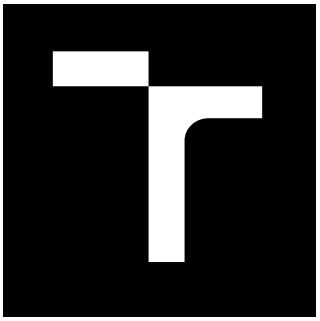


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

DEPARTMENT OF CONTROL AND INSTRUMENTATION

ZAŘÍZENÍ PRO MONITOROVÁNÍ TEPLOTY

TEMPERATURE MONITORING DEVICE

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — **Vladislav Kaderkin**

AUTHOR

VEDOUCÍ PRÁCE — **Ing. Tomáš Macho, Ph.D.**

SUPERVISOR

BRNO 2025

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Automatizační a měřicí technika**

Ústav automatizace a měřicí techniky

Student: Vladislav Kaderkin ***ID:*** 241670

Ročník: 3 ***Akademický rok:*** 2024/25

NÁZEV TÉMATU:

Zařízení pro monitorování teploty

POKYNY PRO VYPRACOVÁNÍ:

1. Seznamte se s problematikou monitorování teploty v serverovnách a s vývojovou deskou TWR-S08 firmy NXP Semiconductors.
2. Navrhněte koncepci zřízení, které bude zaznamenávat datum a čas poklesu teploty pod zadanou mez a překročení teploty nad zadanou mez a tyto datumy a časy ukládat do nevolatilní paměti. Zařízení bude na displeji zobrazovat aktuální teplotu. Uvažujte použití vývojové desky TWR-S08.
3. Řešte napájení zařízení včetně využití baterie pro zálohování.
4. Vyberte vhodný kalendářový obvod a připojte jej k vývojové desce.
5. Vytvořte softwarové vybavení pro mikrokontrolér a odlaďte jej.
6. Řešte vyčítání záznamů o překročení teplotních mezí do počítače PC.
7. Zařízení otestujte a vyhodnoťte dosažené výsledky.

DOPORUČENÁ LITERATURA:

Temperature Sensor for the HCS08 Microcontroller Family. AN3031, Rev. 1, 04/2010.

Termín zadání: 10.2.2025 ***Termín odevzdání:*** 28.5.2025

Vedoucí práce: Ing. Tomáš Macho, Ph.D.

Ing. Miroslav Jirgl, Ph.D.
předseda rady studijního programu

UPOZORNĚNÍ:

Autor bakalářské práce nesmí při vytváření bakalářské práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

ABSTRAKT

Tato bakalářská práce se zabývá problematikou monitorování teploty v serverovnách. Zařízení pro monitorování teploty je založeno na vývojové desce TWR-S08 od společnosti NXP Semiconductors. Práce je rozdělena do následujících částí: teoretická část se stručným úvodem do problematiky, koncepční část popisující princip fungování zařízení a výběr použitých komponentů, realizační část popisující software zařízení a jeho fyzické připojení a kapitola popisující testování zařízení.

KLÍČOVÁ SLOVA

Mikrokontrolér MC9S08LH64, vývojová deska TWR-S08, MCP9808, snímač teploty, DS3231, RTC, kalendářový obvod, FLASH, SCI, I2C, PWM, monitorování teploty, serverovna.

ABSTRACT

This bachelor thesis deals with the problem of temperature monitoring in server rooms. The temperature monitoring device is based on the TWR-S08 development board from NXP Semiconductors. The thesis is divided into the following parts: theoretical part with a brief introduction to the problem, conceptual part describing the principle of operation of the device and the selection of components used, implementation part describing the software of the device and its physical connection and a chapter describing the testing of the device.

KEYWORDS

Microcontroller MC9S08LH64, development board TWR-S08, MCP9808, temperature sensor, DS3231, RTC, Real-Time Clock, FLASH, SCI, I2C, PWM, temperature monitoring, server room.

Vysázeno pomocí balíčku `thesis` verze 4.09; `https://latex.fekt.vut.cz/`

KADERKIN, Vladislav. *Zařízení pro monitorování teploty*. Bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2025. Vedoucí práce: Ing. Tomáš Macho, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Vladislav Kaderkin
VUT ID autora:	241670
Typ práce:	Bakalářská práce
Akademický rok:	2024/25
Téma závěrečné práce:	Zařízení pro monitorování teploty

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Tomáši Machovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci, svému příteli a spolužákovi Volodymyru Mykhailu Horbalovi za technickou a morální podporu a za poskytnutí vybavení pro ladění, také svému příteli a spolužákovi Nikitovi Ivanovovi za morální podporu v průběhu celého studia.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

Moderní serverovny jsou technologicky vyspělé místnosti vybavené zařízením pro zpracování a ukládání dat. Udržování optimálních teplot zajišťuje bezproblémový a spolehlivý chod serverů. Přehřátí nebo nadměrné chlazení může vést k poruchám, ztrátě dat a nákladným odstávkám.

V rámci této práce bude navrženo zařízení pro monitorování teploty v serverovnách založené na vývojové desce TWR-S08 od společnosti NXP Semiconductors. Budou stanoveny požadavky na navrhované zařízení, popsána jeho koncepce, vybrán potřebný kalendářový obvod a snímač teploty a bude zvolen způsob napájení zařízení. Zařízení musí být schopno zaznamenat čas, kdy teplota překročí nastavené limity, uložit tyto údaje do vnitřní nevolatilní paměti mikrokontroléru a zobrazit teplotu na LCD displeji desky.

Dalším krokem bude vývoj softwaru pro toto zařízení, který bude schopen provádět výše uvedené funkce a přenášet monitorované údaje do počítače.

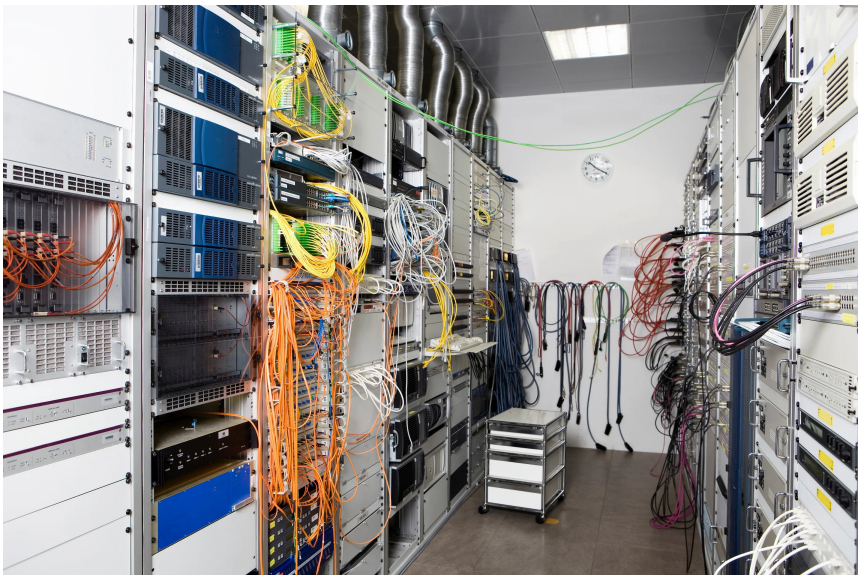
Hotové zařízení bude otestováno, vyhodnocena jeho funkčnost a případně navrženy vylepšení a úpravy.

Taktéž vývoj tohoto zařízení na bázi desky TWR-S08, která se doposud používala ve výukovém procesu v rámci předmětu BPC-MIC, je jednou z možností jejich dalšího využití, protože od příštího akademického roku budou tyto desky nahrazeny jinými.

Monitorování klimatu v serverovnach

Serverovna je speciálně vybavená místnost určená k umístění serverů a dalšího technického vybavení, kde jsou udržovány optimální podmínky. Hlavním úkolem těchto místností je zajistit nepřetržitý a spolehlivý provoz instalovaného zařízení.

Jedním z klíčových prvků serverovny je řízení klimatu. Servery při nepřetržitém provozu generují vysoké množství tepla, což pro jeho správnou funkci vyžaduje neustálé monitorování a regulaci parametrů prostředí. Pro udržování optimálních klimatických podmínek v serverovnách je třeba měřit parametry jako je teplota a vlhkost.



Obr. 1.1: Přehled serverovny [2]

Řízení klimatu

Americká společnost inženýrů v oblasti klimatických systémů (ASHRAE) vypracovala standardy [1], které doporučují pro kvalitní monitorování serverových místností a datových center. Doporučené klimatické podmínky v serverovnách tříd A1-A4 jsou: relativní vlhkost – maximálně 60 %, rosný bod – -9 °C až 15 °C, teplota v rozmezí 18 °C až 27 °C. Tyto parametry zajišťují spolehlivý a stabilní provoz zařízení, zabraňují korozi, kondenzaci a problémům se statickou elektřinou.

Monitorování teploty

V rámci této práce je realizováno pouze sledování teploty, proto se při stanovení požadavků na rozsah měřených hodnot budeme vycházet z výše popsaných doporučených hodnot.

Koncepce zařízení

V této části práce bude popsán princip fungování zařízení pro monitorování teploty. Dále bude proveden výběr modulů pro připojení k zařízení a zvolen způsob napájení.

2.1 Požadavky na monitorování teploty

Pro účinné sledování teploty v serverovně byly stanoveny požadavky na zařízení pro monitorování teploty:

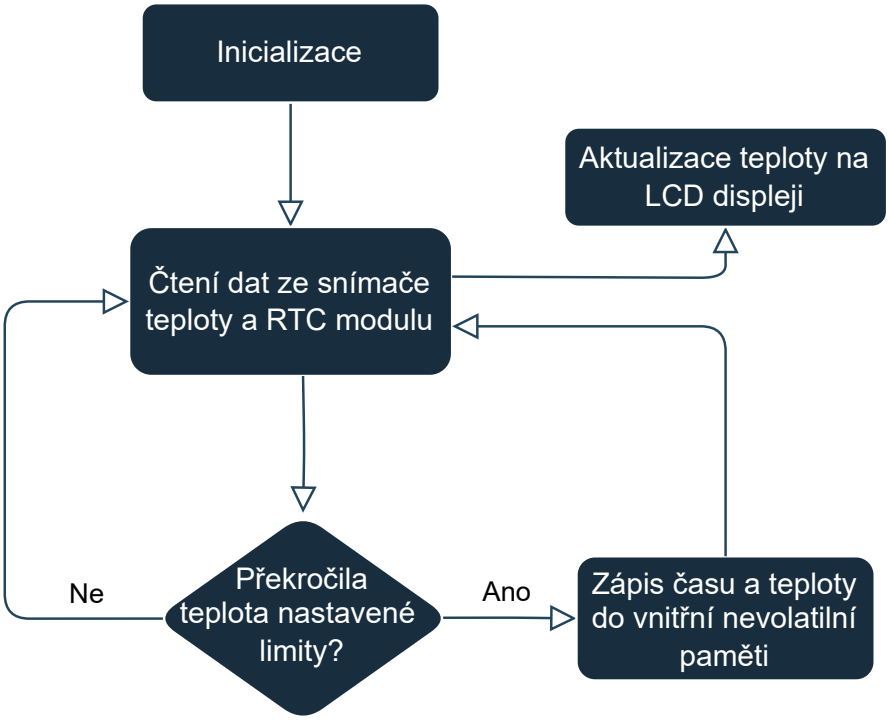
- Minimální požadovaný rozsah měření teploty je v rozmezí **15 °C až 30 °C**. Na základě standardů ASHRAE pro monitorování [1] je přípustný rozsah teplot 18–27 °C.
- Interval monitorování teploty je zvolen na **1 minutu**. Tato hodnota vychází z praktických doporučení a zohledňuje tepelnou setrvačnost zařízení, umožňuje včasné zaznamenávání teplotních změn a nezatěžuje mikrokontrolér.
- Minimální přesnost měření teploty je **±0,5 °C**. Jedná se o standardní hodnotu používanou pro většinu teplotních snímačů, která je dostatečná pro včasnou detekci odchylek od stanoveného teplotního rozsahu v podobných monitorovacích systémech.

Do vnitřní nevolatilní paměti mikrokontroléru se při překročení nastaveného mezního intervalu zapíše údaje o teplotě, minutách, hodinách, sekundách a kalendářním datu (den, měsíc, rok).

2.2 Princip činnosti zařízení

Základní činnost zařízení pro monitorování teploty je znázorněna na obrázku 2.1:

1. **Inicializace** – počáteční nastavení mikrokontroléru a modulů.
2. **Čtení dat z modulů** – komunikace s moduly pro přečtení časových údajů z modulu RTC a teplotních údajů ze snímače teploty. Ke čtení dat bude docházet každou minutu.
3. **Zpracování teploty** – pokud naměřená teplota překročí nastavené meze, její hodnota, čas a datum se zapíší do vnitřní nevolatilní paměti mikrokontroléru.



Obr. 2.1: Vývojový diagram činnosti zařízení

2.3 Vývojová deska TWR-S08

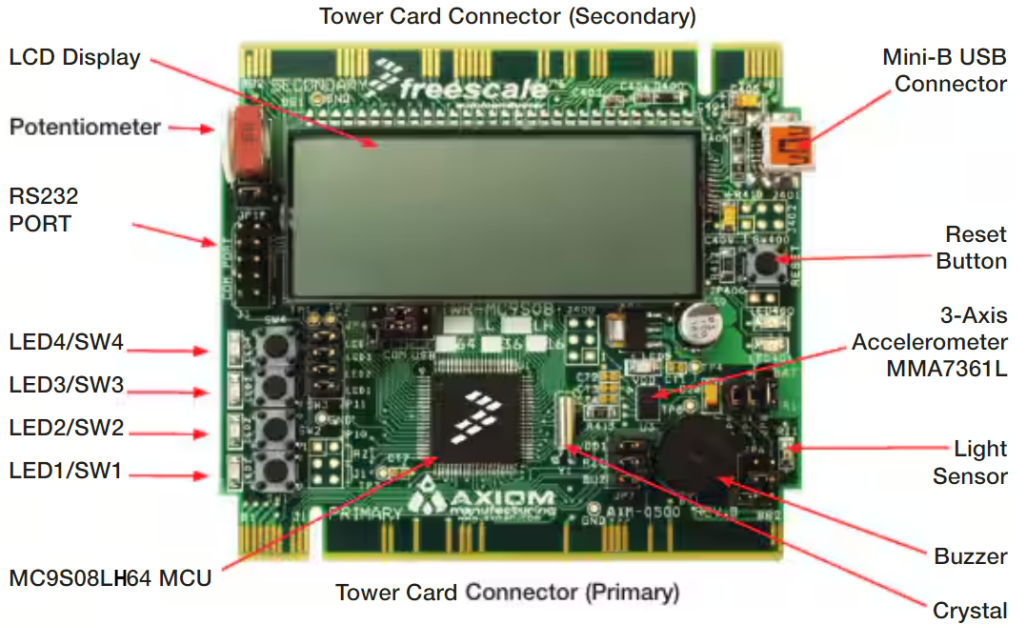
Zařízení bude vytvořeno na bázi vývojové desky TWR-S08 s nainstalovaným mikrokontrolérem **MC9S08LH64**.

Vývojová deska TWR-S08 byla vydána společností Freescale Semiconductor (nyní NXP) pro podporu mikrokontrolérů řady MC9S08LL a MC9S08LH v roce 2010. Byla navržena pro integraci s modulární platformou **Freescale Tower System**, která usnadňuje rychlé prototypování a opětovné využití nástrojů. Všechny signály mikrokontroléru jsou dostupné prostřednictvím krajních konektorů. Součástí této vývojové desky je vývojové prostředí pro vestavěný software CodeWarrior, což je kompletní integrované prostředí pro programování a ladění vestavěného mikrokontroléru.

Vývojová deska TWR-S08 obsahuje [3]:

- Analogový akcelerometr MMA7361L
- Sériové rozhraní RS-232 s 2x5 pinovým konektorem
- Open-Source BDM plně podporovaný prostředím CodeWarrior
- Konektor BDM_PORT pro externí BDM kabel (není osazen)
- Vestavěný regulátor $+3,3\,\mathrm{V}$

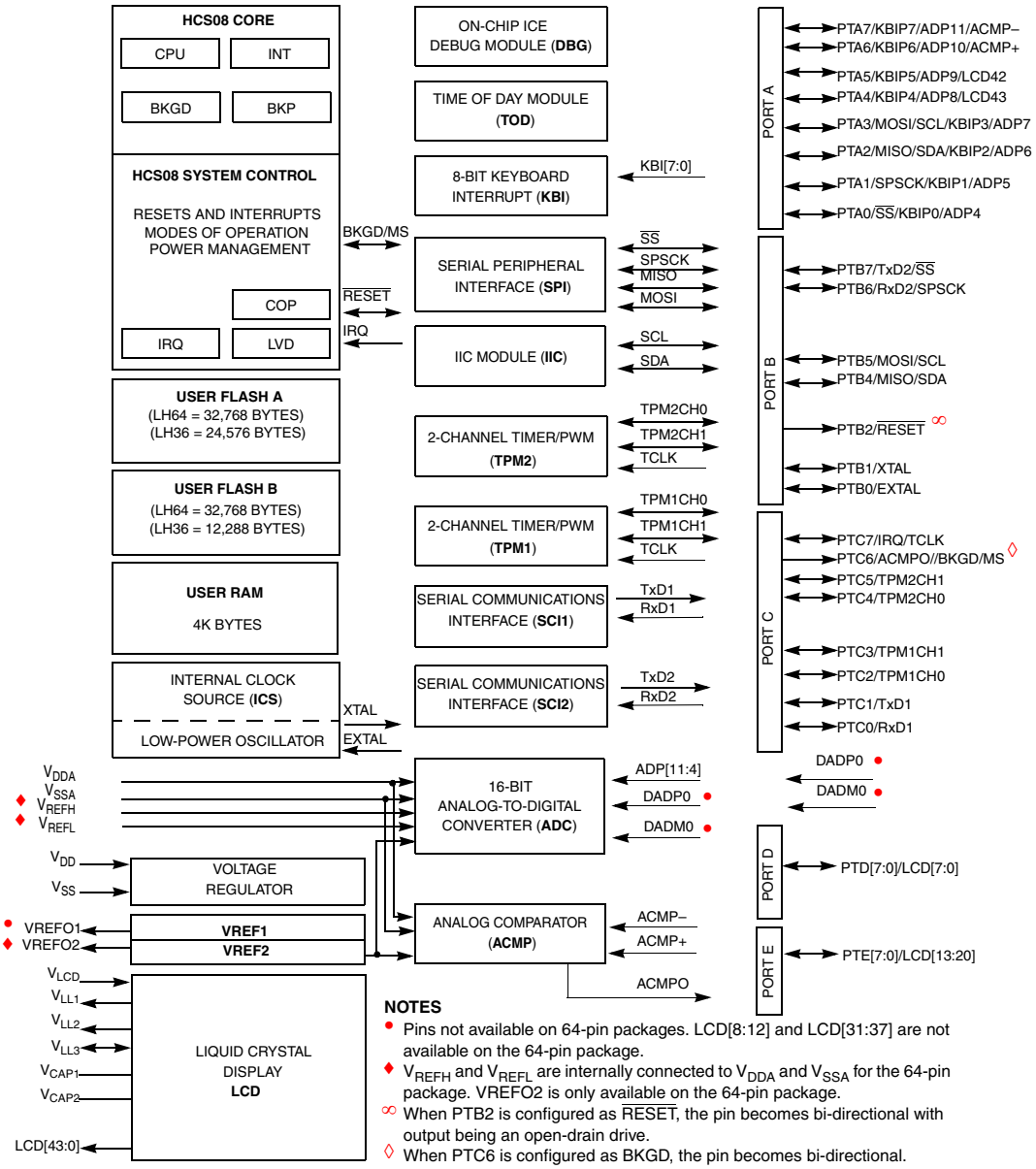
- Držák na knoflíkovou baterii (namontovaný na zadní straně)
- Výběr zdroje napájení
  - Napájení z USB-BDM
  - Napájení z vestavěného regulátoru
  - Napájení z držáku baterie CR2325
  - Napájení z krajních konektorů Tower System
- Uživatelské komponenty
  - 4 tlačítka
  - 4 LED indikátory
  - 5k ohmový potenciometr s dolní propustí
  - Světelný senzor s dolní propustí a operačním zesilovačem
  - 2,4 kHz Piezo bzučák



Obr. 2.2: Komponenty vývojové desky TWR-S08 [4]

2.3.1 Mikrokontrolér MC9S08LH64

Mikrokontrolér MC9S08LH64 patří do řady HCS08, která se vyznačuje nízkou cenou, nízkou spotřebou energie a vysokým výkonem. Všechny mikrokontroléry této řady jsou dostupné v různých konfiguracích, co se týče modulů, velikostí a typů paměti i druhů pouzder. Jádro mikrokontroléru pracuje s logickým napětím 3,3 V a periferní I/O – s napětím 5 V, jeho blokové schéma je znázorněno na obrázku 2.3.



Obr. 2.3: Blokové schéma mikrokontroléru MC9S08LH64 [5]

Hlavní parametry [5]:

- 8-bitová centrální procesorová jednotka (CPU) HCS08
  - Až 40 MHz CPU při napětí 2,1 V až 3,6 V v teplotním rozsahu -40 °C až 85 °C
  - Podpora až 32 zdrojů přerušení/resetu
- Paměť na čipu
  - FLASH: 64 kB (dva bloky o velikosti 32 kB)
  - RAM: 4 kB operační paměti

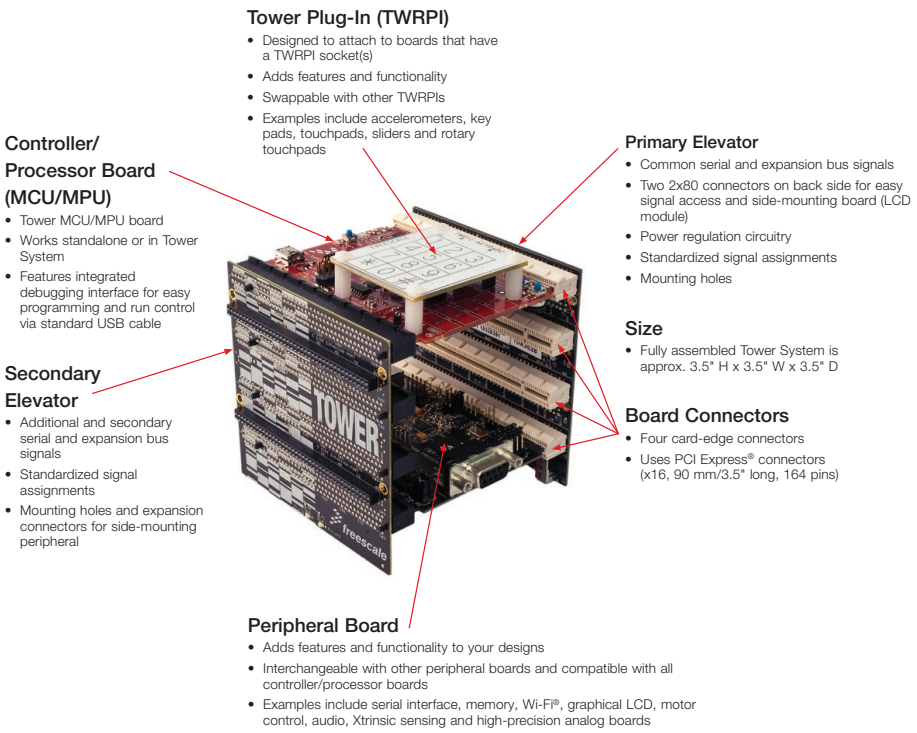
- Periferní zařízení
  - Až 32 GPIO pinů, dva piny pouze pro výstup
  - LCD – ovladač LCD až do 8x36 nebo 4x40 segmentů s interní nábojovou pumpou a možností interně regulované reference LCD, kterou lze trimovat pro řízení kontrastu
  - I^2C – Inter-Integrated Circuit, provoz s rychlostí až 100 kb/s při maximálním zatížení sběrnice
  - SCI – Serial Communication Interface
  - TOD – Time-of-day, 8-bitový čtvrtsekundový čítač se srovnávacím registrem
  - TPMx – Timer Pulse-Width Modulator, dvoukanálový modul časovače a PWM
  - atd.

2.3.2 Freescale Tower System

Freescale Tower System je modulární platforma určená pro vývoj vestavěných aplikací s využitím 8-bitových, 16-bitových a 32-bitových mikrokontrolérů a procesorů. Systém se skládá z hlavních modulů procesorů nebo mikrokontrolérů, periferních modulů a spojovacích modulů nazývaných Elevator Boards (obr. 2.4):

1. Moduly procesorů a mikrokontrolérů – tvoří základ systému a podporují provoz jak v samostatném režimu, tak jako součást Tower System. Tyto moduly zahrnují rozhraní pro programování a ovládání přes USB.
2. Periferní moduly – rozšiřují funkcionalitu pomocí rozhraní, grafických LCD, pamětí, řízení motorů, Wi-Fi a dalších komponent.
3. Elevator Boards – zajišťují standardizovaný přenos signálů mezi moduly. Primary Elevator a Secondary Elevator obsahují konektory PCI Express (164 pinů) [6].

Platforma Freescale Tower System umožňuje podporu různých komunikačních protokolů (I^2C, SPI, UART), možnost připojení až tří periferních modulů současně a napájení přes libovolný připojený modul nebo přes konektory modulu Primary Elevator (v tomto případě bude takové napájení prioritní).



Obr. 2.4: Ukázkový příklad platformy Freescale Tower System [6]

2.4 Kalendářový obvod

Součástí zařízení pro monitorování teploty je pravidelný záznam naměřených hodnot do paměti s přesným datem a časem měření. To je nutné pro následné analýzy teplotních změn a pro záznam překročení mezní hodnoty. K realizaci tohoto úkolu je zapotřebí modul reálného času (RTC), který sleduje aktuální čas a datum pomocí integrovaného krystalového oscilátoru.

Modul Time of Day

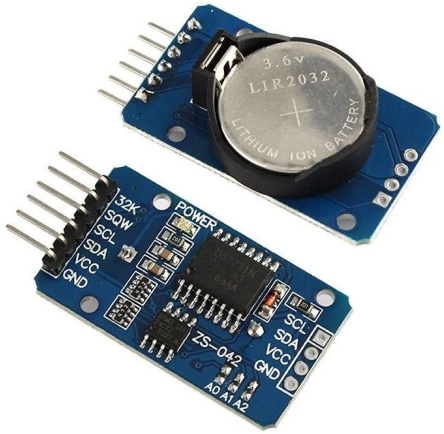
Mikrokontrolér MC9S08LH64 má vestavěný modul Time of Day (TOD), což je 8-bitový čítač s předděličem a číslicovým komparatorem. Tento modul lze použít jako RTC pro základní časování. Použití externího modulu reálného času je však vhodnější z následujících důvodů:

- Vestavěný RTC (TOD) ztratí časové údaje, pokud dojde k výpadku napájení mikrokontroléru.
- Zdroj hodinového kmitočtu mikrokontroléru nemá teplotní kompenzaci.
- Vestavěný modul má základní funkce a nepodporuje sledování složitějších kalendářních údajů, jako je den, měsíc a rok.

Na základě výše uvedeného je nejvhodnější variantou použití externího modulu RTC. Ten zajistí dlouhodobě přesnější a spolehlivější záznam teploty a také uložení údajů o čase a datu i v případě výpadku napájení. Modul TOD však lze použít k zajištění minutového intervalu pro monitorování.

Kalendářový obvod DS3231

V současné době je na trhu mnoho modulů RTC, které lze v našem projektu použít. Při výběru vhodného modulu byla věnována pozornost především takovým kritériím, jako je vysoká spolehlivost, přesnost, široká obliba a přijatelná cena. S ohledem na tyto faktory byl pro realizaci projektu vybrán modul reálného času **DS3231** (obr. 2.5).



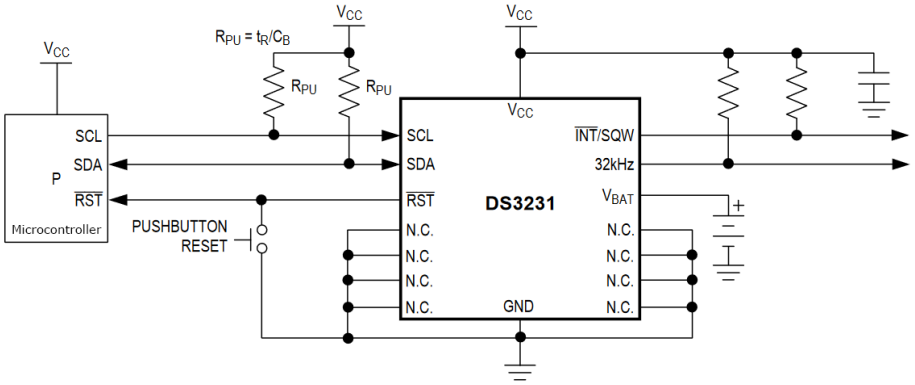
Obr. 2.5: Kalendářový obvod DS3231 [7]

DS3231 je vysoce přesný real-time clock (RTC) modul s integrovaným teplotně kompenzovaným krystalovým oscilátorem a krystalem. Je navržen pro komunikaci prostřednictvím sběrnice I^2C a je schopen uchovávat přesný čas i při výpadku hlavního napájení díky záložní baterii. Modul má registry ve kterých jsou uchovávány sekundy, minuty, hodiny, den, den v týdnu, měsíc a rok (s automatickou korekcí přestupných roků), alarmy a atd. Schéma typického provozního obvodu modulu je znázorněno na obrázku 2.6.

Klíčové vlastnosti:

- Přesnost je ±2 ppm od 0 °C do 40 °C a ±3, 5 ppm v rozmezí od -40 °C do +85 °C.
- Rozsah napájecího napětí VCC a VBAT je 2,3 V až 5,5 V.
- Záložní napájení – automatický přechod na záložní baterii při výpadku hlavního napájení.

- Teplotní kompenzace – vestavěný teplotní snímač zajišťuje automatické upravení frekvence generátoru.
- Komunikace prostřednictvím protokolu I^2C s rychlostí až 400 kHz [8].



Obr. 2.6: Typický provozní obvod DS3231 [8]

Modul DS3231 je pro náš projekt vhodný z následujících důvodů:

- Modul DS3231 je vybaven vestavěnou teplotní kompenzací, která zajišťuje vysokou přesnost a dlouhodobou stabilitu.
- Modul uchovává čas i po odpojení hlavního napájení díky vestavěné baterii. To má zásadní význam pro dlouhodobé monitorování, protože časové údaje zůstávají uloženy i v případě výpadku sítě nebo mikrokontroléru.
- DS3231 je populární modul mezi vývojáři, což zajišťuje dostupnost knihoven a příkladů pro zjednodušení integrace.

2.5 Snímač teploty

Ke sledování teploty v serverovně potřebujeme přímo samotný teplotní snímač, který má přesnost ±0,5 °C nebo lepší, rozsah měření minimálně 15–30 °C a je snadně ovladatelný mikrokontrolérem.

Vestavěný teplotní snímač

Podle dokumentace [10] je vestavěný snímač teploty, který je k dispozici v mikrokontrolérech řady HCS08, včetně MC9S08LH64, součástí periferie ADC (S08ADCV1).

Vestavěný teplotní snímač pracuje na principu P-N přechodu. Pro měření teploty je nutné použít typické parametry uvedené v dokumentaci k mikrokontrolérům řady HCS08. Činnost teplotního snímače je popsána přibližnou přenosovou funkcí [10]:

$$TEMP = 25 - \frac{V_{TEMP} - V_{TEMP25}}{m} \tag{2.1}$$

kde: V_{TEMP} je napětí na vstupu A/D převodníku při teplotě okolí

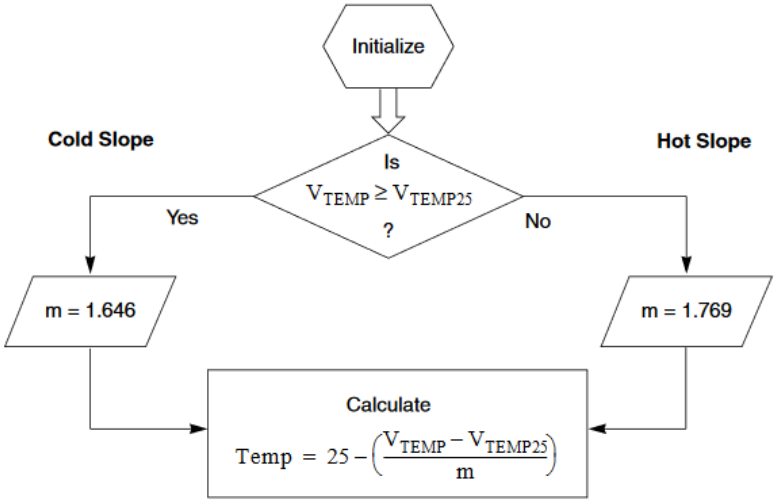
V_{TEMP25} je napětí na vstupu A/D převodníku při 25 °C a VDD = 3 V

m je sklon napětí vůči teplotě pro teplý nebo studený režim ve V/°C

Parametr m je odlišný pro teplý a studený sklon rovnice:

- Teplý sklon m se využívá při teplotách nad 25 °C
- Studený sklon m se využívá při teplotách pod 25 °C

Některé technické specifikace v části elektrických charakteristik poskytují typické parametry pro V_{TEMP25} a m (parametry teplého a studeného sklonu). Princip softwarové obsluhy teplotního snímače je uveden na obrázku 2.7 (na příkladu parametrů pro mikrokontrolér MC9S08QG8).



Obr. 2.7: Vývojový diagram měření teploty HCS08 [10]

V této implementaci (obr. 2.7) software nejprve inicializuje ADC a poté načte data ze snímače teploty. Na základě získané hodnoty se zvolí parametr sklonu: studený nebo teplý a nakonec se vypočítá teplota.

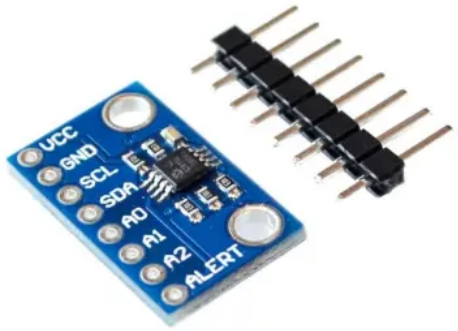
Nicméně tento vestavěný teplotní snímač má několik omezení:

- Je nutná filtrace signálu z ADC pro dosažení přesnějších hodnot, což zvyšuje složitost zpracování dat.
- Pro dosažení vysoké přesnosti je zapotřebí složitá vícestupňová kalibrace při různých teplotách. Bez kalibrace je typická přesnost ±4,5 °C, po kalibraci dosahuje ±2,5 °C, což je stále výrazně méně přesné než u externích snímačů.

Na základě výše uvedených nevýhod bylo rozhodnuto použít externí teplotní snímač.

Externí snímač teploty MCP9808

Při výběru externího teplotního snímače byl kladen hlavní důraz na následující kritéria: požadovaná přesnost měření, snadné připojení k mikrokontroléru a cenová dostupnost. S ohledem na tyto požadavky byl vybrán snímač **MCP9808**.



Obr. 2.8: Snímač teploty MCP9808 [11]

Teplotní snímač MCP9808 převádí teploty v rozsahu -20 °C až +100 °C na digitální hodnotu s přesností ±0,25 °C (typicky), ±0,5 °C (max.). Má uživatelsky nastavitelné rozlišení měření: +0,5 °C, +0,25 °C, +0,125 °C, +0,0625 °C (defaultní). Rozsah pracovního napětí je 2,7 V až 5,5 V, pracovní proud 200 μA (typický) a komunikace prostřednictvím I^2C (SMBus) až 400 kHz.

Snímač využívá zdroj napětí citlivý na teplotu na bázi band-gap, který vydává analogové napětí úměrné absolutní teplotě. Toto napětí se pak převádí na 13-bitovou digitální hodnotu ve formátu dvojkového doplnku pomocí Sigma-Delta ADC. Tato hodnota je zapsána do 16-bitového registru pro následné čtení hodnoty uživatelem.

Po přečtení 16-bitové hodnoty ze snímače je třeba ji zpracovat a převést na decimální format. To se provádí podle následujícího vzorce (kde *UpperByte* je 15-8 bity, *LowerByte* je 7-0 bity) [12]:

$$\text{Temperature} = 256 - (\text{UpperByte} \cdot 16 + \text{LowerByte} \div 16) \quad T < 0°\text{C} \tag{2.2}$$

$$\text{Temperature} = \text{UpperByte} \cdot 16 + \text{LowerByte} \div 16 \quad T \geq 0°\text{C} \tag{2.3}$$

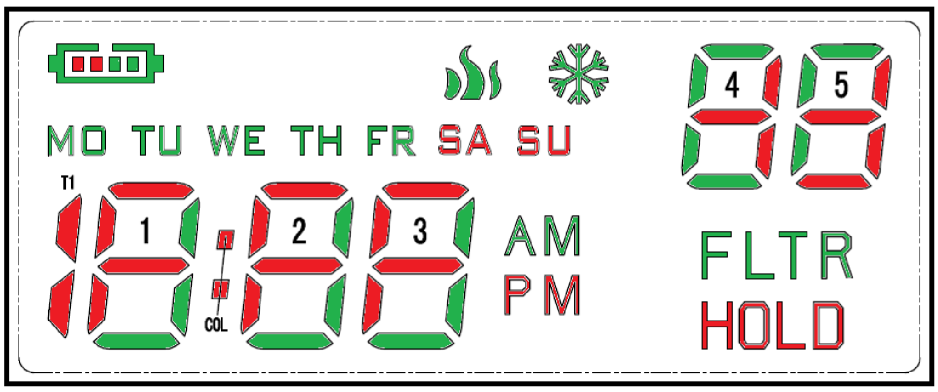
Důvody použití tohoto snímače:

- Přesnost a rozsah měření odpovídající našim požadavkům.
- Protokol přenosu dat je stejný jako u modulu reálného času, což usnadní vývoj softwaru.
- Dobrý poměr ceny a kvality.

2.6 LCD Displej

Součástí práce je zobrazení aktuální teploty prostředí během monitorování. S tím nám pomůže LCD displej zabudovaný do vývojové desky TWR-S08 (obr. 2.2).

TWR-S08 používá LCD displej GD-5306P, 2x28 segmentů chip-on-glass, připojený přímo k mikrokontroléru MC9S08LH64. Mikrokontrolér zajišťuje interní čerpání náboje a regulované referenční napětí potřebné pro provoz LCD [3]. Displej GD-5360P má 30 pinů, které ovládají 55 segmentů, včetně pěti sedmisegmentových symbolů pro indikaci tepla (plamen) a chladu (sněhová vločka), a má indikátor nabití baterie, předdefinované textové symboly AM, PM, FLTR, HOLD a anglické zkratky pro dny v týdnu (MO, TU, WE, TH, FR, SA a SU) (obr. 2.9) [13].



Obr. 2.9: Rozložení segmentů displeje GD-5360P [13]

Z uvedených možností zobrazení se použijí segmenty 1 až 3 pro zobrazení teploty a symboly pro indikaci tepla (plamen) a chladu (sněhová vločka), pokud je teplota mimo nastavené limity (obr. 2.9).

Data budou zobrazena pomocí hotové knihovny pro ovládání LCD displeje GD-5360P, která byla napsána v rámci bakalářské práce "Knihovna pro řízení LCD displeje GD-5360P"[13].

2.7 Napájení

Jak již bylo zmíněno v sekci Vývojová deska TWR-S08, je k dispozici 5 možností napájení zařízení:

- Připojení zdroje napětí přímo k vývojové desce TWR-S08:
  1. přes port USB-BDM
  2. přes napájecí piny
  3. pomocí baterie CR2325
- Prostřednictvím modulu Primary Elevator systému Freescale Tower System:

4. přes port USB
5. přes napájecí svorky

Pro napájení zařízení jsou vybrány 2 možnosti. První je napájení **pomocí baterie CR2325** (nebo podobné) připojené na zadní straně desky TWR-S08. Takové napájení zajišťuje autonomnost a nezávislost na hlavním zdroji napájení serverovny. Druhou možností je napájení **přes port USB-BDM**. V tomto případě lze druhý konec kabelu připojit přes standardní 5 V zdroj do zásuvky. Zde však bude zařízení závislé na hlavním napájení serverovny, což na druhou stranu není závažná nevýhoda, protože v případě výpadku hlavního napájení bude veškeré vybavení serverovny vypnuto a monitorování teploty nemá smysl. Napětí z desky se pomocí modulu Primary Elevator přenáší do všech pracovních modulů zařízení (teplotní snímač a modul RTC).

Obr. 2.10: Baterie CR2325 [14]

Realizace

Na základě koncepce zařízení a s využitím výše popsaných modulů byl vytvořen kompletní funkční prototyp zařízení pro monitorování teploty. Zařízení zahrnuje periodické monitorování teploty, záznam hodnot do nevolatilní paměti a interakci s uživatelem. Během realizace byl koncept dodržen a všechny požadavky byly splněny.

Vlastnosti zařízení:

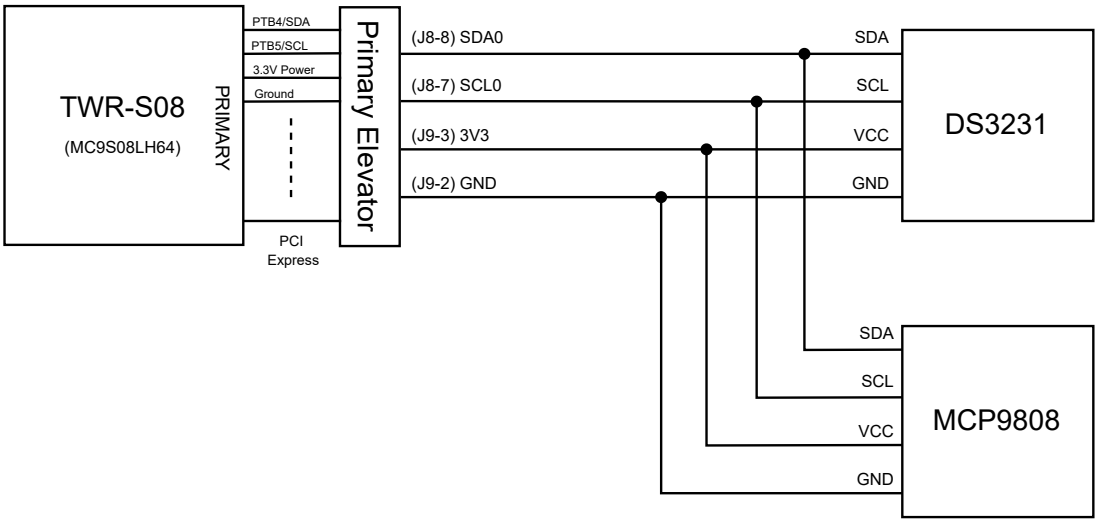
- Možný rozsah monitorování – -20°C až 40°C
  - Defaultní teplotní rozsah (po zapnutí zařízení): dolní mez – 18°C, horní mez – 27°C
- Přesnost měření teploty – ±0,5°C
- Komunikace s uživatelem
  - prostřednictvím konzoly SCI (UART)
  - prostřednictvím tlačítek na desce
  - zvukové signály
  - zobrazení informací na LCD displeji
- Frekvence monitorování – každou 1 minutu
- Výběr zdroje napájení
  - pomocí baterie 3 V
  - přes port USB-BDM
- Uchovává čas a monitorovaná data i bez napájení
- Dostupná paměť pro data – 2 KB

V následujícím textu bude podrobně popsán hardware (schéma zapojení) a software zařízení (knihovny, funkce, algoritmy). Softwarová část byla napsána ve vývojovém prostředí CodeWarrior.

3.1 Schéma a zapojení zařízení

Na obr. 3.1 je schéma zařízení. Vzhledem k tomu, že vyrábíme pouze prototyp zařízení, jsou moduly připojeny k hlavnímu mikrokontroléru MC9S08LH64 pomocí modulu Primary Elevator [9, str. 3, konektory], protože jinak není možné moduly připojit bez vážného zásahu (např. pájení).

Napájení, jak již bylo popsáno, bude zajištěno pomocí baterie CR2325 (nebo podobné) nebo přes port USB-BDM, pro volbu napájení je třeba přepnout jumper do požadované polohy (viz obr. 3.2).

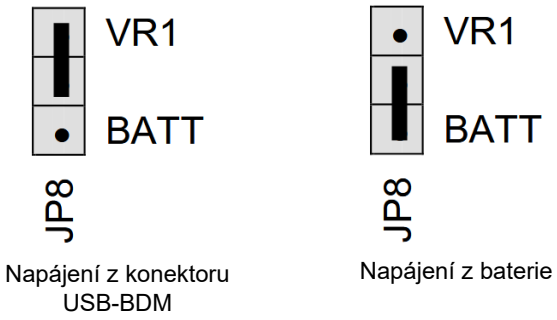


Obr. 3.1: Schéma zapojení zařízení

3.2 Komunikace s uživatelem, přenos dát do počítače

Ke konfiguraci zařízení, získávání stavových a akčních informací a především k získávání monitorovaných dat ve formátu příkazů pro zpracování zadávaných prostřednictvím konzoly se používaly periferie **SCI** (Serial Communication Interface).

Funguje to následovně: uživatel se připojí přímo k desce přes konektor USB-BDM (je nutné na desce umístit jumper na VR1, viz obr. 3.2), otevře terminál UART (Serial) na počítači (např. přes program PuTTY), zadá připojený port a přenosovou rychlost (v našem případě 9600 Bd) a zahájí komunikaci.

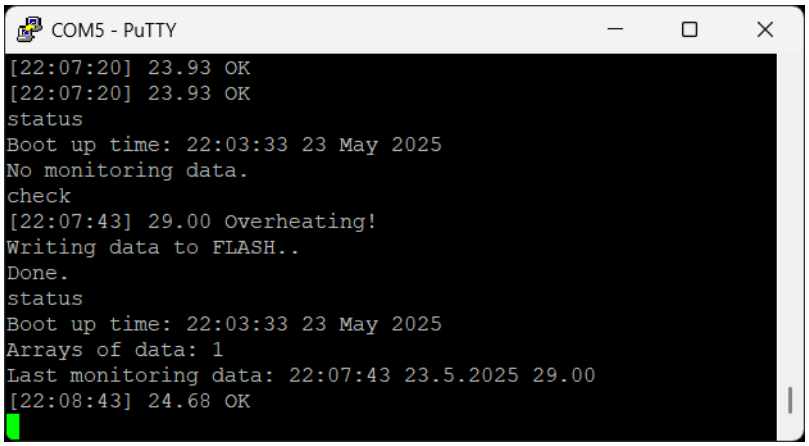


Obr. 3.2: Umístění jumperu [3]

Pro konzolu byl napsán seznam příkazů, které umožňují provádět základní dotazy na mikrokontrolér a spravovat sadu monitorovacích dat. Příkazy jsou následující:

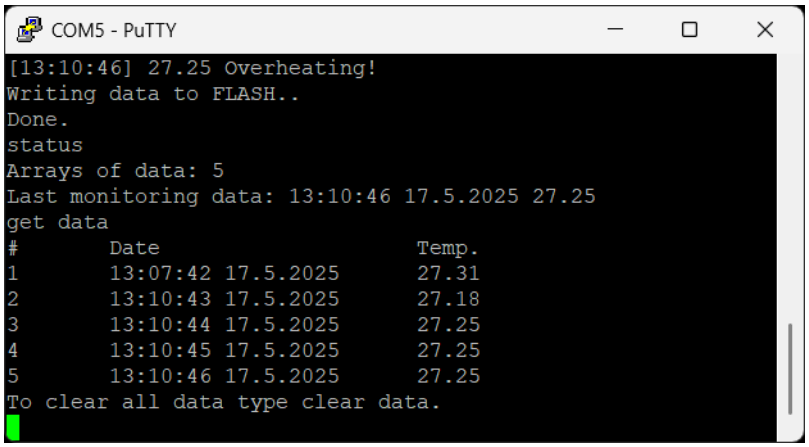
- **help** – získat seznam dostupných příkazů
- **status** – získat stav monitorování: zobrazí čas spuštění zařízení, v případě přítomnosti dat v paměti zobrazí počet zaznamenaných datových sad a poslední

monitorovaná data, pokud je paměť plná, zobrazí zprávu o plné paměti



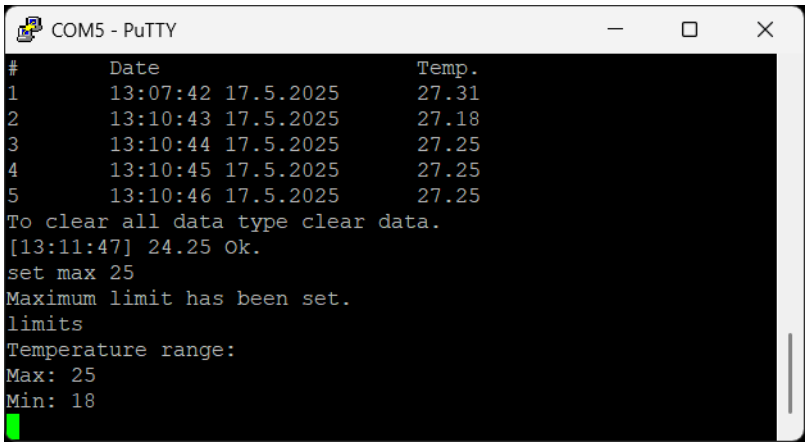
Obr. 3.3: Příklad příkazu *status*

- **check** – manuální kontrola teploty
- **restart** – restartovat zařízení
- **get data** – získat všechna monitorovaná data



Obr. 3.4: Příklad příkazu *get data*

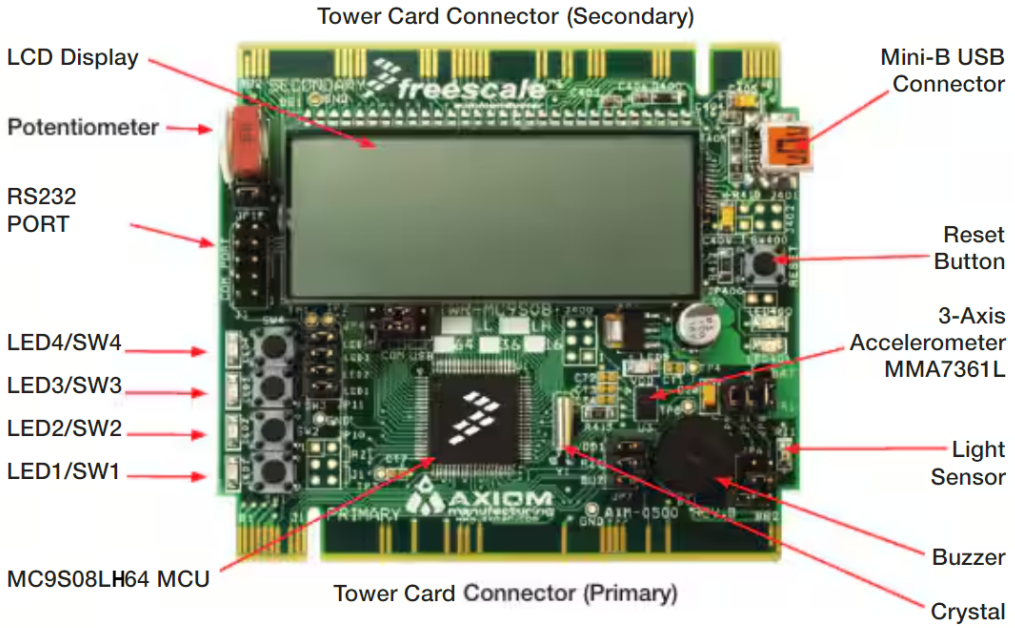
- **clear data** – vymazat všechna monitorovaná data
- **set** – nastavit monitorovací limity, má 3 parametry: *min* (dolní mez), *max* (horní mez) a *default* (defaultní limity), pro nastavení limitu je nutné za příkazem zadat požadovaný parametr a případně číslo požadované teploty, např. *set max 25*



Obr. 3.5: Příklad příkazu *set*

- **limits** – získat aktuální teplotní limity
- **time** – získat aktuální čas z modulu RTC
- **time set** – nastavit čas modulu RTC, zadává se příkaz ve formátu *"time set hh mm ss dd mm yy w"*

Kromě konzolového příkazu pro nastavení limitů má zařízení také možnost jejich nastavení pomocí tlačítek na desce. Pro nastavení vlastních limitů je třeba stisknout tlačítko SW1 a poté pomocí tlačítek SW3 (zvýšení hodnoty) a SW2 (snížení hodnoty) nastavit požadovaný rozsah (viz obr. 3.6).



Obr. 3.6: Komponenty vývojové desky TWR-S08 [4]

V případě nedostatku paměti se na konzolu pošle upozornění a LED2 začne

blikat. Pokud teplota překročí limit, bude na konzolu odesláno upozornění, vydán zvukový signál a na LCD displeji se zobrazí teplota se znakem překročení limitu (sněhová vločka – pokud je překročen dolní limit, plamen – pokud je překročen horní limit). Teplotu je možné také kontrolovat manuálně stisknutím tlačítka SW2.

3.2.1 Knihovna SCI.h

K práci s periferním zařízením SCI byla napsána příslušná knihovna. Při tvorbě této knihovny byly využity informace z návodu k práci s periferiemi mikrokontrolérů rodiny HCS08 [15]. Knihovna obsahuje následující funkce.

SCI_Init

Inicializace modulu SCI: zapnutí přerušení při příjmu, zapnutí funkcí příjmu a odesílání. Funkci jsou předány proměnné pro nastavení registrů SCI1BDH a SCI1BDL pro nastavení rychlosti (Baud rate). Rychlost se vypočítá podle následujícího vzorce [15, str. 65]:

$$\text{Baud Rate} = \frac{f_{\text{Bus}}}{[\text{SBR12:SBR0}] \cdot 16} \tag{3.1}$$

Bity SBR12 až SBR0 se nacházejí ve výše zmíněných registrech SCI1BDH a SCI1BDL [5, str. 298].

```
void SCI_Init(unsigned char divH, unsigned char divL)
{
  SCI1C1 = 0x00;  /* Vypnutý Loop mode,  Tx výstup není
     invertován, 8-bitové znaky, probuzení volné linky,
     vypnutí paritního bitu  */
  SCI1C2 = 0x2C;  /* Povolit přerušení SCI Receive,
     transmitter a receiver */
  SCI1C3 = 0x00;  /* Zakázat všechna chybová přerušení */

  SCI1BDH = divH; /* Baud rate dělič */
  SCI1BDL = divL;
}
```

Výpis 3.1: Funkce SCI_Init

ReadByte

Čtení jednoho bajtu, vrátí hodnotu registru datového SCI1D.

```
unsigned char ReadByte(void)
{
```

```
  unsigned char Temp;
  // Vymazání příznaku plného příjmového registru
  Temp = SCI1S1;
  // Počkat, až bude přijímací buffer prázdný
  while (!SCI1S1_RDRF);
  return SCI1D;   // Čtení a vrácení datového bufferu
}
```

Výpis 3.2: Funkce ReadByte

SendByte

Odeslat jeden bajt, zapíše předávanou hodnotu do datového registru SCI1D.

```
void SendByte(const unsigned char byte)
{
  // Čekání na vyprázdnění vysílacího bufferu
  while (!SCI1S1_TDRE);
  SCI1D = byte;     // Odeslat bajt
}
```

Výpis 3.3: Funkce SendByte

SendString

Odeslat řetězec bajtů, volá funkci SendByte pro každý bajt předaného řetězce.

```
void SendString(const unsigned char *array)
{
  while (*array && *array != '\0')
  SendByte(*array++);
}
```

Výpis 3.4: Funkce SendString

3.3 Práce s pamětí FLASH

Nevolatilní paměť mikrokontroléru je paměť FLASH o celkové kapacitě 64 KB. Pro práci s touto pamětí poskytuje výrobce řadu příkazů: *Byte program* – zápis bajtu, *Byte program (burst)*, *Page erase* – vymazání stránky, *Mass erase* – vymazání celé paměti, *Blank check* – kontrola, zda je paměť prázdná.

Pro data je vyhrazena paměť o velikosti 4 stránek FLASH (každá 512 B), celkem **2 KB**. Rozsah adres paměti: 0xC000 - 0xC7FF, tento rozsah lze programově

změnit zapsáním požadované adresy místo 0xC000 (soubor Terminal.c, viz příslušná kapitola):

```
#define FIRST_PAGE  0xC000
#define SECOND_PAGE (FIRST_PAGE+0x0200)
#define THIRD_PAGE  (SECOND_PAGE+0x0200)
#define FOURTH_PAGE (THIRD_PAGE+0x0200)
#define FIFTH_PAGE  (FOURTH_PAGE+0x0200)
```

Výpis 3.5: Definování adres paměti

Jedno datové pole obsahuje **8 bajtů**: hodiny, minuty, sekundy, den, měsíc, rok a 2 bajty teploty. Celkem lze do paměti současně uložit až 255 polí monitorovacích dat (proč ne 256, je popsáno dále v části Terminal.c, popis funkce writeData).

3.3.1 Knihovna flash.h

V programu zařízení byly použity pouze 2 příkazy – *Byte program (burst)* a *Page erase*, ale v knihovně napsané pro práci s pamětí FLASH byly implementovány funkce pro provádění všech příkazů kromě *Mass erase*.

FlashInit

Inicializace modulu pro práci s pamětí FLASH, nastavení děliče frekvence procesoru pro dosažení požadované frekvence modulu. Pro správnou funkci příkazů při práci s pamětí FLASH musí být dělička frekvence nastavena tak, aby modul pracoval na frekvenci 150 kHz až 200 kHz [5, str. 79]. Vzorec pro výpočet frekvence modulu je následující (pro bit PRDIV8 = 0 v registru FCDIV – *Prescale (Divide) Flash Clock by 8*):

$$f_{\mathrm{FCLK}} = f_{\mathrm{Bus}} \div (\mathrm{DIV} + 1) \tag{3.2}$$

Funkci je předána hodnota děliče.

```
void FlashInit(unsigned char div)
{
    FCDIV = div;
}
```

Výpis 3.6: Finkce FlashInit

FlashCommand

Provedení předaného příkazu na předané adrese (je-li to nutné) se zápisem předané hodnoty (je-li to nutné). Algoritmus byl převzat z manuálu k mikrokontroléru [5, str. 74, Figure 4-9]:

1. Zakázání přerušení.
2. Čekání na flag FCBEF (*Flash Command Buffer Empty*), který signalizuje připravenost k zápisu příkazu do příkazového registru.
3. Vymazání flagů FPVIOL (*Protection Violation* – porušení ochrany) a FACCERR (*Access Error* – chyba přístupu).
4. Zápis přenášené hodnoty do přenášené adresy (pokud je vyžadován):

```
    *(volatile uint8_t *)address = data;
```

5. Zápis přenášeného příkazu do příkazového registru a jeho spuštění zápisem 1 do FCBEF.
6. Pokud dojde k chybám FPVIOL nebo FACCERR, ukončí se funkce s kódem chyby.
7. Čekání na dokončení programu (dokud není příznak FCCF = 1 – *Flash Command Complete*).
8. Zapnutí přerušení a návrat z funkce.

Úplný kód funkce naleznete v příloze.

FlashProg

Funkce zápisu jednoho bajtu do adresové buňky. Provádí se voláním funkce FlashCommand s příkazem *Byte program* (0x20), adresou a hodnotou, která se má zapsat:

```
uint8_t FlashProg(uint16_t address, uint8_t data)
{
  return FlashCommand(mByteProg, address, data);
}
```

Výpis 3.7: Funkce FlashProg

FlashErase

Funkce pro vymazání jedné stránky. Provádí se také voláním funkce FlashCommand s příkazem *Page erase* (0x40), libovolnou adresou na požadované stránce a fiktivní hodnotou:

```
uint8_t FlashErase(uint16_t address)
{
  return FlashCommand(mPageErase, address, 0xFF);
}
```

Výpis 3.8: Funkce FlashErase

FlashBlank

Funkce kontroly, zda je celá paměť FLASH prázdná. Podobně jako předchozí funkce se provádí voláním funkce FlashCommand s předáním příkazu *Blank check* (0x05), fiktivní adresy a fiktivní hodnoty.

FlashRead

Čtení paměťové buňky podle adresy, vrátí obsah paměťové buňky přenášené adresy:

```
uint8_t FlashRead(uint16_t address)
{
  return *(volatile uint8_t *)address;
}
```

Výpis 3.9: Funkce FlashRead

FlashBurst

Program pro provedení "burst"zápisu do paměti. Tento program se používá při zápisu pole dat do paměťových buněk, kdy bude zápis rychlejší než zápis každého bajtu zvlášť (např. pomocí funkce FlashProg). Tohoto efektu se dosáhne tím, že se mezi jednotlivými operacemi zápisu udržuje v modulu vysoké napětí, takže se neztrácí čas na dobíjení.

Funkce provádí postupný zápis hodnot z předaného pole s předanou adresou jako počátečním bodem. Algoritmus byl převzat z manuálu k mikrokontroléru [5, str. 75, Figure 4-10]:

1. Zakázání přerušení.
2. Vymazání chybových příznaků FPVIOL a FACCERR.
3. Pro každý prvek přenášeného pole se po dobu délky přenášeného pole provede následující:
   - 3.1 Čekání na příznak FCBEF, který signalizuje připravenost k zápisu příkazu do příkazového registru.
   - 3.2 Zápis hodnoty z předaného pole na předanou adresu.
   - 3.3 Zápis příkazu *Burst program* (0x25) do příkazového registru a jeho spuštění zápisem 1 do FCBEF.
   - 3.4 Pokud dojde k chybám FPVIOL nebo FACCERR, ukončí se funkce s kódem chyby.
4. Čekání na dokončení programu (dokud není příznak FCCF = 1).
5. Povolení přerušení a návrat z funkce.

Úplný kód funkce naleznete v příloze.

3.4 Knihovny pro práci s moduly

Všechny moduly použité v zařízení přenášejí data prostřednictvím protokolu I^2C, takže pro realizaci přenosu dat je nutné napsat knihovnu pro přímou práci s linkou I^2C a pomocí této knihovny pak přímo nastavit přenos dat jednotlivých modulů.

Mikrokontrolér MC9S08LH64 použitý na desce má podporu periferií I^2C. Pro práci s touto periferií má vyhrazené registry a vektor přerušení pro zpracování akcí a chybových stavů. Ve výchozím nastavení je SDA na pinu PTB4, SCL na pinu PTB5 (obr. 3.1).

Více o fungování protokolu na tomto mikrokontroléru se dozvíme níže při popisu funkcí knihovny pro podporu komunikace I^2C.

3.4.1 Knihovna i2c.h

Knihovna pro implementaci komunikačního protokolu I^2C. Základní kód knihovny byl převzat z návodu k práci s periferiemi mikrokontrolérů rodiny HCS08 [15], který byl následně mírně upraven a dopracován.

Proměnné knihovny potřebné k implementaci protokolu jsou následující:

- **I2C_STEP** – příznaková proměnná označující aktuální krok komunikace, může nabývat následujících hodnot:
  - **IIC_ERROR_STATUS** (0) – chybový stav
  - **IIC_READY_STATUS** (1) – stav připravenosti k akci
  - **IIC_HEADER_SENT_STATUS** (2) – stav odeslání adresy sběrnice s bitem R/$\overline{\text{W}}$
  - **IIC_DATA_TRANSMISION_STATUS** (3) – stav přenosu dat
  - **IIC_DATA_SENT_STATUS** (4) – stav dokončeného přenosu dat
- **I2C_DATA_DIRECTION** – příznaková proměnná určující směr přenosu dat: 1 – vysílání, 0 – příjem
- **I2C_LENGTH** – počet bajtů pro vysílání/příjem.
- **I2C_COUNTER** – čítač
- **I2C_DATA[8]** – pole bajtů pro vysílání/příjem

Dále si popíšeme funkce knihovny.

I2C_Init

Inicializace modulu pro práci s I^2C. Hlavní inicializační kroky jsou následující:

1. Zapnutí I^2C.
2. Nastavení děliče frekvence mikrokontroléru. Rychlost sběrnice se vypočítá podle následujícího vzorce [5, str. 228, Eqn. 12-1]:

$$f_{\text{I2C}} = \frac{f_{\text{Bus}}}{\text{mul} \cdot \text{SCLdivider}} \tag{3.3}$$

Hodnota děliče (*mul* a *SCLdivider*) se předává funkci jako parametr.

3. Zapnutí přerušení I²C.

```
void I2C_Init(unsigned char div)
{
    IICC1_IICEN = 1;  /* Zapnutí I2C */
    IICF = div;       /* Nastavení frekvence */
    I2C_STEP = IIC_READY_STATUS;
    IICC1_IICIE = 1;  /* Zapnutí přerušení */
}
```

Výpis 3.10: Funkce I2C_Init

WriteBytesI2C

Odeslání předaného počtu bajtů do zařízení s předanou adresou. Funkce odešle na periferii předanou adresu zařízení s bitem $R/\overline{W} = 0$, nastaví režim vysílání a zahájí přenos dat nastavením režimu master. Po prvním odeslání adresy zařízení s bitem $R/\overline{W}$ je přijat požadavek na přerušení, který je dále zpracováván příslušnou rutinou přerušení, o níž si povíme později.

```
void WriteBytesI2C(unsigned char slaveAddress, unsigned char
   numberOfBytes)
{
  I2C_LENGTH = numberOfBytes;
  I2C_COUNTER = 0;
  I2C_STEP = IIC_HEADER_SENT_STATUS;
  I2C_DATA_DIRECTION = 1;  /* Vysílání dat */

  /* Formátování adresy pro přidání bitů R/W = 0 */
  slaveAddress = slaveAddress << 1;
  slaveAddress &= 0xFE;

  IICC1_TX = 1;  /* Výběr režimu vysílání */
  IICC1_MST = 1; /* Zvolit režim Master (Start condition) */

  /* Zápis adresy do datového registru I2C */
  IICD = slaveAddress;
}
```

Výpis 3.11: Funkce WriteBytesI2C

ReadBytesI2C

Příjem předaného počtu bajtů ze zařízení s předanou adresou. Funkce je podobná jako u předchozího postupu, pouze při přenosu předané adresy s bitem R/$\overline{\text{W}}$ je tento bit nastaven na 1 a proměnná *I2C_DATA_DIRECTION* je nastavena na 0, což znamená příjem dat. Ostatní akce jsou rovněž řešeny rutinou přerušení.

Přerušení isrIIC

Přerušovací rutina pro zpracování všech akcí souvisejících s periferií I^{2}C. K rutině přerušení se dostanete zápisem do datového registru I^{2}C (IICD), což se děje v předchozích dvou funkcích. Zbytek akcí se pak provádí přímo v rutině.

Kód rutiny přerušení má 4 části: zpracování ACK, potvrzení odeslání adresy s bitem R/$\overline{\text{W}}$ (Header), akce při odesílání/příjmu dat a ukončení komunikace. Níže popíšeme každou sekci podrobněji.

Zpracování ACK. Zde se potvrdí komunikace s podřízeným zařízením. Pokud je přijat bit ACK/NACK, komunikace pokračuje až do logického ukončení. Pokud není přijat žádný bit, komunikace se zastaví a zapíše se příznak chyby.

```
/* Ukončení komunikace, pokud nebyl přijat žádný ACK/NACK */
if(IICS_RXAK == 1){
    IICC1_MST = 0;
    I2C_STEP = IIC_ERROR_STATUS;
    return;
}
```

Výpis 3.12: Zpracování ACK (přerušení isrIIC)

Potvrzení o odeslání Header. Po odeslání adresy zařízení s bitem R/$\overline{\text{W}}$ se nastaví následný směr komunikace.

```
if(I2C_STEP == IIC_HEADER_SENT_STATUS){
    IICC1_TX = I2C_DATA_DIRECTION;
    I2C_STEP = IIC_DATA_TRANSMISION_STATUS;
    if(IICC1_TX == 0){ // Pokud je režim příjmu dat, vymazání
        datového registru pro následný příjem dat
        Temp = IICD;
        return;
    }
}
```

Výpis 3.13: Potvrzení o odeslání Header (přerušení isrIIC)

Vysílání/příjem dat. Na základě aktuálního stavu směru přenosu dat se do datového registru načítají/zapisují bajty z/do datového pole. Při čtení dat je v případě potřeby odeslán také bit ACK/NACK.

```
if(I2C_STEP == IIC_DATA_TRANSMISION_STATUS){
    if(IICC1_TX == 1){ /* Pokud odesíláme data */
      IICD = I2C_DATA[I2C_COUNTER]; /* Odeslat bajt */
      I2C_COUNTER++;
      if(I2C_LENGTH <= I2C_COUNTER){
        /* Všechna data byla odeslána */
        I2C_STEP = IIC_DATA_SENT_STATUS;
      }
      return;
    }
    else{ /* Pokud přijímáme data */
      if((I2C_COUNTER+1) == I2C_LENGTH){
        /* Odeslání NACK na znamení ukončení čtení */
        IICC1_TXAK = 1;
      }
      I2C_DATA[I2C_COUNTER] = IICD; /* Přečíst bajt */
      I2C_COUNTER++;
      if(I2C_LENGTH <= I2C_COUNTER){
        /* Všechna data byla přijata */
        I2C_STEP = IIC_DATA_SENT_STATUS;
      }
      return;
    }
}
```

Výpis 3.14: Vysílání/příjem dat (přerušení isrIIC)

Ukončení komunikace. To se provádí vypnutím režimu Master (Stop condition).

```
if(I2C_STEP == IIC_DATA_SENT_STATUS){
  I2C_STEP = IIC_READY_STATUS;
  IICS_IICIF = 1;
  IICC1_MST = 0; /* Stop condition */
  return;
}
```

Výpis 3.15: Ukončení komunikace (přerušení isrIIC)

3.4.2 Knihovna ds3231lib.h

Knihovna pro komunikaci s RTC modulem DS3231. Při práci s touto knihovnou je nutné připojení knihovny i2c.h.

Podle manuálu [8] je adresa zařízení na lince I^2C **0x68**. Adresy pro ukládání času a jejich obsah jsou uvedeny v tabulce 3.1.

Funkce registru	Adresa (Hex)	Data							
		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Sekundy	0x00	0	vyšší řád			nižší řád			
Minuty	0x01	0	vyšší řád			nižší řád			
Hodiny	0x02	0	0	vyšší řád		nižší řád			
Den v týdnu	0x03	0	0	0	0	0	Číslo		
Den	0x04	0	0	vyšší řád		nižší řád			
Měsíc	0x05	Století	0	0	vyšší ř.	nižší řád			
Rok	0x06	vyšší řád				nižší řád			

Tab. 3.1: Obsah časových registrů DS3231 [16]

Poznámka k tabulce 3.1:

- V registrech jsou uložena desetinná čísla ve formátu vyššího a nižšího řádu.
- Obsah hodinového registru je aktuální při konfiguraci 24-hodinového časového formátu.
- Registr dne v týdnu obsahuje číslo od 1 do 7 (kde 1 je pondělí, 7 je neděle).

readRTC

Čtení hodnot sekund, minut, hodin, dne v týdnu, dne, měsíce, roku z modulu. Princip načítání dat z modulu [16]:

1. Odeslání adresy zařízení s bitem $R/\overline{W} = 0$.
2. Odeslání adresy registru, který má být přečten.
3. Odeslání signálu Repeated Start (nebo signálu Stop následovaného signálem Start).
4. Odeslání adresy zařízení s bitem $R/\overline{W} = 1$.
5. Příjem obsahu registrů, v tomto případě se bude jednat o sekvenční čtení registrů.

Časové registry, které potřebujeme, jsou v rozsahu 0x00-0x06 (tab. 3.1), takže na základě algoritmu čtení dat stačí jednou odeslat adresu sekundového registru (0x00) a poté 7-krát přijmout data. Po přijetí příslušných bajtů je třeba je zpracovat a

získat časové údaje v desítkové soustavě. Zpracované hodnoty se zapíší do datové struktury (výp. 3.16), kterou pak vrátí funkce.

```
typedef struct time{
    unsigned char seconds;
    unsigned char minutes;
    unsigned char hours;
    unsigned char dayOfWeek;
    unsigned char day;
    unsigned char month;
    unsigned char year;
} time;
```

Výpis 3.16: Struktura time

```
time readRTC(void){
    time now;

    I2C_DATA[0] = 0x00; // Počáteční adresa
    WriteBytesI2C(DS3231_ADDRESS, 1);
    ReadBytesI2C(DS3231_ADDRESS, 7);

    /* Zpracování dat */
    now.seconds =
      (((I2C_DATA[0]>>4)&0x0F)*10)+(I2C_DATA[0]&0x0F);
    now.minutes =
      (((I2C_DATA[1]>>4)&0x0F)*10)+(I2C_DATA[1]&0x0F);
    now.hours =
      (((I2C_DATA[2]>>4)&0x0F)*10)+(I2C_DATA[2]&0x0F);
    now.dayOfWeek = I2C_DATA[3];
    now.day =
      (((I2C_DATA[4]>>4)&0x0F)*10)+(I2C_DATA[4]&0x0F);
    now.month =
      (((I2C_DATA[5]>>4)&0x0F)*10)+(I2C_DATA[5]&0x0F);
    now.year =
      (((I2C_DATA[6]>>4)&0x0F)*10)+(I2C_DATA[6]&0x0F);

    return now;
}
```

Výpis 3.17: Funkce readRTC

setRTC

Nastavení času modulu. Odesílají se hodnoty sekund, minut, hodin, dne v týdnu, dne, měsíce, roku. Princip je následující:

1. Odeslání adresy zařízení s bitem $\mathrm{R}/\overline{\mathrm{W}} = 0$.
2. Odeslání adresy registru, do kterého se má zapisovat.
3. Odeslání dat pro daný registr.
4. Odeslání dalších dat, která se mají zapsat do dalšího registru v pořadí.

Zápis probíhá na dříve uvedených adresách registrů ve stejném algoritmu. Časové údaje pro zápis jsou funkci předávány strukturou (výp. 3.16), před odesláním musí být data zpracována, aby byla uvedena do požadované podoby.

```
void setRTC(time now)
{
    /* Kontrola a oprava dat */
    if (now.seconds > 59)    now.seconds = 59;
    if (now.minutes > 59)    now.minutes = 59;
    if (now.hours > 23)      now.hours = 23;
    if (now.dayOfWeek < 1)   now.dayOfWeek = 1;
    if (now.dayOfWeek > 7)   now.dayOfWeek = 7;
    if (now.day < 1)         now.day = 1;
    if (now.day > 31)        now.day = 31;
    if (now.month < 1)       now.month = 1;
    if (now.month > 12)      now.month = 12;
    if (now.year > 99)       now.year = 99;

    /* Zpracování dat */
    I2C_DATA[0] = 0x00; // Počáteční adresa
    I2C_DATA[1] = ((now.seconds/10)<<4)|(now.seconds%10);
    I2C_DATA[2] = ((now.minutes/10)<<4)|(now.minutes%10);
    I2C_DATA[3] = ((now.hours/10)<<4)|(now.hours%10);
    I2C_DATA[4] = now.dayOfWeek;
    I2C_DATA[5] = ((now.day/10)<<4)|(now.day%10);
    I2C_DATA[6] = ((now.month/10)<<4)|(now.month%10);
    I2C_DATA[7] = ((now.year/10)<<4)|(now.year%10);

    WriteBytesI2C(DS3231_ADDRESS, 8);
}
```

Výpis 3.18: Funkce setRTC

3.4.3 Knihovna mcp9808lib.h

Knihovna pro komunikaci se snímačem teploty MCP9808. Při práci s touto knihovnou je nutné připojit knihovnu i2c.h.

Podle manuálu výrobce [12] je adresa zařízení na lince I^{2}C **0x18**. Řídicí registr a registr okolní teploty, se kterými budeme pracovat, jsou na adresách 0x01 a 0x05.

T_A vs. T_C	T_A vs. T_U	T_A vs. T_L	Znak	2^7 °C	2^6 °C	2^5 °C	2^4 °C
Bit 15							Bit 8
2^3 °C	2^2 °C	2^1 °C	2^0 °C	2^{-1} °C	2^{-2} °C	2^{-3} °C	2^{-4} °C
Bit 7							Bit 0

Tab. 3.2: Obsah registru okolní teploty MCP9808 (0x05) [12, REGISTER 5-4]

Poznámka k tabulce 3.1:

- T_A – aktuální okolní teplota, T_L, T_U, T_C – dolní mez, horní mez a kritická hodnota teploty zapsaná v snímači (detaily ukládání viz manuál [12]). Bity 15, 14 a 13 se v rámci naší práce s modulem nepoužívají.
- Znak – znaménko teploty: 0 – teplota rovná nebo vyšší než nula, 1 – nižší než nula.

Dále jsou popsány funkce pro práci s modulem.

getTemp

Příjem údajů o teplotě z modulu. Podobně jako u modulu DS3231 (viz výše) se nejprve modulu předá adresa registru a poté se z něj získají data. Z příslušného teplotního registru (0x05) jsou získány dva bajty – *UpperByte* a *LowerByte*. Hodnota teploty je vrácena prostřednictvím struktury (výp. 3.19), která má celočíselnou a zlomkovou část hodnoty teploty. Celočíselná hodnota teploty může být kladná nebo záporná.

```
typedef struct temp{
    signed char integer;
    unsigned char fractional;
} temp;
```

Výpis 3.19: Struktura temp

```
temp getTemp(void)
{
  unsigned short raw;
  unsigned long frac;
  temp result;
```

```
  I2C_DATA[0] = 0x05;    // Registr teploty
  WriteBytesI2C(MCP9808_ADDRESS, 1);
  ReadBytesI2C(MCP9808_ADDRESS, 2);

  /* Zpracování dat */
  raw = ((unsigned short)I2C_DATA[0]<<8) | I2C_DATA[1];

  /* Celočíselná část teploty */
  result.integer = (signed char)(raw >> 4);
  /* Zlomková část teploty */
  frac = raw & 0x0F;
  result.fractional = (unsigned char)((frac * 625) / 100);
  /* Korekce zlomkové části při záporné teplotě */
  if ((I2C_DATA[0] & 0x10) == 0x10)
  {
    if (result.fractional > 0)
      result.fractional = 100 - result.fractional;
    else
      result.integer--;
  }

  return result;
}
```

Výpis 3.20: Funkce getTemp

V této funkci může zpracování přijatých bajtů (zde I2C_DATA[0] je *UpperByte*, I2C_DATA[1] je *LowerByte*) vypadat jinak než ve vzorci (2.2), ale podstata je stejná. Pro získání celočíselné části teploty provedeme 4 bitové posuny doprava, čímž se zbavíme zlomkové části a získáme hned celočíselnou část se znaménkem. Pro získání zlomkové části je třeba vynásobit poslední 4 bity přijatých dat nastavenou přesností, v našem případě 0,0625 (defaultní hodnota). Abychom se vyhnuli použití zlomkových čísel (mikrokontrolér nemá hardwarovou podporu pro čísla s plovoucí desetinnou čárkou), vynásobíme číslo nejprve číslem 625 a poté vydělíme číslem 100, což je ekvivalentní násobení číslem 0,0625 (protože se jedná o zlomkovou část).

shutdownMode

Funkce pro volbu provozního režimu modulu (SHDN). Tento snímač může pracovat ve dvou režimech: normální režim sériového měření (0) a režim úspory energie (1). První režim je defaultní provozní režim s normální spotřebou proudu a provozem

všech procesů. V režimu úspory jsou všechny procesy spotřebovávající energii (včetně měření teploty) vypnuty a aktivní zůstává pouze komunikační rozhraní.

Řídicí registr (0x01) má velikost 2 bajty, volba režimu SHDN je 8 bit. Proto se při přenosu hodnoty do registru přenášejí 2 bajty, první bajt (*UpperByte*) obsahuje přenášený režim (0 nebo 1). Předávaná hodnota *mode* může být 0 (normální režim) nebo 1 (úsporný režim).

```
void shutdownMode(unsigned char mode)
{
  if (mode > 1)
  return;

  I2C_DATA[0] = 0x01;   // Řídicí registr
  WriteBytesI2C(MCP9808_ADDRESS, 1);
    ReadBytesI2C(MCP9808_ADDRESS, 2);

  /* Zpracování UpperByte (I2C_DATA[0]),
   * LowerByte (I2C_DATA[1]) zůstane stejný */
  I2C_DATA[0] = (mode == 0x00) ? (I2C_DATA[0] & 0xFE) :
  (I2C_DATA[0] | 0x01);

  WriteBytesI2C(MCP9808_ADDRESS, 2);
}
```

Výpis 3.21: Funcke shutdownMode

3.5 Hlavní soubory

3.5.1 main.c

Tento hlavní soubor obsahuje počáteční inicializaci modulů, přerušení SCI, KBI (přerušení tlačítek) a TOD (časovač) a kruhovou smyčku, která kontroluje příznaky pro zpracování příkazu zadaného prostřednictvím konzoly a požadavku na kontrolu hodnoty teploty.

Příznaky (tj. proměnné tohoto souboru) jsou následující:

- **PROCESS_READY** – příznak připravenosti ke zpracování příkazu zadaného prostřednictvím konzoly, tento příznak má hodnotu 1, když je příkaz konzoly sestavený (odesláním Enteru).
- **CHECK_REQ** – příznak požadavku na kontrolu hodnoty teploty, tento příznak se každou minutu nastaví na hodnotu 1 pomocí přerušení časovače (TOD) nebo stisknutím tlačítka SW2 (přerušení KBI).

Funkce main inicializuje moduly:

- **MCUinit** – inicializace procesoru a periferií (více informací v části Knihovna mcu.h).
- **FlashInit** – inicializace modulu pro práci s pamětí FLASH (viz sekce Práce s pamětí FLASH), funkci je předána hodnota 20 – dělič frekvence procesoru (bus clock), která je $\approx 4{,}2\,\text{MHz}$ (3.2):

$$\text{DIV} + 1 = \frac{f_{\text{Bus}}}{f_{\text{FCLK}}} = \frac{4,2 \cdot 10^6}{200 \cdot 10^3} = 21 \rightarrow \text{DIV} = 20$$

- **SCI_Init** – inicializace SCI modulu (viz část Knihovna SCI.h), funkci jsou předány hodnoty 0 (SCI1BDH) a 0x1B (SCI1BDL) – horní a dolní bajt děliče frekvence procesoru pro získání požadované přenosové rychlosti, v našem případě **9600 Bd**. Pro výpočet předaných hodnot byl použit vzorec (3.1):

$$[\text{SBR12:SBR0}] = \frac{f_{\text{Bus}}}{\text{Baud Rate} \cdot 16} = \frac{4,2 \cdot 10^6}{9600 \cdot 16} \approx 27 = \text{0x1B}$$

- **I2C_Init** – inicializace modulu I^2C (viz část Knihovna i2c.h), do funkce se předává hodnota 0x07 – dělič frekvence procesoru pro dosažení požadované rychlosti komunikace: požadovaná rychlost pro přenos dat je **100 kHz**, takže pro dosažení požadované rychlosti musí být hodnota *SCLdivider* rovna (*mul* je 1) (3.3):

$$\text{mul} \cdot \text{SCLdivider} = \frac{f_{\text{Bus}}}{f_{\text{I2C}}} = \frac{4,2 \cdot 10^6}{100 \cdot 10^3} = 42$$

  Podle tabulky [5, str. 229, Table 12-5] odpovídá přibližná hodnota *SCLdivider* hodnotě ICR = 0x07, tato hodnota je předána funkci.
- **lcd_init_clock**, **lcd_init** – inicializace LCD displeje (podrobný popis najdete v [13]).

Dále se provedou funkce **dataCheck** a **checkTemp** (budou popsány v části Terminal.c) – počáteční kontrola přítomnosti dat v paměti a kontrola teploty.

V nekonečné smyčce se kontrolují hodnoty výše uvedených příznaků, a pokud jsou nastaveny na 1, provedou se příslušné funkce.

```
for(;;) {
  if (PROCESS_READY)
  { // Pokud je požadavek na zpracování příkazu
    ProcessCommand();
    PROCESS_READY = 0;
  }
  if (CHECK_REQ)
  { // Pokud je požadavek na měření teploty
    checkTemp();
```

```
    CHECK_REQ = 0;
  }

  __RESET_WATCHDOG();
}
```

Výpis 3.22: Nekonečná smyčka (main.c)

Knihovna mcu.h

Knihovna konfigurace mikrokontroléru (frekvence procesoru), časovače (TOD), tlačítek a LED, modulu TPM (Timer Pulse-Width Modulator). V této části budou stručně popsány hlavní funkce knihovny.

MCUinit. Nastavení frekvence procesoru (bus clock) na $\approx 4{,}2$ MHz, modul TOD s přerušením jednou za 1 minutu, přerušení tlačítka (KBI) a modul TPM pro dva kanály: první kanál bude sloužit k přehrávání určité frekvence na bzučáku, který je instalován na desce, druhý kanál bude sloužit jako časovač zpoždění, zpoždění je realizováno prostřednictvím přerušení modulu TPM2, což umožňuje paralelní použití bzučáku nebo zpoždění akcí bez pozastavení hlavního programu.

MCUreset. Funkce resetování mikrokontroléru "nenakrmením watchdogu", používá se pro příkaz reset (viz sekce Komunikace s uživatelem, přenos dát do počítače).

pwmTone. Funkce pro aktivaci bzučáku s předanou frekvencí.

notePlay. Funkce pro přehrávání předané frekvence s předanou dobou trvání.

Přerušení isrTPM2. Přerušení modulu TPM2, které umožňuje zpoždění bez pozastavení hlavního programu a přehrávání melodií pomocí bzučáku.

Podrobnější informace o tom, jak všechny funkce této knihovny fungují, naleznete v projektu zařízení v příloze.

3.5.2 Terminal.c

Tento soubor obsahuje všechny hlavní funkce pro provoz zařízení.

Významné proměnné:

- **rxBuffer** – pole bajtů přijatých přes SCI
- **addressPnt** – ukazatel na adresu v paměti FLASH pro zápis pole dat
- **addressCache** – dodatečný ukazatel na adresu v paměti FLASH, v případě přetečení paměti ukazuje na začátek úseku paměti s daty

Dále si stručně projdeme nejdůležitější funkce souboru.

ProcessInput

Tato funkce pomocí přerušení (výp. 3.23) umístěného v souboru main.c zpracovává bajty přicházející přes modul SCI a ukládá je do pole *rxBuffer* pro další vytváření příkazu a jeho zpracování.

```
interrupt VectorNumber_Vsci1rx void isrSCI(void)
{
    PROCESS_READY = ProcessInput();
    SCI1S1_RDRF = 0;
}
```

Výpis 3.23: Přerušení isrSCI

```
uint8_t ProcessInput(void)
{
  uint8_t ReceivedByte;
  ReceivedByte = ReadByte(); // Přečíst bajt

  switch (ReceivedByte) // Zpracování přijatého bajtu
  {
  case '\r': // Enter symbol
    /* Dokončit pole a vrátit 1 */
    rxBuffer[rxIndex] = '\0';
    rxIndex = 0;
    return 1;
  case 0x7f: // Backspace symbol
    /* Odstranění posledního pruku z pole */
    if (rxIndex != 0)
      rxBuffer[--rxIndex] = '\0';
    SendByte(ReceivedByte);
    return 0;
  case 0x1b: // Escape symbol
    /* Nic se neděje */
    return 0;
  default:
    /* Zápis přijatého bajtu do pole */
    if (rxIndex < MAX_BUFFER_SIZE - 1)
    {
      rxBuffer[rxIndex++] = ReceivedByte;
      SendByte(ReceivedByte);
    }
    return 0;
```

```
  }
}
```

Výpis 3.24: Funkce ProcessInput

Když přijde znak, zapíše se do pole *rxBuffer*, kromě znaku Enter, který signalizuje konec příkazu a předá pole funkci obsluhy příkazu, znaku Backspace, který odstraní poslední bajt z pole, a znaku Escape, po jejím přijetí se nic neděje.

ProcessCommand

Tato funkce porovná dokončený *rxBuffer* jako řetězec s řetězci příkazů, a pokud je shoda, zahájí provádění funkce vázané na tento příkaz.

```
if (strcmp(rxBuffer, "check") == 0)
{
  checkTemp();
  return;
}
```

Výpis 3.25: Část funkce ProcessCommand

checkTemp

Funkce pro komunikaci s modulem reálného času a teplotním snímačem pro získání a kontrolu hodnoty teploty.

Pokud přijatá hodnota teploty překročí nastavené limity, provedou se následující akce: odeslání zprávy uživateli prostřednictvím konzoly s upozorněním na překročení limitu, zobrazení hodnoty teploty a příslušné ikony na displeji LCD (pod dolním limitem – sněhová vločka, nad horním limitem – plamen), zápis časových a teplotních údajů do paměti (funkce writeData) a zvukový signál. V opačném případě, pokud je teplota v mezích, se hodnota teploty na LCD displeji aktualizuje a uživateli se prostřednictvím konzoly zobrazí zpráva o normálním stavu.

```
void checkTemp(void)
{
  /* Načtení dat z modulů */
  temperature = getTemp(0);
  time = readRTC();

  /* ..Výpis dat o čase a teplotě do konzoly.. */

  /* Zpracování teploty */
  if (temperature.integer < MIN_TEMP)
```

```
    {
        SendString("Overcooling!\r\n");
        LCD_SEG_BLINK_COLD;
        LCD_SEG_OFF_HEAT;
        writeData(&time, &temperature);
        soundPlay(Freq_low, TDur, 2);
    }
  /* ..Další zpracování.. */

  /* Zobrazení teploty na displeji */
  lcd_show_small_int16_dec((int16_t)temperature.integer);
}
```

Výpis 3.26: Funkce checkTemp

writeData

Jak již je popsáno v sekce Práce s pamětí FLASH, do paměti se zapíše 8 bajtů dat:

Sekundy	Minuty	Hodiny	Den	Měsíc	Rok	Teplota	
1 bajt	2 bajt	3 bajt	4 bajt	5 bajt	6 bajt	7 bajt	8 bajt

Tab. 3.3: Datový řetězec pro zápis

Algoritmus je následující:

1. Vytvoření a zápis do pole časových a teplotních údajů předaných funkci.
2. Zápis pole do paměti na adresu v proměnné *addressCnt*. Data se zapisují pomocí funkce FlashBurst (viz FlashBurst v části Knihovna flash.h).
3. Zvýšení ukazatele a příprava paměti pro další zápis: pokud *addressCnt* ukazuje na další stránku v paměti, tak se stránka připraví (kontrola prázdnosti stránky, pokud není prázdná – vymazání stránky), v případě přetečení paměti se ukazatel *addressCnt* vrátí na začátek paměti (na začátek první stránky, první stránka se připraví, data se vymažou), začátek další stránky se zapíše do *addressCache* (obsahuje data). V tomto případě, pokud dojde k přetečení první stránky, bude ukazatel *addressCache* přesunut na další stránku, což nám zajistí kruhový záznam dat a přístup ke všem zaznamenaným datům. Při jakékoli přípravě stránky bude na konzolu odesláno oznámení.
4. Oznámení prostřednictvím konzoly o nedostatku volného místa v paměti, pokud je dosaženo poslední (čtvrté) volné stránky pro zápis nebo pokud hodnota *addressCache* není nulová. V takovém případě začne na desce blikat LED2 (obr. 3.6), což znamená, že paměť je plná.

Při tomto algoritmu práce s pamětí do ní lze uložit maximálně 255 polí dat monitorování. Pokud je tedy zapsáno 255 polí a přijde požadavek na zápis, zapíše se další pole a ukazatel se přesune na začátek paměti, čímž se vymaže první stránka (vymaže se 64 datových polí), v paměti pak zůstane 192 polí.

getData

Funkce výpisu přijatých dat ve formě tabulky do konzoly (obr. 3.4).

Nejprve se kontroluje, zda proměnná *addressCache* není nulová, v případě, že není – vypisují se pole od adresy v této proměnné až do konce dostupné paměti, dále pokud je adresa v proměnné *addressCnt* větší než adresa začátku první stránky – vypisují se hodnoty od začátku dostupné paměti až po adresu v proměnné *addressCnt*. Pokud je proměnná *addressCache* nulová, začíná výpis okamžitě ukazatelem *addressCnt*.

Tento algoritmus zajišťuje, že všechna data jsou načtena z paměti při zachování časové posloupnosti – od starých po nové.

dataCheck

Tato funkce se provede jednou při spuštění zařízení. Jejím účelem je zkontrolovat přítomnost dat v paměti FLASH. Algoritmus projde celou dostupnou paměť, a pokud jsou v ní pole dat, zapíše adresy do proměnných *addressCnt* a *addressCache*:

1. Procházení paměti (čtení paměťových buněk pomocí funkce FlashRead, viz sekce Knihovna flash.h) a kontrola první prázdné adresy, pokud je taková adresa nalezena – zápis do proměnné *addressCnt*.
2. Procházení paměti až na konec čtvrté stránky v případě, že hodnota *addressCnt* je menší než adresa začátku čtvrté stránky, a kontrola první neprázdné adresy. V tomto případě se kontrola provádí od adresy v proměnné *addressCnt*.

```
for (i = FIRST_PAGE; i < FIFTH_PAGE; i += ARRAY_SIZE)
{
  if (FlashRead(i) == 0xFF) // Pokud je buňka prázdná
  {
    addressPnt = i;
    break;
  }
}
if (addressPnt < FOURTH_PAGE)
{
  for (i = addressPnt; i <= FOURTH_PAGE; i+= ARRAY_SIZE)
  {
    if (FlashRead(i) != 0xFF) // Pokud buňka není prázdná
```

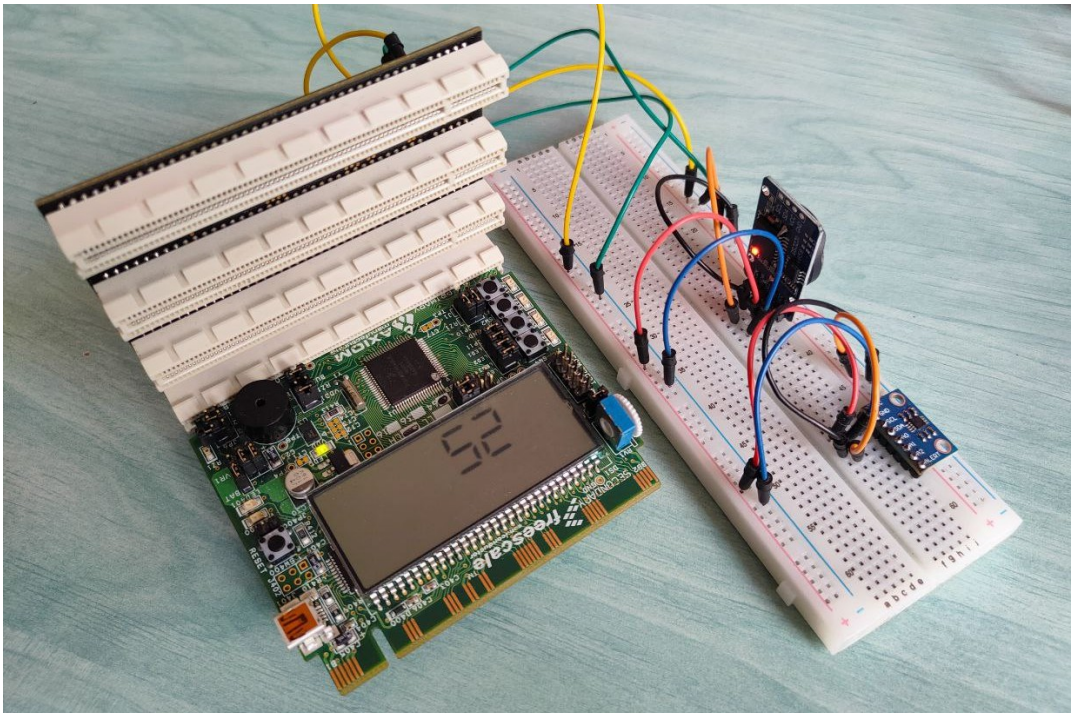
```
    {
      addressCache = i;
      break;
    }
  }
}
```

Výpis 3.27: Kontrola přítomnost dat (funkce dataCheck)

3. Oznámení konzole, že data jsou k dispozici, pokud byla zapsána alespoň jedna ze dvou proměnných.
4. Oznámení přes konzoli o nedostatku volného místa v paměti, pokud je dosaženo poslední (čtvrté) volné stránky pro zápis nebo pokud hodnota *addressCache* není nulová. V takovém případě začne na desce blikat LED2 (obr. 3.6), což znamená, že paměť je plná.

Testování

Nakonec bylo sestaveno samotné zařízení pro monitorování teploty (obr. 4.1), nahrán software a otestována jeho funkčnost.



Obr. 4.1: Zařízení pro monitorování teploty

Prvním a jednorázovým krokem je počáteční nastavení modulu reálného času – nahrání aktuálních časových údajů. To bylo provedeno odesláním příkazu pro nastavení času s časovými parametry (viz příkaz *time set* v sekce Komunikace s uživatelem, přenos dát do počítače). Po tomto nastavení je zařízení plně připraveno k monitorování a neztratí časové údaje ani v případě výpadku napětí díky baterii na modulu RTC.

Při testování s napájením z baterie byla na držáku desky místo požadované CR2325 použita podobná baterie CR2354 tabletového typu, která má stejné jmenovité napětí 3 V, ale větší kapacitu 560 mAh [17] (CR2325 má 210 mAh [14]). Baterie CR2354 byla vybrána z důvodu její vyšší maloobchodní dostupnosti (jinými slovy, je snadněji k sehnání v obchodech). Neodpovídá však zcela rozměrům držáku na desce.

Bohužel zařízení nebylo testováno ve skutečné serverovně, pro kterou je určeno, a to z několika důvodů: k takové místnosti nebyl přístup a vyhodnocení fungování zařízení v reálném prostředí serverovny by bylo časově náročné. Účinnějším způsobem

testování bylo umělé zvýšení teploty v blízkosti snímače nebo změna monitorovacích limitů (snížení maximální hranice pro spuštění záznamu, obr. 4.2) pro posouzení správnosti komunikace se snímačem teploty, modulem RTC a paměťovým modulem mikrokontroléru (záznam hodnot). Zařízení byla také testována a kalibrována při teplotách pod bodem mrazu.

Obr. 4.2: Testování zobrazení oznámení překročení teploty

Během testování a drobných softwarových oprav bylo zařízení zkalibrováno a vyladěno. Zařízení plní svůj hlavní účel, kterým je monitorování a záznam dat.

Aktuální konfigurace zařízení má však závažnou nevýhodu – velmi nízkou provozní dobu v případě napájení z baterie. Důvodem je použití baterie tabletového typu, v našem případě CR2354. Během testování bylo zjištěno, že napětí baterie časem klesne na kriticky nízkou úroveň, která nestačí k napájení modulů a která nakonec donutí mikrokontrolér přejít do cyklického resetu. Po vypnutí napájení zařízení se však baterie po určité době obnoví na svou jmenovitou hodnotu a po opětovném zapnutí napájení zařízení pokračuje v činnosti.

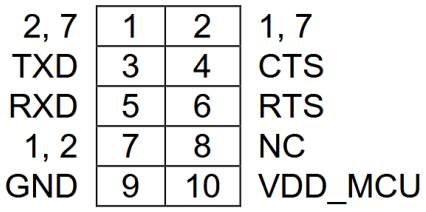
Toto chování lze zdůvodnit přetížením napájecího zdroje a efektem obnovení náboje [18], který je charakteristický pro lithiové baterie. Jmenovitý provozní proud baterie CR2354 je 0,2 mA [17], naměřený proud na výstupu 3V3 pro napájení modulů (obr. 3.1) při naměřeném napětí baterie 2,9 V je přibližně 1,3 mA, což více než 6-krát překračuje jmenovitý proud. Tento rozdíl způsobí, že napětí baterie klesne během přibližně hodiny a půl z 2,9 V na 2,4 V, což nestačí k napájení modulů a zařízení přestane správně fungovat.

4.1 Možné úpravy

Následující návrhy na vylepšení zařízení nebyly v rámci cílů této práce a jsou pouze doplněním základního konceptu zařízení, které z těch či oněch důvodů nebyly realizovány.

Jiný zdroj napájení

Na základě problému nízké doby provozu zařízení při napájení z baterie by zřejmým zlepšením bylo nahradit současný napájecí zdroj výkonnějším zdrojem (jmenovitý proud alespoň 2 mA) a případně zdrojem s vyšší kapacitou, aby se prodloužila doba provozu zařízení. Nový napájecí zdroj nemusí být připojen přímo k zadní straně desky, lze jej připojit buď k napájecím pinům na desce TWR-S08 (obr. 4.3, piny GND a VDD_MCU), nebo k hlavní napájecí lince modulů (obr. 3.1, vodiče 3V3 a GND), která bude současně napájet moduly i mikrokontrolér.



Obr. 4.3: Konektor PCI Express [3, Figure 5]

Jako nový zdroj napájení lze použít dvě baterie AA (jmenovité napětí jedné je 1,5 V) zapojené do série. Toto řešení zajistí výstupní napětí 3 V a vyhovující proud – ačkoli baterie tohoto typu nemají jednotný standard jmenovitého proudu, obvykle jsou určeny pro provoz do 50-200 mA nebo více (v závislosti na typu baterie), což je pro napájení zařízení dostačující. Při výběru baterie AA je také třeba věnovat pozornost jejímu rozsahu provozních teplot, aby odpovídal provoznímu rozsahu zařízení (-20°C až 40°C), protože tento rozsah se u výrobců liší. Příkladem vhodného modelu je baterie Panasonic Evolta AA (obr. 4.4) [19].

Obr. 4.4: Panasonic Evolta AA [19]

Režim úspory energie

Jednou z možností, jak vyřešit problém s nízkou dobou provozu z baterie tabletového typu nebo jak zvýšit dobu provozu výběrem jiného zdroje napájení (jak je naznačeno v předchozím bodě), je nastavit zařízení do úsporného režimu, který sníží spotřebu proudu zařízení, a tím udrží zařízení déle v provozu.

Náhrada modulu Primary Elevator

Pro připojení modulů jsme použili modul Primary Elevator, jak bylo popsáno dříve, který využívá pouze 1 ze 4 konektorů PCI Express a 4 ze $80 \cdot 2 = 160$ konektorů na zadní straně a který v praxi zbytečně zabírá mnoho místa. Proto není v rámci tohoto zařízení praktická. Místo tohoto modulu můžeme uvažovat o jiném způsobu připojení periferií k desce, např. prostřednictvím adaptérů PCI Express (obr. 4.5), který zabere mnohem méně místa a umožní nám realizovat následující bod.

Obr. 4.5: Konektor PCI Express [20]

Pouzdro pro zařízení

Významným doplňkem koncepce zařízení je vytvoření pouzdra, které skryje a ochrání všechny jeho vnitřní propojení, zdůrazní autonomii a mobilitu zařízení. Toto řešení také usnadní ovládání zařízení, např. přepínání zdroje napájení lze provést pomocí přepínače na pouzdře namísto ručního připojení jumperu.

Závěr

V této bakalářské práci byl prostudován problém monitorování teploty v serverovnách a také vývojová deska TWR-S08 od společnosti NXP Semiconductors s vestavěným mikrokontrolérem MC9S08LH64, modul reálného času DS3231, který byl zvolen jako kalendářový obvod pro zařízení, a externí teplotní snímač MCP9808.

Byla zpracována koncepce zařízení pro monitorování teploty, které je schopno zaznamenávat čas, kdy teplota překročí nastavené limity, ukládat tyto údaje do vnitřní nevolatilní paměti mikrokontroléru a zobrazovat informace na LCD displeji. Bylo zváženo a vybráno vhodnější napájení zařízení, které by vyhovovalo této koncepci.

Je realizována softwarová část zařízení zahrnující práci s pamětí FLASH, protokol přenosu dat I^2C pro komunikaci s moduly, periferie SCI pro komunikaci s počítačem a komunikaci s uživatelem jak prostřednictvím prvků na samotné vývojové desce (tlačítka, LED a LCD displej), tak prostřednictvím konzoly při připojení zařízení k počítači, která umožňuje nejen přijímat data monitorování, ale také nastavovat zařízení.

Během implementace softwarové části byly napsány knihovny pro práci s paměťovým modulem FLASH, I^2C a SCI, které lze využít v jiných projektech při práci s mikrokontrolérem rodiny HCS08.

Konečným výsledkem je zařízení pro monitorování teploty, které lze použít k monitorování serveroven i jiných místností nebo prostor. Zařízení je schopno monitorovat teploty v rozmezí -20°C až 40°C, zaznamenávat data, pokud teplota překročí nakonfigurované limity, a uchovávat tato data i při vypnutém napájení.

V průběhu testování zařízení se potvrdila jeho funkčnost. Byl zjištěn a popsán závažný nedostatek zařízení, bylo uvedeno jeho teoretické řešení a navrženy možnosti jeho zlepšení.

Vyvinuté zařízení je jednou z možností použití vývojových desek TWR-S08 poté, co byly nahrazeny jinými deskami, které již nebudou používány v průběhu výuky.

Literatura

[1] ASHRAE. Data Center Power Equipment Thermal Guidelines and Best Practices. Online. Dostupné z: https://www.ashrae.org/File%20Library/Technical%20Resources/Bookstore/ASHRAE_TC0909_Power_White_Paper_22_June_2016_REVISED.pdf. [cit. 2024-12-10].

[2] RIPPLEIT. Modernizing Your Small Business Server Room. Online. Dostupné z: https://www.rippleit.com/blog/how-to-setup-a-small-business-server-room. [cit. 2024-12-15].

[3] AXMAN. TWR-S08 User Guide. Online. Dostupné z: https://www.axman.com/sites/default/files/TWR-S08LL_LH_UG_0.pdf. [cit. 2024-12-09].

[4] NXP. Get to know the TWR-S08LL64. Online. Dostupné z: https://www.nxp.com/docs/en/supporting-information/TWRS098LL64LAB2.pdf. [cit. 2024-12-09].

[5] NXP. MC9S08LH64 Reference Manual. Online. Dostupné z: https://www.nxp.com/docs/en/reference-manual/MC9S08LH64RM.pdf. [cit. 2024-12-10].

[6] FARNELL. Freescale Tower System Development Board Platform. Online. Dostupné z: https://www.farnell.com/datasheets/1918683.pdf. [cit. 2024-12-09].

[7] LASKAKIT. RTC Hodiny reálného času DS3231 AT24C32. Online. Dostupné z: https://www.laskakit.cz/arduino-rtc-hodiny-realneho-casu-ds3231-at24c32/. [cit. 2024-12-10].

[8] ANALOG. DS3231 Datasheet. Online. Dostupné z: https://www.analog.com/media/en/technical-documentation/data-sheets/DS3231.pdf. [cit. 2024-12-09].

[9] NXP. TWR-ELEV Primary. Online. Dostupné z: https://www.nxp.com/downloads/en/schematics/TWR-ELEV-PRI-SCH.pdf. [cit. 2024-12-24].

[10] NXP. Temperature Sensor for the HCS08 Microcontroller Family. Online. Dostupné z: https://www.nxp.com/docs/en/application-note/AN3031.pdf. [cit. 2024-12-10].

[11] ALIBABA. MCP9808 Accuracy Temperature Sensor I2C Breakout Board Module 2.7V-5V Logic Voltage. Online. Dostupné z: https://www.alibaba.com/product-detail/MCP9808-Accuracy-Temperature-Sensor-I2C-Breakout_62382442140.html. [cit. 2025-05-03].

[12] MICROCHIP. MCP9808 ±0.5°C Maximum Accuracy Digital Temperature Sensor. Online. Dostupné z: https://ww1.microchip.com/downloads/en/DeviceDoc/25095A.pdf. [cit. 2025-05-04].

[13] ŠTIBRANÝ, Miroslav. Knihovna pro řízení LCD displeje GD-5360P. Online, Bakalářská práce, vedoucí Tomáš Macho. Brno: Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií. Ústav automatizace a měřicí techniky, 2014. Dostupné z: http://hdl.handle.net/11012/31327. [cit. 2025-05-24].

[14] MADE-IN-CHINA. UL Approved Li-Mno2 Cr2325 3V Coin Cell Battery Directly Sold by Factory. Online. Dostupné z: https://naccon.en.made-in-china.com/product/EsLJpjBMENch/China-UL-Approved-Li-Mno2-Cr2325-3V-Coin-Cell-Battery-Directly-Sold-by-Factory.html. [cit. 2025-05-19].

[15] NXP. HCS08 Peripheral Module Quick Reference. Online. Dostupné z: https://www.nxp.com/docs/en/user-guide/HCS08QRUG.pdf. [cit. 2025-05-17].

[16] IARDUINO. Sverkhtochnyy RTC modul' real'nogo vremeni na baze chipa DS3231. Online. Dostupné z: https://iarduino.ru/lib/412a3f4a96a8fb2cb6a995eb5d9cbc8e.pdf. [cit. 2025-05-17].

[17] RS-ONLINE. Manganese Dioxide Lithium Coin Batteries: Individual Specifications. Online. Dostupné z: https://docs.rs-online.com/0a96/0900766b8061c9cc.pdf. [cit. 2025-05-19].

[18] WIKIPEDIA. Recovery effect. Online. Dostupné z: https://en.wikipedia.org/wiki/Recovery_effect. [cit. 2025-05-19].

[19] LYGTE-INFO. Panasonic Evolta AA. Online. Dostupné z: https://lygte-info.dk/review/batteries2012/Panasonic%20Evolta%20AA%20UK.html. [cit. 2025-05-19].

[20] ELECBEE. PCIE Express Connector 164 Pin Plug-in PCI-e 16X Memory Card Slot Connector. Online. Dostupné z: https://www.elecbee.com/en-13520-pcie-express-connector-164-pin-plug-in-pci-e-16x-memory-card-slot-connector. [cit. 2025-05-19].

A Obsah elektronické přílohy

```
/ ..... kořenový adresář
├── Knihovny periferií
│   ├── Knihovna FLASH
│   │   ├── flash.h
│   │   └── flash.c
│   ├── Knihovna I2C
│   │   ├── i2c.h
│   │   └── i2c.c
│   └── Knihovna SCI
│       ├── SCI.h
│       └── SCI.c
└── Projekt zařízení
    └── Temperature Monitoring Device ..... projekt zařízení v CodeWarrioru
```

A.1 Knihovny periferií

Knihovny pro práci s periferiemi paměti FLASH, I^2C a SCI. Knihovny jsou přiloženy samostatně a zároveň jsou součástí projektu zařízení.

A.2 Projekt zařízení

Projekt zařízení napsaný ve vývojovém prostředí CodeWarrior. Obsahuje všechny knihovny a soubory uvedené v kapitole Realizace. Označuje software zařízení.