ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA ELEKTROTECHNICKÁ

KATEDRA APLIKOVANÉ ELEKTRONIKY A TELEKOMUNIKACÍ

BAKALÁŘSKÁ PRÁCE

Inteligentní termostat

Petra Kristová **2018**

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta elektrotechnická
Akademický rok: **2017/2018**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Petra KRISTOVÁ**
Osobní číslo:	**E15B0013P**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Elektronika a telekomunikace**
Název tématu:	**Inteligentní termostat**
Zadávající katedra:	**Katedra aplikované elektroniky a telekomunikací**

Zásady pro vypracování:

1. Porovnejte vlastnosti a možnosti dostupných senzorů teploty.
2. Vytipujte vhodnou platformu řídicího mikrokontroléru.
3. Navrhněte uživatelské rozhraní a způsob zobrazení zadávaných parametrů.
4. Navrhněte hardwarové uspořádání zařízení pro lokální měření teploty s ovládacím výstupem pro dvoustavovou regulaci.
5. Implementujte firmware termostatu.

Rozsah grafických prací: **podle doporučení vedoucího**

Rozsah kvalifikační práce: **30 - 40 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

Student si vhodnou literaturu vyhledá v dostupných pramenech podle doporučení vedoucího práce.

Vedoucí bakalářské práce: **Ing. Jiří Basl, Ph.D.**
Katedra aplikované elektroniky a telekomunikací

Datum zadání bakalářské práce: **10. října 2017**

Termín odevzdání bakalářské práce: **7. června 2018**

Doc. Ing. Jiří Hammerbauer, Ph.D.
děkan

L.S.

Doc. Dr. Ing. Vjačeslav Georgiev
vedoucí katedry

V Plzni dne 10. října 2017

Abstrakt

Práce pojednává o návrhu aplikace inteligentního termostatu pro lokální měření teploty. Nejprve jsou porovnány vlastnosti různých dostupných senzorů pro měření teploty a zvolena platforma řídicího mikrokontroléru pro aplikaci. Dále se práce věnuje návrhu uživatelského rozhraní (způsobu zobrazení a zadávání jednotlivých parametrů) a návrhu hardwarového uspořádání celého zařízení s ovládacím výstupem pro dvoustavovou regulaci. Nakonec je implementován firmware dané aplikace.

Klíčová slova

displej, mikrokontrolér, paměť, program, rozhraní, řízení, senzor, teplota, termostat

Abstract

The theses deals with smart thermostat design intended for local temperature measurement. The features of the available temperature sensors are evaluated and the platform of the control application microcontroller is chosen. Next, the user interface and the hardware arrangement of the device was designed, including the control output. Finally, the application firmware is implemented.

Key words

control, display, interface, memory, microcontroller, thermostat, temperature, sensor

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně, s použitím odborné literatury a pramenů uvedených v seznamu, který je součástí této bakalářské práce.

Dále prohlašuji, že veškerý software, použitý při řešení této bakalářské práce, je legální.

.....

podpis

V Plzni dne 5.6.2018 Petra Kristová

Poděkování

Tímto bych ráda poděkovala vedoucímu bakalářské práce Ing. Jiřímu Baslovi, Ph.D. za cenné profesionální rady, připomínky a metodické vedení práce.

Obsah

Úvod

Měření a regulace teploty jsou dnes běžně využívané a hlavně potřebné v mnoha aplikacích a nejrůznějších odvětvích lidské činnosti. Od řízení složitých průmyslových procesů, přes lékařské aplikace, zpracování potravin, sportovní a zájmovou činnost, až po domácí využití. Právě v oblasti domácího využití a z pohledu koncového uživatele má měření a regulace velký význam, protože je součástí každého topného systému. Složitost takového zařízení se velmi různí, od nejjednodušších pokojových termostatů, provádějících dvoustavovou regulaci teploty na základě měření jedné lokální teploty, až po pokročilé regulátory, které vyhodnocují současně venkovní teplotu i teploty v jednotlivých místnostech a obsahují složité řídicí algoritmy.

Obsahem bakalářské práce je návrh pokojového termostatu pro dvoustavovou regulaci lokální pokojové teploty. I přes tuto základní funkci je však zařízení navrženo moderním způsobem. Je zde využit výkonný 32-bitový mikrokontrolér s grafickým dotykovým LCD displejem, což umožňuje příjemné uživatelské rozhraní. Navíc byla při výběru použité hardwarové platformy zvolena varianta možného budoucího rozšíření (např. měření z více vzdálených čidel).

V první kapitole práce jsou zmíněny základní principy měření teploty a probrány jednotlivé typy dostupných čidel - jak analogových, tak digitálních. V této souvislosti jsou pro každý z uvedených druhů vybrány příklady součástek dostupných na trhu. Na základě toho je pak vybrán konkrétní senzor, který je v aplikaci použitý. Další kapitola 2 se zabývá rozborem požadavků na použitý řídicí mikrokontrolér a specifikuje konkrétní vývojový kit zvolený pro realizaci. Obsahem kapitoly 3 je zdokumentovaný návrh grafického uživatelského rozhraní aplikace, obsahující popis jednotlivých obrazovek a grafických prvků, ze kterých se skládají. Návrh blokové struktury a popis jednotlivých hardwarových bloků je obsahem kapitoly 4. Nakonec je v kapitole 5 popsán firmware termostatu.

Seznam symbolů a zkratek

DSI……. Display Serial Inteface (Specifikace rychlého sériového rozhraní pro připojení grafických displejů s vysokým rozlišením)

e ..... Elementární náboj, $e = 1{,}602 \cdot 10^{-19}\,C$

EEPROM…… Electrically Erasable Programmable Read Only Memory (Elektricky mazatelná a programovatelná non-volatile paměť pouze pro čtení)

I^2C..... Inter Integrated Circiut (Přístrojová sériová sběrnice)

k ..... Boltzmannova konstanta, $k = 1{,}38 \cdot 10^{-23}\,J \cdot K^{-1}$

LCD..... Liquid Crystal Display (Displej z tekutých krystalů)

MIPI..... Mobile Industry Processor Interface (Specifikace rozhraní pro připojení displejů v mobilních systémy)

RAM…… Random Access Memory (Paměť s náhodným přístupem - volatile datová paměť)

SMD…… Surface Mount Device (Součástka pro povrchovou montáž plošných spojů)

SPI..... Serial Peripheral Interface (Přístrojová sériová sběrnice)

TFT..... Thin Film Transistor (Technologie LCD displeje)

THT……. Through Hole Technology (Technologie osazování plošných spojů součástkami s drátovými vývody)

1 Teplotní senzory

Jedním z nejpodstatnějších problémů, kterými se musíme při návrhu termostatu zabývat, je měření teploty. Bez něj by zařízení vůbec nemohlo fungovat, protože by mu chyběly potřebné údaje o tom, kdy zapnout natápění, kdy ho naopak vypnout atd. K získání těchto údajů slouží teplotní senzory, kterých je na trhu celá řada.

Teplotu lze měřit metodou přímou, čímž se rozumí přímo měření teploty nějakým teploměrem. Dále existují metody nepřímé, kdy se teplota převede na nějakou jinou veličinu, např. odpor, napětí nebo proud, a pak se dále zpracovává, na čemž jsou založeny teplotní senzory.

Obecně je lze rozdělit na senzory pro bezdotykové měření teploty, které jsou schopny změřit teplotu povrchu určitého tělesa na nějakou vzdálenost. Z důvodu jiného zaměření této práce jim nebude věnováno více prostoru. Dalším typem jsou senzory pro dotykové měření teploty, kdy se měří teplota prostředí, ve kterém se čidlo nachází, tedy přímo se ho „dotýká“. A právě jimi se zabývá první část tohoto textu.

1.1 Analogové teplotní senzory

Jak vyplývá z názvu, výstupem analogových čidel je analogová veličina, tzn. veličina spojitá v čase i v úrovni. Nedochází tedy ke kvantizační chybě (alespoň ne přímo při měření), ta vznikne až při převodu na digitální veličinu v AD převodníku (bude vysvětleno později v kapitole *1.2 Digitální teplotní senzory*). To ale neznamená, že je měření bezchybné. Kromě toho, že záleží na přesnosti konkrétního čidla, může se k jeho výstupnímu signálu přidat nějaké rušení při přenosu k následnému zpracování. To platí zejména v případě, pokud je výstupní signál senzoru slabší, nebo pokud signál putuje nějakou delší přenosovou cestou.

1.1.1 Termočlánky

Termočlánek využívá temoelektrického jevu, tedy přeměny rozdílu teplot na elektrické napětí. Pokud bychom utvořili obvod ze dvou různých vodičů či polovodičů a zároveň by

jejich oba spoje měly rozdílnou teplotu, začal by protékat proud. Pokud bychom obvod přerušili, naměřili bychom mezi rozpojenými konci určitou hodnotu napětí, a právě na tom jsou senzory tohoto typu založeny. Jejich citlivost se pohybuje v desítkách mikrovoltů na 1 °C. Z toho důvodu nachází lepší využití spíše pro měření vysokých teplot v řádu stovek °C.

Podstatnou nevýhodou termočlánků pro některé aplikace však může být skutečnost, že za jejich pomoci jsme schopni změřit pouze rozdíl dvou teplot. To znamená, že jeden spoj musí mít v průběhu měření nějakou známou referenční teplotu, jejíž hodnota se nebude měnit. Pro aplikaci pokojového termostatu se tedy daný typ čidel příliš nehodí.

1.1.2 Odporové senzory

Tyto senzory využívají toho, že se změnou teploty se mění i odpor kovu či polovodiče, což závisí na koncentraci volných nosičů elektrického náboje a jejich pohyblivosti u konkrétního materiálu.

Obecně u odporových čidel může při měření docházet k podstatným chybám. První je způsobena tím, že se čidlo zahřívá, pokud jím protéká proud, také má určitý vliv odpor přívodů dané součástky. Tyto chyby však lze snížit využitím vhodného zapojení, řadu jich lze najít např. v [1].

1.1.2.1 Odporové kovové senzory

Kovy jsou tvořeny souborem kladných iontů v krystalové mřížce a chaoticky se pohybujícími elektrony (tzv. elektronovým plynem). Se zvyšující se teplotou roste amplituda kmitů iontů v mřížce a tím i pravděpodobnost srážky s volnými elektrony, snižuje se tedy střední doba mezi těmito srážkami, což má za následek zvyšování odporu materiálu. Teplotní závislost odporu kovů je možné celkem přesně vyjádřit polynomem druhého stupně, v mnoha případech lze však kvadratický člen zanedbat a využít pouze lineárního vztahu:

$$R = R_0 \cdot \left[1 + \alpha \cdot \left(t - t_0\right)\right] \tag{1.1}$$

kde R_0 je odpor při teplotě t_0 a α teplotní součinitel odporu.

Abychom mohli takového zjednodušení využít, musí být samozřejmě lineární i teplotní součinitel α, a to nejlépe pro co nejširší teplotní interval, ve kterém pak bude čidlo použitelné. Dále od materiálu vyžadujeme i nějakou mechanickou odolnost, odolnost proti korozi a také chemickou a časovou stabilitu. Pro možnost měření vysokých teplot musí mít kov i vysokou teplotu tání. V praxi se pro výrobu těchto senzorů často využívají čisté kovové materiály, což znamená, že dalším požadavkem je i možnost produkce daného materiálu ve velmi čistém stavu.

Největší využití zde nachází platina, která je teplotně stálá a má vysokou teplotu tání (1772 °C). Platinová čidla jsou velmi přesná a rozšířená. Na trhu je můžeme najít s označením PT100, což znamená, že při teplotě 0 °C je hodnota jejich odporu 100 Ω. Lze zakoupit i jiné hodnoty: 50, 200, 500, 1000 a 2000. Také existuje několik tříd přesnosti, které udávají konkrétní rozsahy teplot, které mohou být daným senzorem měřeny, a tolerance.

Niklové senzory se vyznačují vysokou citlivostí, rychlostí odezvy na změnu teploty a svými malými rozměry. Bohužel se zde najdou i nevýhody v podobě nelinearity a nižšího teplotního rozsahu, navíc z dlouhodobého hlediska nejsou příliš stabilní.

Dále existují i měděné senzory, které však kvůli své nízké rezistivitě a náchylnosti k oxidaci nejsou moc rozšířené. Většinou se používají při přímém měření teploty vinutí elektromotorů.

1.1.2.2 Odporové polovodičové senzory

Polovodiče obecně dokáží měnit své vlastnosti na základě působení nějakých vnějších vlivů, např. při změně osvětlení, přiloženého elektrického napětí, magnetického pole, nebo při změně tlaku či teploty.

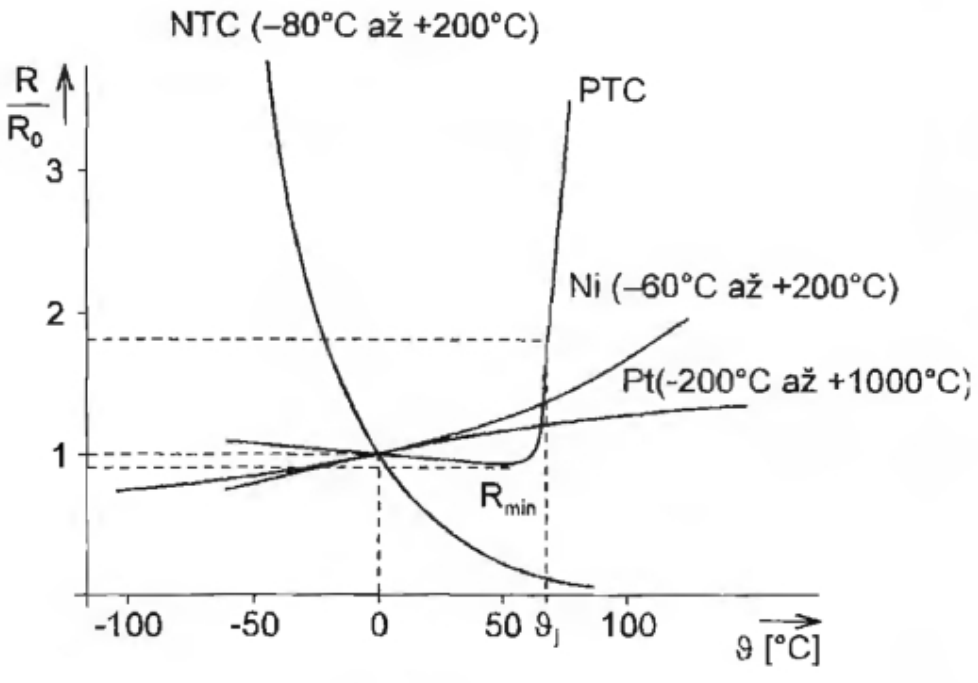
Stejně jako u kovů záleží na pohyblivosti nosičů náboje, navíc se zde ale uplatňuje i změna jejich koncentrace. Při teplotě rovné absolutní nule (tzn. 0 K nebo také -273,15° C) se v polovodiči nevyskytují žádné volné nosiče, což znamená, že tudy nemůže téct proud,

materiál má tedy vysoký odpor. Se zvyšující se teplotou dochází k tepelné aktivaci, tzn. určitá část elektronů získá dostatečnou energii k tomu, aby přeskočila z valenčního do vodivostního pásu. Odpor se tedy s teplotou snižuje (teplotní koeficient může být v některých případech i kladný, což bude vysvětleno dále v této kapitole).

Do kategorie odporových polovodičových senzorů se řadí termistory, což jsou teplotně závislé rezistory. Podle jejich teplotního koeficientu je lze rozdělit na NTC (negative thermal coefficient) a PTC (positive thermal coefficient). U NTC termistorů dochází k tepelnému vybuzení nosičů (viz předchozí popis), odpor se tedy s teplotou snižuje. U PTC termistorů má naopak teplota v určitém intervalu vliv jen na pohyblivost nosičů a odpor s teplotou roste.

Dalším typem jsou monokrystalické senzory z vlastních polovodičů, které opět využívají závislosti odporu polovodiče na teplotě. Vyrábí se např. z křemíku, germania, india a jejich slitin. V praxi jsou však nejvíce rozšířená křemíková čidla, jejichž odpor se s teplotou zvyšuje podobně jako u kovů, výhodou je jejich časová stálost.

Porovnání několika odporových senzorů ukazuje obr. 1.1.



Obr. 1.1 Teplotní závislosti odporových senzorů teploty (převzato z [1])

1.1.3 Monokrystalické PN senzory

Tyto polovodičové senzory využívají teplotní závislosti napětí PN přechodu dané součástky v propustném směru. Jejich výhodou je možnost měření i velmi nízkých teplot.

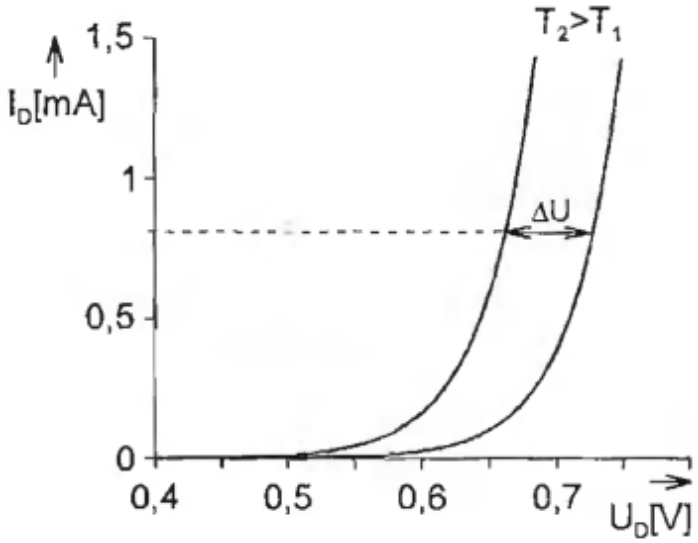
Proud PN přechodem diody v propustném směru lze vyjádřit Shockleyho rovnicí:

$$I_D = I_S \cdot \left(e^{\frac{U_D}{m \cdot U_T}} - 1 \right) \tag{1.2}$$

kde I_D je proud PN přechodem v propustném směru, I_S saturační proud PN přechodem v závěrném směru, U_D napětí na PN přechodu v propustném směru, m rekombinační koeficient a U_T je teplotní napětí dané vztahem $U_T = \frac{k \cdot T}{e}$

zde k je Boltzmannova konstanta, T teplota v Kelvinech a e zde vyjadřuje elementární náboj.

Jak je patrné z obr. 1.3, napětí U_D se s rostoucí teplotou snižuje.



Obr. 1.3 Teplotní závislost U_D diody (převzato z [1])

Tranzistorové senzory fungují podobně jako diodové, je zde využívána teplotní závislost PN přechodu báze-emitor v propustném směru. Toto napětí se dá vyjádřit vztahem:

$$U_{BE} = \frac{k \cdot T}{e} \cdot \ln \frac{I_C}{I_S} \tag{1.3}$$

Podobně jako u diod napětí klesá se zvyšující se teplotou.

1.2 Digitální teplotní senzory

Druhou velkou skupinu teplotních čidel představují digitální senzory, které rovnou převedou danou analogovou veličinu do digitální (tzn. nespojité) podoby. Takový převod se skládá ze tří částí – vzorkování, kvantování a kódování. Vzorkování znamená, že spojitý signál se nejprve rozdělí v čase, je tedy definován jen v určitých časových okamžicích, které jsou dány vzorkovací frekvencí. Poté následuje kvantování, kdy se signál diskretizuje i v amplitudě. Protože existuje jen omezené množství kvantizačních úrovní, vzniká zde určitá kvantizační chyba, jejíž velikost záleží právě na tom, jak velké jsou jednotlivé kvantizační stupně. Nakonec přichází na řadu kódování, které dané hodnotě signálu přiřadí určitý kód (většinou binární).

Ke zpracování se posílá (ve většině případů sériově) sled logických úrovní „0“ a „1“ (neboli L a H), tento signál je tedy odolnější vůči nějakému vnějšímu rušení, které se k němu cestou může přidat, než signál analogový. Z toho vyplývá, že digitální senzory lze použít i pro delší přenosové cesty v zarušenějším prostředí. Komunikace s těmito čidly je však realizována přes různá rozhraní, kde je třeba využít nějakých komunikačních protokolů, což by se někomu mohlo zdát jako určitá nevýhoda.

1.2.1 Typy digitálního rozhraní

V následujících podkapitolách budou stručně vypsány principy funkce digitálních rozhraní. V praxi lze jistě nalézt i další, z důvodu rozsahu této práce však bude pozornost věnována pouze některým.

1.2.1.1 1-Wire

Sběrnici 1-Wire navrhla firma Dallas Semiconductor. Její výhodou je možnost připojení nějakého zařízení s využitím pouze dvou vodičů. Jeden slouží jako zem, po druhém probíhá obousměrná komunikace mezi řídicím obvodem (master) a nějakým menším ovládaným zařízením (slave), což může být v tomto případě například nějaký teploměr. Takových zařízení se však může na sběrnici připojit i více.

Před začátkem komunikace vyšle master tzv. reset pulz, který na datovém vodiči nastaví logickou „0“ na dobu minimálně 480 μs, poté se vrátí zpět do logické „1“. Připojené zařízení tuto změnu zaznamená a opět vodič uzemní na 60 – 240 μs. Tím master zjistí, že je ke sběrnici připojeno nějaké zařízení a může začít přenos dat. Ten probíhá v časových slotech, mezi kterými vždy musí být mezera (log. „0“) minimálně 1 μs. Během jednoho časového slotu trvajícího 60 – 120 μs se přenese 1 bit.

Mohou být přijaty nebo vyslány 4 druhy těchto slotů – zápis „0“ nebo „1“ a čtení „0“ nebo „1“. Pokud chce master něco vyslat do připojeného zařízení, nastaví na sběrnici maximálně na 15 μs logickou „0“. Podle toho, co se má zapsat, ji buď opět uvolní do „1“, nebo „0“ ponechá po zbytek daného slotu. Pokud se mají číst data od připojeného zařízení, datový vodič se minimálně na 1 μs uzemní, opět uvolní a dále se časový slot doplní hodnotou, kterou chce zařízení vyslat. Data se posílají po bajtech, přičemž napřed je odeslán nultý bit a sedmý bit jako poslední.

Na výše popsaném principu funguje sběrnice pouze s jedním připojeným zařízením. Pokud by jich bylo více, komunikace by vypadala o něco složitěji.

1.2.1.2 SPI

Další velmi rozšířenou sběrnicí je SPI (Serial Peripheral Interface). Opět se jedná o propojení řídicí jednotky (master) s jedním či více ovládanými zařízeními (slave). Od řídicího obvodu jsou ke všem zařízením rozvedeny hodinové impulzy SCK, které zajišťují synchronizaci. Dalšími dvěma vodiči se přenáší vstupní a výstupní sériová data – MISO (Master In, Slave Out), MOSI (Master Out, Slave In). Protože tyto vodiče jsou společné pro všechna připojená zařízení, musí se před začátkem přenosu nejprve zvolit,

které bude vysílat nebo přijímat data od mastera. K tomu slouží signál CS (Chip Select), který vždy vybere nanejvýš jedno z těchto zařízení. Nemůže jich vysílat více najednou, jinak by na vodiči MISO došlo ke kolizi.

Výhoda SPI je v jednoduchosti a rychlosti. Protože komunikace probíhá v každém směru po samostatném vodiči, není zde nutný složitý protokol, který by řešil přepínání mezi příjmem a vysíláním dat, jako tomu bylo u 1-Wire. Díky volbě zařízení pomocí CS navíc odpadá nutnost posílat adresu tohoto konkrétního zařízení mezi sériovými daty, což zrychlí přenos. Rychlost se odvíjí od vlastností daného obvodu, hodinová frekvence se pohybuje od stovek kHz až do několika MHz.

Hlavní nevýhoda této sběrnice spočívá v tom, že pro každý signál CS je potřeba jeden samostatný vodič. To znamená, že pokud bude k řídicímu obvodu připojeno hodně zařízení typu slave, počet obsazených výstupních pinů na obvodu master také bude narůstat. Jako další komplikace by se mohla nakonec zdát i synchronizace, protože přenos pak nemůže fungovat na dlouhé vzdálenosti – zpoždění hodinového signálu musí být shodné se zpožděním přenášených dat. Další komplikací by se v některých případech mohla jevit i neexistence signálu ACK (Acknowledge), který by potvrdil příjem dat na straně zařízení typu slave, není tedy možné snížit rychlost přenosu v případě, že by toto zařízení nestíhalo data včas zpracovávat.

1.2.1.3 I^2C

I^2C (Inter-Integrated Circuit Bus) je sběrnice, kterou původně vyvinula firma Philips. Opět propojuje více zařízení dvěma vodiči, přičemž jeden slouží k rozvodu hodinových pulzů (SCL) a druhý k obousměrnému přenosu sériových dat (SDA). Protože je sběrnice založena na budičích s otevřenými kolektory, nedochází k desktruktivním konfliktům, může být tedy připojeno více stanic najednou. Přístup ke sběrnici může být řízen i více zařízeními (multimaster), v jeden okamžik však může jako master pracovat nanejvýš jedno, jinak by došlo ke kolizi.

Komunikace probíhá následovně. Pokud se nevysílají žádná data, díky kolektorovým rezistorům je klidová úroveň na obou vodičích log. „1“. Přenos se zahájí tzv. start bitem (S), což prakticky znamená kombinace doběžných hran - master nastaví „0“ nejprve

na vodiči SDA, poté i na SCL. Následně začíná přenos po 8 bitech (1 byte) s MSB (Most Significant Bit – nejvyšší bit) napřed. První byte představuje adresu zařízení, se kterým chce obvod typu master komunikovat, poté následují data. Po osmi hodinových pulzech SCL vyšle příjemce potvrzovací bit ACK (Acknowledge) tak, že stáhne vodič SDA do „0“, zatímco na vodiči SCL je od řídicího zařízení „1“. Tím se potvrdí příjem osmice bitů a může se odeslat další. Když se přenesou všechna data, vyšle se tzv. stop bit (P), což je kombinace náběžných hran nejprve signálu SCL a vzápětí i SDA. Pokud však následuje nějaká další zpráva, může se místo stop bitu odeslat rovnou start bit a poté opět byte s adresou.

Adresace připojených zařízení může být dvojího typu. Sedmibitová umožňuje připojení zařízení o maximálním počtu 2^7 (128) a adresa se při přenosu vejde do jednoho bytu. Prvních 7 bitů představuje právě adresu a poslední bit R/W určuje směr toku dat dalších dat: $A_6\ A_5\ A_4\ A_3\ A_2\ A_1\ A_0$ R/W

Desetibitová adresace umožňuje připojení až 2^{10} (1024) zařízení, je však nutné tuto adresu při odesílání rozdělit do 2 bytů, které vypadají následovně:

1 1 1 1 0 $A_9\ A_8$ R/W $A_7\ A_6\ A_5\ A_4\ A_3\ A_2\ A_1\ A_0$

Co se týče rychlosti, je dáno několik standardů – 10 kbit/s (Low speed mode), 100 kbit/s (Standard mode), 400 kbit/s (Fast mode), 1 Mbit/s (Fast mode+), nebo až 3,4 Mbit/s (High speed mode).

Výhodou sběrnice I^2C je možnost zpoždění dalšího pulzu SCL od zařízení typu slave, který strhne vodič SCL do „0“. Protože master vysílá „1“, tento konflikt zaznamená a přestane vysílat nová data, dokud na SCL nebude opět „1“. To je velmi užitečné v případě, kdy slave nestíhá zpracovávat přijímaná data. Další výhodou oproti SPI je nižší počet potřebných vodičů.

Za nevýhodu by se dala považovat složitost komunikačního protokolu a nutnost posílání adres mezi sériovými daty. Podobně jako u SPI také nelze použít I^2C na delší vzdálenosti kvůli synchronizaci přenášených dat a hodinových pulzů.

1.2.1.4 Pulzní výstup

Posledním rozhraním, kterému bude v této kapitole věnován prostor, je pulzní výstup některých digitálních teplotních senzorů. V principu jde o generování signálových pulzů (proudových nebo napěťových), jejichž počet či frekvence je dána teplotou. Nejprve se přímo v integrovaném čidle provede analogově-digitální převod, tzn. naměřená teplota se převede do digitální podoby. Na základě této hodnoty se na výstupu čidla vygeneruje určitý počet pulzů, které se odešlou k dalšímu zpracování řídicímu obvodu. Během toho už se převádí další hodnota naměřené teploty na digitální veličinu. Po uběhnutí periody (času na odeslání pulzů udávajících jeden vzorek teploty) už se vysílají další pulzy dány novým naměřeným vzorkem.

Výhodou těchto čidel je, že některá mají proudový výstup, který je mnohem odolnější vůči okolnímu rušení než výstup napěťový. Navíc některé typy těchto čidel lze připojit přímo na GPIO (General Purpose Input-Output) pin řídicího zařízení, kde není potřeba řešit nějaké složité komunikační protokoly, stačí zde využít čítač, který bude počítat vstupní pulzy.

Jako nevýhoda by se mohla zdát menší vzdálenost, na kterou lze tato čidla použít, v porovnání s ostatními rozhraními to však není téměř žádný rozdíl.

1.3 Senzory dostupné pro realizaci

V této kapitole budou uvedeny základní parametry několika konkrétních senzorů dostupných pro realizaci dané aplikace. Opět lze na trhu jistě najít i další různá čidla, z důvodu rozsahu této práce však bude vybráno pouze několik příkladů.

1.3.1 Analogové

1.3.1.1 Termistor NTC 10 kΩ

Jako první teplotní analogový senzor byl vybrán termistor NTC (tedy se záporným tepelným koeficientem) v provedení THT (Through Hole Technology). Dle [10] má tato

konkrétní součástka hodnotu 10 kΩ s přesností ±1% při teplotě 25 °C. S její pomocí lze měřit teploty v intervalu -55 až +125 °C. Závislost odporu na teplotě je dána vztahem:

$$R = 10 \cdot 10^3 \cdot e^{3380\,(1/25 - 1/T)} \tag{1.4}$$

kde R je odpor termistoru při teplotě T.

1.3.1.2 Texas Instruments LM35CAZ/NOPB

Druhým analogovým příkladem je integrované čidlo od firmy Texas Instruments, opět v provedení THT. Výstupní napětí není závislé na napájecím napětí v rozsahu 4 až 30 V, závislost na teplotě lze vyjádřit takto:

$$V_{out} = 10\,mV/°C \tag{1.5}$$

S tímto čidlem lze měřit teploty v rozmezí -40 až 110 °C s výrobcem garantovanou přesností ±0,5 °C při 25 °C. Pro bližší specifikace viz [11].

1.3.2 Digitální

1.3.2.1 Maxim Dallas DS18B20+

Toto integrované čidlo od firmy Dallas se vyrábí jak v provedení THT, tak v provedení SMD (Surface Mounted Device), navíc je k dostání více typů pouzder. Každý typ obsahuje 3 vývody, z nichž jeden je zem, druhý napájecí napětí a třetí slouží pro datový vstup/výstup, který komunikuje přes rozhraní 1-Wire. Měřená teplota je přímo vyjádřena binárním číslem včetně znaménka. Převodník má nastavitelné rozlišení 9 až 12 bitů, odkud se odvíjí přesnost měřené teploty. Rozsah se pohybuje v rozmezí od -55 do +125 °C a výrobce garantuje přesnost ±0,5 °C pro rozsah -10 až +85 °C při rozlišení 9 bitů. Více informací je k nalezení v [12].

1.3.2.2 Texas Instruments TMP275AQ

Dalším vybraným je integrované čidlo od firmy Texas Instruments, které komunikuje přes rozhraní I^2C, v provedení SMD. Teplota je převáděna pomocí AD převodníku s rozlišením 12 bitů a poté odeslána jako binární číslo. Takto lze měřit teploty od -40 do +125 °C, přičemž výrobce udává odchylku max. ±0,5 °C pro rozsah -10 až +80 °C. Pro bližší specifikace viz [13].

1.3.2.3 Texas Instruments LMT01LPG

Posledním z vybraných příkladů je integrované čidlo s pulzním výstupem, opět od firmy Texas Instruments. Během jedné periody se vždy odesílají proudové pulzy, jejichž počet udává jeden vzorek teploty, zároveň se už převádí do digitální podoby vzorek další. Doba této periody je pevně daná a trvá 50 ms, frekvence pulzů 88 kHz je také neměnná.

Tento senzor lze připojit přímo k GPIO pinu a jak už bylo řečeno, jeho výstup je proudový. Lze s ním měřit teploty od -50 do +150 °C s výrobcem uvedenou přesností ±0,5 °C v rozsahu -20 až +90 °C. Počet pulzů během jedné periody je v teoretickém rozsahu 1 až 4095, reálný počet v rozsahu měřitelných teplot je od 15 do 3228 pulzů. Pro více informací viz [14].

1.4 Volba čidla pro aplikaci

Jako čidlo pro danou aplikaci bylo vybráno analogové integrované čidlo LM35, a to hned z několika důvodů.

Protože čidlo bude využíváno pro měření pokojové teploty, bylo by adekvátní požadovat přesnost např. ±0,5 °C. Reálně však měření budou ovlivňovat i další faktory - např. čidlo bude uložené někde poblíž mikrokontroléru, který se po nějaké době zahřeje a následně se zvýší i teplota vzduchu kolem čidla atd. Tyto problémy budou řešeny v rámci dalších kapitol.

Jak již bylo řečeno, čidlo bude umístěné někde v blízkosti mikrokontroléru, tzn. přenosová cesta bude dlouhá cca do 10 cm, navíc o prostředí se nedá říct, že by bylo nějak příliš zarušené. Pro danou aplikaci se tedy hodí i čidlo analogové.

Co se týče rozsahu měřených teplot, pro pokojový termostat požadujeme interval např. od 5 do 30 °C. I tak už zacházíme do extrémů, kterých při běžném užívání jistě nebude dosaženo. I tomu zvolený senzor vyhovuje, navíc s velkou rezervou.

Největší výhodou vybraného čidla je, že výstupní napětí není závislé na napájecím napětí v rozsahu 4 až 30 V. Pokud tedy čidlo připojíme k 5 V nebo ke 4 V, vždy bychom měli dostat stejnou hodnotu výstupního napětí, nebude mít tedy vliv nějaké kolísání napájecího napětí. Další výhodou je (až na malé nepřesnosti) linární závislost výstupního napětí na teplotě v celém měřicím rozsahu.

2 Volba platformy řídicího mikrokontroléru

Další důležitou částí je volba řídicího mikrokontroléru, na kterém bude aplikace termostatu realizována. Stejně jako u výběru teplotního senzoru platí i zde, že v dnešní době lze na trhu nalézt mnoho možností, proto je nutné si nejprve určit nějaké požadavky.

2.1 Požadavky na systémové prostředky mikrokontroléru

Aby bylo možné pomocí daného mikrokontroléru měřit teplotu pomocí zvoleného analogového čidla, měl by disponovat A/D převodníkem. Také potřebujeme spínání natápění, tzn. potřebujeme nějaký výstupní pin, který sepneme/vypneme. Obě tyto záležitosti jsou však samozřejmostí.

Protože v aplikaci termostatu bude den rozdělen do časových pásem, je nutné, aby byl znám aktuální čas. Mikrokontrolér tedy musí mít i možnost získávat čas z RTC (Real Time Clock). To je dnes u mikrokontrolérů na trhu také standard.

Zařízení nebude mít žádný záložní napájecí zdroj z baterie. Všechny parametry se však ukládají do paměti RAM, což znamená, že při výpadku elektřiny by se všechny údaje ztratily. To není pro uživatele úplně pohodlné, měly by zůstat zachovány alespoň nastavené parametry programů (teploty a časy pro daná časová pásma). Z toho vyplývá, že by měl mikrokontrolér umožňovat ukládání parametrů do trvalé paměti (EEPROM, FLASH). Tomu však opět z hlediska výběru mikrokontroléru vyhovuje velký počet možností.

Dalším požadavkem je rozhraní pro LCD displej, který bude sloužit ke komunikaci uživatele se zařízením. Displeje lze vybrat dle způsobu připojení k mikrokontroléru (sériové, paralelní) a dle formátu zobrazovaného obsahu (znakové/grafické, monochromatické/barevné). Tento požadavek už výběr lehce zužuje, lze totiž využít i řadu vývojových kitů od různých výrobců, které již disponují přímo LCD grafickým displejem.

Díky těmto požadavkům byl pro aplikaci termostatu vybrán vývojový kit STM32F469I-DISCO, jehož výkon by se dal jistě využít i na daleko náročnější aplikace.

Při volbě však byl zohledněn LCD dotykový displej, který bude využit pro uživatelské rozhraní. Podrobné informace jsou k nalezení v [15].

3 Uživatelské rozhraní aplikace

3.1 Specifikace požadavků na komunikaci uživatele se zařízením

Jak již bylo řečeno v předchozí kapitole, bylo by vhodné, aby aplikace měla nějaké grafické rozhraní, pomocí kterého bude uživatel schopný vyčíst užitečné údaje – např. o aktuální měřené teplotě v místnosti, nastaveném programu, aktuálním čase atd. Také je důležité, aby se uživateli s aplikací pracovalo pohodlně, tedy aby mu i grafická část poskytla příjemný dojem. K tomu bude sloužit již zmíněný grafický LCD displej.

Dále je nutné, aby se uživatel mohl přepínat mezi jednotlivými obrazovkami a aby mohl zadávat určité parametry – např. při nastavení teploty či začátku časového pásma konkrétního programu atd. To lze samozřejmě realizovat pomocí tlačítek, v tomto případě však bude vhodnější (a pro uživatele pohodlnější) využít dotykového displeje, který je k dispozici.

3.2 Návrh struktury uživatelských obrazovek

Následující podkapitoly obsahují stručný popis jednotlivých uživatelských obrazovek. Ty by měly být navrženy s ohledem na jednoduché a intuitivní ovládání aplikace s plynulým přechodem mezi nimi.

3.2.1 Ovládací prvky

Nejprve budou rozebrány základní ovládací prvky, ze kterých se uživatelské obrazovky skládají. Veškeré dále uvedené datové typy či proměnné jsou definovány ve zdrojových a hlavičkových souborech projektu *Thermal* na přiloženém CD.

- ***Statický text***

Slouží k vypisování řetězců, které se nemění (např. „Teplota:“, „Program:“). V kódu je tento text definován jako struktura *label_t* (viz kapitola *5.1.14 Statický text (label_t)*). Tyto struktury jsou uloženy do pole *labels[]*, které se pak při výpisu celé prochází.

- ***Nadpis obrazovky***

Opět se jedná o určitou formu neměnného textu, který se vypisuje jako nadpis na některých uživatelských obrazovkách. Oproti statickému textu mají však tyto nadpisy jiné parametry. Jsou definovány jako struktury *screenText_t* (viz kapitola *5.1.11 Nadpis uživatelské obrazovky (screenText_t)*), které jsou uloženy do pole *screenTexts[]*. To se prochází při každém vykreslování nové obrazovky.

- ***Dynamický text***

Tento text se mění v průběhu chodu programu a slouží pro výpis proměnných hodnot – např. měřené teploty, stavu natápění, aktuálního času atd. Tyto hodnoty jsou uloženy v nějakých proměnných, které mohou být různého datového typu - např. měřená teplota typu *float*, aktuální rok typu *uint16_t*. Konkrétní proměnná se tedy nejprve převede na textový řetězec typu *char[]*, poté se rovnou vypíše na displej.

- ***Tlačítko***

Každé tlačítko je v programu definováno jako struktura *button_t* (viz kapitola *5.1.12 Uživatelské tlačítko (button_t)*). Struktury jednotlivých tlačítek jsou opět uloženy do pole *buttons[]*, které se prochází při vykreslování nové uživatelské obrazovky nebo při zjištění dotyku na displeji, jehož souřadnice se porovnávají s parametry tlačítek na dané obrazovce.

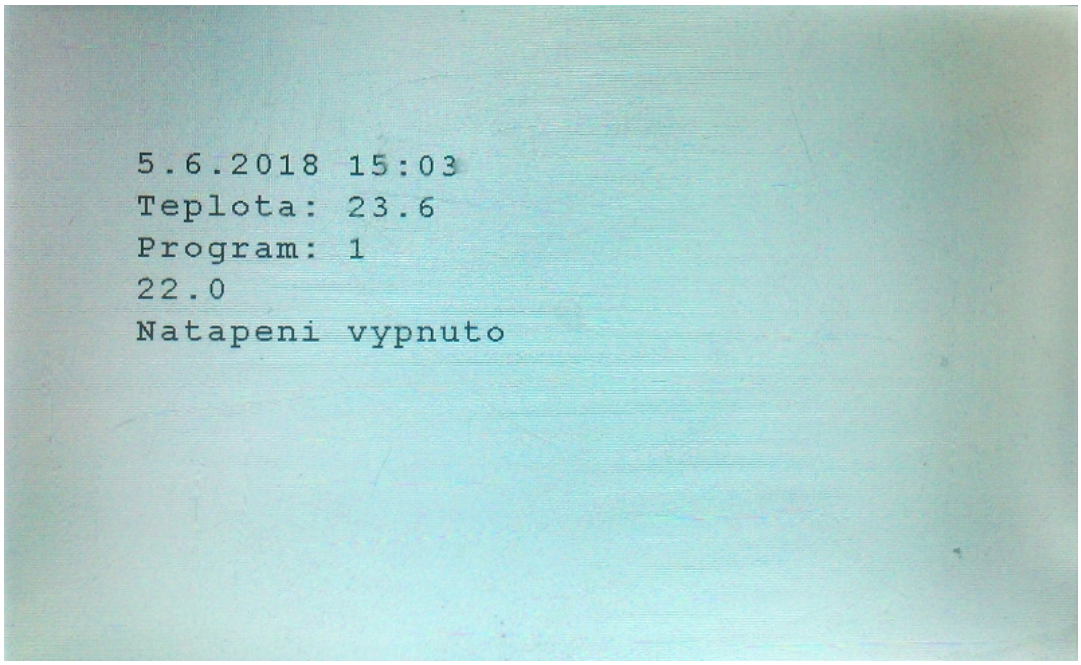
- ***Tlačítko v kalendáři***

Zvláštním typem tlačítka je tlačítko v kalendáři, které v sobě navíc nese informaci o číslu dne, na který odkazuje. Každé z těchto tlačítek je definováno strukturou *calendarBtn_t* (viz kapitola *5.1.13 Tlačítko v kalendáři (calendarBtn_t)*), ty jsou dále uloženy do pole *calendarBtn[]*, které se prochází vždy při listování kalendářem.

3.2.2 Typy obrazovek

3.2.2.1 Úvodní obrazovka

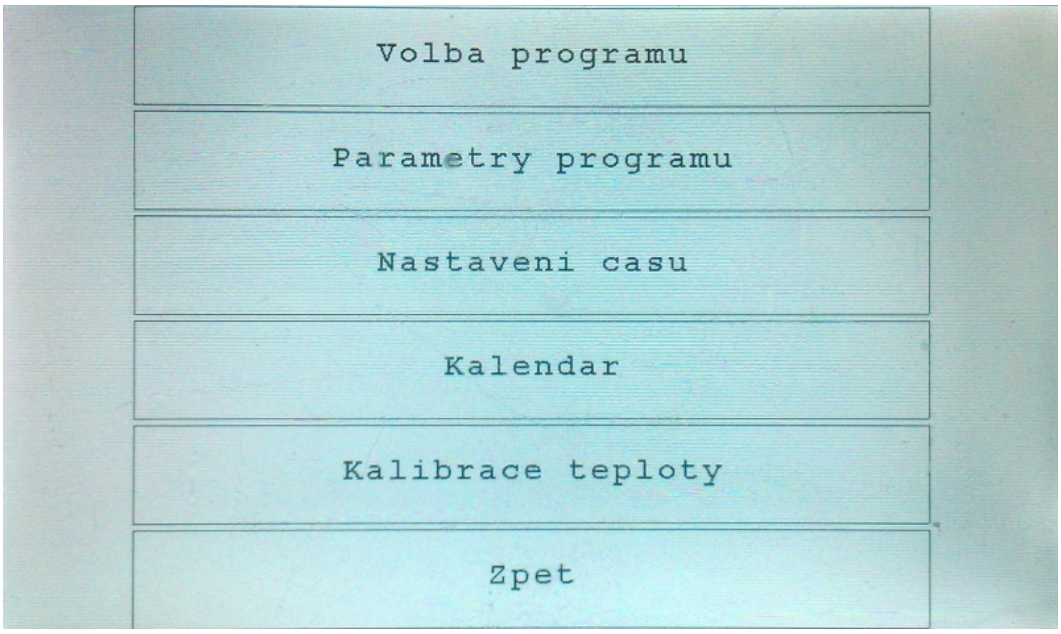
Úvodní obrazovka poskytuje uživateli základní informace o aktuálních údajích. Nejprve je vypsán aktuální datum a čas, měřená teplota, momentálně vybraný program s požadovanou teplotou, která by se v místnosti měla udržovat, a nakonec momentální stav natápění (vypnuto/zapnuto). Při dotyku na jakémkoliv místě displeje se obrazovka přepne na *Menu*.



Obr. 3.1 Úvodní obrazovka

3.2.2.2 Menu

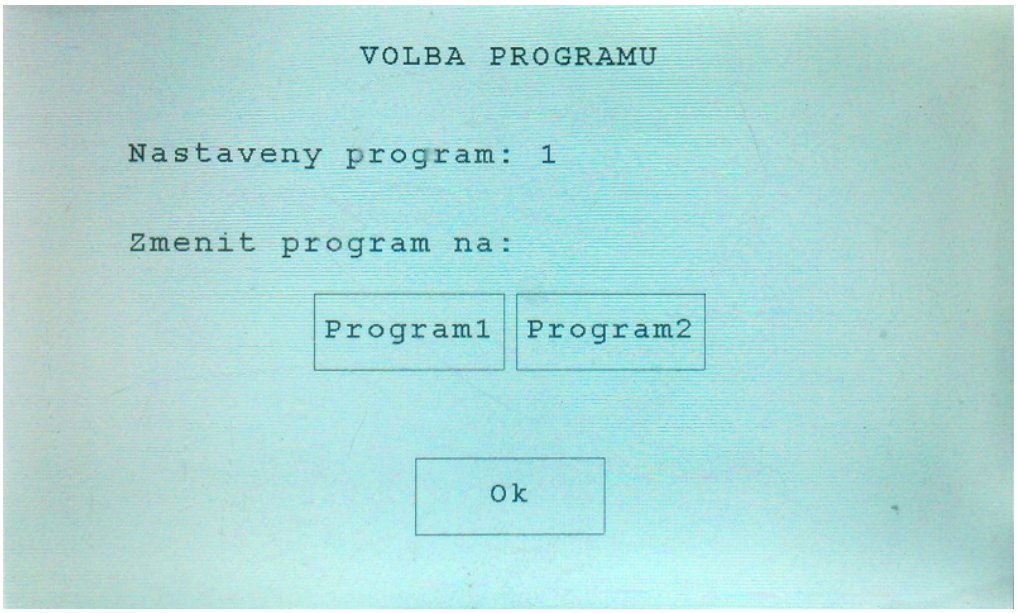
Menu slouží pouze jako rozcestník. Obsahuje 6 tlačítek, která uživatele přepnou na další obrazovku – *Volba programu*, *Parametry programu*, *Nastavení času*, *Kalendář*, *Nastavení kalibrace* a dále tlačítko *Zpět*, kterým se uživatel vrátí na úvodní obrazovku.



Obr. 3.2 Menu

3.2.2.3 Volba programu

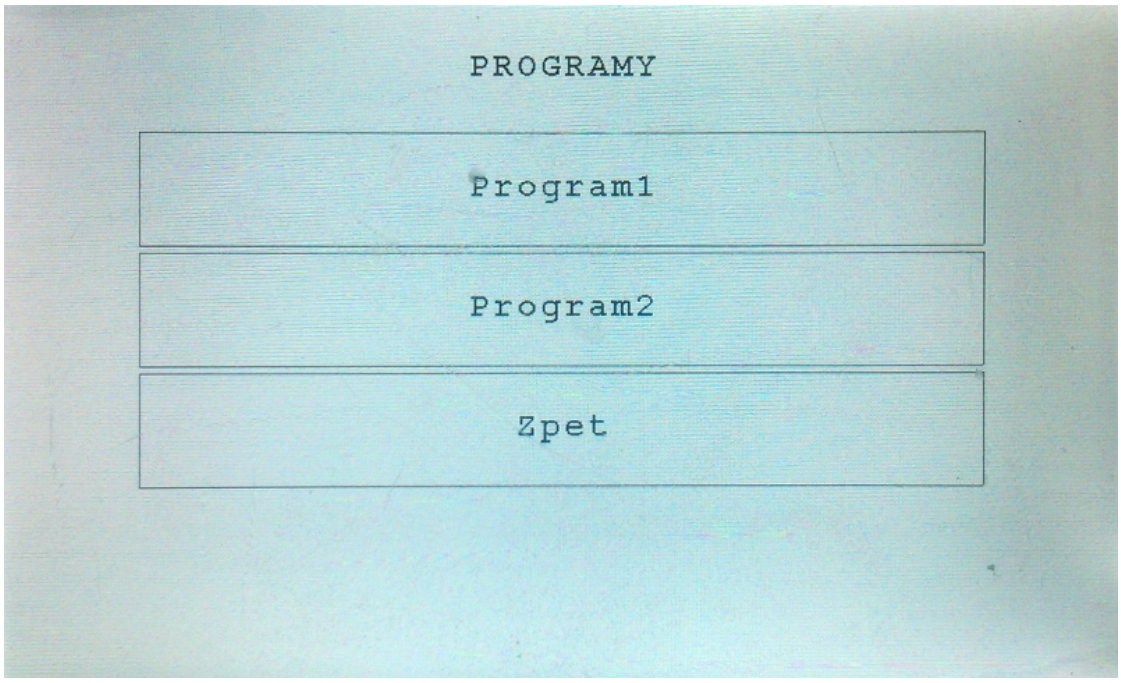
Tato obrazovka slouží pro změnu aktuálního programu, podle kterého termostat reguluje teplotu v místnosti. Je zde vypsán aktuálně nastavený program a dále vykreslena tlačítka, kterými lze přenastavit. Stisknutím tlačítka *Ok* se zvolený program uloží a uživatel je přepnut zpět do *Menu*.



Obr. 3.3 Volba programu

3.2.2.4 Nastavení parametrů programu – menu

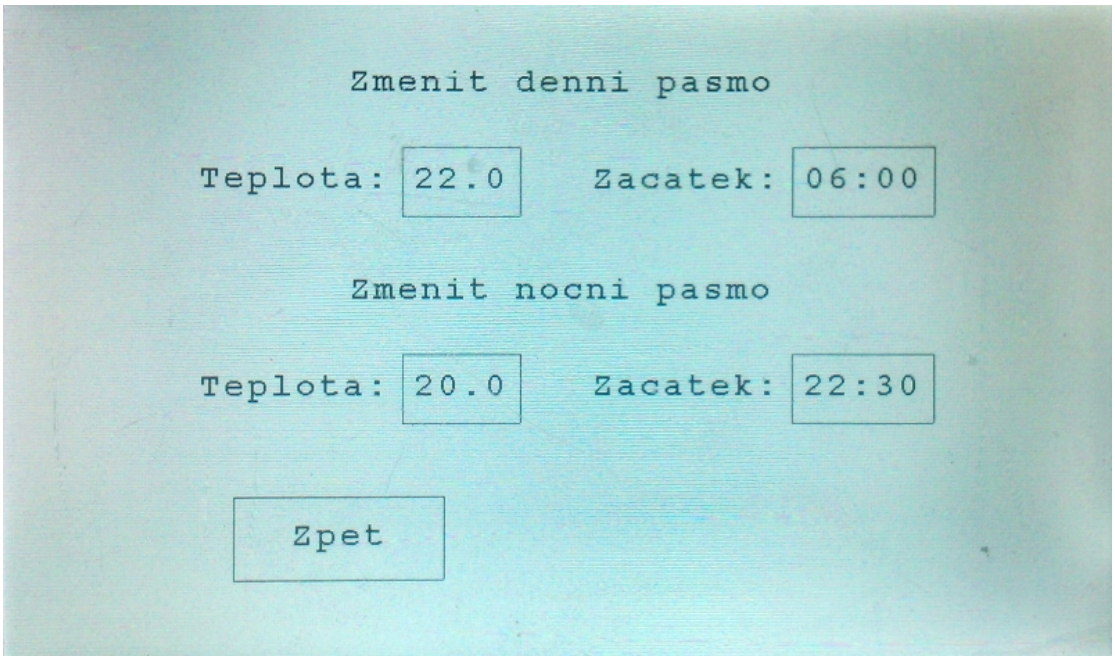
Tato obrazovka opět slouží spíše jako rozcestník, uživatel si zde vybere program, jehož parametry by si přál změnit. Po stisknutí jednoho z tlačítek programů se obrazovka přepne na *Parametry programu*, tlačítko *Zpět* uživatele vrátí do *Menu*.



Obr. 3.4 Nastavení parametrů programu – menu

3.2.2.5 Parametry programu

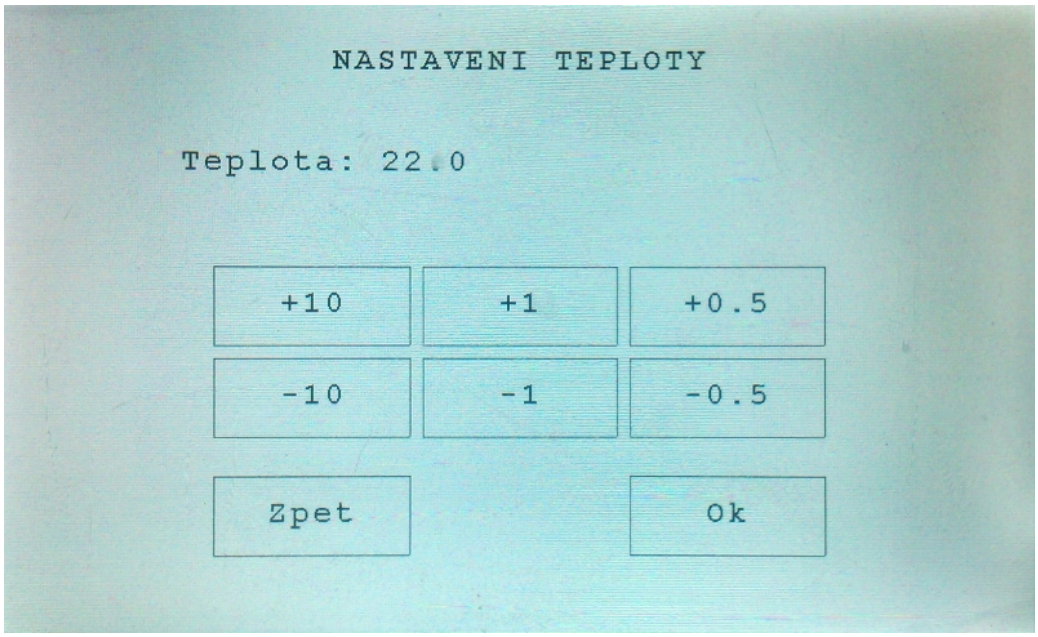
Zde jsou zobrazeny veškeré parametry daného programu – požadovaná teplota, která se by se v pokoji měla udržovat, a začátek časového pásma. Každý program má tato časová pásma dvě – denní a noční. Hodnoty těchto parametrů zároveň slouží jako tlačítka pro jejich změnu. Při stisknutí hodnoty teploty se obrazovka přepne na *Nastavení teploty programu*, při stisknutí času na *Nastavení začátku časového pásma*. Tlačítkem *Zpět* se uživatel opět vrátí do *Nastavení parametrů programu – menu*.



Obr. 3.5 Parametry programu

3.2.2.6 Nastavení teploty programu

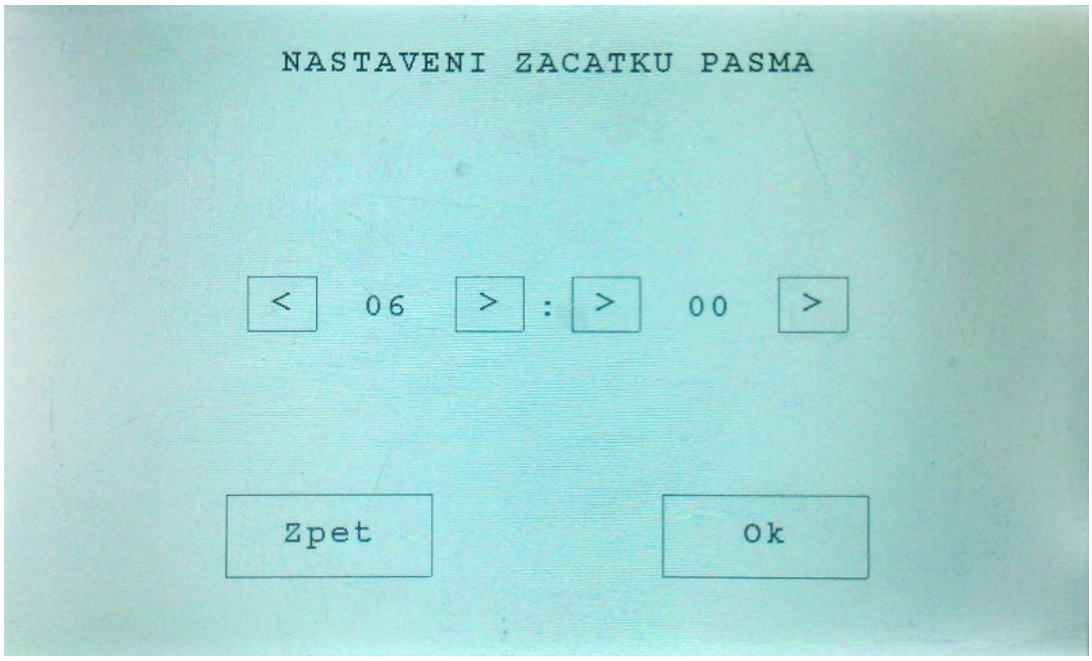
Nejprve je vypsána hodnota dané teploty, pod ní se nachází 6 tlačítek, které umožňují její změnu. Nastavená teplota se však může pohybovat pouze v intervalu 5 až 30 °C. Tlačítko *Zpět* vrátí uživatele do přehledu *Parametry programu*, tlačítko *Ok* novou teplotu navíc uloží.



Obr. 3.6 Nastavení teploty programu

3.2.2.7 Nastavení začátku časového pásma

Zde je vypsán začátek časového pásma, který uživatel může změnit pomocí tlačítek ve formě šipek. Hodiny lze přičítat/odečítat po jedné, minuty po 15. Je zde samozřejmě ošetřeno přetečení, tzn. pokud hodiny zespoda přesáhnou 0, přepne se hodnota na 23 a naopak. To samé s minutami, pokud zespoda přesáhnou 0, objeví se hodnota 45 a naopak. Také je zde nutné počítat s tím, že každé časové pásmo programu musí mít alespoň nějaké trvání. Z toho důvodu nelze nastavit stejný čas začátku denního i nočního pásma. Proto se při zjištění jejich shody čas automaticky přepne na nejbližší možný (např. 6:00 rovnou na 6:30). Tlačítko *Zpět* uživatele znovu přepne do přehledu *Parametry programu*, tlačítko *Ok* nový čas uloží.

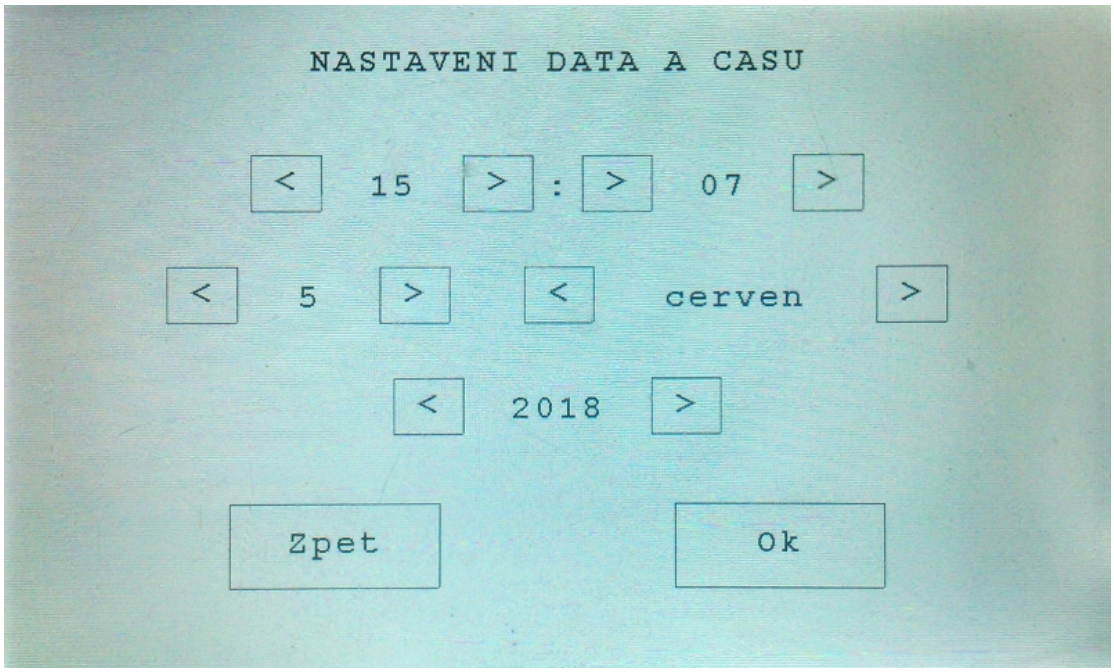


Obr. 3.7 Nastavení začátku časového pásma

3.2.2.8 Nastavení data a času

Další uživatelskou obrazovkou, na kterou se lze dostat z hlavního menu, je *Nastavení data a času*. Nahoře je vypsán aktuální čas, pod ním datum. Všechny tyto parametry lze přenastavit opět pomocí šipek. Samozřejmě i zde musí být nějak ošetřeny hranice zadávaných hodnot. Pokud by měl den překročit zespoda hodnotu 1, přepne se na poslední den v měsíci (tento údaj se pro každý měsíc mění) a naopak. Stejně tak u měsíce, pokud by

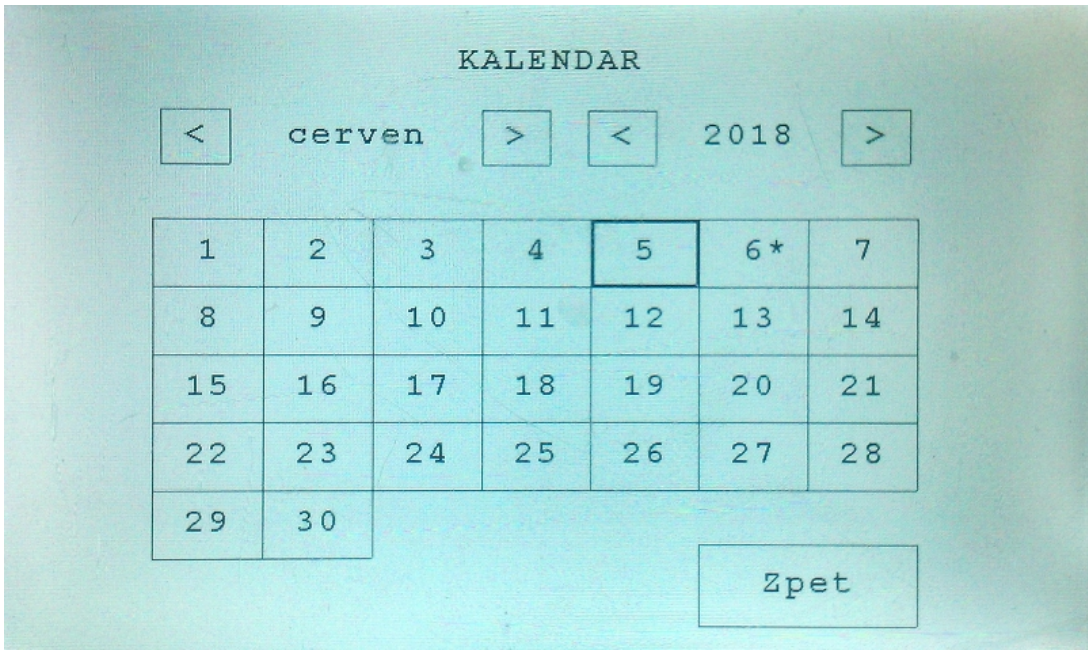
se měla jeho hodnota snížit z ledna, přepne se na prosinec a naopak. Roky jsou omezené od hodnoty 2000 do 3000. Tlačítkem *Zpět* se uživatel vrátí do *Menu*, tlačítkem *Ok* nově nastavený čas uloží.



Obr. 3.8 Nastavení data a času

3.2.2.9 Kalendář

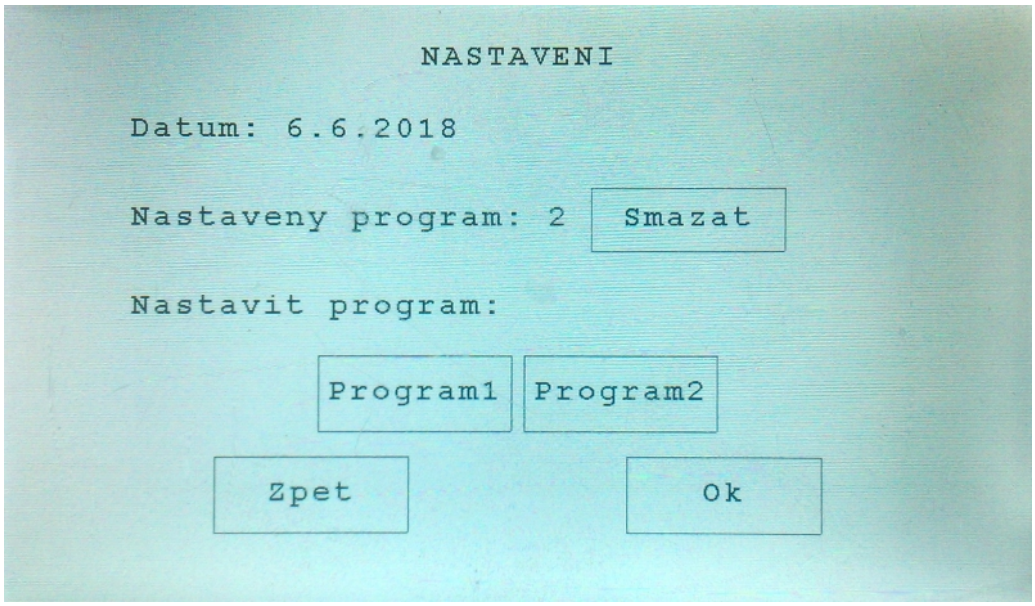
Zde se vykresluje tabulka tlačítek, které označují den v měsíci, nad nimi je vypsán měsíc a rok. Uživatel může kalendářem listovat pomocí šipek (u roku je opět omezení od 2000 do 3000). Při vstupu z *Menu* se vykreslí tabulka pro aktuální měsíc a rok, aktuální den je zvýrazněn tlustším rámečkem tlačítka. Pokud uživatel pro nějaký datum provedl výběr programu, tento den je označen hvězičkou. Při doteku na nějaký konkrétní den se obrazovka přepne na *Kalendář – výběr programu*, při stisku tlačítka *Zpět* opět na *Menu*.



Obr. 3.9 Kalendář

3.2.2.10 Kalendář – výběr programu

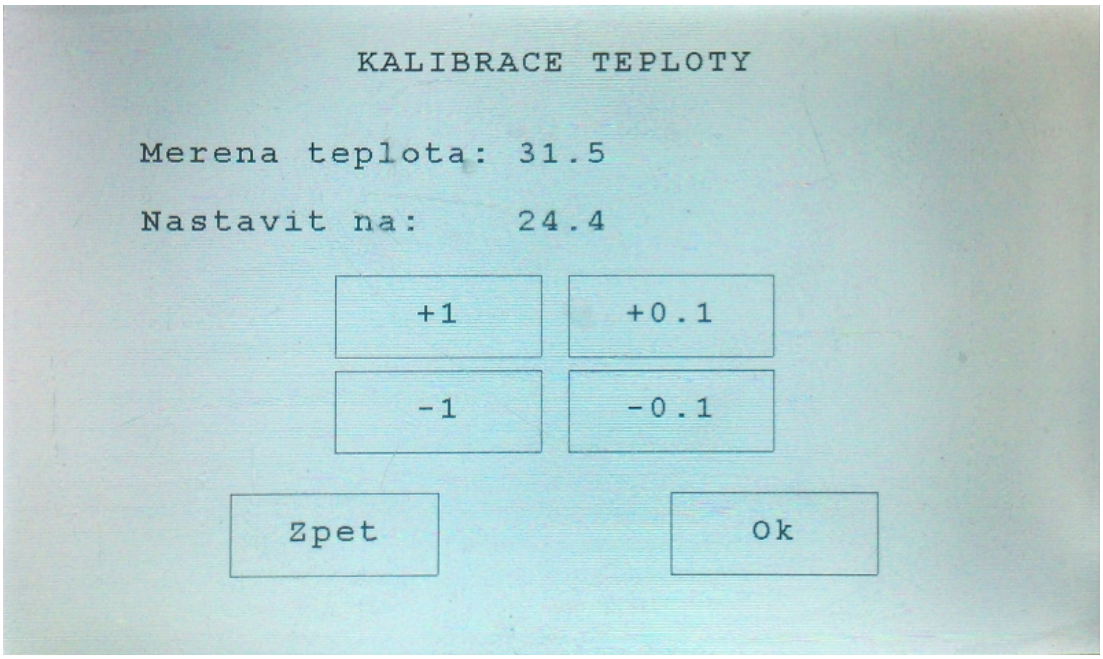
Tato obrazovka slouží k výběru programu pro konkrétní den. Nejprve je vypsán datum, pod ním zvolený program a tlačítko, které umožňuje smazat již provedené nastavení. Dále jsou vykreslena tlačítka *Program1* a *Program2*, díky kterým lze provést výběr. Tlačítkem *Zpět* se obrazovka překreslí na *Kalendář*, tlačítko *Ok* uloží provedené změny.



Obr. 3.10 Kalendář – výběr programu

3.2.2.11 Kalibrace měřené teploty

Základní menu dále nabízí možnost *Kalibrace teploty*, kde je napřed vypsána aktuálně měřená teplota a pod ní hodnota, na kterou si uživatel přeje tuto teplotu zkalibrovat. Dále jsou zde vykreslena 4 tlačítka, pomocí kterých lze kalibrovaná teplota měnit. Po stisku tlačítka *Zpět* se uživatel vrátí do *Menu*, po stisku *Ok* se navíc spočítá a uloží kalibrační konstanta.



Obr. 3.11 Kalibrace měřené teploty

4 Fyzické uspořádání termostatu

4.1 Návrh blokové struktury

Dále následuje popis fyzického uspořádání termostatu. Pro lepší představu by se celé zařízení dalo rozdělit do několika bloků, znázorněných na obr. 4.1.

Obr. 4.1. Bloková struktura termostatu

4.1.1 Mikrokontrolér – vývojový kit

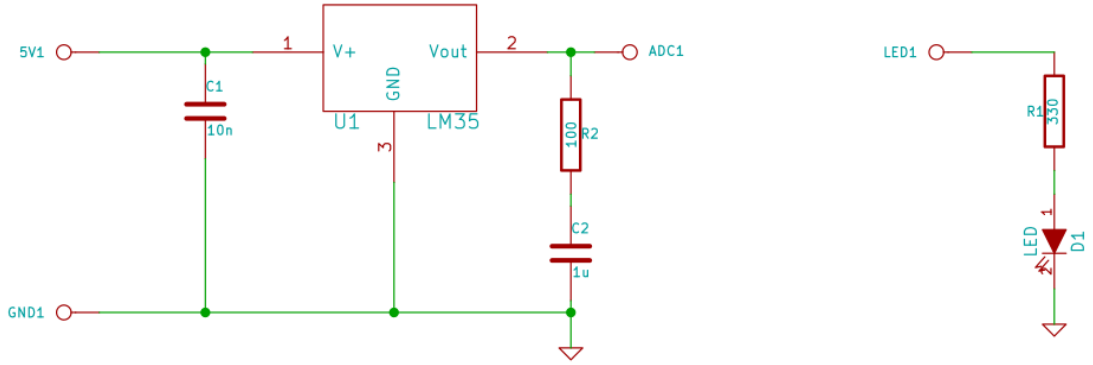
Hlavním blokem je vývojový kit STM32F469I-DISCO, který již byl zmíněn v kapitole *2 Volba platformy řídicího mikrokontroléru*. Jeho základními vlastnostmi jsou: výkonný 32-bitový mikrokontrolér STM32F469NIH6 s vnitřní pamětí FLASH o velikosti 2 MB, pamětí RAM o velikosti 324 KB, čtyřpalcový grafický TFT LCD displej o rozlišení 800x480 pixelů, připojený přes rozhraní MIPI DSI, podporovaný externími pamětmi SDRAM o velikosti 4Mx32 bitů a 128-Mbit Quad-SPI NOR FLASH. Dále kit disponuje i dalšími vlastnostmii, které v této aplikaci nejsou využity, pro bližší informace viz [15]. Blokové schéma převzaté z uvedeného zdroje je připojeno v příloze *A*.

4.1.2 Napájení

Dalším blokem je napájení celého termostatu. Základem zařízení je především vývojový kit, který lze napájet několika způsoby a který již přímo obsahuje potřebné napájecí obvody a stabilizátory. Pro danou aplikaci bylo zvoleno napájení stejnosměrným napětím 5 V přes uživatelský konektor microUSB. K tomu poslouží již hotová nabíječka k mobilnímu telefonu s výstupním proudem 500 mA. Napájení dalších bloků (měření teploty a výstupní obvod) pak už zajistí samotný vývojový kit.

4.1.3 Měření teploty a výstupní obvod

Posledním a velmi důležitým blokem je obvod sloužící pro měření teploty a výstupní obvod pro simulaci spínání kotle. Ty budou na jedné desce, která se k mikrokontroléru připojí pomocí konektorů pro arduino. Schéma viz obr. 4.2.



Obr. 4.2. Schéma měření teploty a výstupního obvodu

Co se týče měření teploty, čidlo je připojeno k napájecímu napětí 5 V dle katalogového listu (viz [11]), jeho výstup vede do A/D převodníku (pin mikrokontroléru *A0*). Spínání kotle je zde simulováno LED diodou (ovládanou pinem *A1*), která se rozsvítí při zapnutí natápění, při vypnutí opět zhasne.

5 Aplikační firmware

5.1 Enumerační a datové typy

Před začátkem popisu samotného firmwaru se přímo nabízí možnost nejprve popsat vlastní nadefinované datové typy, které se vyskytují napříč celou aplikací.

5.1.1 Stav natápění (*heatSet_t*)

Jedná se o dvouhodnotovou enumeraci, jejíž hodnota udává stav natápění. V programu vypadá takto:

```
typedef enum
{
	HEAT_OFF,
	HEAT_ON
} heatSet_t;
```

kde *HEAT_OFF* udává stav vypnuto, *HEAT_ON* značí zapnuto.

5.1.2 Program pro regulaci natápění (*prg_t*)

Další enumerací je *prgt_t*, která slouží k výběru jednoho z možných programů. V kódu je definována jako:

```
typedef enum
{
	PRG_ERR = 0,
	PRG1,
	PRG2,
	PRG_NUM
} prg_t;
```

kde stav *PRG_ERR* určuje chybnou hodnotu (např. v kalendáři bylo smazáno nastavení pro konkrétní den – využití tohoto stavu bude ještě probráno v rámci dalších kapitol). Konkrétní programy zde udávají stavy *PRG1* a *PRG2*. *PRG_NUM* udává počet programů zvýšený o 1, což by mohlo být praktické, pokud by někdy v budoucnu bylo potřeba přidat další programy a jejich počet se vyskytoval na více místech kódu. Odpadá tak totiž nutnost tuto hodnotu ručně přepisovat.

5.1.3 Parametry programu (*prgStruct_t*)

Tato struktura obsahuje údaje o daném programu pro regulaci natápění.

```
typedef struct
{
      float temp1;
      float temp2;
      uint8_t hour1;
      uint8_t minute1;
      uint8_t hour2;
      uint8_t minute2;
} prgStruct_t;
```

Zde *temp1* a *temp2* jsou teploty, které se budou v místnosti udržovat pro daná časová pásma (každý program má denní a noční pásmo). Začátek denního pásma udává *hour1* a *minute1*, začátek nočního pásma *hour2* a *minute2*.

5.1.4 Časové pásmo (*prgZone_t*)

Tato enumerace určuje časové pásmo dne. *DAY_ZONE* udává denní pásmo, *NIGHT_ZONE* noční pásmo.

```
typedef enum
{
      DAY_ZONE = 0,
      NIGHT_ZONE
} prgZone_t;
```

5.1.5 Parametry dne v kalendáři (c*alendarSettings_t*)

Tato struktura slouží pro ukládání nastavených parametrů konkrétního dne v kalendáři. V kódu je typ definován jako:

```
typedef struct
{
      uint8_t day;
      uint8_t month;
      uint16_t year;
      prg_t prgSet;
} calendarSettings_t;
```

kde *day* je den v měsíci, který může nabývat hodnot 1 až 31 (zde samozřejmě záleží na počtu dní v daném měsíci). Číslo měsíce (1 až 12) představuje *month*, dále struktura obsahuje rok (*year*) a program zvolený pro daný den (*prgSet*).

5.1.6 Návratový stav (*state_t*)

Další jednoduchou dvouhodnotovou enumerací je *state_t* sloužící k určení návratového stavu nějaké funkce.

```
typedef enum
{
      STATE_OK = 0,
      STATE_ERR
} state_t;
```

STATE_OK udává, že se požadovaná akce vydařila, *STATE_ERR* že se naopak nevydařila. Toho může být využito např. při ukládání parametrů pro další den v kalendáři a ověření, zda je pro tato data ještě místo v paměti pro to určené.

5.1.7 Datum a čas (*timeStruct_t*)

Do této struktury lze uložit datum a čas. V kódu je definována jako:

```
typedef struct
{
  uint8_t seconds;
  uint8_t minutes;
  uint8_t hours;
  uint8_t day;
  uint8_t month;
  uint16_t year;
} timeStruct_t;
```

Jejím obsahem jsou sekundy, minuty, hodiny ve formátu 0 až 23, den v rozmezí hodnot 1 až 31 (záleží na počtu dní v daném měsíci), měsíc ve formátu 0 až 12 a rok.

5.1.8 ID uživatelských obrazovek (*idScreen_t*)

Tato enumerace slouží k rozlišení indexů jednotlivých uživatelských obrazovek, čehož pak program velmi často využívá při komunikaci aplikace s uživatelem.

```
typedef enum
{
      SCR_ERR = 0,
      BACKGROUND,                   //uvodni obrazovka
      MENU,                         //menu
      PRG_MENU,                     //volba programu
      PRG_SET_MENU,                 //parametry programu - menu
      PRG_SET,                      //prehled parametru programu
      PRG_SET_TEMP,                 //nastaveni teploty programu
      PRG_SET_TIME,                 //nastaveni zacatku casoveho pasma
```

```
      TIME_SET,                    //nastaveni data a casu
      CALENDAR,                    //kalendar
      CALENDAR_SET,                //kalendar - vyber programu
      CALIB                        //kalibrace merene teploty
} idScreen_t;
```

SCR_ERR značí nějakou chybnou hodnotu, která by později mohla být někde použita. *BACKGROUND* udává úvodní obrazovku, *MENU* je obrazovka základního menu, *PRG_MENU* obrazovka pro volbu programu, *PRG_SET_MENU* představuje obrazovku menu pro výběr programu, kterému by si uživatel přál změnit parametry. *PRG_SET* je obrazovka přehledu parametrů daného programu, *PRG_SET_TEMP* obrazovka pro změnu teploty, *PRG_SET_TIME* obrazovka pro změnu začátku daného časového pásma. *TIME_SET* představuje obrazovku pro nastavení data a času, *CALENDAR* obrazovku kalendáře, *CALENDAR_SET* udává obrazovku nastavení parametrů pro určitý den v kalendáři a *CALIB* obrazovku pro nastavení kalibrace měřené teploty.

5.1.9 ID uživatelských tlačítek (*buttonId_t*)

Jedná se o enumeraci sloužící k indexaci všech uživatelských tlačítek. Protože v celé aplikaci je jich opravdu mnoho, nebude zde uvedena takto dlouhá definice dané enumerace. Lze ji však nalézt ve zdrojových kódech přiložených na CD, konkrétně v souboru *graphics.h*.

5.1.10 ID statických textů (*labelId_t*)

Tato enumerace slouží k indexování statických textů. Opět se jedná o delší enumeraci o více prvcích, proto ani zde nebude uvedena její definice. Taktéž ji lze nalézt v přiložených zdrojových kódech, opět v souboru *graphics.h*.

5.1.11 Nadpis uživatelské obrazovky (*screenText_t*)

Tato jednoduchá struktura spojuje konkrétní uživatelskou obrazovku s jejím nadpisem. V kódu je definována jako:

```
typedef struct
{
      idScreen_t idScreen;
      char *text;
} screenText_t;
```

Zde *idScreen* udává ID uživatelské obrazovky a ukazatel *text* odkazuje na textový řetězec, který představuje nadpis dané obrazovky.

5.1.12 Uživatelské tlačítko (*button_t*)

Jedná se o strukturu obsahující údaje o konkrétním uživatelském tlačítku.

```
typedef struct
{
	idScreen_t idScreen;
	idScreen_t nextIdScreen;
	uint16_t x;
	uint16_t y;
	uint16_t width;
	uint16_t height;
	char *text;
} button_t;
```

Zde *idScreen* představuje ID uživatelské obrazovky, na které se tlačítko nachází, a *nextIdScreen* obrazovku, na kterou se má displej po stisku tlačítka překreslit. Parametry *x* a *y* udávají souřadnice tlačítka, parametry *width* a *height* jeho rozměry. Ukazatel *text* odkazuje na textový řetězec, který vykreslí uprostřed tlačítka.

5.1.13 Tlačítko v kalendáři (*calendarBtn_t*)

Struktura tlačítka v kalendáři vypadá velmi podobně jako struktura klasického uživatelského tlačítka.

```
typedef struct
{
	idScreen_t idScreen;
	idScreen_t nextIdScreen;
	uint16_t x;
	uint16_t y;
	uint16_t width;
	uint16_t height;
	uint8_t date;
} calendarBtn_t;
```

Tlačítko v kalendáři však nese navíc údaj o dni v měsíci, na který odkazuje (*date*). Jeho hodnota může nabývat hodnot 1 až 31.

5.1.14 Statický text (*label_t*)

Tato struktura obsahuje údaje o statickém textu. V kódu je definována jako:

```
typedef struct
{
      idScreen_t idScreen;
      uint16_t x;
      uint16_t y;
      char *text;
} label_t;
```

kde *idScreen* představuje ID uživatelské obrazovky, na které se text vypíše. Parametry *x* a *y* udávají souřadnice nápisu a ukazatel *text* opět odkazuje na textový řetězec, který má být vypsán.

5.2 Popis zdrojových kódů

Před vlastním popisem zdrojových kódů by ještě bylo vhodné zmínit použitý vývojový nástroj pro jejich tvorbu. Zde byl využit System Workbench for STM32 verze 1.14.0.

5.2.1 Soubor adc.c

Funkce *void adcInit()* slouží k inicializaci A/D převodníku *ADC1* pro měření pokojové teploty pomocí teplotního čidla.

Funkce *void adcStart()* provádí analogově-digitální převod a přepočet získaných dat na potřebné hodnoty. Nejprve se zapne start konverze, počká se na její konec a poté se z registru vyčtou potřebná data, která se následně přepočítají na výstupní napětí teplotního čidla a z něj na změřenou teplotu *adcTemp*. Dále se tento nově naměřený vzorek započítá do průměrné teploty *adcTempAvg*, ze které se rovnou vypočítá i zkalibrovaná teplota *adcTempCalib* na základě uživatelem nastavené kalibrační konstanty (offsetu) *adcCalibConst*.

5.2.2 Soubor thermo.c

Funkce *void fillPrograms()* naplní pole *programs[]* nějakými výchozími počátečními hodnotami – každému programu přiřadí požadovanou teplotu a začátek časového pásma pro denní i noční pásmo.

Funkce *void heatOutputInit()* provede inicializaci pinu *PC2*, který ovládá spínání výstupního obvodu, tzn. pro simulaci zapínání/vypínání kotle (v tomto případě rozsvěcení LED diody).

Funkce *void cntTempLimits()* vypočítá meze hystereze z aktuálně nastavené požadované teploty *tempSet* ± 0,5 stupňů. Tyto hodnoty uloží do proměnných *tempLow* a *tempHigh*.

Funkce *void compTemp()* slouží k vyhodnocení stavu natápění. Testuje se, jaký je aktuální stav natápění *heating* a zda zkalibrovaná měřená teplota dosáhla některé z mezí hystereze. Na základě toho se rozhoduje o vypnutí či zapnutí natápění (a rozsvícení LED diody).

Funkce *void compTime(prg_t prg)* vyhodnocuje časové pásmo momentálně nastaveného programu a na základě toho požadovanou teplotu v místnosti. Aktuální čas a začátky obou časových pásem daného programu se přepočítají na minuty, jejichž hodnoty se pak mezi sebou porovnávají. Z toho se vyhodnotí, ve kterém časovém pásmu se aktuální čas nachází a jaká by pro toto pásmo měla být požadovaná teplota v místnosti. Pokud je to nutné, změní se hodnota teploty pro regulaci *tempSet*, zároveň se přepočítají meze hystereze. K této změně dochází pokaždé, když se přepne časové pásmo, nebo když se aktuální program změní na jiný.

Funkce *void compDate(prg_t prg)* slouží k porovnání programu aktuálního dne s daty uloženými v kalendáři. Pomocí funkce *findCalendarPrg()* (viz následující kapitola) se projdou uložené parametry a pokud se najdou platná data, porovná se aktuálně nastavený program *prgSet* s nalezeným pro daný den. Pokud se neshodují, *prgSet* se změní.

5.2.3 Soubor calendar.c

Funkce *prg_t findCalendarPrg()* hledá program konkrétního dne v údajích uložených v kalendáři. Vstupními parametry jsou den a měsíc typu *uint8_t* a rok typu *uint16_t*. Pro tyto údaje se pak hledá záznam v poli *calendarSettings[]*. Pokud se požadovaný záznam najde, funkce vrátí program, který se má pro daný den nastavit. Pokud tento záznam nebyl nalezen nebo pokud není platný, vrátí se chybová hodnota *PRG_ERR*.

Funkce *state_t findCalendarIndex()* slouží k nalezení indexu určitého záznamu v poli *calendarSettings[]* (např. při změně údajů pro den, který se již v poli nachází). Vstupními parametry jsou zde *settings* typu *calendarSettings_t **, který představuje ukazatel na určitou strukturu obsahující datum pro porovnání, a *index* typu *uint8_t **, což je ukazatel na proměnnou, do které se má uložit hledaný index. Opět se prochází celé pole *calendarSettings[]* a hledá se záznam s datem, který obsahuje struktura *settings*. Pokud je nalezen, index se uloží a funkce vrátí hodnotu *STATE_OK*. Pokud požadovaný záznam není nelezen, funkce vrátí hodnotu *STATE_ERR*.

Funkce *state_t findEmptyCalendarPrg()* hledá index první volné struktury v poli *calendarSettings[]*. Vstupními parametry jsou opět *calendarSettings_t * settings* a *uint8_t * index*. Pokud se při procházení pole *calendarSettings[]* narazí na takovou strukturu, která v proměnné zvoleného programu *prgSet* nese hodnotu *PRG_ERR* (chybovou hodnotu), znamená to, že tato struktura je buď prázdná nebo neplatná (tzn. jedná se o již zastaralý datum nebo byl tento záznam odstraněn uživatelem). V tom případě se index uloží a funkce vrátí hodnotu *STATE_OK*. Pokud se v poli nenajde ani jedna prázdná či neplatná struktura, funkce vrátí hodnotu *STATE_ERR*.

Funkce *state_t saveCalendarPrg()* slouží k uložení parametrů konkrétního dne do pole *calendarPrg[]*. Vstupním parametrem je *settingsToSave* typu *calendarSettings_t **, což je ukazatel na strukturu obsahující parametry k uložení. Nejprve se pomocí funkce *findCalendarIndex()* zjistí, zda už pole náhodou neobsahuje záznam se stejným datem. Pokud ano, pouze se změní hodnota nastaveného programu *prgSet* v dané struktuře. Pokud pole ještě neobsahuje žádný záznam s tímto datem, pomocí funkce *findEmptyCalendarPrg()* se pole znovu prochází a hledá se volné místo pro nový záznam.

Pokud je nalezeno, zkopírují se sem parametry struktury *settingsToSave*. Pokud ukládání proběhlo v pořádku, celá funkce vrátí hodnotu *STATE_OK*, v opačném případě hodnotu *STATE_ERR*.

Funkce *void deleteOldCalendarPrg()* maže zastaralé záznamy v poli *calendarSettings[]*. To se celé prochází a porovnává se aktuální datum s daty jednotlivých záznamů. Pokud se zjistí, že datum některého záznamu je už minulost, hodnota nastaveného programu *prgSet* dané struktury se nastaví na *PRG_ERR*, což záznam označí jako neplatný.

5.2.4 Soubor rtc.c

Funkce *void rtcSetTime()* uloží čas z určité struktury do registrů RTC (Real Time Clock). Vstupním parametrem je *time* typu *timeStruct_t **, což je ukazatel na strukturu, ze které se čas nastavuje.

Funkce *void rtcGetTime()* naopak uloží aktuální čas z registrů RTC do nějaké struktury. Vstupním parametrem je opět *time* typu *timeStruct_t **, což je v tomto případě ukazatel na strukturu, do které se má aktuální čas uložit.

Funkce *void rtcInit()* slouží k inicializaci RTC. Nejprve se povolí přístup do „backup“ domény mikrokontroléru, což je nezbytné pro to, aby se dalo zapisovat do registrů této periférie. Poté je povolen přesný externí oscilátor LSE a vybrán jako zdroj hodin pro RTC. Nakonec se do registrů zapíše výchozí inicializační hodnota času.

Funkce *void timePartToString()* převádí část času (hodiny nebo minuty) na textový řetězec. Vstupními parametry jsou číslo *num* typu *uint8_t* a *str* typu *char **, což je ukazatel na text, kam se má převedený řetězec uložit. Při převodu se používá funkce *itoa()*, deklarována v souboru *stdlib.h*. Pokud je číslo jednociferné (tzn. menší než 10), přidá se navíc na začátek „0“.

Funkce *void timeToString()* převádí čas na textový řetězec. Vstupními parametry jsou hodiny *hour* a minuty *min* typu *uint8_t*, dále *str* typu *char **, což je ukazatel na textový řetězec, do kterého se má uložit převedený text. Jednotlivé složky (hodiny, minuty) se

převedou pomocí funkce *timePartToString()*, výsledný textový řetězec vznikne jejich spojením pomocí funkce *strcat()*, deklarována v souboru *string.h*.

5.2.5 Soubor graphics.c

Na začátku by bylo vhodné zmínit, že tento soubor slouží pro obsluhu grafického rozhraní, tzn. je zde realizováno vykreslování ovládacích prvků (tlačítka, dynamické a statické texty). K tomu jsou zde využity funkce *BSP_LCD_DisplayStringAt()*, *BSP_LCD_DrawRect()* a další, které byly převzaty z knihoven od výrobce STMicroelectronics. Z důvodu přehlednosti textu zde nejsou uvedeny vstupní či výstupní parametry funkcí, lze je však dohledat v příslusných hlavičkových souborech.

Funkce *uint16_t cntTextLength()* počítá délku textového řetězce v pixelech, což je často využíváno při výpočtech souřadnic některých ovládacích prvků na displeji. Vstupními parametry jsou šířka fontu *width* typu *uint16_t* (předpokládáme, že všechny znaky fontu jsou stejně široké) a dále *text* typu *char* *, což je ukazatel na textový řetězec, jehož délku chceme spočítat.

Funkce *void fillScreenTexts()* naplní struktury *screenText_t*, které obsahují informace o nadpisech jednotlivých uživatelských obrazovek, a uloží je do pole *screenTexts[]*. Jako indexů je zde využito enumerace ID obrazovek *idScreen_t*.

Funkce *void newLabel()* slouží k vytvoření nového statického textu typu *label_t* a jeho uložení do pole *labels[]*. Vstupními parametry jsou zde *i* typu *labelId_t*, což je index daného textu, a jeho souřadnice *x* a *y* typu *uint16_t*. Dále *idScr* typu *idScreen_t*, což vyjadřuje obrazovku, na které se nápis nachází, a *txt* typu *char* *, což je ukazatel na textový řetězec nápisu.

Funkce *void fillLabels()* naplní pole statických textů *labels[]*. Využívá při tom funkci *newLabel()*, kterou volá při ukládání každého textu, jako vstupní parametry *i* a *idScr* zadává hodnoty enumerací ID jednotlivých nápisů a uživatelských obrazovek.

Funkce *void floatToStr()* převádí desetinné číslo typu *float* na textový řetězec. Vstupními parametry jsou desetinné číslo *num* typu *float* a *str* typu *char* *, což je odkaz

na textový řetězec, kam se má převedená hodnota uložit. Číslo se zaokrouhlí na jedno desetinné místo a poté je uloženo do *str*.

Funkce *void newButton()* slouží k vytvoření nového tlačítka *button_t* a jeho uložení do pole *buttons[]*. Vstupními parametry jsou zde index *i* typu *buttonId_t*, pozice tlačítka *x* a *y* typu *uint16_t*, jeho rozměry *width* a *height* opět typu *uint16_t* a *idScr* typu *idScreen_t*, což vyjadřuje obrazovku, na které se tlačítko nachází. Dále *txt* typu *char **, což je ukazatel na text, který se vypíše uprostřed tlačítka, a *nextIdScr* typu *idScreen_t*, což je obrazovka, na kterou se uživatel přepne po stisknutí daného tlačítka.

Funkce *void newCalendarButton()* vytvoří nové tlačítko v kalendáři typu *calendarBtn_t* a uloží jej do pole *calendarBtn[]*. Vstupní parametry jsou stejné jako u funkce *newButton()*, je zde však navíc přidán parametr *date* typu *uint8_t*, který určuje den v měsíci, na který dané tlačítko bude odkazovat.

Funkce *void fillButtons()* naplní pole *buttons[]* uživatelskými tlačítky typu *button_t*. Využívá se zde funkce *newButton()*, která je opakovaně volána s různými vstupními parametry pro každé nové tlačítko. Jako indexy pole *i* a uživatelské obrazovky *idScr* a *nextIdScr* jsou zadávány enumerace *buttonId_t* a *idScreen_t*.

Funkce *void fillCalendarButtons()* naplní pole *calendarBtn[]* strukturami kalendářních tlačítek typu *calendarBtn_t*. K vytvoření těchto tlačítek je využita funkce *newCalendarButton()*, která je volána v cyklu, kde se na základě čísla průchodu pokaždé dosadí jiné vstupní parametry. Vznikne tak 31 tlačítek pro maximální možný počet dní v měsíci, které při vykreslení na obrazovce kalendáře vytvoří tabulku.

Funkce *void drawButtons()* slouží k vykreslení tlačítek a dalších ovládacích prvků na displej. Nejprve se testuje, jaká je aktuální obrazovka. Pokud se uživatel nachází na úvodní obrazovce, vypíší se pouze statické texty. Pokud se jedná o jakoukoliv jinou obrazovku, projde se pole uživatelských tlačítek *buttons[]* a u požadovaných se dle jejich parametrů vykreslí rámeček a vypíše text uprostřed. Dále se projde pole statických textů *labels[]*, kde se opět hledají takové prvky, které se mají vypsat na dané obrazovce. Pokud se jedná o obrazovku kalendáře, projde se navíc i pole kalendářních tlačítek *calendarBtn[]*. Zde se vykreslí tabulka tlačítek pro aktuální měsíc (je zde samozřejmě ošetřen počet dní

v měsíci, který se počítá znovu pro každý měsíc pomocí funkce *cntCalendarDays()*), pomocí tlustšího rámečku se zvýrazní aktuální den a u dní, pro která již bylo uživatelem provedeno nějaké nastavení, se vedle čísla dne vykreslí hvězdička.

Funkce *void dispTexts()* vypíše dynamické (proměnné) texty na displej. Ve funkci se nachází velký *switch*, který testuje obrazovku, na které se uživatel právě nachází. Na základě toho se převedou hodnoty určitých proměnných na textové řetězce (pokud je to potřeba), které jsou následně vypsány na určité souřadnice.

Funkce *void btnFunction()* vykoná nějakou akci při stisku uživatelského tlačítka typu *button_t*. Vstupním parametrem je *index* typu *uint8_t*, který udává index daného tlačítka v poli *buttons[]*. Opět se zde nachází velký *switch*, který podle indexu testuje, o které tlačítko se jedná. Na základě toho se vykoná určitá operace.

Funkce *void touchButtonTest()* porovnává souřadnice dotyku na displeji se souřadnicemi a rozměry tlačítek na určité obrazovce. Nejprve se zjistí, zda se uživatel nachází na úvodní obrazovce – pokud ano, nezáleží na tom, kde byl dotyk zaznamenán, obrazovka se automaticky překreslí na hlavní menu. Pokud se uživatel nachází na jiné obrazovce, nejprve se otestuje, zda se jedná o kalendář. Projde se pole kalendářních tlačítek *calendarBtn[]* a pokud je zjištěn stisk některého z nich, uloží se datum (aktuální měsíc a rok z kalendáře, den z parametrů tlačítka) do pomocných proměnných, se kterými se pak bude dále pracovat. Přepne se uživatelská obrazovka a další průběh funkce se ukončí příkazem *return* (další testování již není potřeba). Pokud se uživatel nenachází na úvodní obrazovce ani v kalendáři, projde se pole uživatelských tlačítek *buttons[]*. Pokud se tlačítko nachází na aktuální obrazovce a je zjištěn jeho stisk, zavolá se funkce *btnFunction()* (vstupní parametr je index daného tlačítka), díky čemuž se vykoná nějaká akce, a přepne se uživatelská obrazovka.

5.2.6 Soubor main.c

Ve funkci *main()* nejprve proběhnou veškeré počáteční inicializační kroky – např. Inicializace A/D převodníku, časovače TIM4, RTC a dále nastavení všech použitých proměnných do výchozího stavu.

V nekonečné smyčce se odstartuje AD konverze teploty a získá se aktuální čas z RTC. Poté se na základě těchto údajů pomocí funkce *compDate()* porovná, jaký program má být v daném čase nastaven dle záznamů v kalendáři, a pomocí funkce *compTime()* se určí, jaké je aktuální časové pásmo (a tedy jaká teplota se má v místnosti udržovat). Pomocí funkce *compTemp()* se poté vyhodnotí stav natápění a funkce *deleteOldCalendarPrg()* zkontroluje a případně vymaže staré záznamy v kalendáři.

Protože na úvodní obrazovce je nutné přepisovat údaje na aktuální (měřená teplota, čas atd.), kontroluje se, zda časovač TIM4 již napočítal čas 1 sekundu. Pokud ano a uživatel se právě nachází na úvodní obrazovce, údaje na ní jsou přepsány.

Dále se testuje stav dotyku na displeji. Pokud je detekován, čeká se tak dlouho, dokud ho uživatel znovu uvolní – to pro případ, že by změnil své rozhodnutí, co má zrovna v plánu, a dotyk by uvolnil někde na jiném tlačítku. Zjištěné souřadnice se pak otestují funkcí *touchButtonTest()*, která rozhodne, zda bylo stisknuto tlačítko, případně jaké.

Závěr

Práce se zabývala návrhem aplikace inteligentního termostatu. V první části byla provedena rešerše dostupných teplotních senzorů a na jejím základě bylo vybráno konkrétní čidlo pro realizaci měření teploty. Dále bylo navrženo uživatelské rozhraní a hardwarová struktura celého zařízení. V návrhu byla realizována pouze simulace spínání akčního členu pomocí dvoustavové regulace pomocí rozsvěcení LED diody – v reálu by bylo nutné připojení nějakého skutečného spínacího prvku, který by odpovídal parametrům topného systému (např. optotriak). Nakonec byl implementován aplikační firmware termostatu. Tím byly splněny veškeré body zadání práce.

Hardwarové uspořádání zařízení bylo navrženo s ohledem na budoucí případné rozšíření jeho dalších funkcí – použitý vývojový kit umožňuje např. ukládání libovolných parametrů, ukládání teplot v průběhu času a jejich následné zobrazení na displeji, ovládání a monitorování pomocí vzdáleného přístupu, případě připojení dalších teplotních čidel umístěných v jiných místnostech (či dokonce venku). Dalších možných rozšíření by mohla být celá řada.

Seznam literatury a informačních zdrojů

[1] ĎAĎO, Stanislav a KREIDL, Marcel. *Senzory a měřící obvody*. 1. vyd. Praha: ČVUT, 1996. 315 s. ISBN 80-01-01500-9.

[2] KUČEROVÁ, Eva. *Elektrotechnické materiály.* 1. vyd. Plzeň: Západočeská univerzita, 2002. 174 s. ISBN 80-7082-940-0.

[3] TÖLG, Tomáš a kol. *Fyzikální praktikum*. 5. přeprac. vyd. Plzeň: Západočeská univerzita, 2002. 184 s. ISBN 80-7082-851-X.

[4] REICHL, Jaroslav a VŠETIČKA, Martin. *Encyklopedie fyziky: Termoelektrické články* [online]. [cit. 2018-06-05]. Dostupné z: http://fyzika.jreichl.com/main.article/view/909-termoelektricke-clanky

[5] ADÁMEK, Martin. *Odporové senzory teploty* [online]. [cit. 2018-06-05]. Dostupné z: http://www.umel.feec.vutbr.cz/~adamek/uceb/DATA/s_3_2_4.htm

[6] ŠPRINGL, Vít. *Měření teploty – polovodičové odporové senzory* [online]. [cit. 2018-06-05]. Dostupné z: https://vyvoj.hw.cz/teorie-a-praxe/dokumentace/mereni-teploty-polovodicove-odporove-senzory-teploty.html

[7] PINKER, Jiří. *Sériové vstupní a výstupní obvody.* Materiály k předmětu KAE/MPP, rok 2018.

[8] MALÝ, Martin. *Sběrnice 1-Wire* [online]. [cit. 2018-06-05]. Dostupné z: https://vyvoj.hw.cz/navrh-obvodu/rozhrani/sbernice-1-wiretm.html

[9] TIŠNOVSKÝ, Pavel. Komunikace po sériové sběrnici I2C [online]. [cit. 2018-06-05]. Dostupné z: https://www.root.cz/clanky/komunikace-po-seriove-sbernici-isup2supc/

[10] Datasheet NTC: NTC MF52 A2 103 F 3380 [online]. [cit. 2018-06-05]. Dostupné z: https://www.tme.eu/cz/Document/7ce1f75ee09bde1018469502c704066e/NTCM-HP-10K-1.pdf

[11] *LM35 Datasheet: LM35 Precision Centigrade Temperature Sensors* [online]. Texas Instruments, 2017 [cit. 2018-06-05]. Dostupné z: http://www.ti.com/lit/ds/symlink/lm35.pdf

[12] *Datasheet DS18B20: Programmable Resolution 1-Wire Digital Thermometer* [online]. Maxim Integrated Products, 2015 [cit. 2018-06-05]. Dostupné z: https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

[13] *Datasheet TMP275-Q1: TMP275-Q1 Automotive Grade ±0.75°C Temperature Sensor with I 2 C and SMBus Interface in Industry-Standard LM75 Form Factor and Pinout* [online]. Texas Instruments, 2017 [cit. 2018-06-05]. Dostupné z: http://www.ti.com/lit/ds/sbos760b/sbos760b.pdf

[14] *Datasheet LMT01: LMT01 0.5°C Accurate 2-Pin Digital Output Temperature Sensor With Pulse Count Interface* [online]. Texas Instruments, 2017 [cit. 2018-06-05]. Dostupné z: http://www.ti.com/lit/ds/symlink/lmt01.pdf

[15] *Datasheet STM32F469xx* [online]. [cit. 2018-06-05]. Dostupné z: http://www.st.com/resource/en/datasheet/stm32f469ni.pdf

[16] *Reference Manual RM0368* [online]. [cit. 2018-06-05]. Dostupné z: http://www.st.com/resource/en/reference_manual/dm00127514.pdf

Přílohy

Příloha A – Blokové schéma mikrokontroléru