATMega128 [8]

## 6.2 Jádro procesoru AVR

Procesor ATMega128 je založen na AVR jádru typu RISC Harvardské architektury. To se stará o provádění programu, přístup k pamětem a periferiím a obsluhu přerušení. Pro zrychlení provádění instrukcí má jednoúrovňovou pipeline, kdy během provádění jedné instrukce se následující instrukce načítá z programové paměti.



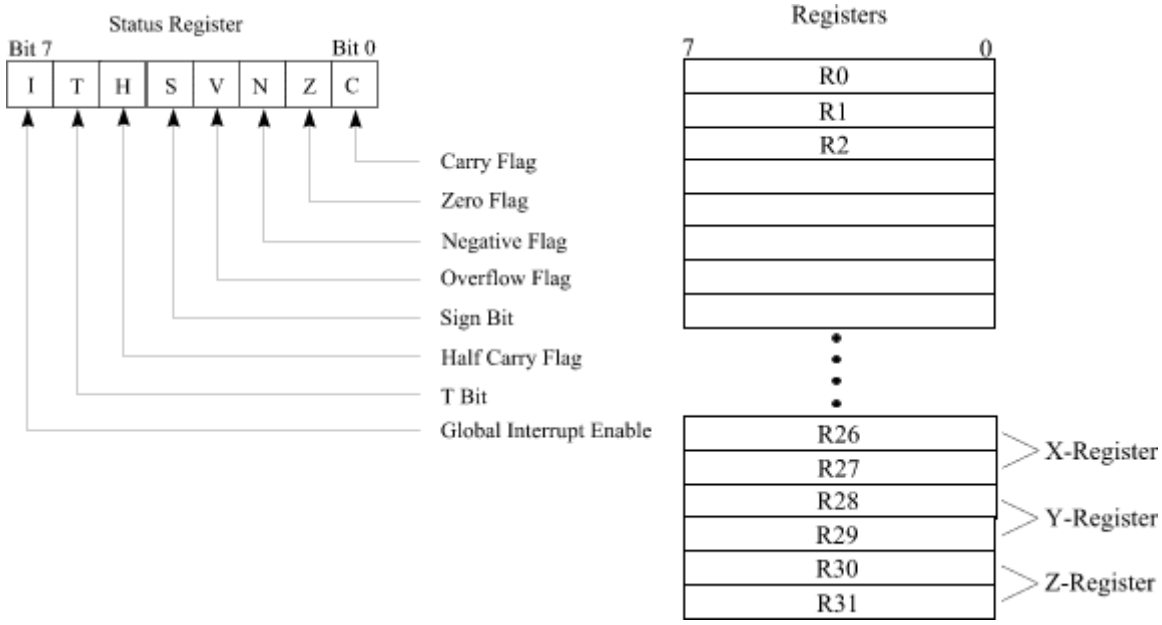

Obr. 6.3 – Blokové znázornění jádra AVR [8]

Jádro AVR-série se podobá jádru většiny RISC-procesorů, které jsou dostupné na trhu. AVR jádro se skládá ze 32 stejných 8-bitových registrů, které mohou obsahovat jak data tak adresy. Posledních 6 registrů můžeme ve dvojici použít jako ukazatele adresy pro nepřímé adresování paměti dat. Tyto registry označované písmeny X, Y a Z dovolují libovolné ukládací operace (Load/ Store). Programátor má například na výběr, zda ukazatel adresy bude po zpracování určité instrukce inkrementovat nebo před zpracováním této instrukce dekrementovat. Užitečné je pro adresování využít možnosti 6-bitového posunu v ukazateli adresy v dvojitých registrech Y a Z.

AVR architektura má 5 adresovacích módů pro paměť dat:

- přímé adresování
- nepřímé adresování s posunutím (6-bitový posun)
- nepřímé adresování
- nepřímé adresování s dekrementací ukazatele adresy před zpracováním instrukce
- nepřímé adresování s inkrementací ukazatele adresy po zpracování instrukce

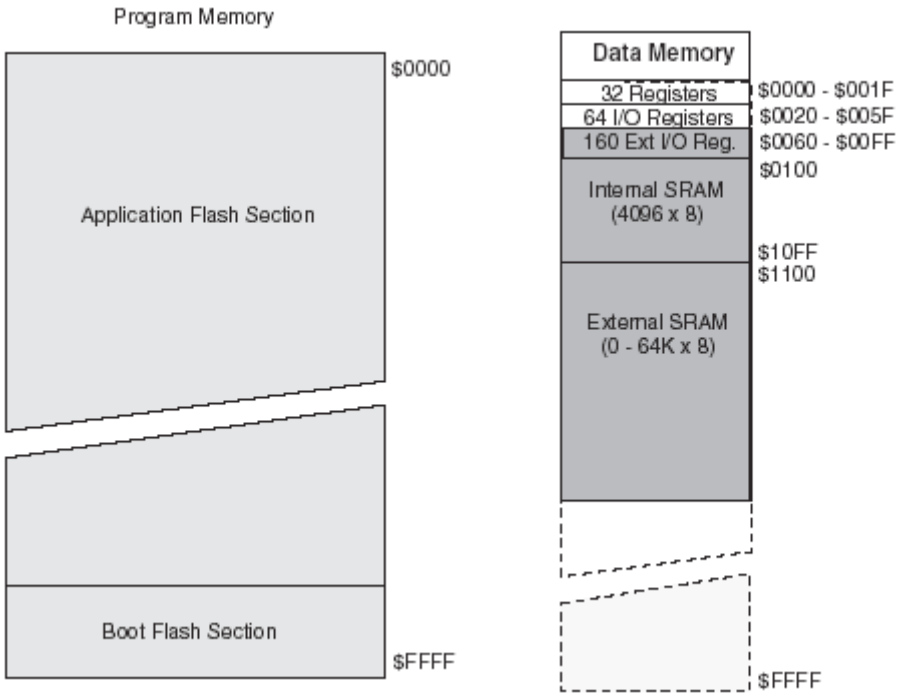
Status registr obsahuje příznak přetečení, přenos do vyššího řádu, znaménko a další příznaky, jak je znázorněno na následujícím obrázku:




Obr. 6.4 – Status registr a 32 víceúčelových registrů [8]

Jak je u mnoha jednoduchých mikroprocesorů obvyklé, registry jsou zobrazovány přímo v adresovém prostoru dat. Prvních 32-bytů paměti (0x00 až 0x1F) odpovídá registrům R0 až R31., ty jsou také přímo zamapovány do paměti dat. Proto je možno s každým registrem zacházet použitím standardních odkazů bez toho, aby programátor potřeboval znát řídící instrukce registrů. AVR používá Harvardskou architekturu, tj. paměť programu a paměť dat jsou odděleny. Paměť programu je přístupná pomocí dvoustupňové pipeline. Když se určitá instrukce začíná vykonávat, další instrukce je připravována k zpracování. Tato konstrukce dovoluje zpracovávání instrukcí vždy v jednom hodinovém cyklu.

Programová i datová paměť má lineární adresování. Jelikož všechny instrukce AVR jsou dlouhé 16 nebo 32 bitů, je paměť organizována jako 64K x 16. Programová paměť Flash se dělí na aplikační a oblast zavaděče (bootcode). Každá oblast má svoji pojistku proti čtení/zápisu. V zaváděcí oblasti může být uložen program zavaděče, který přes nějaké rozhraní přijme aplikační program a pomocí instrukce SPM (nelze ji volat programem z aplikační oblasti) jej zapíše do aplikační paměti a pak mu předá řízení. Tento mechanismus lze využít pro programování i přes jiné rozhraní, než k tomu určené SPI/JTAG. Zásobník sdílí interní paměť SRAM a roste směrem dolů. Před voláním podprogramů nebo obsluh přerušení je nutné nastavit ukazatel zásobníku (SP) tak, aby byl k dispozici dostatek paměti pro návratové adresy. Každé přerušení lze individuálně nebo globálně povolit/zakázat. Priorita přerušení je daná adresou vektorů přerušení (čím nižší adresa vektoru, tím vyšší priorita).



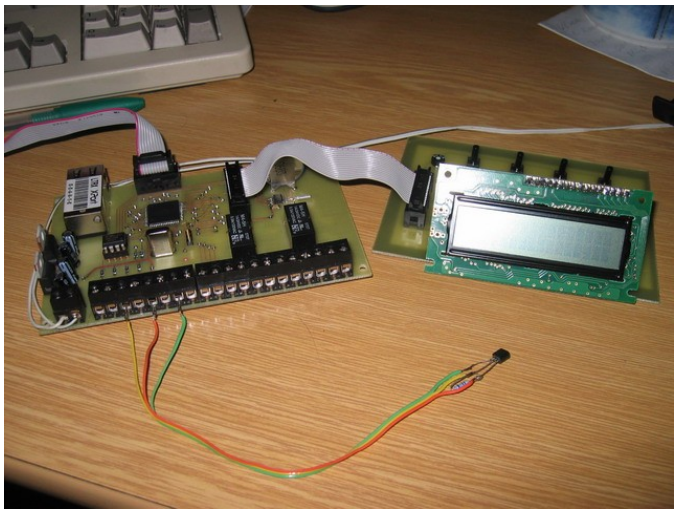

Obr. 6.5 – Mapa paměti ATMega128 [8]

ATMega128 dále obsahuje 4kB datové paměti EEPROM. Ta je organizována jako oddělený datový prostor, v němž může být každý byte adresován jednotlivě.

# 7. Realizace měřícího systému

## 7.1 Návrh hardware

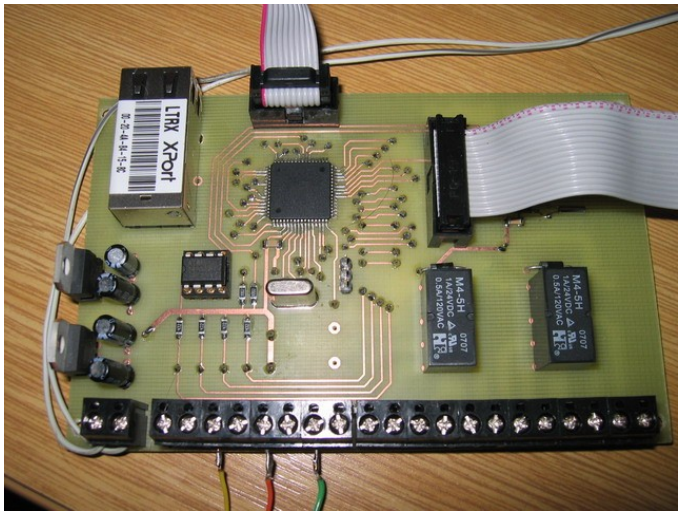
Hardware měřícího systému se skládá ze dvou části – hlavní desky a klávesnice s displejem.



Obr. 7.1 – Měřící systém

## 7.2 Hlavní deska

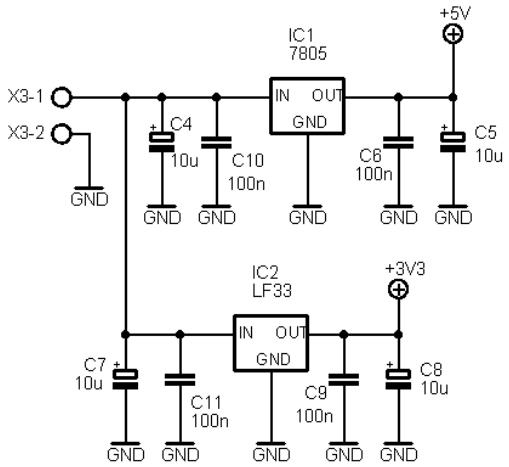
Tvoří „srdce“ měřícího systému. Obsahuje řídící mikroprocesor, hodiny, paměť pro uchování naměřených dat, měřící vstupy, alarmové výstupy, interface pro komunikaci po síti Ethernet.




Obr. 7.2 – Detailní pohled na hlavní desku

### 7.2.1 Napájení

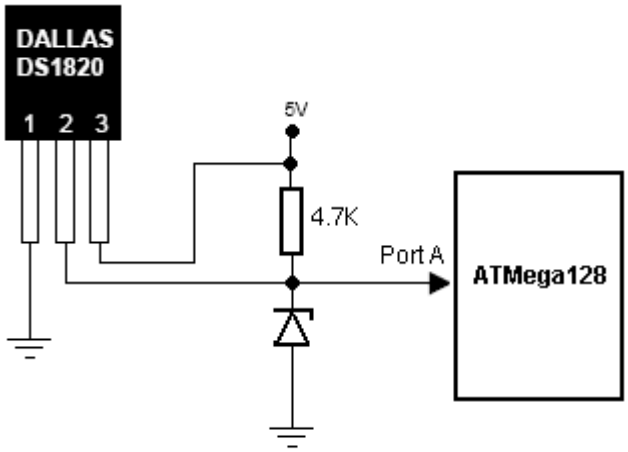
Napájení modulu je možné realizovat prakticky jakýmkoliv zdrojem napětí v rozsahu 6 – 15V, případně z baterie. O stabilizaci a filtraci napětí pro chod součástek na desce se starají běžné stabilizátory. Typ LF33 je stabilizátorem napětí 3,3V, které je potřeba pro napájení XPortu, 7805 je klasický 5V stabilizátor potřebný pro napájení všech ostatních aktivních součástek.




Obr. 7.3 – Schéma napájení modulu

### 7.2.2 Měřící vstupy

Modul obsahuje celkem 4 vstupy, kdy ke každému z nich je možno připojit (skrze sběrnici 1-wire) až 8 teplotních čidel DS18S20. Celkem tedy může modul měřit až 32 různých teplot.

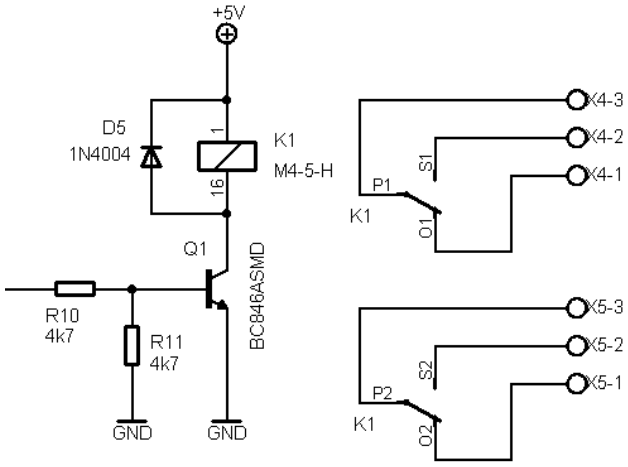



Obr. 7.4 – Připojení teploměru DS18S20 k modulu

Vstupy jsou připojeny na port A procesoru, konkrétně na piny 4 – 7. Sběrnice je typu ??, na desce jsou tedy ke vstupům připojeny 4k7 pullup rezistory, ochrana před nežádoucím přepětím na vedení je zajištěna zenerovými diodami.

### 7.2.3 Alarmové výstupy

Modul dále obsahuje dva alarmové výstupy. Ty složí jako spínané kontakty, jenž mohou obsluhovat např. akustické nebo světelné výstražné znamení, upozorňující na překročení mezí některé z měřených teplot.




Obr. 7.5 – Alarmový výstup

Alarmové výstupy jsou tvořeny běžným 5V relé, spínaným z portu D5 (resp. D6) přes NPN tranzistor.. Relé je dvou kontaktové, může tedy spínat/rozpínat dva výstupy současně.

### 7.2.4 Další periferie

Hlavní deska modulu dále obsahuje konektor pro připojení desky s displejem a klávesnicí, konektor JTAG pro připojení programovacího kabelu, modul XPort pro ethernetovou konektivitu a vyvedený druhý sériový port procesoru, pro případné připojení dalších zařízení, jako např. GSM telefonu pro odesílání alarmových SMS zpráv.

## 7.3 Softwarové řešení

### 7.3.1 Vývojové nástroje

Pro mikroprocesory řady Atmel AVR je k dispozici široká řada vývojových nástrojů a kompilátorů, ať už zdarma vyvíjených komunitou jako open-source software nebo dodaných výrobcem – firmou Atmel, anebo komerčních, např. CodeVision.

Programování probíhá v jazyce C/C++, je proto poměrně intuitivní, navíc je překlad pro AVR procesory dobře optimalizovaný, jelikož bylo při jejich vývoji počítáno s tímto programovacím jazykem.

**AVR Studio**

Je balík nástrojů pro práci s mikroprocesory AVR dodávaný jejich výrobcem – firmou Atmel. Obsahuje vývojové prostředí (IDE), kompilátor kódu v Assembleru a programátor jednočipů AVR.

AVR studio také umožňuje ladění externích kódů, takže je možné využít jeho schopností pro ladění již zkompilovaných programů např. z jiných vývojových prostředí. Zpracovává formáty UBROF (IAR), Noric (AVR Assembler), COFF (GCC, ImageCraft, Codevision, ELAB, atd.) a Intel-HEX.

Kromě editace a ladění programu umožňuje prostředí i samotné programování mikrokontroleru. Podporovány jsou systémy ICE50, JTAGICE, STK500/5001 a AVRISP

**WinAVR**

Balík WinAVR obsahuje vše potřebné pro přeložení zdrojového kódu a naprogramování jednočipu:

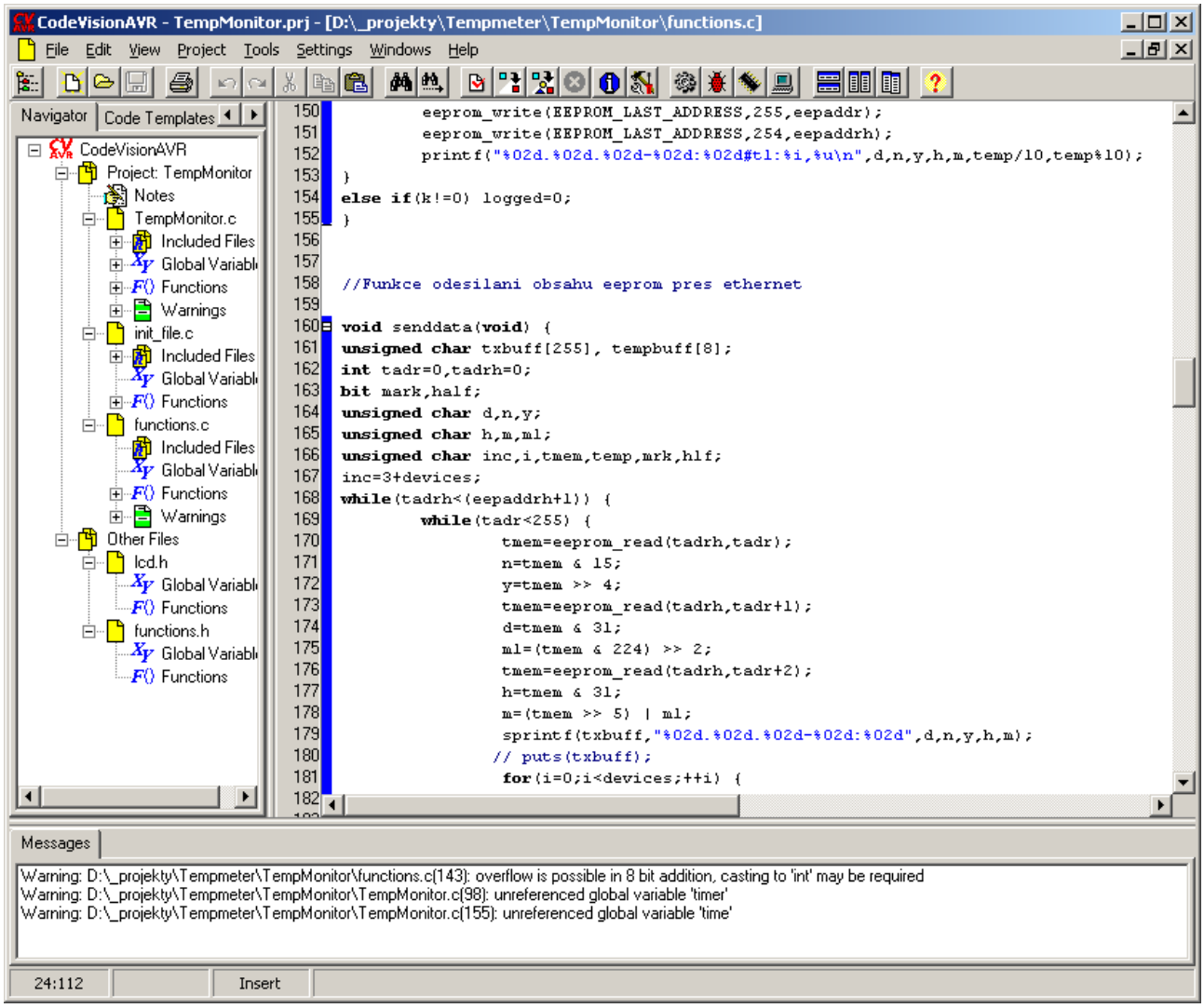
- GCC – překladač ANSI C a C++,
- Binutils – nástroje pro tvorbu a práci s binárními soubory (linker, assembler, atd.),
- AVR-libC – knihovna standardních C funkcí pro AVR (detailní popis v [27]),
- AVRdude – programátor jednočipů AVR s širokou podporou hardware + GUI,
- GDB – debugger + GUI,
- AvarICE – podpora pro debugování přes JTAG kabel pro GDB,
- SimulAVR – podpora simulace pro GDB,
- Programmer's Notepad – textový editor,
- PDF/HTML dokumentace a další.

Je vidět, že tento balík je v podstatě poskládaný z různých dalších převážně open source projektů. Jako takový nemá žádné vlastní vývojové prostředí, pro tyto účely lze využít přiložený Programmer's Notepad, nebo jej propojit s AVR Studiem, které při kompilaci kódu v jazyku C využije kompilátor a knihovny z balíku WinAVR.

## CodeVision AVR

Je komerční překladač a vývojové prostředí od Pavla Haiduca z firmy HP InfoTech. Je znám též pod zkratkou CVAVR nebo jen jako Codevision. Jedná se o plnohodnotný vývojový systém pro mikroprocesory AtmelAVR, dostupný ve dvou verzích - plná (Standard), která generuje kód pro všechny klasické AVR a ATMega série, a odlehčená (Light) verze, která vytváří kód pouze pro klasickou AVR řadu (čísla součástek začínající AT90S....). K dispozici bude též verze pro součástky bez statické paměti SRAM (TinyAVR a AT90S1200) nazývaná CodeVisionAVR Tiny. Tato verze je ke stažení bezplatně z webu Codevision a je omezená velikostí programu, který je možné kompilovat. Limit velikosti programu však stále ještě dovoluje velmi slušnou práci se systémem. Pro vyzkoušení programu je k dispozici zkušební verze s omezenou velikostí výsledného kódu.

Velká obliba vývojových prostředků CodeVisionAVR je bezesporu dána také jejich nízkou cenou, respektive skvělým poměrem cena/výkon.




Obr. 7.6 - Ukázka vývojového prostředí CVAVR

**Základní vlastnosti**

- 32bitová aplikace, spustitelná pod Windows 95/98/NT4.0/2000/XP;
- integrované vývojové prostředí a kompilátor jazyka C se snadným použitím;
- editor s automatickým odrážkováním a zvýrazněním syntaxe;
- podpora datových typů bit, char, int, short, long, float;
- podpora specifických rozšíření mikrokontrolérů AVR
- přístup k paměťovým polím EEPROM & FLASH
- přístup k registrům na bitové úrovni
- podpora přerušení;
- rozsáhlé možnosti optimalizace výsledného kódu kompilátoru včetně
- odstranění přebytečného kódu
- optimalizace pro paměťové modely Tiny (8 bit ukazatel pro obvody s pamětí RAM do 256 bytů) a Small (16 bit ukazatel pro obvody s více než 256 byty RAM)
- volby optimalizace výsledného kódu pro rychlost nebo velikost programu;
- možnost vkládání asembleru do zdrojového kódu C;
- ladění aplikace na úrovni zdrojového kódu C s výstupem do formátu COFF umožňující využití I/O terminálu z debugeru Atmel AVR Studia;
- plně slučitelný s emulátory In-Circuit ATMEL ICE200, JTAG-ICE a dalšími;
- vestavěný sériový komunikační terminál RS232, RS422, RS485 pro ladění aplikací;
- vestavěný programátor In-System pro AVR s automatickým programováním po úspěšné kompilaci, kompatibilní s
- ATMEL STK500/STK501/STK502/AVRISP
- Kanda Systems STK200+ a STK300
- Vogel Elektronic VTEC-ISP
- Dontronics DT006
- Tietomyrsky EXB2313
- 4Ahead AVR Board 1
- Futurlec JR-AVR AT90S2313 a AT90S8535

### 7.3.2 Popis funkcí a knihoven

**Základní princip obsluhy 1-wire sbernice**

Jak již bylo nastíněno v popisu výše, je komunikace po 1-wire sběrnici z velké části založena na správném časování. Proto je potřeba nejprve správně navrhnout funkce pro časové zpoždění.

Pakliže máme relativně přesné funkce pro časové zpoždění, přichází na řadu samotná komunikace, nejprve je potřeba provést reset sběrnice:

```
//---------------------------------------
// Reset DS1820
//---------------------------------------
Bool ResetDS1820(void)
{
       Bool presence;
       DQ = 0;             // prizemneni sbernice
       Delay(480);         // a nechat tak po cca 480us
       DQ = 1;             // navraceni zpet na napajeci napeti
       Delay(60);          // cekani na odezvu 60us
       presence = DQ;      // precteni odezvy
       DelayUs(316);       // cekani na konec timeslotu 316us
       return(presence);   // vraci stav odezvy
}      // 0 = pritomno, 1 = nic tam neni
```

Funkce nejprve vyšle na sběrnici 0 po dobu 480 μs a poté ji vrátí zpět na výchozí stav, tedy log. 1. Poté se čeká 60 μs a pokud se na sběrnici vyskytuje nějaké 1-wire zařízení, odpoví v této době 0 V presence pulsem. Vrácená hodnota této funkce pak odpovídá zjištěnému stavu: 0 – čidlo nalezeno, 1 - na sběrnici se nevyskytuje 1-wire zařízení.

Pro čtení a zápis bitu na sběrnici slouží tyto funkce:

```
//---------------------------------------
// Precte 1 bit z DS1820
//---------------------------------------
Bool ReadBit(void)
{
       unsigned char i=0;
       DQ = 0;      // prizemneni sbernice pro spusteni timeslotu
       DQ = 1;
       Delay(50); // cekani 50us od startu timeslotu
       return(DQ); // vrati hodnotu DQ
}
```

Komunikace začíná vysláním 0 V pulsu na sběrnici a poté se čeká 50 μs a vrátí se hodnota stavu sběrnice. Pokud chce senzor odeslat nulový bit, bude v této době držet sběrnici na hodnotě 0, jinak sběrnice zůstává na hodnotě napájecího napětí.

```
//----------------------------------------
// Zapise 1 bit do DS1820
//----------------------------------------
void WriteBit(Bool Dbit)
{
       unsigned char i=0;
       DQ = 0;
       DQ = Dbit ? 1:0;
       Delay(60);                // zpozdeni 60us
       DQ = 1;
}
```

Princip je podobný funkci předchozí, opět se začíná 0 V pulsem, který se v případě odesílání nulového bitu protáhne na 60 μs, jinak se sběrnice hned vrátí na hodnotu 1.

```
//----------------------------------------
// Precte 1 byte z DS1820
//----------------------------------------
unsigned char ReadByte(void)
{
       unsigned char i;
       unsigned char Din = 0;
       for (i=0;i<8;i++)   // precte byte, bit po bitu
       {
              Din|=ReadBit()? 0x01<<i:Din;
              DelayUs(6);
       }
       return(Din);
}

//----------------------------------------
// Zapise 1 byte
//----------------------------------------
void WriteByte(unsigned char Dout)
{
  Bool bit = 1;
       unsigned char i;
       for (i=0; i<8; i++) // zapise byte, bit po bitu
       {
              WriteBit((Bool)(Dout & 0x1));
              Dout = Dout >> 1;
       }
       DelayUs(5);
}
```

Tyto funkce slouží k odeslání celého datového slova, tedy postupně pomocí funkcí ReadBit/WriteBit odešlou/přijmou všech jednotlivých 8 bitů.

Tyto výše uvedené postupy používá hlavní funkce pro vlastní měření teploty:

```
//---------------------------------------
// Hlavni funkce cteni teploty
//---------------------------------------
void ReadTemp(unsigned char * buff)
{
unsigned char n;

if (ResetDS1820() == 0) {
    WriteByte(0xcc);  // skip ROM
    WriteByte(0x44);  // konverze teploty
    while (ReadByte()==0xff); // cekani na dokonceni konverze
    ResetDS1820();
    WriteByte(0xcc);  // skip ROM
    WriteByte(0xbe);  // precteni vysledku

    for (n=0; n<9; n++)     // precte 9 bytu a nasklada se do pole
    {
       buff[n]=ReadByte();  // cteni z DS1820
    }
}
}
```

Funkce provede reset sběrnice a pokud je na ní přítomen teploměr (návratová hodnota funkce ResetDS1820() je 0), realizuje vlastní měření, jehož výsledky uloží do pole bufferu.

Výše uvedený kód slouží pouze pro ilustraci možné obsluhy sběrnice a integrovaného teploměru. Při vývoji projektu bylo využito již hotových knihoven pro obsluhu 1-wire sběrnice a teploměru DS1820, jenž jsou součástí kompilátoru CVAVR.

**Knihovna pro obsluhu teploměru DS18S20**

Knihovna DS1820 (hlavičkový soubor ds1820.h) obsahuje veškeré funkce potřebné pro obsluhu tohoto teploměru.

Funkce `unsigned char ds1820_select(unsigned char *addr)` zajišťuje výběr konkrétního čidla. Jejím parametrem je hexadecimální adresa čidla, zjištěná dříve. Funkce vrací hodnotu nula, pokud se nepodařilo inicializovat 1-wire sběrnici, hodnotu jedna pokud inicializace a zápis adresy proběhl v pořádku. Tato funkce je používána dalšími obslužnými funkcemi teploměru.

Funkce `unsigned char ds1820_read_spd(unsigned char *addr)` čte ScratchPad, tedy vnitřní paměť čidla (viz kap. 3.2). Jejím parametrem je adresa čidla, kterou předává funkci pro výběr. V případě neúspěchu výběru čidla vrací hodnotu nula, jinak vrátí obsah paměti ScratchPad. Tato funkce je opět používána ve funkcích dále.

Funkce `int ds1820_temperature_10(unsigned char *addr)` je v zásadě hlavní a nejpoužívanější funkcí této knihovny. Přečte z čidla, definovaného adresou jako parametr funkce, aktuálně naměřenou teplotu. Ta je čtena s přesností na jednu desetinu, ovšem jako celé číslo. Je tedy nutné zjištěnou hodnotu vydělit deseti pro korektní zobrazení teploty ve stupních celsia.

Funkce `unsigned char ds1820_set_alarm(unsigned char *addr,signed char temp_low,signed char temp_high)` slouží k nastavení vnitřních alarmů teplotního čidla. Jejími parametry jsou adresa čidla, spodní teplotní limit pro alarm a horní teplotní limit pro alarm.

Po nezbytné inicializaci periferií dochází periodicky po jedné sekundě ke čtení teploty z čidla a jejímu zobrazení na sériové lince.

Tento princip obsluhy 1-wire sběrnice je velmi jednoduchý, ovšem na druhou stranu také velmi transparentní a tím také vhodný pro ověření funkce této komunikace.

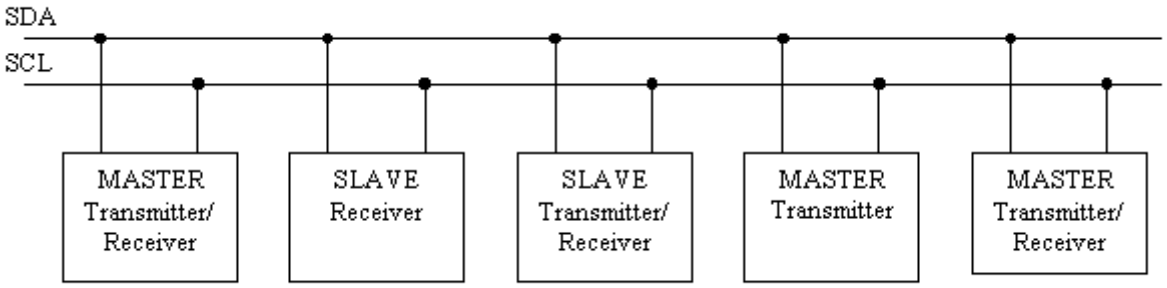
**Funkce pro komunikaci po $I^2C$ sběrnici**

Sběrnice $I^2C$ ($I^2C$-bus, Inter-IC-bus) je dvouvodičové datové propojení mezi jedním nebo několika procesory (Masters) a speciálními periferními součástkami (Slaves). Všechny součástky jsou připojeny na téže sběrnici a jsou cíleně vybírány svými adresami. Adresy i data se přenášejí týmiž vodiči. Sběrnice umožňuje velmi jednoduché propojení mezi několika integrovanými obvody a bezproblémové dodatečné rozšiřování.

Protokol sběrnice

K této sběrnici mohou být připojeny všechny integrované obvody, které zvládají její speciální protokol. Mimo integrovaných obvodů RAM, EEPROM , obvodů pro rozšíření portů, A/D a D/A převodníků a obvodů hodinových signálů existuje ještě celá řada speciálních integrovaných obvodů, jako například budiče displejů nebo integrovaných obvodů pro televizní a audio techniku. Sběrnice $I^2C$ používá sériovou datovou linku SDA a linku hodinového signálu SCL. Data a adresy

se přenášejí podobně jako v posuvných registrech společně hodinovými impulsy. Obě linky je možno používat jako obousměrné. Jsou vybaveny zvyšovacím (pull-up) odporem a mohou být každým účastníkem sběrnice staženy na nízkou úroveň výstupem s otevřeným kolektorem nebo drainem.

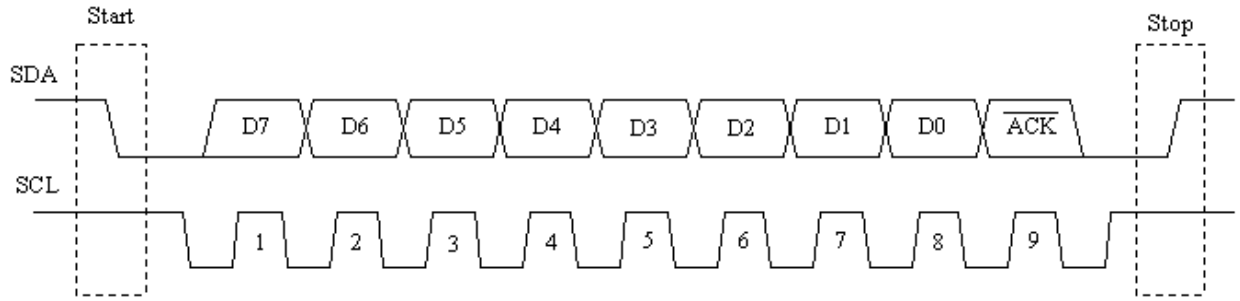



Obr. 7.7 – Propojení sběrnice $I^2C$ [10]

Obrázek ukazuje princip propojení sběrnice. Neaktivní účastníci sběrnice mají vysokou impedanci, neustále však vyhodnocují signály na sběrnici. Je-li použit jen jeden master, vydává hodinový signál jen on. Data však může vysílat jak master, tak slave.

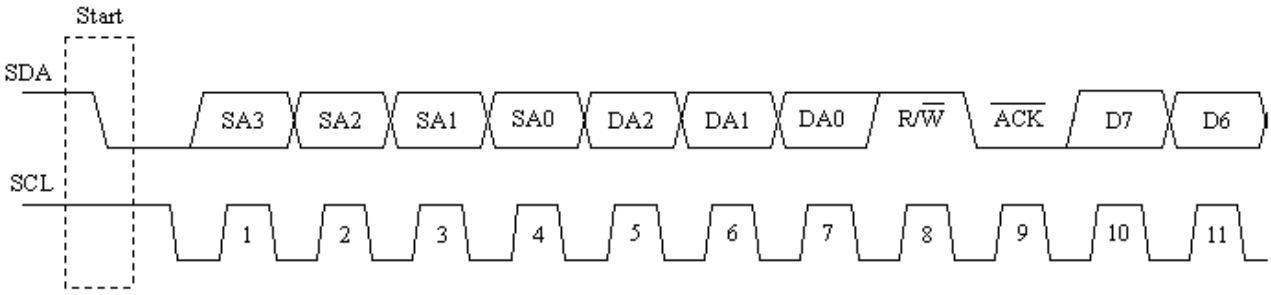
Protokol $I^2C$ rozeznává řadu přesně definovaných situací, které každému účastníkovi umožňují rozeznat začátek a konec přenosu a také své možné adresování:

- Klidový stav: SDA i SCL jsou na vysoké úrovni (HIGH) a tím neaktivní.
- Podmínka startu: SDA je masterem stažena na nízkou úroveň, zatímco SCL zůstává na úrovni HIGH.
- Podmínka stopu: SDA přejde z LOW na HIGH, SCL zůstává na úrovni HIGH. Přenos dat: Příslušný vysílač přivede na datovou linku SDA osm datových bitů, které jsou hodinovými impulsy na lince SCL vysílanými masterem posouvány dále. Přenos začíná bitem s nejvyšší váhou.
- Potvrzení (acknowledge): Příslušný přijímač potvrzuje příjem bytu nízkou úrovní na SDA, dokud master nevyšle devátý hodinový impuls na SCL. Potvrzení současně znamená, že se má přijímat další byte. Požadované ukončení přenosu se musí ohlásit neexistencí potvrzení. Vlastního ukončení přenosu se dosahuje podmínkou stopu.




Obr. 7.8 – Průběh signálů na sběrnici při přenosu adresy [10]

Přenos a potvrzování adres se provádí přesně stejně jako přenos dat. V nejjednodušším případě přenosu dat od mastera k podřízenému zařízení (slave), např. výstupnímu portu, probíhají následující děje: master vyrobí podmínku startu a pak v bitech 7 až 1 přenese adresu portu (součástky) a v bitu 0 požadovaný směr přenosu dat, totiž 0 pro "zápis". Podřízené zařízení (slave) adresu potvrdí. Pak master vyšle datový byte, který rovněž bude potvrzen. Master nyní může spojení přerušit zasláním podmínky stopu nebo může témuž zařízení slave posílat další byty.




Obr. 7.9 – Průběh signálů na sběrnici při příjmu dat [10]

Mají-li se číst data od zařízení slave, musí se adresa přenést s nahozeným bitem přenosu R/W. Master vždy vydá osm hodinových impulsů a dostane osm datových bitů . Potvrdí-li příjem vysláním devátého hodinového impulsu, může přijímat další byty. Přenos je nakonec masterem ukončen vynecháním potvrzení a podmínkou stopu. Každá součástka $I^2C$ má stanovenou svoji adresu, která je zčásti pro daný typ specificky stanovená (SA0...SA3), zčásti proměnná (DA0...DA2). Při třech vyvedených adresních linkách může být na jedné sběrnici $I^2C$ až osm součástek téhož typu. Maximální hodinový kmitočet pro sběrnici $I^2C$ je pro většinu integrovaných obvodů 100 kHz.

Pro typický příklad použití sběrnice $I^2C$ si vezměme paměť EEPROM typu 24cxx. Pro komunikaci postačí pouze dva vodiče (SCL a SDA), na jednu komunikační linku je možno umístit až 8 pouzder (podle kapacity a výrobce!). Používají zápisovou stránku až 64B, což umožňuje rychlé uložení velkého množství dat. Jsou dostupné od mnoha výrobců (asi nejvíce ATMEL a MICROCHIP), pro dosavadní typy 24C postačí pouze dva typy komunikačních protokolů, je velmi vhodný poměr cena/kapacita.

**Funkce pro komunikaci s RTC**

Knihovna pro komunikaci s RTC DS1307 využívá funkcí pro obsluhu $I^2C$ sběrnice a také funkcí pro převod BCD kódu.

Základními funkcemi jsou unsigned `char rtc_read(unsigned char address)` a `void rtc_write(unsigned char address,unsigned char data)`, používané hlavně dalšími funkcemi pro práci s RTC. Provádí zápis nebo čtení dat z/do RTC. Jejími parametry jsou adresy paměti RTC, u funkce pro zápis ještě hodnota dat k zápisu. Funkce pro čtení pak tuto hodnotu vrací. Adresa RTC na $I^2C$ sběrnici je 0xd0 pro zápis, 0xd1 pro čtení (poslední bit adresního slova RTC je read/write bit). Např. pro čtení dat je tedy potřeba, po inicializaci $I^2C$ sběrnice, adresovat RTC v režimu zápisu, odeslat adresu dat v jeho paměti, potom jej přepnout do režimu čtení a přečíst z $I^2C$ sběrnice požadovaná data.

Funkce `void rtc_init(unsigned char rs,unsigned char sqwe,unsigned char out)` provede prvotní inicializaci obvodu a nastaví jeho volitelné parametry (výstup square-wave, jeho prvotní hodnotu...).

Funkce `void rtc_set_time(unsigned char hour,unsigned char min,unsigned char sec)` a `void rtc_get_time(unsigned char *hour,unsigned char *min,unsigned char *sec)` slouží k nastavení nebo zjištění aktuálního času. Jejími parametry nebo návratovými hodnotami jsou hodiny, minuty a sekundy ve dvoumístném decimálním formátu. Pro údaj hodin je použit 24 hodinový formát času.

Funkce void `rtc_set_date(unsigned char date,unsigned char month,unsigned char year)` a `void rtc_get_date(unsigned char *date,unsigned char *month,unsigned char *year)` potom nastaví nebo přečtou aktuální datu. Parametry anebo návratovými hodnotami jsou obdobně jako v předchozím případě hodnoty roku, měsíce a dne v měsíci, vše ve dvoumístném decimálním formátu.

**Obsluha $I^2C$ eeprom**

Použitá EEPROM sídlí na $I^2C$ sběrnici na adrese 0xA0, přičemž poslední bit, stejně jako v případě RTC, určuje zdali je do EEPROM zapisováno, nebo je z ní čteno.

`unsigned char eeprom_read(unsigned char address, unsigned char address2)` je funkcí pro čtení dat z paměti. Jako parametr se uvede adresa části paměti, ze které chceme přečíst data, u větších EEPROM je nejvyšší adresa větší, než jeden byte, z toho důvodu je předávána ve dvou proměnných. Návratová hodnota jsou pak přečtená data. Vlastní komunikace probíhá obdobně jako v případě RTC, tedy po inicializaci sběrnice se odešle adresa EEPROM v režimu zápisu a odešle se adresa paměti. Poté se přepne do režimu čtení a ze sběrnice se přečtou požadovaná data.

Funkce pro zápis dat `void eeprom_write(unsigned char address, unsigned char address2, unsigned char data)` funguje obdobným způsobem

**Organizace paměti**

Pro účely kontinuálního záznamu teplot je žádoucí, aby bylo do paměti zaznamenáno co nejvíce hodnot, tedy co nejdelší časový interval měření. I v případě použití největší dostupné EEPROM paměti je k dispozici relativně omezený prostor, proto je potřeba rozvrhnout optimální rozložení dat v paměti, aby byla co nejlépe využita.

| | Bity | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Byte | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | ROK | | | | MĚSÍC | | | |
| 1 | MINUTY (Hi) | | | DATUM | | | | |
| 2 | MINUTY (Lo) | | | HODINY | | | | |
| 3 | TEPLOTA (1) (celé č.) | | | | | | 0,5 (1) | +/- (1) |
| 4 | TEPLOTA (2) (celé č.) | | | | | | 0,5 (2) | +/- (2) |

Obr. 7.10 – Rozvržení dat v paměti

Jako první je vždy uložená časová značka. Pro hodnoty měsíce a roku postačí 4 bitové hodnoty, v případě roku to tedy bude měření na max. 15 let dopředu, což pro danou aplikaci postačuje. Den v měsíci a hodiny potřebují pro svůj rozsah pětibitové číslo, u každé hodnoty tak v datovém prostoru zůstanou 3 bity volné. Do tohoto volného prostoru byla rozdělena hodnota minut. Poté již následují jednotlivé údaje o naměřených teplotách (na obrázku pro dvě čidla, pro další hodnoty by jejich záznamy pokračovaly dále). Ty jsou ukládány v rozsahu ± 63°C s přesností na 0,5°C, tedy opět s přesností dostačující pro danou aplikaci. Celé číslo teploty se ukládá v šestibitovém rozsahu, poté následuje bit rozlišující půl anebo celý stupeň. Poslední bit je příznakem znaménka, tedy jestli uložená teplotní hodnota je kladná nebo záporná. Takto se naměřené hodnoty ukládají postupně za sebou, po kompletním zaplnění paměti se začne zapisovat znovu od začátku, tedy od první adresy, čímž dojde k přepisování nejstarších hodnot.

Do posledních dvou byte EEPROM je průběžně zapisována aktuální adresa paměti, kam se zapisují data, čímž se zajistí že software i po restartu (např. selhání napájení) bude pokračovat v zápisu hodnot od poslední použité pozice, nedojde tedy ze ztrátě dat. Adresa se nezapisuje vždy, ale pouze při každém čtvrtém zápisu. Toto řešení bylo použito kvůli prodloužení teoretické životnosti EEPROM. Ta má udávaných 100 000 zápisů. Při uvažované průměrné periodě zápisu teplot po půl hodině, dojde za den ke 48 zápisům, tedy při 100 000 cyklů je to kolem 5,5 roku teoretické životnosti při neustálém zapisování adresy. V případě zápisu každé čtvrté hodnoty adresy je tato hodnota 4x vyšší, tedy kolem 22 let, což je více než dostačující. V tomto případě se při resetu ztratí v nejhorším případě poslední 4 měřené hodnoty.

**Funkce pro zápis dat (logování) do paměti**

Jedná se o základní funkci celého zařízení, která provádí zápis naměřených hodnot ve správném formátu do externí EEPROM paměti.

```
void logdata(void) {
bit j;
int i,k,temp,tmp;
unsigned char h,m,s;
unsigned char d,n,y;
unsigned char yearmonth,mindate,minhour;

rtc_get_time(&h,&m,&s);
k=m%period;
if(k==0 && logged==0) {
        rtc_get_date(&d,&n,&y);
        yearmonth=((y & 15) << 4) | (n & 15);
        mindate=((m & 56) << 2) | (d & 31);
        minhour=((m & 7) << 5) | (h & 31);
        eeprom_write(eepaddrh, eepaddr, yearmonth);
        eeprom_write(eepaddrh, eepaddr+1, mindate);
        eeprom_write(eepaddrh, eepaddr+2, minhour);
        for (i=0;i<devices;++i) {
          temp=ds1820_temperature_10(&rom_code[i][0]);
          j=0;
          if (temp<0)
             {
             j=1;
             temp=-temp;
             };
        tmp=(((temp/10) & 63) << 2) | (((temp%10) & 1) << 1) | j;
        eeprom_write(eepaddrh,eepaddr+i+3,tmp);
        };
        eepaddr=eepaddr + (3 + devices);
        if(eepaddr > 255) {
                eepaddrh=eepaddrh+1;
                eepaddr=0;
                }
        logged=1;
```

```
            eeprom_write(EEPROM_LAST_ADDRESS,255,eepaddr);
            eeprom_write(EEPROM_LAST_ADDRESS,254,eepaddrh);

    }
    else if(k!=0) logged=0;
    }
```

V první fázi dojde k přečtení aktuálního času a v případě, že je hodnota minut násobkem dříve definované hodnoty (tj. měřící perioda, nastavitelná v konfiguraci zařízení), pokračuje se dále. Dojde k zjištění aktuálního data a k rozdělení, posunutí a sloučení hodnot do jednotlivých byte připravených k zápisu a k jejich zapsání do EEPROM. Dále následuje cyklus, v němž se pro všechna aktivní čidla zjistí jejich teplota, oddělí se znaménko teploty a desetinné číslo, opět se vše poskládá do jednoho byte a uloží na aktuální adresu EEPROM. Poté dojde k posunu ukazatele adresy a nastavení příznaku, že log pro tuto periodu již proběhl a zapíše se aktuální adresa na poslední datové místo EEPROM..

**Funkce pro alarm**

```
void checkalarm(void)
```

Tato funkce kontroluje teplotu, zdali je v předem nastavených mezích (údaje pro každé čidlo jsou uložené v interní EEPROM paměti použitého CPU). Pokud teplota není v definovaných mezích, dojde k aktivaci jednoho ze dvou alarmových výstupů, tedy k sepnutí nebo rozepnutí (dle zapojení) kontaktů relé. Dále může být odeslán informační e-mail o aktivaci alarmu. Alarm je možno zrušit stiskem potvrzovacího tlačítka.

**Funkce odesílání dat**

Další používanou funkcí je tato funkce, jenž odešle všechna data nacházející se v datové EEPROM v určitém formátu na vzdálené PC přes ethernetový port.

```
void senddata(void) {
unsigned char txbuff[255], tempbuff[8];
int tadr=0,tadrh=0;
bit mark,half;
unsigned char d,n,y;
unsigned char h,m,m1;
unsigned char inc,i,tmem,temp,mrk,hlf;
inc=3+devices;
while(tadrh<(eepaddrh+1)) {
        while(tadr<255) {
                tmem=eeprom_read(tadrh,tadr);
                n=tmem & 15;
```

```
                    y=tmem >> 4;
                    tmem=eeprom_read(tadrh,tadr+1);
                    d=tmem & 31;
                    m1=(tmem & 224) >> 2;
                    tmem=eeprom_read(tadrh,tadr+2);
                    h=tmem & 31;
                    m=(tmem >> 5) | m1;
                    sprintf(txbuff,"%02d.%02d.%02d-%02d:%02d",d,n,y,h,m);
                    for(i=0;i<devices;++i) {
                            tmem=eeprom_read(tadrh,tadr+3);
                            mark=tmem & 1;
                            half=(tmem & 2) >> 1;
                            temp=tmem >> 2;
                            if(mark==0) mrk='+'; else mrk='-';
                            if(half==0) hlf=0; else hlf=5;
                            sprintf(tempbuff,"#%c%02d.%u",mrk,temp,hlf);
                            sprintf(txbuff,"%s%s",txbuff,tempbuff);
                            }
                    puts(txbuff);
                    tadr=tadr+inc;
                    if (tadrh==eepaddrh) {
                            if (tadr==eepaddr) tadr=255;
                            }
                    }
            tadrh++;
            tadr=0;
            }
    printf("\n");

    }
```

Aby se zajistilo odeslání všech dat v paměti, je potřeba v cyklu postupně vyčíst všechny měřené hodnoty, tedy až do aktuální nejvyšší adresy zápisu EEPROM. Začíná se od nejnižší adresy, prvně dojde k přečtení prvních tří byte, kde je uložena časová značka, a k poskládání dat do původní podoby, tj. rok, měsíc, den, hodiny a minuty. Následuje cyklus, v němž jsou čteny následující byte obsahující naměřené teploty z jednotlivých čidel. Toto vše je poskládáno do řetězce ve tvaru RR.MM.DD;HH:MM#T1#T2#T3...., který je takto pomocí standardních funkcí odesílán na sériové rozhraní procesoru a přes modul X-Port ethernetovým rozhraním do klientského PC.

**Komunikační funkce**

Funkce komunikace s PC jsou vyvolávány přerušením z USART rozhraní procesoru, tedy skrze X-port přes ethernetové rozhraní.

```
interrupt [USART0_RXC] void usart0_rx_isr(void)
{
char status,data;
status=UCSR0A;
data=UDR0;
if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
   {
   rx_buffer0[rx_wr_index0]=data;
   if (++rx_wr_index0 == RX_BUFFER_SIZE0) rx_wr_index0=0;
   if (++rx_counter0 == RX_BUFFER_SIZE0)
      {
      rx_counter0=0;
      rx_buffer_overflow0=1;
      };
   if (rx_buffer0==0xE0) senddata();
   if (rx_buffer0==0xE5) readcfg();
   if (data==0xE6) {
        rx_wr_index0=0;
        rx_buffer0[0]=0;
        }
   if (data==0xE7) {
        updatecfg(rx_buffer0);
        rx_wr_index0=0;
        }
   };
}
```

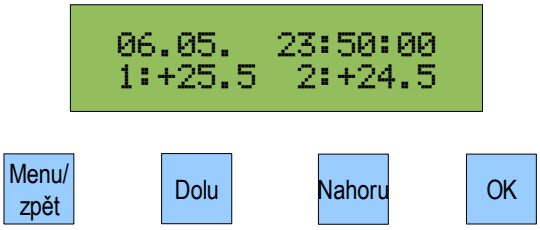
Pokud je přijatý datový rámec v pořádku, rozhoduje se dle přijatého datového slova o další akci, tedy o odeslání všech dat v případě přijetí 0xE0h, přečtení aktuálního nastavení modulu při 0xE5h a nebo o aktualizaci nastavení z klientského PC při 0xE6h.

**Funkce hlavního menu**

Slouží pro základní ovládání modulu. Umožňuje ověření aktuálního stavu naplnění paměti, nastavení data a času, nastavení používaných měřících portů, nastavení alarmů, prohlížení naměřených hodnot a reset paměti.

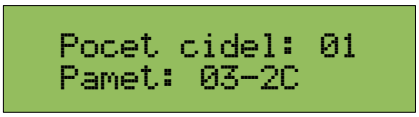
### 7.3.3 Popis funkce hlavního programu

Při startu zařízení dojde k prohledání 1-wire sběrnice a k detekci všech připojených čidel. Tím dojde, v případě změny od posledního stavu, k úpravě konfigurace a rozvržení dat v záznamové EEPROM. Poté dojde k zobrazení hlavní obrazovky s aktuálním datem a časem a se zvolenými dvěma měřenými teplotami.



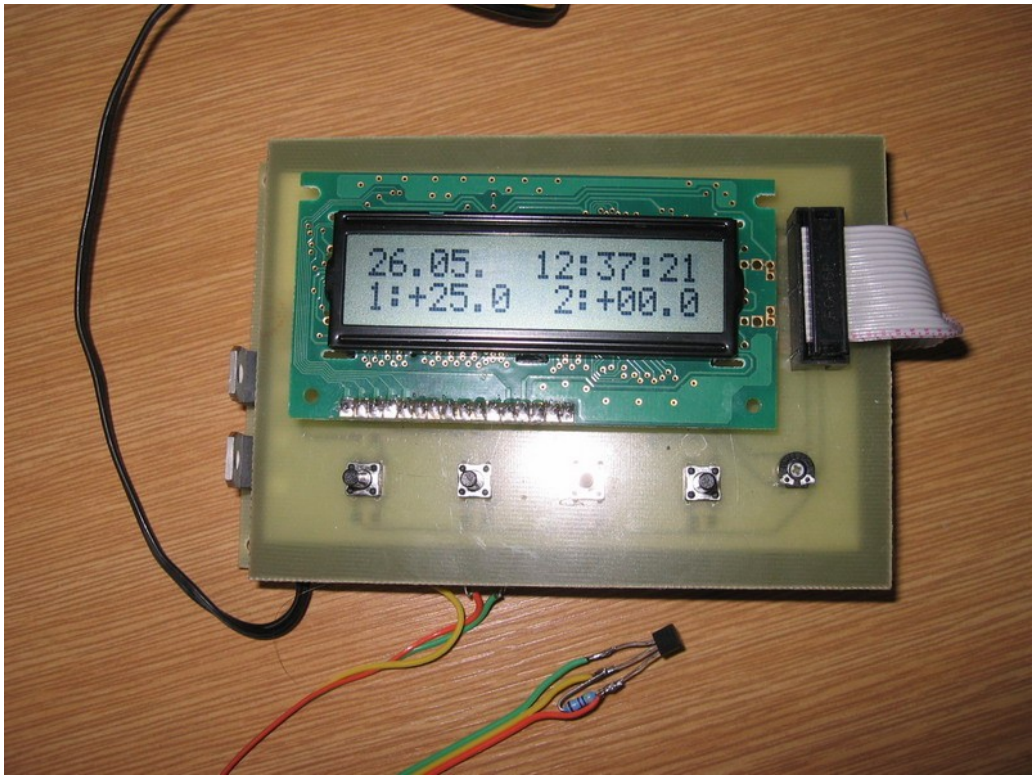

Obr. 7.11 – Ukázka hlavní obrazovky a rozložení kláves

Zde je možno klávesami Nahoru a Dolu změnit čísla čidel z nichž jsou zobrazovány teploty, pomocí tlačítka OK zrušit případný alarm a klávesou Menu vstoupit do hlavního menu programu. Zde je možno ověřit aktuálního stavu zařízení (počet aktivních portů, zaplnění paměti), nastavení data a času, nastavení používaných měřících portů, nastavení alarmů, prohlížení naměřených hodnot a reset paměti. Klávesami Nahoru a Dolu se pohybuje mezi jednotlivými položkami menu, klávesa OK vstoupí do dalšího podmenu nebo potvrdí hodnotu. Klávesou menu se pohybujeme o krok zpět, až do „vyskočení“ z menu na hlavní obrazovku. Menu je cyklické, tedy při pohybu kurzorovými klávesami se při dosažení poslední položky dostaneme na položku první. Obdobná funkce je zajištěna i při prohlížení hodnot, lze tedy jednoduše začít prohlížet od hodnot nejstarších i nejnovějších.




Obr. 7.12 – Ukázka zobrazení stavu modulu v menu


Obr. 7.13 – Ukázka čtení naměřených dat z menu




Obr. 7.14 – Funkční prototyp modulu

## 7.4 Klientský software

Klientský software pro MS Windows umožňuje načíst data z modulu, vykreslit je do grafu, uložit do souboru, dále též zjištění a změnu konfigurace modulu. Byl vytvořen pod vývojovým prostředím Borland C++ Builder verze 6.

### 7.4.1 Řešení síťové komunikace

V systémech Windows se síťová komunikace protokoly TPC/IP realizuje pomocí socketů (knihovna winsock.dll). Socket interface nebo zkráceně jen sockets je rozhraní, které zpřístupňuje služby protokolů TCP/IP jednotlivým aplikacím. Jinak řečeno - pokud chceme ve své aplikaci použít protokolu IP, resp. protokolu TCP, nemusíme znát žádné podrobnosti o těchto protokolech, nemusíme vědět, jak se navazuje TCP spojení, jak probíhá potvrzování přijatých paketů, nemusíme znát princip posuvného okénka (sliding window): stačí použít rozhraní sockets a všechny důležité služby protokolů TCP/IP máme rovnou k dispozici.

Samotná síťová komunikace je řešena pomocí komponenty ClientSocket. Této komponentě je potřeba nejprve specifikovat požadovaný server, ke kterému se má připojit. To je možné buďto jeho IP adresou (vlastnost `ClientSocket -> Address`) nebo doménovým jménem (vlastnost `ClientSocket -> HostName`). Kromě serveru je nutné zvolit i port, k němuž se připojujeme – známe-li jeho číslo (číslo služby), použijeme vlastnost `ClientSocket -> Port`, známe-li jeho název (název služby), bude vhodnější vlastnost `ClientSocket -> Service`.

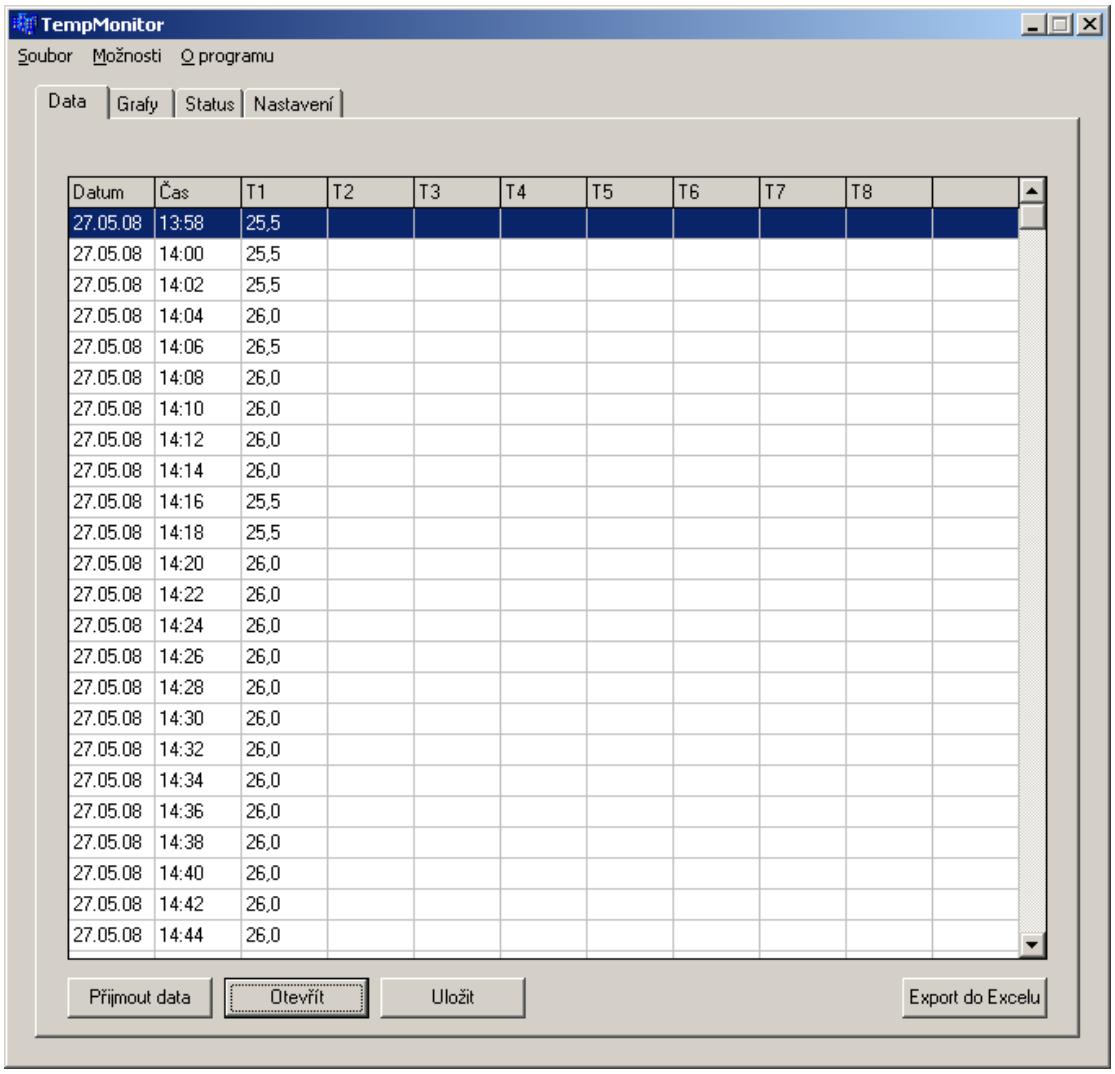
Pokud máme specifikovaný název (adresu) serveru a název (číslo) služby – portu, můžeme navázat se vzdáleným serverem síťové spojení. K tomuto účelu slouží metoda `ClientSocket -> Open`. Stejný význam má nastavení vlastnosti `ClientSocket -> Active` na hodnotu True. Po navázání spojení je již možno se serverem komunikovat následujícím způsobem:

Pro odesílání dat poslouží funkce `ClientSocket -> Socket -> SendBuf(void *Buf, int BufSize, int Flags = 0);`, tato funkce odešle jakoukoliv posloupnost bytů z definované proměnné.

Pro příjem dat slouží funkce `ClientSocket -> Socket -> ReceiveBuf(void *Buf, int BufSize, int Flags = 0);`, která příjme data odesílané serverem..

Přijímání dat z měřícího modulu tedy potom probíhá tak, že se po vytvoření spojení s modulem odešle z klientského SW klíčové slovo `0xE0h`. Na to modul zareaguje tím, že začne ihned odesílat data ze své paměti, toto odesílání uzavře také klíčovým slovem `0xE0h`. Klientský softvare v reakci na událost `ClientSocketRead` začne přijímat data z modulu a plnit jimi datovou mřížku. Obdobně probíhá komunikace i při čtení/aktualizaci nastavení modulu.

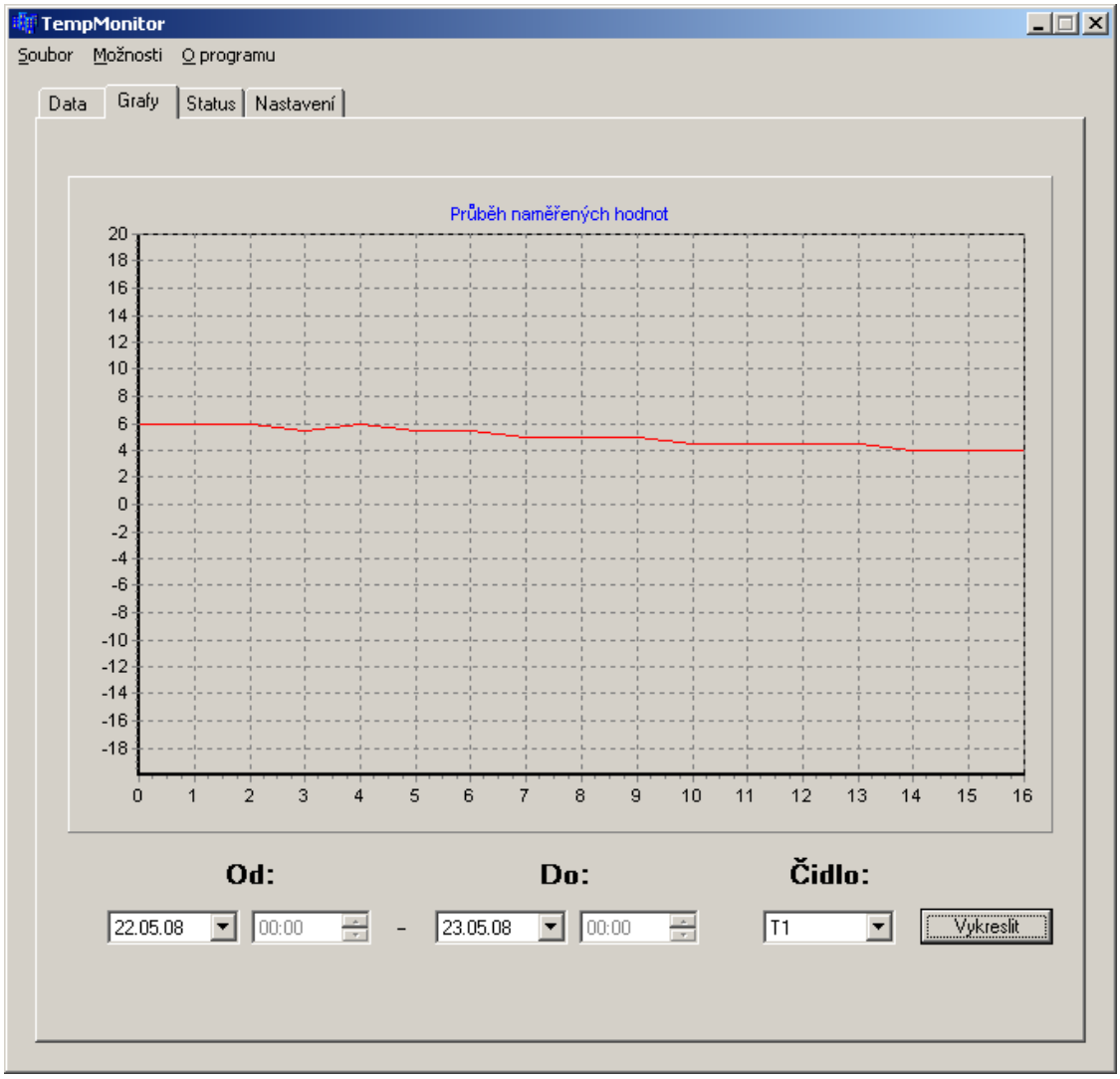
### 7.4.2 Uživatelské rozhraní




Obr. 7.15 – Zobrazení přijatých dat v tabulce

Záložka „Data“ obsahuje tabulku naměřených hodnot. Data je možno získat přes síť přímo z modulu tlačítkem „Přijmout data“. Je možno ji uložit anebo načíst ze souboru (CSV formát oddělený středníkem), také exportovat do tabulkového procesoru (MS Excel) pro podrobnější zpracování.

Grafické zobrazení ve formě grafů je realizováno pomocí komponenty TeeChart, která umožňuje pohodlné vytváření různých druhů grafů (sloupcový, 3D, spojnicový atd). Její základní verze je přímo součásti Borland C++ Builderu.

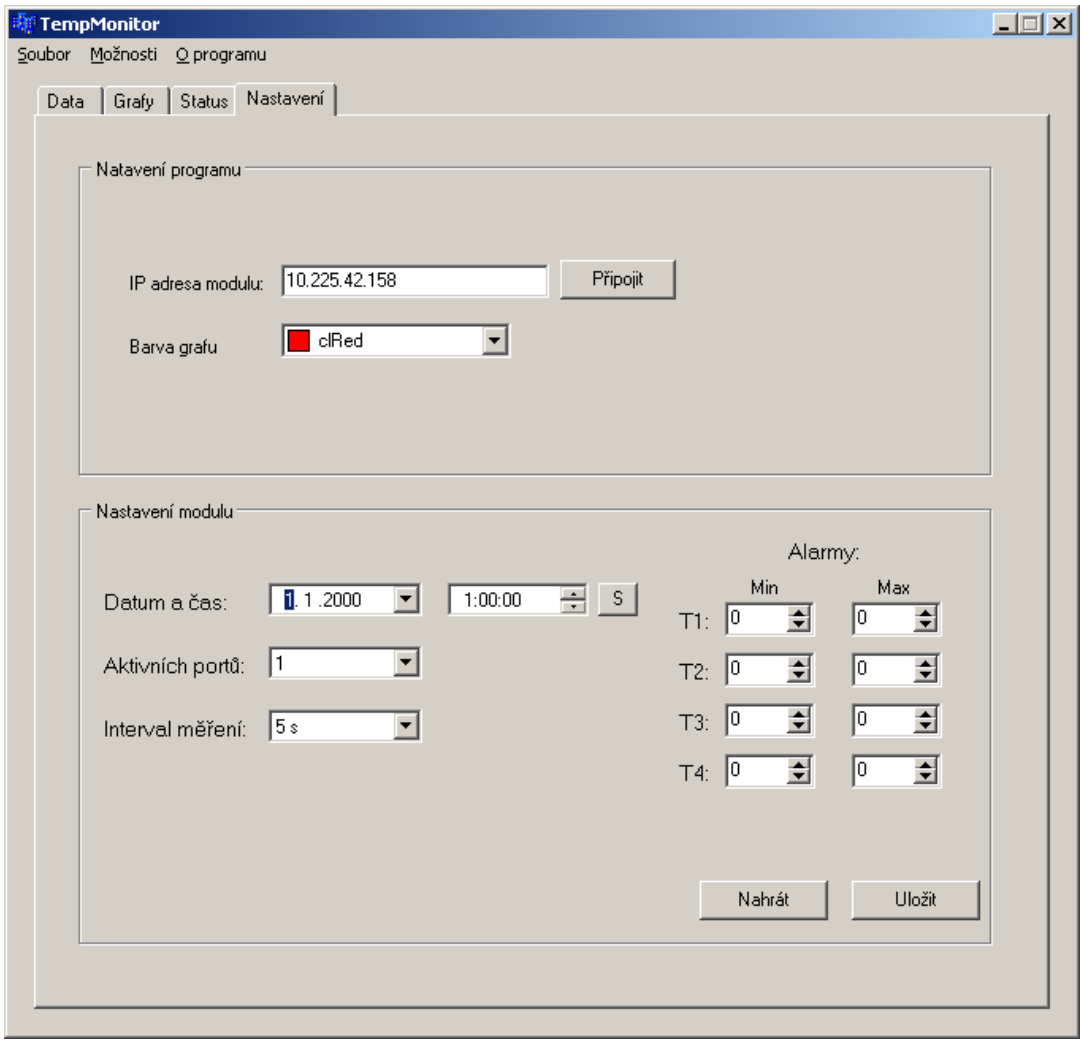



Obr. 7.16 – Grafické zobrazení dat

V záložce „Grafy“ je tedy možno vykreslovat grafy pro různá čidla z přijatých naměřených hodnot v zadaném časovém intervalu.

Záložka „Status“ slouží ke zobrazování stavových informací, jako např. stav spojení s modulem, aktuální činnost atd.

V poslední záložce „Nastavení“ je možno provádět dálkovou konfiguraci některých parametrů modulu a také měnit nastavení vlastního programu.




Obr. 7.17 – Záložka Nastavení

V horní části okna je možno měnit IP adresu, na které má být aktivní měřící modu a dále barvu průběhu v grafu naměřených hodnot.

Ve spodní části je potom možno upravovat nastavení modulu. Kliknutím na tlačítko „Nahrát“ se do okna načtou aktuální parametry z modulu, kliknutím na „Uložit“ dojde k aktualizaci námi upravených parametrů v modulu. Tlačítko „S“ slouží k převzetí systémového času PC do nastavení modulu.

# 8. Závěr

Cílem diplomové práce bylo navrhnout a zrealizovat autonomní modul pro monitorování teplot v mrazících a chladících zařízeních ve skladech potravin tak, aby byl modul schopen pracovat samostatně a data mohla být přenášena do klientského PC k dalšímu zpracování.

Jako měřiče teplot (čidla) byly zvoleny digitální teploměry DS18S20 od firmy Dallas Semiconductors, připojitelné přes 1-wire sběrnici. Důvodem byla jednoduchost jejich připojení (pomocí dvou vodičů), jednoduchost obsluhy (čidla vrací přímo hodnotu měřené teploty) a jejich dostačující přesnost bez nutnosti kalibrace. V souvislosti s připojením těchto čidel byly v diplomové práci prostudovány principy 1-wire sběrnice a možnosti její softwarové obsluhy.

Jádrem měřícího modulu je jednočipový mikrokontrolér ATMega128 od firmy Atmel, patřící do rodiny mikrokontrolérů AVR. Jeho vlastnosti (rychlost, velikost paměti, počet datových portů, sériová rozhraní) jsou více než postačující pro danou aplikaci. Modul tak obsahuje celkem 4 datové vstupy pro připojení čidel (při maximálním počtu 8 čidel na jedné sběrnici to dává ve výsledku celkem 32 připojitelných čidel) a 2 alarmové spínané výstupy. Jako zdroj přesného času byl zvolen integrovaný obvod hodin přesného času DS1307, z důvodu jeho přesnosti a jednoduchosti obsluhy. Oproti softwarovým hodinám řešeným přímo v procesoru je tento způsob daleko přesnější a také lépe uzpůsoben pro bateriové zálohování. Vnitřní paměť modulu je běžná sériová EEPROM, kde se v závislosti na nastaveném intervalu měření, počtu monitorovaných míst a v neposlední řadě také na velikosti samotné EEPROM může vejít kontinuální záznam teplot v časovém intervalu v řádu měsíců až roku, což je pro danou aplikaci postačující. Součástí modulu je také deska s alfanumerickým displejem a tlačítky pro jednoduchou uživatelskou obsluhu.

Propojení modulu s klientským PC je realizováno pomocí modulu XPort firmy Lantronix. Jedná se v podstatě o převodník RS232 na Ethernet, jeho obsluha je tedy velice jednoduchá. Umožňuje jednoduchou konfiguraci pomocí WWW rozhraní anebo přiloženého firemního softwaru. Je též využito jeho možnosti odesílání e-mailových zpráv v případě alarmových stavů. Na modulu je dále vyvedeno druhé sériové rozhraní pro případné připojení dalších periferních zařízení, např. GSM telefonu pro odesílání informačních SMS zpráv.

Návrh schémat a desek plošných spojů probíhal v programu Eagle Layout Editor firmy CadSoft, při návrhu byl modul pro jednoduchost rozdělen na dvě desky – základní měřící desku a desku s displejem a tlačítky.

Programování modulu probíhalo v jazyce C++ za pomoci překladače CodeVision AVR. Obslužný software provádí detekci připojených čidel, periodická měření, spolupráci s RTC, ukládání naměřených dat do vnitřní paměti a kontrolu a vyhodnocování alarmových stavu. Také obsluhuje komunikaci s klientským PC a obsahuje jednoduché uživatelské menu, umožňující uživateli základní nastavení modulu, zjištění jeho stavu a prohlížení naměřených hodnot.

Klientský software, vytvořený v prostředí Borland C++ Builder umožňuje příjem naměřených dat z modulu, jejich zobrazení v tabulce a ve formě grafu, jejich ukládání do souboru a také může sloužit pro konfiguraci některých parametrů měřícího modulu.

# Seznam použité literatury


[1] MAXIM-DALLAS Semiconductor, 1-Wire Communication Throug Software, 2000

[2] MAXIM-DALLAS Semiconductor, Public Domain 1-Wire Net Functions, 2000

[3] Wikipedia [online]: 1-Wire, 2005, [cit. 2007]. Dostupné na WWW <http://en.wikipedia.org/wiki/1-Wire>

[4] Malý, M.: Algoritmus procházení sběrnice 1-wire, HW server [online], 2005, [cit. 2007]. Dostupné na WWW <http://www.hw.cz/Teorie-a-praxe/Dokumentace/ART1249-Algoritmus-prochazeni-sbernice-1-Wire.html>

[5] Láník, V.: MicroLan - A jde to i s jedním vodičem!, HW server [online], 2005, [cit. 2007]. Dostupné na WWW <http://www.hw.cz/Rozhrani/ART1240-MicroLan---A-jde-to-i-s-jednim-vodicem.html>

[6] MAXIM-DALLAS Semiconductor, DS1307 Datasheet, 2006

[7] Lantronix, Inc, XPort Product Brief, 2008

[8] Atmel Corporation, ATMega128 datasheet, 2008

[9] KABELOVÁ, A., DOSTÁLEK, L. Velký pruvodce protokoly TCP/IP a systémem DNS. Brno: Computer Press, 2002.

[10] Hankovec, D.: Sběrnice $I^2C$, o co jde a jak pracuje, DH Servis [online], 2005, [cit. 2008]. Dostupné na WWW <http://www.dhservis.cz/iic.htm>


# Seznam použitých zkratek a symbolů

| | |
|---|---|
| LSB | Least Significant Bit |
| MSB | Most Significant Bit |
| CRC | Cyclic Redundancy Check |
| NV RAM | Non Volatile Random Access Memory |
| EEPROM | Electronically Erasable Programmable Read-Only Memory |
| LAN | Local Area Network |
| TCP | Transmission Control Protocol |
| IP | Internet Protocol |
| RISC | Reduced Instruction Set Computer |
| MIPS | Million Instructions Per Second |
| JTAG | Joint Test Action Group |
| RTC | Real Time Clock |
| CPU | Central Processing Unit |

# Seznam příloh

- Schémata zapojení modulu
- Návrh desek plošného spoje
- Seznam součástek
- CD s elektronickou podobou textu a zdrojovými kódy software

# Schéma zapojení základní desky modulu

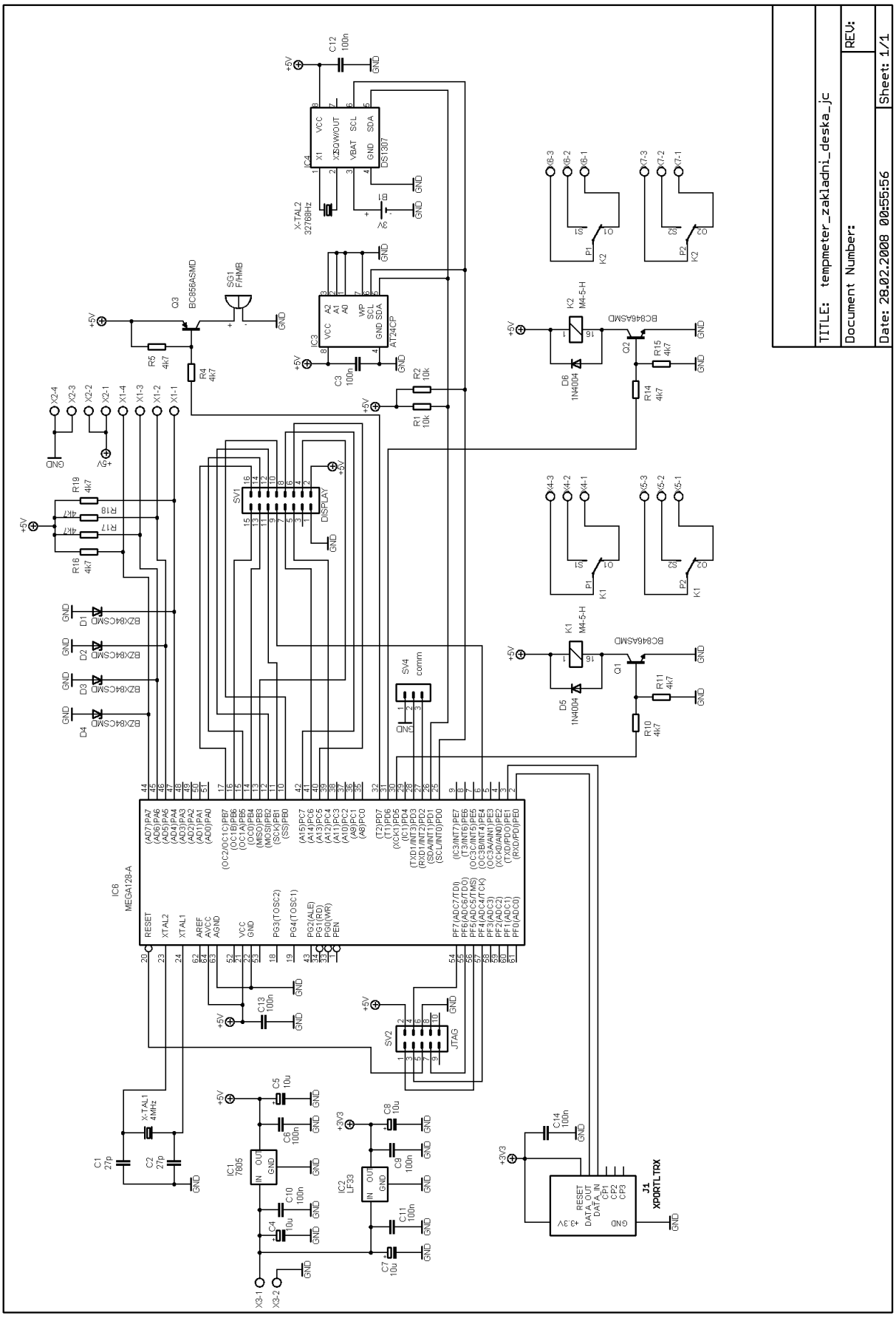
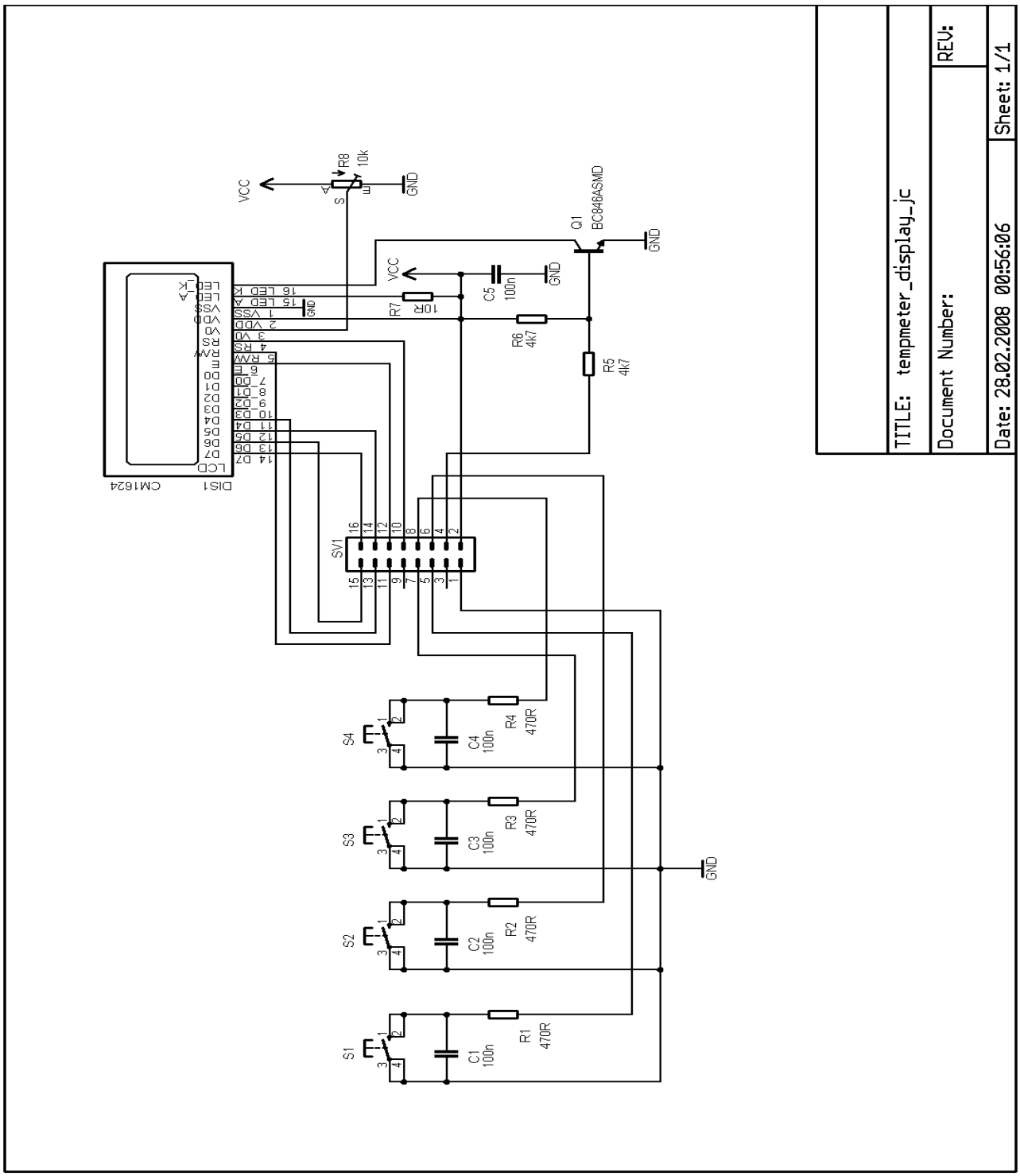
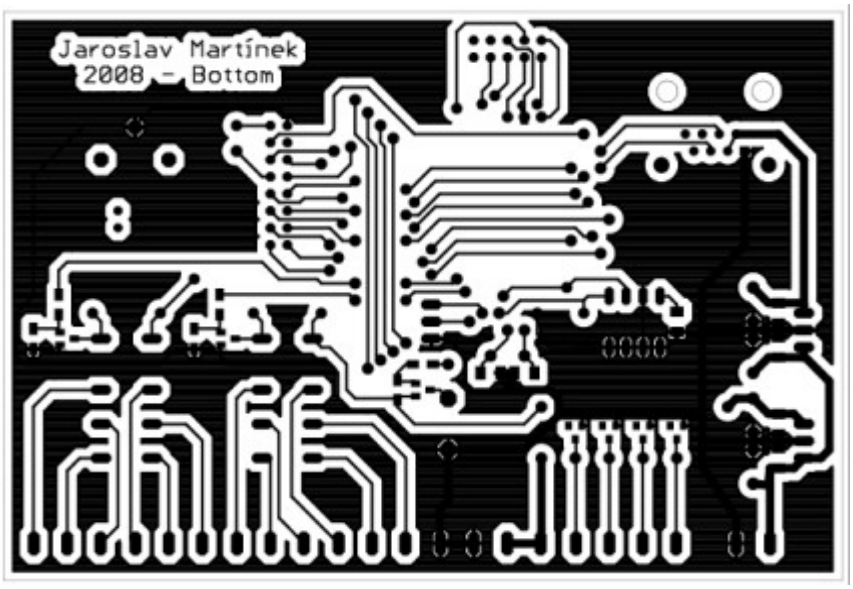


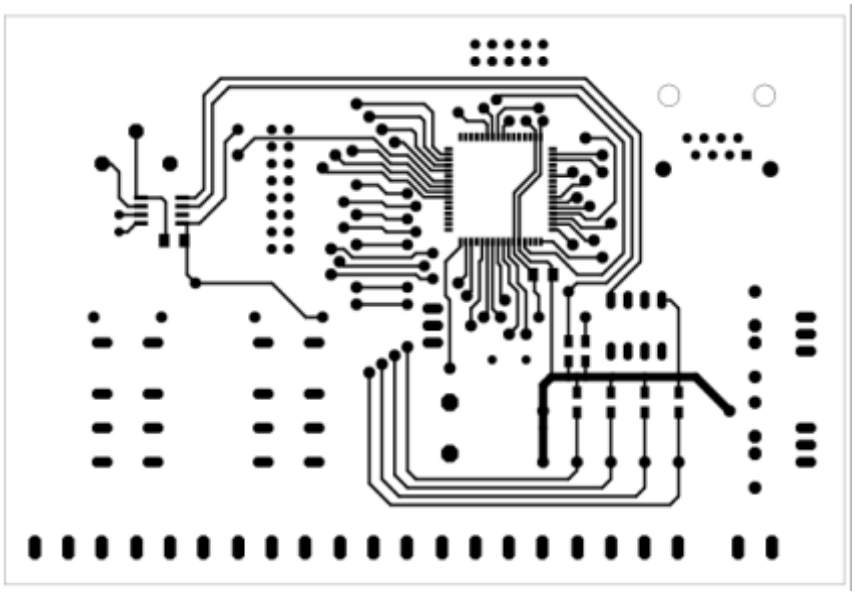
**Schéma zapojení desky s tlačítky a displejem**



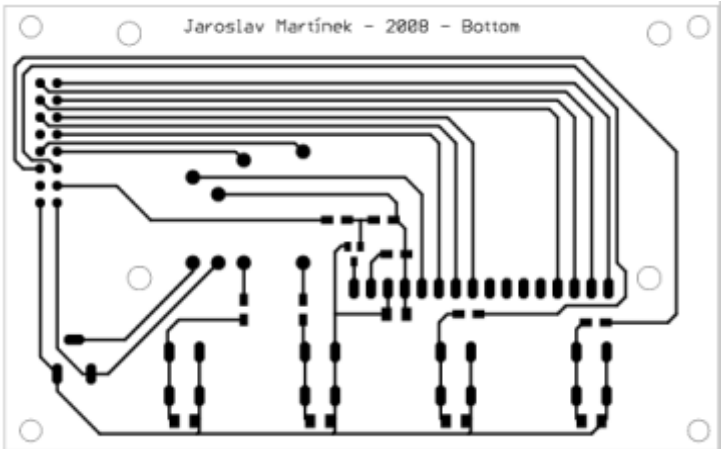
**Návrh desek plošného spoje**




*hlavní deska – spodní strana*



*hlavní deska – horní strana*




*deska displeje a tlačítek*

## Seznam součástek – hlavní deska

| Part | Value | Package | Library | Position (mil) | Orientation |
|---|---|---|---|---|---|
| B1 | 3V | CR2032V | BATTERY | (4150 2550) | R90 |
| C1 | 27p | C1206K | rcl | (1850 1200) | MR0 |
| C2 | 27p | C1206K | rcl | (2050 1200) | MR180 |
| C3 | 100n | C1206K | rcl | (950 1500) | MR270 |
| C4 | 10u | E5-6 | rcl | (500 650) | R90 |
| C5 | 10u | E5-6 | rcl | (500 950) | R270 |
| C6 | 100n | C1206K | rcl | (350 900) | MR270 |
| C7 | 10u | E5-6 | rcl | (500 1300) | R90 |
| C8 | 10u | E5-6 | rcl | (500 1600) | R270 |
| C9 | 100n | C1206K | rcl | (350 1550) | MR270 |
| C10 | 100n | C1206K | rcl | (350 1350) | MR90 |
| C11 | 100n | C1206K | rcl | (350 700) | MR90 |
| C12 | 100n | C1206K | rcl | (3925 2000) | R0 |
| C13 | 100n | C1206K | rcl | (1750 1800) | R0 |
| C14 | 100n | C1206K | rcl | (600 2400) | MR0 |
| D1 | BZX84CSMD | TO236 | diode | (950 850) | MR180 |
| D2 | BZX84CSMD | TO236 | diode | (1150 850) | MR180 |
| D3 | BZX84CSMD | TO236 | diode | (1350 850) | MR180 |
| D4 | BZX84CSMD | TO236 | diode | (1550 850) | MR180 |
| D5 | 1N4004 | DO41-10 | diode | (3250 1550) | R0 |
| D6 | 1N4004 | DO41-10 | diode | (4200 1550) | R0 |
| IC1 | 7805 | 78XXS | v-reg | (100 800) | R90 |
| IC2 | LF33 | 78XXS | v-reg | (100 1450) | R90 |
| IC3 | AT24CP | DIL08 | atmel | (1200 1500) | R0 |
| IC4 | DS1307 | SOIC-8 | dallas-1 | (4000 2150) | R180 |
| IC6 | MEGA128-A | TQFP64 | atmel | (2000 2300) | R0 |
| J1 | XPORTLTRX | XPORT | xport | (550 2500) | R0 |
| K1 | M4-5-H | 351 | relay | (3250 1250) | R270 |
| K2 | M4-5-H | 351 | relay | (4200 1250) | R270 |
| Q1 | BC846ASMD | SOT23 | transistor-npn | (3600 1400) | MR270 |
| Q2 | BC846ASMD | SOT23 | transistor-npn | (4550 1400) | MR270 |
| Q3 | BC856ASMD | SOT23-BEC | transistor-pnp | (2450 1100) | MR270 |
| R1 | 10k | R1206W | rcl | (1500 1350) | R270 |
| R2 | 10k | R1206W | rcl | (1600 1350) | R270 |
| R4 | 4k7 | R1206W | rcl | (2450 1250) | MR0 |
| R5 | 4k7 | R1206W | rcl | (2600 1100) | MR270 |
| R10 | 4k7 | R1206W | rcl | (3650 1600) | MR90 |
| R11 | 4k7 | R1206W | rcl | (3800 1400) | MR270 |
| R14 | 4k7 | R1206W | rcl | (4600 1600) | MR90 |
| R15 | 4k7 | R1206W | rcl | (4750 1400) | MR270 |
| R16 | 4k7 | R1206W | rcl | (1550 1050) | R270 |
| R17 | 4k7 | R1206W | rcl | (1350 1050) | R270 |
| R18 | 4k7 | R1206W | rcl | (1150 1050) | R270 |
| R19 | 4k7 | R1206W | rcl | (950 1050) | R270 |
| SG1 | F/HMB | F/HMB | buzzer | (2300 900) | R90 |
| SV1 | DISPLAY | ML16 | con-harting-ml | (3300 2300) | R270 |
| SV2 | JTAG | ML10 | con-harting-ml | (1950 3100) | R0 |
| SV4 | comm | MA03-1 | con-lstb | (2400 1500) | R90 |
| X-TAL1 | 4MHz | QS | 0JIRKA | (1950 1300) | R0 |
| X-TAL2 | 32768Hz | TC38H | crystal | (4250 2100) | R270 |
| X1 | | W237-4 | con-wago-500 | (1250 250) | R0 |
| X2 | | W237-4 | con-wago-500 | (2050 250) | R0 |
| X3 | | W237-132 | con-wago-508 | (500 225) | R0 |
| X4 | | W237-103 | con-wago-500 | (2750 250) | R0 |
| X5 | | W237-103 | con-wago-500 | (3350 250) | R0 |
| X6 | | W237-103 | con-wago-500 | (3950 250) | R0 |
| X7 | | W237-103 | con-wago-500 | (4550 250) | R0 |

## Seznam součástek – deska displeje a tlačítek

| Part | Value | Device | Package | Library | Sheet |
|---|---|---|---|---|---|
| C1 | 100n | C-EUC1206K | C1206K | rcl | 1 |
| C2 | 100n | C-EUC1206K | C1206K | rcl | 1 |
| C3 | 100n | C-EUC1206K | C1206K | rcl | 1 |
| C4 | 100n | C-EUC1206K | C1206K | rcl | 1 |
| C5 | 100n | C-EUC1206K | C1206K | rcl | 1 |
| DIS1 | CM1624 | CM1624 | CM1624 | display-lcd | 1 |
| Q1 | BC846ASMD | BC846ASMD | SOT23 | transistor-npn | 1 |
| R1 | 470R | R-EU_R1206W | R1206W | rcl | 1 |
| R2 | 470R | R-EU_R1206W | R1206W | rcl | 1 |
| R3 | 470R | R-EU_R1206W | R1206W | rcl | 1 |
| R4 | 470R | R-EU_R1206W | R1206W | rcl | 1 |
| R5 | 4k7 | R-EU_R1206W | R1206W | rcl | 1 |
| R6 | 4k7 | R-EU_R1206W | R1206W | rcl | 1 |
| R7 | 10R | R-EU_R1206W | R1206W | rcl | 1 |
| R8 | 10k | TRIM_EU-CA6V | CA6V | pot | 1 |
| S1 | | 10-XX | B3F-10XX | switch-omron | 1 |
| S2 | | 10-XX | B3F-10XX | switch-omron | 1 |
| S3 | | 10-XX | B3F-10XX | switch-omron | 1 |
| S4 | | 10-XX | B3F-10XX | switch-omron | 1 |
| SV1 | | ML16 | ML16 | con-harting-ml | 1 |