Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tomaž Tomažič

Avtomatizacija ogrevanja

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Patricio Bulić

Ljubljana 2013

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00128/2013
Datum: 11.04.2013

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **TOMAŽ TOMAŽIČ**

Naslov: **AVTOMATIZACIJA OGREVANJA**
HEATING AUTOMATION

Vrsta naloge: Diplomsko delo univerzitetnega študija prve stopnje

Tematika naloge:

Načrtujte avtomatizacijo ogrevanja v večetažnih stanovanjskih hišah. Krmilna elektronika naj bo zgrajena okoli mikrokrmilnika STM32F407 z jedrom ARM Cortex-M4. Zaradi potreb po nadzoru etažnih temperatur in krmiljenju črpalk v realnem času, uporabite realno-časovni operacijski sistem FreeRTOS. Sistem naj omogoča interakcijo z uporabnikom preko zalona na dotik.

Mentor:
izr. prof. dr.Patricio Bulić

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:
prof. dr. Nikolaj Zimic

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Tomaž Tomažič, z vpisno številko **63100281**, sem avtor diplomskega dela z naslovom:

Avtomatizacija ogrevanja

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom izr. prof. dr. Patricia Bulića,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 11. avgust 2013 Podpis avtorja:

Zahvaljujem se mentorju izr. prof. dr. Patriciu Buliću za pomoč in vodenje pri opravljanju diplomskega dela. Zahvaljujem se tudi ing. Vojtěch Vignerju za dovoljenje uporabe njegovih knjižnic. Prav tako se zahvaljujem Maši Gabrijel za grafično podporo uporabniškega vmesnika.
Posebna zahvala gre mojim staršem, še posebej pa očetu ki mi je dal idejo za projekt in mi vse skozi pomagal in me podpiral.

Kazalo

Povzetek

V diplomskem delu je predstavljena uporaba in delovanje vhodno izhodnih naprav skupaj z mikrokrmilnikom za avtomatizacijo ogrevanja. Cilj diplomske naloge je narediti kvalitetno krmiljenje sistema ogrevanja tri etažne hiše in s tem povečati izkoristek ogrevalnih naprav in znižanje stroškov ogrevanja. V sistemu je potrebno krmiliti toplotno črpalko, peč, črpalko za bojler, dve črpalki talnega ogrevanja in dve črpalki radiatorskega ogrevanja. Izbrana je bila razvojna plošča `stm32f4-discovery`, na katero je priklopljenih pet temperaturnih senzorjev, na dotik občutljiv zaslon LCD, osem relejev in štirje termostati. Zasnovana je bila aplikacija s preprostim uporabniškim vmesnikom, ki omogoča avtomatsko in ročno vklapljanje ali izklapljane naprav z uporabo operacijskega sistema FreeRTOS.

Ključne besede:

centralno ogrevanje, `stm32f4-discovery`, mikrokrmilnik, temperaturni senzor DS18B20, FreeRTOS,

Abstract

This degree paper presents usage and operation of peripheral devices with microcontroller for heating automation. The main goal is to make a quality system control for heating three house floors and with that, increase efficiency of heating devices and lower heating expenses. Heat pump, furnace, boiler pump, two floor-heating pumps and two radiator pumps need to be controlled by this system. For work, we have chosen a development kit `stm32f4-discovery` with five temperature sensors, LCD display with touch screen, eight relays and four thermostats. An application with a simple user interface has also been designed. It enables turning devices on and off automatically or non-automatically with help from FreeRTOS operating system.

Key words:

central heating, `stm32f4-discovery`, microcontroller, temperature sensor DS18B20, FreeRTOS

Seznam uporabljenih kratic

- **LED** - Light Emitting Diode
- **TFT** - Thin Film Transistor
- **LCD** - Liquid Crystal Display
- **I2C** - Inter Integrated Circuit
- **CRC** - Cyclic Redundancy Check
- **ARM** - Advanced RISC Machine
- **MAC** - Media Access Control address
- **RTOS** - Real Time Operating System
- **GCC** - GNU Compiler Collection
- **CDT** - C/C++ Development Tooling
- **OCD** - On Chip Debbuger

Poglavje 1

Uvod

Že pred začetkom mojega študija, je bila očetova želja, da bi avtomatizirali ogrevanje pri nas doma. Zaradi te njegove in moje želje po takšnem praktičnem znanju, smo se odločili za realizacijo projekta in izdelavo tega diplomskega dela.

Cilji dela so:

- Spoznati se z delovanjem vgrajenih sistemov in pisanjem programov za mikrokrmilnike.
- Narediti EKO ogrevanje.
- Narediti kvalitetno in poceni krmiljenje za tri etažno hišo.
- Zmanjšati porabo energije.
- Povečati izkoristek naprav.
- Znižati stroške ogrevanja.

V diplomskem delu smo najprej utemeljili izbiro strojne opreme, s katero smo lahko avtomatizirali ogrevanje, nato pa še njeno programsko realizacijo. Izbirali smo predvsem opremo, ki je bila cenovno najugodnejša in je v polni meri zadostovala za realizacijo projekta. Rešitve, ki so že na trgu, so najmanj 10-krat dražje kot vsa oprema, ki smo jo za projekt uporabljali mi. Poleg

tega jih ni možno tako prilagoditi našim željam [13, 22]. Problem obstoječe rešitve, je ročno vklapljanje in izklapljane toplotne črpalke ali peči glede na zunanjo temperaturo. Drugi problem je, ker je toplotna črpalka starejša in nima inverterskega kompresorja, ter jo zaradi prevelikih pritiskov varnostni senzorji izključijo. Obtočna črpalka pa je še vedno vključena in po nepotrebnem izrablja energijo. Ker želimo povečati izkoristek in s tem pocenitev ogrevanja, potrebujemo veliko pogojev, ki jih z relejsko logiko ni enostavno doseči. Projekt smo izvedli s pomočjo razvojne plošče `stm32f4-discovery`. Nanjo je bilo potrebno priključiti vse zunanje naprave, ki smo jih potrebovali. Preučiti smo morali tudi njihovo delovanje in programsko napisati opravila za njihovo uporabo.

Poglavje 2

Delovanje in opis orodij

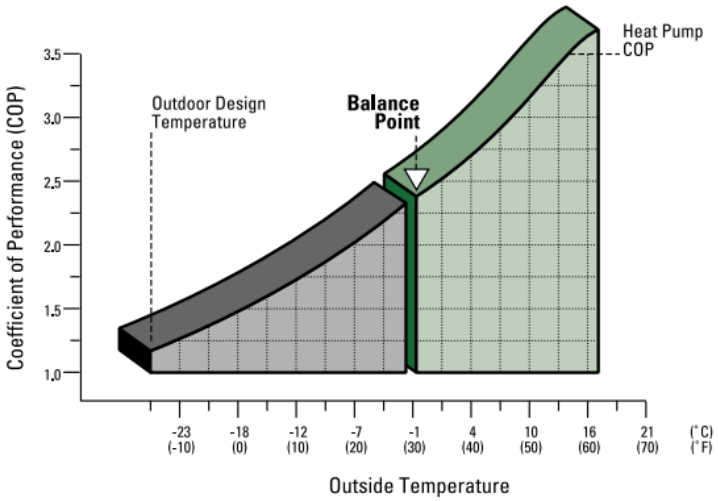
2.1 Opis sistema

Komponente ki sestavljajo sistem za ogrevanje so: peč, bojler, toplotna črpalka, štirje termostati in pet vodnih črpalk. Dosedanja rešitev ogrevanja uporablja samo relejsko logiko, s katero ni mogoče doseči optimalne izrabe energije, ki si jo želimo [20]. V sistemu sta torej dva vira energije, kurilna peč in toplotna črpalka.

Toplotna črpalka je kompresorska. Ogrevanje je z njo cenejše, če jo uporabljamo v temperaturah nekje do 0 °C [21]. Kurilna peč pa je na nafto. Njena poraba energije je veliko večja, vendar tudi učinkovitejša. Torej želimo uporabo toplotne črpalke kjer je to mogoče in smiselno, v nasprotnem pa uporabimo kurilno peč oziroma oboje istočasno.

Dve črpalki sta namenjeni za talno ogrevanje v pritličju in v nadstropju, dve pa za radiatorsko ogrevanje v kopalnici in mansardi. Za ustrezen pregled nad dogajanjem v sistemu smo uporabili pet temperaturnih senzorjev: zunanji, sistemski ki je na ceveh med pečjo in bojlerjem, senzor na izmenjevalcu, v peči in v bojlerju. Z releji vklapljamo in izklapljamo vse vodne črpalke, peč, toplotno črpalko in njen reset signal.

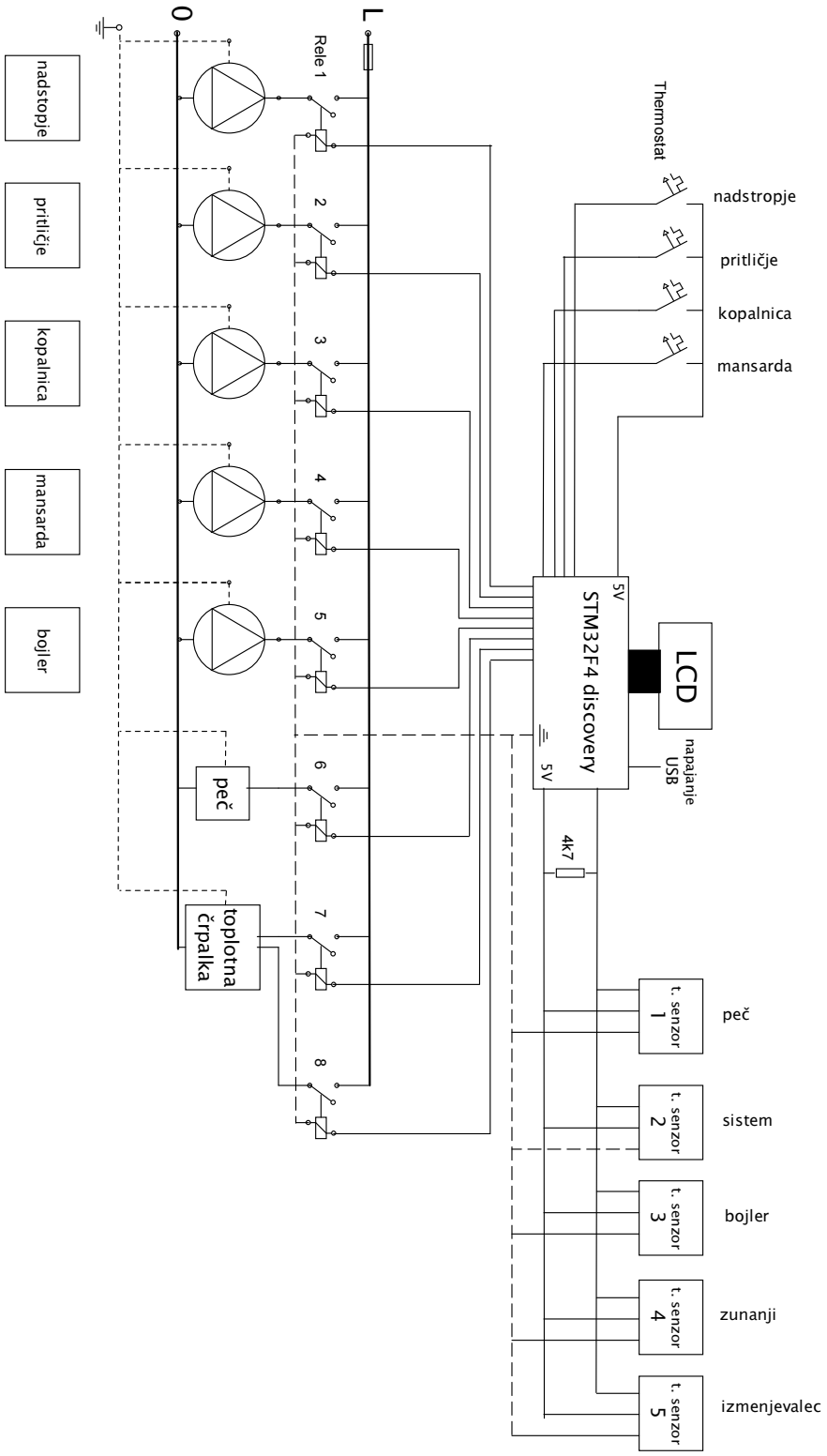
Slika 2.1: Toplotna črpalka.



Slika 2.2: Karakteristika tipične kompresorske toplotne črpalke. (vir: [21])

2.2 Naloge sistema

Glavna naloga sistema je, da zagotavlja dovoljšno količino tople vode v bojlerju, kar se da optimalno. Za potrebe naše hiše je dovolj, če drži to minimalno temperaturo na 45 °C. Sistem mora tudi vklapljati črpalke glede na ukaze iz termostatov. Ko se sistem segreje na dovoljšno temperaturo, se peč ali toplotna črpalka ustavita, v ceveh pa imamo še veliko tople vode. Ker je pozimi pri nas veliko burje, se ta ohladi in pride do tako imenovanega vleka dimnika. Obstoječe rešitve takšnih sistemov običajno ščitijo peč in v njej ohranjajo neko minimalno temperaturo. Vendar prav zaradi tega efekta in ker je naša peč nizko toplotna, to ne pride v poštev. Če je pozimi v sistemu topla voda, katera se nikjer ne uporablja, jo v tem primeru izkoristimo za talno gretje v pritličju, in hišo s tem ohranjamo toplejšo. Velikokrat pride tudi do tega, da ja v ceveh toplejša voda kot v bojlerju. Takrat vedno želimo, da se vključi črpalka za bojler. Sistem mora tudi zagotavljati varno delovanje, na primer če se peč preveč segreje, jo izklopi. Potrebno je tudi predvideti delovanje v primeru, če kateri od temperaturnih senzorjev odpove. Zavedamo se legionela bakterije, ki lahko nastopi v bojlerju. Zaradi velikega pretoka



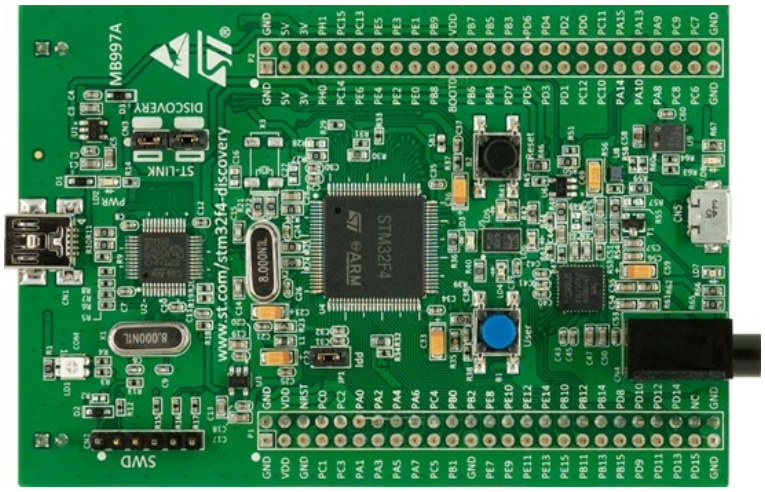
Slika 2.3: Celotna shema sistema.

vode in prihajanja do velikih temperatur v bojlerju pozimi, se nam ni zdelo potrebno reševanje tega problema.

2.3 Razvojni sistem STM32F4 discovery

Za realizacijo takšnega sistema, je potreba po uporabi mikrokrmilnika nujna. Najprej smo želeli kar uporabiti mikrokrmilnik arduino. Njegova prednost pred drugimi, je velika razširjenost in zato je na spletu tudi na voljo veliko že napisanih knjižnic za vse mogoče projekte. Po posvetovanju z mentorjem smo ugotovili da uporaba takega mikrokrmilnika ne bi bila primerna. Mikrokrmilnik Arduino je prepočasen in premalo fleksibilen za resnejše projekte. Zato smo izbrali mikrokrmilnik `STM32F407`. Kaj kmalu smo ugotovili, da je bila takšna izbira veliko bolj pametna, saj je kompleksnost projekta skozi sam razvoj stalno naraščala. Razvojna plošča `stm32f4-discovery` ima [23, 2]:

- 168 MHz takt procesorja
- 1 MB bralnega pomnilnika
- 192 KB bralno pisalnega pomnilnika
- 9 splošno namenskih vhodno izhodnih vmesnikov
- ethernet naslov MAC
- 17 časovnikov
- 3 analogno digitalne pretvornike
- enoto za računanje v plavajoči vejici
- 15 komunikacijskih vmesnikov
- vmesnik za kamero
- ...



Slika 2.4: Mikrokrmilnik `stm32f4-discovery`. (vir: [23])

Osnovana je na ARM-u Cortex-M4 arhitekturi v7E-M [1]. Napajanje je realizirano kar prek univerzalnega serijskega vodila. Na `stm32f4-discovery` najdemo že priklopljene naprave kot je razvidno iz slike 2.4. Te so: diode LED, dva gumba, senzor gibanja in konektor za zvočni vhod.

Za razvoj smo uporabljali razvojni orodji IAR Embedded Workbench for ARM, Eclipse Helios in Eclipse Kepler [16, 9]. Z orodjem IAR smo bili bolj zadovoljni zaradi hitrejšega in boljšega razhroščevanja krmilnika, vendar smo zaradi plačljive licence uporabljali le preizkusno različico. Kot brezplačno alternativo smo naložili razvojno okolje Eclipse [9]. Ta žal ni delal "iz škatle". Potrebno je bilo še namestiti zbirko orodij GCC za ARM (toolchain)[10], vtičnik Eclipse CDT, in strežnik OpenOCD [7]. Prednost Eclipsa je tudi, da nam je okolje bolj poznano, saj ga uporabljamo že več let.

2.4 FreeRTOS

Za lažje delo smo na `stm32f4-discovery` uporabljali operacijski sistem FreeRTOS [12]. Aplikacija je tako boljše organizirana in omogoča enostavnejše hkratno opravljanje več opravil. Opravila lahko dinamično ustvarjamo ali

brišemo, jim spreminjamo prioriteto in postavljamo v različna stanja. Za sinhronizacijo in komunikacijo med procesi je implementiran tudi sporočilni sistem, binarni semaforji, rekurzivni semaforji in drugo. Na voljo je veliko alternativ kot so ChibiOS/RT [5] ali Coocox [6] vendar smo uporabili FreeRTOS, ker je splošno bolj razširjen in kompatibilen tudi z drugimi arhitekturami in je bil že vključen v začetni projekt.

Za izvedbo projekta smo poleg `stm32f4-discovery` kot zunanje naprave uporabil tudi modul TFT LCD , 5 temperaturnih senzorjev in 8 relejev.

2.5 LCD

Na portalu ebay smo nabavili barvni zaslon LCD velikosti 3.2 inča s 320 x 240 pikslov. Zaslon je občutljiv na dotik, kar nam omogoča interakcijo z sistemom brez dodatnih gumbov. Ima 34 pinov prek katerih komuniciramo z njim. Prikazovalni krmilnik za zaslon LCD je SSD1289, krmilnik za zaznavanje dotikov pa je XPT2046 čigar gonilniki so kompatibilni z ADS7843 [24]. Zaslon LCD in razvojno ploščo na katero smo priključili zaslon, je naredil proizvajalec Waveshare Electronics. [25].

2.6 Temperaturni senzorji

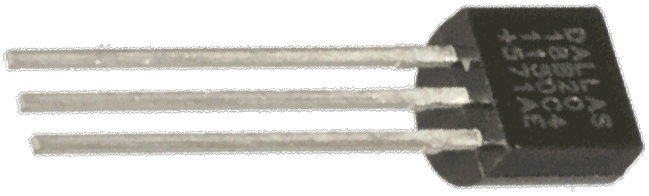
Pri nakupu temperaturnih senzorjev je na voljo zelo veliko izbire. V grobem se delijo na analogne in digitalne senzorje, načeloma pa sta nam pomembna tudi natančnost merjenja in cena. Pri analognih je prednost da so izjemno preprosti, cenovno ugodni in imajo samo dve žici. Ker je bilo na spletu za malo denarja več analognih senzorjev z ohišjem kot pa digitalnih z ohišjem, nam je ta izbira bolj ustrezala. Njihova slabost je, da potrebujemo analogno digitalni pretvornik, katerega `stm32f4-discovery` na srečo ima. Temperatura se torej meri glede na napetost na žici. Problem lahko nastane pri zelo dolgih žicah, kjer je izpostavljenost na zunanje motnje večja in napetost z dolžino žice pada. Profesor nam je svetoval izbiro digitalnega senzorja, s

Slika 2.5: LCD zaslon priključen na razvojno ploščo Open407V-D.

katerim v veliki meri slabosti analognega senzorja odpravimo. Kljub temu smo zaradi uporabe samo dveh vodnikov preizkusili tudi nekaj analognih senzorjev. Ker smo kmalu ugotovili, da bi bila njihova uporaba precej bolj nerodna, smo se odločili za digitalne senzorje. Digitalni temperaturni senzorji uporabljajo različne protokole. Najbolj razširjen je I2C. Za uporabo le teh se navadno potrebuje vsaj štiri vodnike. Ker so ti senzorji dražji, smo kot alternativo našli senzor Dallas DS18B20.

2.6.1 Senzor DS18B20

Senzor DS18B20 uporablja 1-žični protokol (1-wire). Torej lahko za svoje delovanje potrebuje samo dve žici. Ime 1-wire je zato, ker se podatki lahko prenašajo samo po eni žici in pri tem napajajo senzor. To delovanje imenujemo parazitno napajanje [17]. Na podatkovno linijo je potrebno priključiti pullup upor. Lahko uporabimo upor na krmilniku `stm32f4-discovery`, vendar je priporočljiv zunanji 4700 ohm-ski upor. Za temperature nad 80 °C specifikacije [17] priporočajo napajanje senzorja po ločenem vodniku. Sen-



Slika 2.6: Temperaturni senzor DS18B20.

zor deluje od -55 °C do +125 °C, od -10 °C do +85 °C pa na natančnosti do 0.5 °C. Ima devet bajtnih EEPROM registrov. Iz prvih dveh prebiramo temperaturo. Druga dva lahko uporabimo za alarm ali pa ga izrabimo kot navadni pomnilnik. Registri 5, 6, 7 in 8 so samo bralni, obenem ima zadnji CRC kodo ostalih registrov. Temperaturo lahko merimo z največ 12 bitno natančnostjo, vendar njeno pretvarjanje traja do 750 ms. Ker nam časovna omejitev ni ovira, smo uporabili kar privzeto, 12 bitno natančnost pretvarjanja. Iz drugega registra je prvih pet bitov uporabljenih za predznak, ostali trije pa za zgornje bite števila temperature. V prvem registru so prvi štirje biti uporabljeni za spodnje bite števila temperature, ostali pa so za vrednost, ki je za decimalno vejico. Ker se senzor dobi tudi v vodoodpornem ohišju in zaradi masovne proizvodnje po izjemno ugodni ceni, je bila izbira tega senzorja očitna. Možno je tudi brezplačno naročilo nekaj takšnih preizkusnih senzorjev brez ohišja. Za merjenje zunanje temperature, temperature peči in bojlerja nujno potrebujemo senzor v vodoodpornem ohišju. Zaradi poenostavitve dela, smo vseh pet senzorjev uporabili v ohišju.

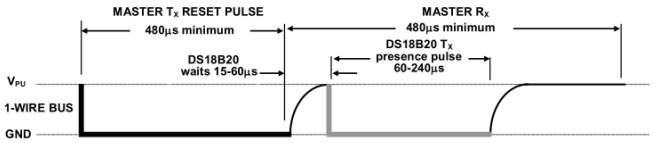
2.6.2 1-žični protokol

1-žični protokol je protokol, ki omogoča komunikacijo med napravami s podatki in napajanjem po eni žici. Protokol je naredila korporacija Dallas. Vsaka naprava v sistemu ima svoj 64 - bitni naslov. Obstaja ukaz s katerim lahko dobimo naslove vseh naprav priključenih na isto podatkovno linijo.

Slika 2.7: Temperaturni senzor DS18B20 v vodoodpornem ohišju.

Naprave so lahko priključene linearno ali v obliki zvezde, vendar se dolžina vodnika ne gleda na najdaljši krak, ampak na celotno dolžino vseh krakov zvezde skupaj. S preprostim pullup uprom, po zaviti parici kategorije 5, je dolžina omejena na 200 metrov. Dolžino je možno povečati z aktivnimi pullup upori tudi do 500 metrov, vendar so pri teh omejitvah prisotni tudi drugi dejavniki, npr. število naprav [18]. Prenos podatkov navadno poteka najprej z inicializacijo, v kateri gospodar vodila pošlje reset signal, ki traja minimalno 480 μs, na katerega sužnji odgovorijo s pulzom prisotnosti, ki traja od 60 μs do 240 μs. Tako lahko gospodar ve, da so sužnji na vodilu pripravljeni za nadaljnje ukaze. Ko gospodar prejme pulz prisotnosti, izstavi ukaz kateremu lahko sledi 64 - bitni naslov če je to potrebno. Tipični ukazi so:

- iskanje naprave, ki omogoča zaznavo števila naprav
- branje naslova, če je na vodilu samo ena naprava
- preskok naprave, ki omogoča naslavljanje vseh naprav naenkrat
- iskanje naprave z omogočenim alarmom
- ...



Slika 2.8: Časovni diagram inicializacije po 1-žičnem protokolu. (vir: [17])

Nazadnje se pošlje ukaz, ki je specifičen glede na napravo. Senzor DS18B20 ima ukaze kot so npr. pretvori temperaturo in beri registre.

2.7 Releji

Za vklapljanje in izklapljanje črpalk, peči in bojlerja smo nabavili 8 relejev s kontaktom napetosti 250 V in 10 A [8]. Vsak rele ima mirovni in delovni kontakt [14]. Vsi skupaj so na tiskanem vezju in se krmilijo z napetostjo 5 V, kar omogoča enostaven priklop in nadzor z razvojno ploščo `stm32f4-discovery`. Vsak rele ima svojo diodo LED, ki sveti, ko je rele vklopljen. Nakup relejev z diodami LED, se je izkazal kot zelo dober, ker nam je to omogočalo hitro testiranje pravilnosti delovanja aplikacije. V opisu izdelka je bilo navedeno, da deluje z originalnimi arduino ploščami, vendar je zadeva delovala tudi na `stm32f4-discovery` brez težav.

Slika 2.9: 8 relejev na enem vezju. (vir: [8])

Poglavje 3

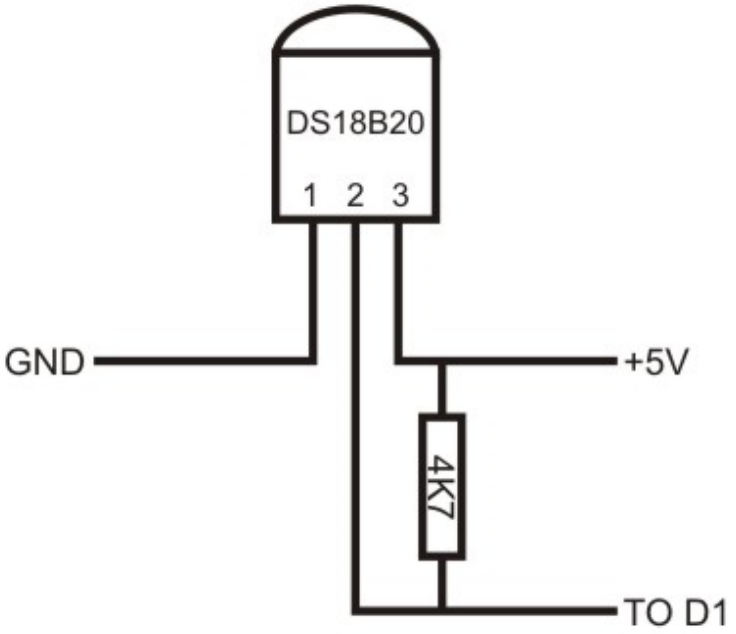
Razvoj aplikacije

3.1 Strojna oprema

Na začetku je bilo potrebno priključiti vso kupljeno opremo. Za hitrejši razvoj smo priključili zaslon LCD na razvojno ploščo. Za priključitev temperaturnih senzorjev, pa smo si pomagali z testno ploščo in sponkami. Med podatkovno in napajalno linijo smo dali 4700 ohmski upor, kot je to razvidno is slike 3.1 in je zahtevano iz specifikacije [17]. Za priklop relejev smo izrabili 8 prostih pinov od PA0 do PA7 in jih nastavili kot digitalni izhod, za prilkop termostatov pa 4 proste pine in jih nastavili kot digitalni vhod.

3.2 Programska oprema

Na začetku smo uvozili v Eclipse začetni projekt, ki nam ga je dal profesor. V projektu so že bili uvoženi gonilniki za zaslon LCD [3] in panel na dotik [4]. Skupaj z operacijskim sistemom FreeRTOS je že bil implementiran prikaz in možnost nastavljanja ure, kakor tudi ikone za sprehajanje v druge načine zaslona. Ker nam ikone v aplikaciji niso bile všeč, smo jih zamenjali z novimi. V pomoč nam je bil program Image Convert, ki je na voljo brezplačno za nekomercialne namene. Z njim smo pretvorili slike v polje znakov, v katerem je vsak piksel predstavljen v šestnajstiški obliki uint16_t. Te lahko enostavno



Slika 3.1: Primer vezave senzorja DS18B20. (vir: [19])

izrišemo na zaslon.

Takoj po zagnanem projektu smo opazili, da se narisani gumbi na zaslonu ne odzivajo pravilno. Ugotovili smo, da je potrebna kalibracija panela na dotik. V gonilnikih je že napisana metoda za pomoč pri kalibraciji. Ta na zaslon zaporedno izriše 3 križce na katere pritisnemo. S podatki kje so dejansko bili izrisani križci in kje smo mi pritisnili, se izračuna 6 koeficientov matrike. Te si shranimo in jih fiksno vkodiramo tako, da je ob naslednjem zagonu zaslon že skalibriran.

Za razvojni kit `stm32f4-discovery` žal še ni tako veliko napisanih knjižnic kot npr. za Arduino, zato je bilo zelo težko najti implementacijo 1-žičnega protokola. Na spletni strani github.com smo našli verzijo, s katero nam zadeva ni delovala. Kontaktirali smo češkega avtorja kode [26], ki nam je poslal delujočo novejšo verzijo protokola in knjižnico za DS1820 senzor. DS1820 senzor je zelo podoben senzorju DS18B20. Potrebno je bilo le spremeniti branje temperature, saj se biti ki jo predstavljajo drugače shranjujejo v registre

senzorja. Za lažje nadaljnjo delo in manjšo porabo pomnilnika, si temperaturo shranimo kot celoštevilsko desetkratno vrednost in tako ohranjamo eno decimalno mesto, ki ga upoštevamo pri izpisu, brez predstavitve v plavajoči vejici. Senzorje ki smo jih dobili, nimajo nikjer vpisanega ali vgraviranega 64 - bitnega naslova na ohišju. Priključili smo po en sam senzor na vodilo in pognali metodo `OW_SearchNext()`. Tako smo dobili naslov naprave in si ga fiksno vkodirali. Postopek smo ponovili za vsak senzor. Nujno moramo namreč vedeti kateri senzor kaj meri.

```
1  int iBinaryToIntTemperature(uint8_t *iSPad) {
2
3    //most and least significant bits
4    uint8_t lsb = iSPad[0];
5    uint8_t msb = iSPad[1];
6
7    //integer part
8    int temperature = ((lsb >> 4) | ((msb << 4) )) &
         0b01111111;
9
10   //negative part
11   if ( (( msb >> 3 ) & 0b1 ) == 1 )
12         temperature = (~temperature & 0b01111111) * -1;
13
14   //avoid decimal
15   temperature *= 10;
16
17   /* "Decimal" part */
18   lsb &= 0xF;
19   temperature += (lsb == 0) ? 0 : (( lsb ) * 10 / 16) ;
20
21   return temperature;
22 }
```

Koda 3.1: Metoda ki pretvori temperaturo iz prebrane binarne vrednosti v

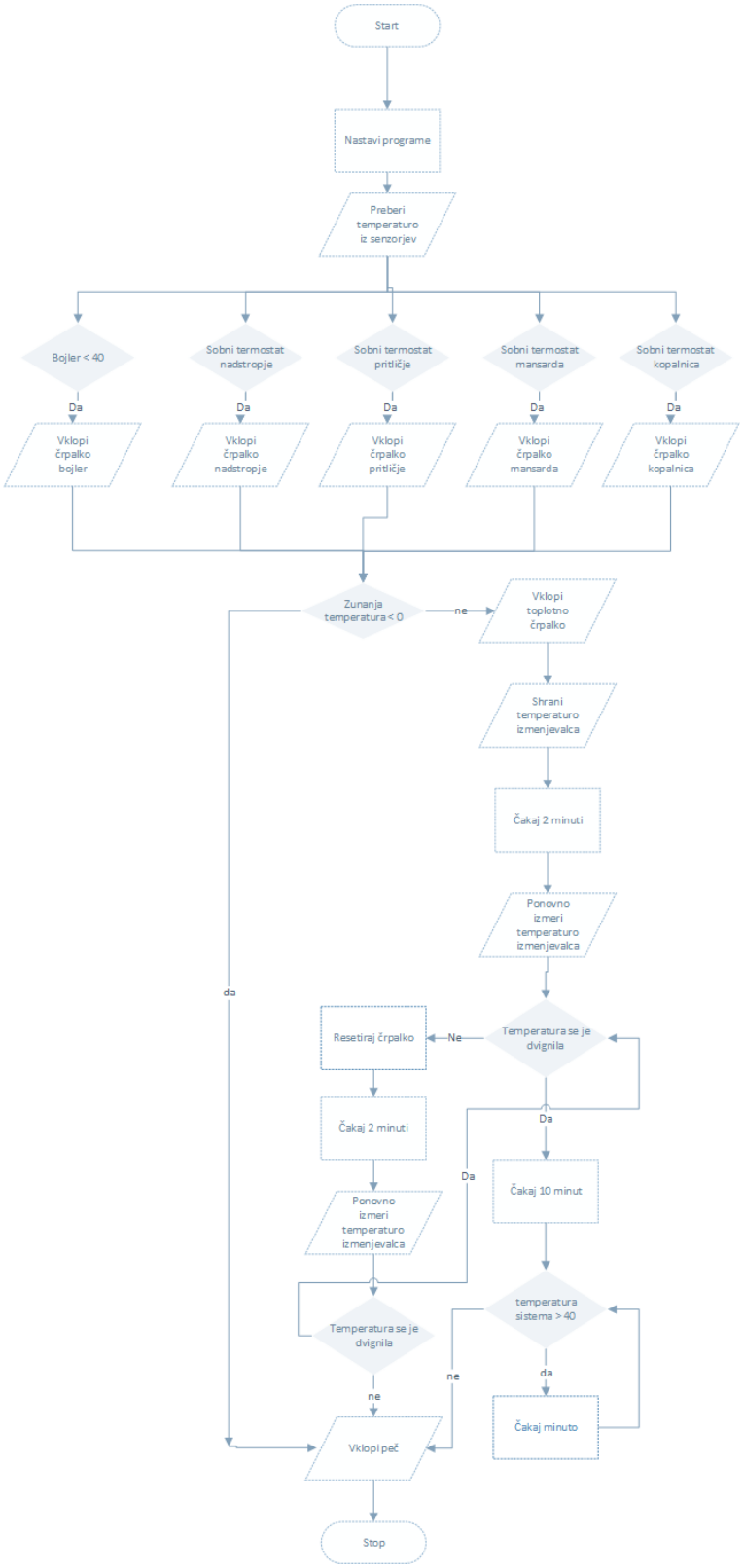
celo številsko.

Začeli smo z risanjem postopkovnih diagramov. Z dodajanjem novih funckionalnosti in spreminjanjem načina delovanja, smo v roku enega meseca morali program večkrat v celoti prekonstruirati, preden smo sploh začeli kodirati. Šele ko smo bili z njim zadovoljni, smo začeli z implementacijo.

Celoten program smo razdelili na 7 opravil, ki se navzven izvajajo istočasno. Ta opravila so:

- spreminjanje ure
- upodabljanje na zaslon
- branje koordinat pritiska na panel
- branje temperature
- vklapljanje in izklapljanje relejev
- sistemsko opravilo gretja
- osveževanje zaslona

Vsa opravila imajo enako prioriteto in se izvajajo za določeno časovno rezino. S tem preprečimo stradanje določenega opravila. Opravilo začne svoje izvajanje ob začetku časovne rezine in se izvaja do njenega konca, ko preide v blokirano ali pa v suspendirano stanje. Čas časovne rezine nastavimo s konstanto `configTICK_RATE_HZ` [11]. Ob kreiranju opravila določimo velikost sklada, ki ga bo imel na voljo. Določitev velikosti sklada ni enostavna. Priporočena je uporaba minimalne velikosti, kot jo ima opravilo v praznem teku (idle task) in je določena s konstanto `configMINIMAL_STACK_SIZE` [11]. S tako nastavitvijo smo imeli precej težav, ko smo začeli v opravilih uporabljati operacije za delo z nizi. Aplikacija je postala neodzivna. Izkazalo se je, da funkcija `sprintf()`, ki omogoča združevanja nizov, potrebuje sklad večje velikosti. Za takšne primere se lahko izračuna točno porabo pomnilnika ki jo potrebujemo, vendar se v praksi kar določi neko razumno vrednost.



Slika 3.2: Postopkovni diagram sistemskega opravila.

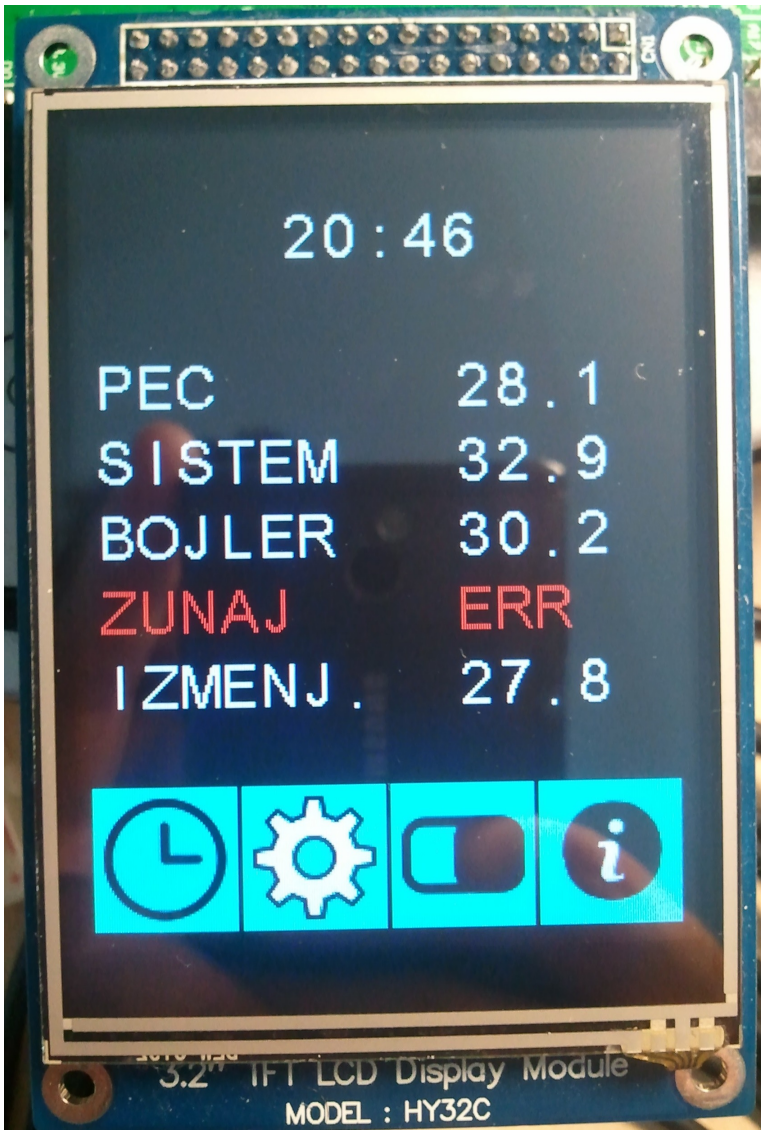
Na domačem zaslonu imamo prikaz ure, izpis temperaturnih senzorjev z izmerjeno temperaturo in preprost meni s štirimi gumbi. Uro je mogoče nastaviti z pritiskom na prvi gumb, kjer se nam odpre nov zaslon za nastavitev ure. Lahko povečamo ali zmanjšamo ure ali minute in novi čas shranimo ali zavržemo. Za dogodek ob pritisku na gumb skrbi opravilo `vTouchPanelTask`. To opravilo v zanki preverja vsake 300 ms, če se je zgodil dogodek dotika na zaslon. Glede na pozicijo v meniju in koordinate zaslona, lahko ugotovimo kateri gumb je uporabnik želel pritisniti. Ker so gumbi večinoma na enakem mestu, smo tudi v podmenijih naredili metodo `iButton`, za lažje ugotavljanje za kateri gumb želimo izvesti neko akcijo. V naši aplikaciji je možnih 5 stanj zaslona:

- `LCD_NORMAL`, ki predstavlja domači zaslon.
- `LCD_SETTIME`, ki predstavlja nastavitve ure.
- `LCD_SETTINGS`, ki predstavlja splošne nastavitve.
- `LCD_RELAY`, ki predstavlja prikaz in nastavitev stanja relejev.
- `LCD_INFO`, ki predstavlja informacije o aplikaciji.

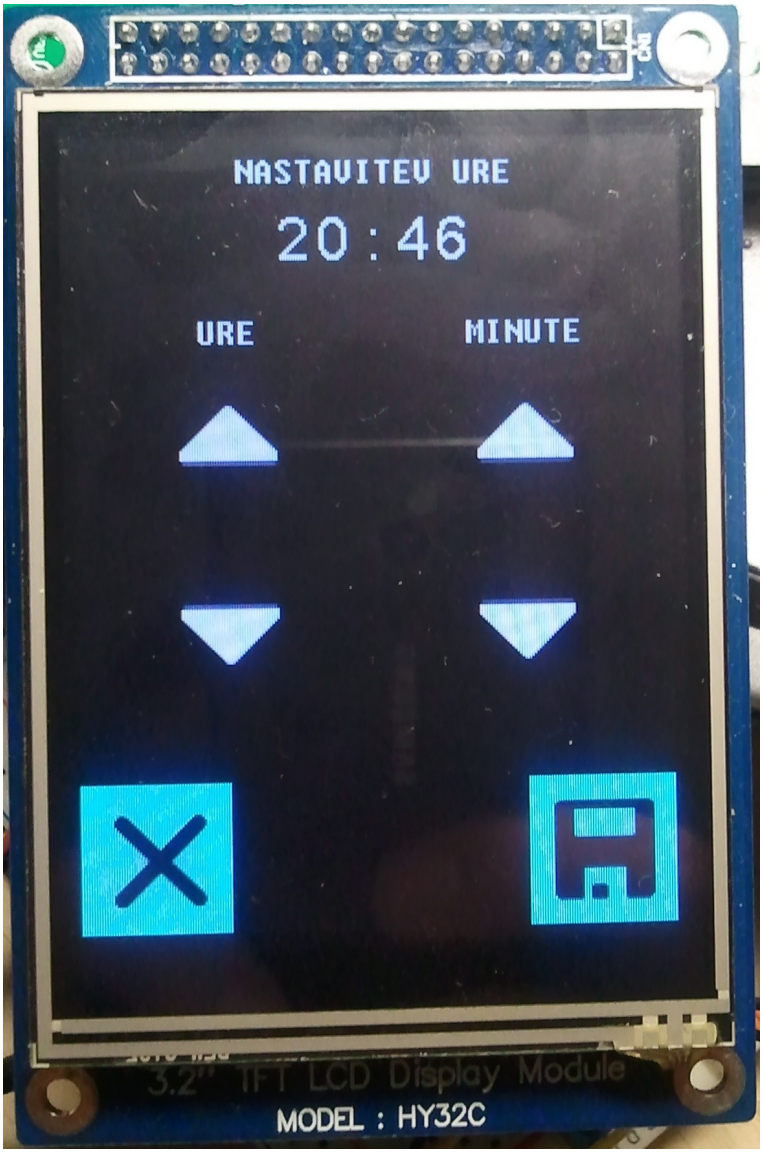
```
1  void ScreenNormal (xLCDMessage *xpMessage) {
2
3  vInitIcons();
4
5  switch (xpMessage->xMsgType) {
6  case TEMP:
7  LCD_CharSize(24);
8  int i;
9  char string[15];
10 char text[MAX_TEMP_SESNORS][10] = {
11   "PEC     ",
12   "SISTEM  ",
13   "BOILER  ",
```

```
14   "ZUNAJ   ",
15   "IZMENJ. "
16 };
17 uint8_t temperatureConst[MAX_TEMP_SESNORS] = {
18   TFURNACE, TSYSTEM, TBOILER, TOUTSIDE, TEXCHANGER
19 };
20
21 for (i=0; i<MAX_TEMP_SESNORS; i++){
22
23   //get temperature
24   int temp = iTemperature[temperatureConst[i]];
25
26   //red
27   if(temp == DS1820_TEMP_ERROR){
28     sprintf(string, "%sERR ", text[i]);
29     LCD_SetTextColor(RED);
30     LCD_StringLine(5, 90 + i*30, string);
31     LCD_SetTextColor(WHITE);
32     continue;
33   }
34
35   //white
36   sprintf(string, "%s%d.%d", text[i], iTemperature[i]/10,
37     abs(iTemperature[i]-iTemperature[i]/10*10));
38
39   //draw
40   LCD_StringLine(5, 90 + i*30, string);
41 }
42
43 break;
44 }}
```

Koda 3.2: Izvleček kode, ki izpiše temperaturo na LCD zaslonu.



Slika 3.3: Domači zaslon v primeru odklopljenega temperaturnega senzorja v bojlerju.

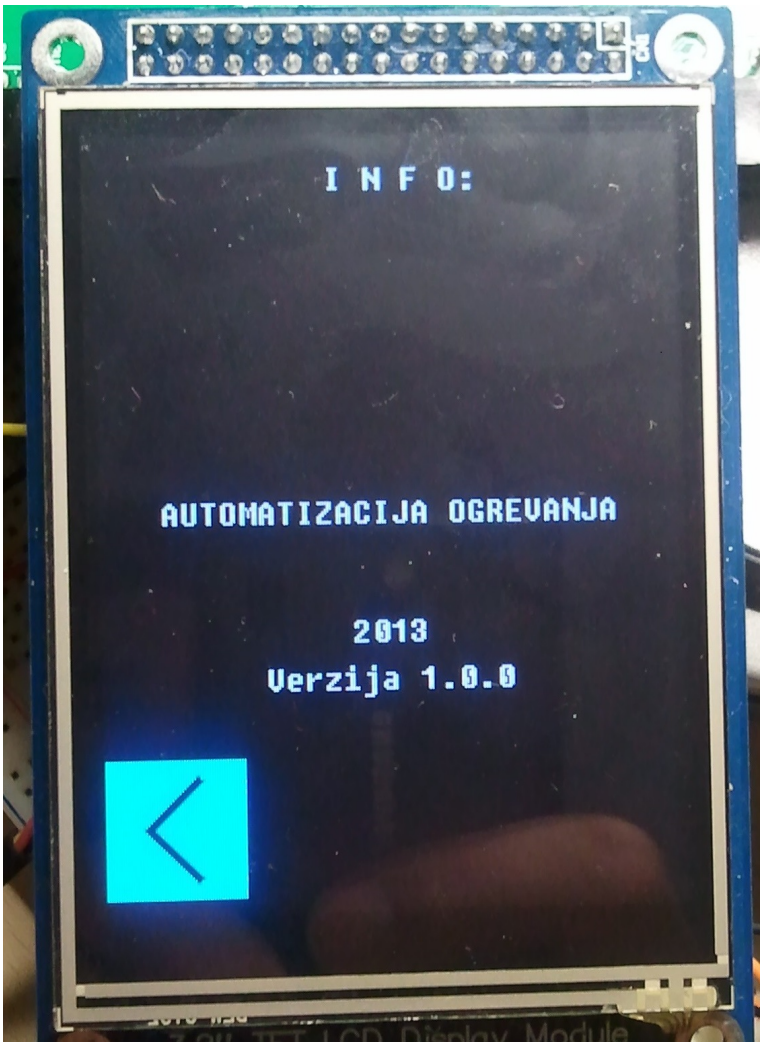


Slika 3.4: Zaslon `LCD_SETTIME` v katerem je možno nastavljanje ure.

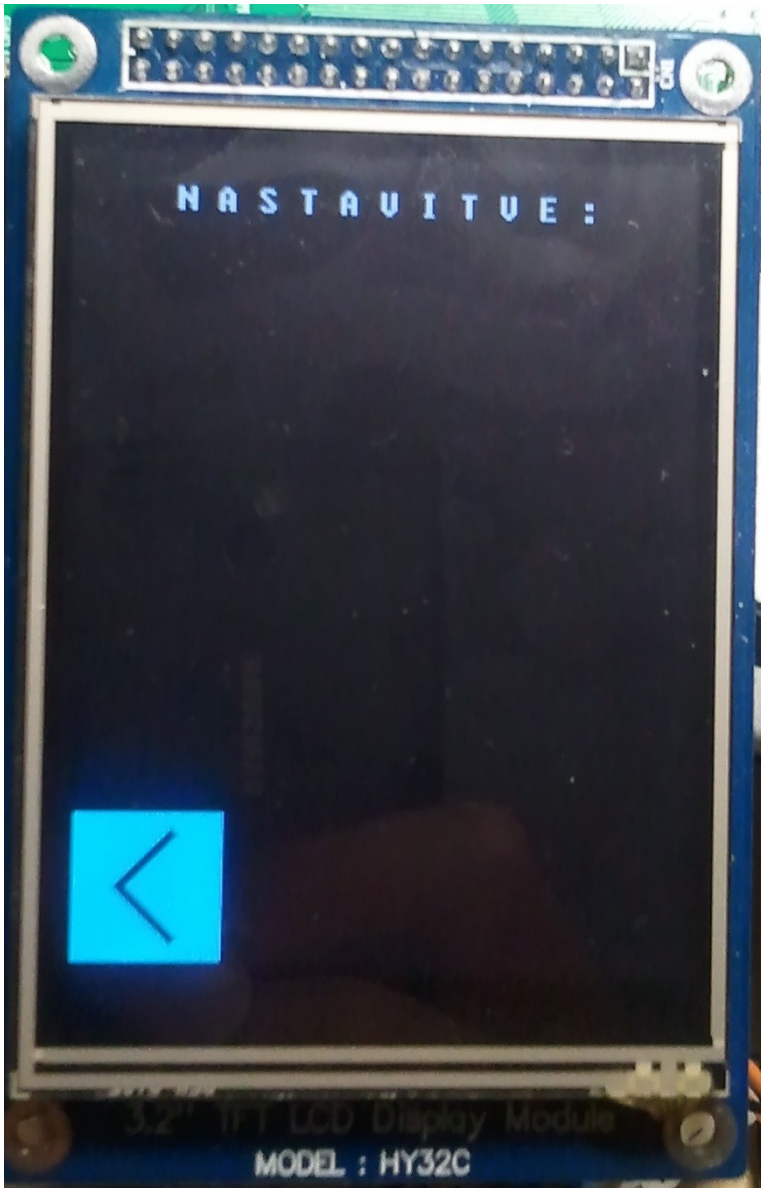
`LCD_SETTINGS` še ni uporaben in je narejen le za lažjo nadgradnjo aplikacije v prihodnosti. Aplikaciji smo dodali ohranjevalnik zaslona, ki izbriše vse kar je trenutno prikazano na njem. Ohranjevalnik zaslona se vključi po eni minuti neaktivenga pritiskanja na zaslon in če vsi temperaturni senzorji normalno delujejo. Tako lažje zaznamo če je prišlo do napake, že ob samem vstopu v prostor kjer imamo `stm32f4-discovery` saj vidimo vključen zaslon. Za posodabljanje temperature na domačem zaslonu skrbi opravilo `vTemperatureTask()`. Opravilo vsake 5 sekund prebere temperaturo iz senzorjev in obvesti drugo opravilo `vLCDTask()` za njen izris na zaslon, z vstavljanjem sporočila v vrsto z metodo `xQueueSend()`. V primeru izpada senzorja ali druge napake, se za njeno lažje prepoznavanje, namesto temperature, izpiše ERR v rdeči barvi.

V stanju zaslona `LCD_RELAY` imamo izpisanih vseh 8 relejev in 4 gumbe. Za vsak rele lahko vidimo ali je vklopljen ali izklopljen in ali je voden avtomatsko ali ročno. Gumb `AUTO` je namenjen za vrnitev releja v avtomatsko vodenje. Zeleni gumb je namenjen za ročni vklop releja, rdeči pa za ročni izklop. Rele moramo najprej izbrati s pritiskom na zaslon kjer je izpisano njegovo ime, če želimo da gumbi nanj delujejo. Da vemo kateri je izbran, se njegovo ozadje obarva modro. Da vemo kateri je ročno krmiljen, pa se napis obarva rdeče namesto bele barve. Prvi gumb s puščico je namenjen za vrnitev v domači meni. Za osveževanje stanja relejev v `LCD_RELAY` zaslonu, poskrbimo s kreiranjem opravila `vRefreshTask()`. Ta vsake tri sekunde pošlje novo stanje v izris na zaslon. Ob vrnitvi na domači zaslon to opravilo izbrišemo. Ročno krmiljenje je predvsem koristno, za lažje popravljanje naprave v sistemu, npr. gorilca.

V opravilu `vConditionTask()` preverjamo vse mogoče pogoje, za vsak rele posebej. Tu se prebira tudi stanje pinov, na katere so priključeni termostati. Glede na pogoje, katerih nekaj je opisanih v poglavju 2.2, želimo krmiliti releje. Če je prenizka temperatura v bojlerju ali pa je vključen katerikoli izmed termostatov, ustvarimo opravilo `vSystemTask()`. V primeru da je zunanja temperatura višja, vključimo najprej toplotno črpalko in počakamo



Slika 3.5: Slika zaslona `LCD_INFO`

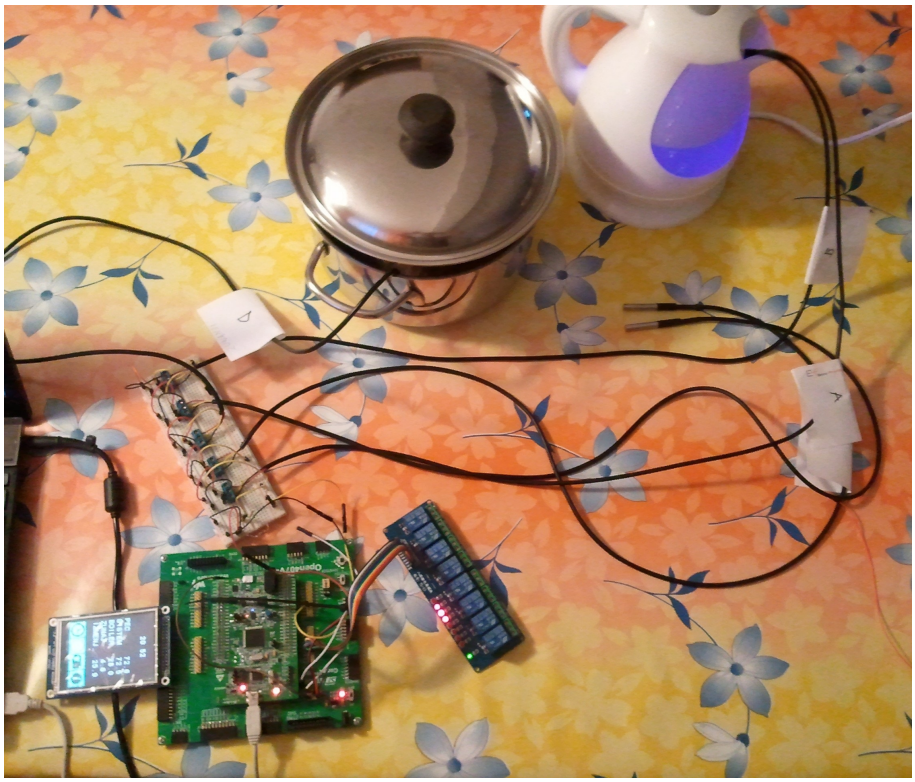


Slika 3.6: Slika zaslona `LCD_SETTINGS`

Slika 3.7: Primer zaslona `LCD_RELAY` ko je rele pritličje ročno vključen, označen je pa rele kopalnica.

2 minuti. Nato preverimo temperaturo v izmenjevalcu da ugotovimo, če toplotna črpalka deluje. Črpalka je namreč precej stara in je bila prvotno uporabljena za hlajenje. Ko smo jo predelali za gretje, smo jo napolnili z nekaj več plina kot je navedeno v specifikacijah. S tem smo pridobili večji izkoristek pri gretju, pri nižji zunanji temperaturi. Njena slabost je, da v višjih temperaturah tlačno stikalo javi napako in toplotno črpalko izklopi. Črpalka je premočna glede na odvzem, kadar je v delovanju samo bojler. Rešitev bi bila možna z frekvenčnim regulatorjem kompresorja (inverter), ki bi glede na odvzem temperature na izmenjevalcu delal z nižjimi obrati in s tem manjšo močjo. Vendar je ta rešitev draga in ker kompresor tega ne podpira bi zaribal, saj nima pravega mazanja. Zato smo uporabili en rele na katerega je priklopljen reset črpalke. Če torej ugotovimo da toplotna črpalka ne deluje, vključimo in izključimo reset. Po dveh minutah ponovno preverimo delovanje črpalke in v primeru ne delovanja vključimo peč. Rešitev z ročnim resetiranjem se je izkazala kot zelo učinkovito. Ko pride sistemsko opravilo do stanja kjer vključi peč, se suspendira. V trenutku ko se izključijo vsi pogoji ki držijo sistem, torej nizka temperatura v bojlerju ali pulz termostata, se sistemsko opravilo izbriše. V opravilu pogojev je tudi varnostni pogoj najvišje prioritete, ki v primeru prevelike temperature v peči, izključi. Vsi pogoji, ki se pojavijo skozi program, ne direktno krmilijo releja, saj bi prišlo do nepotrebnega vklapljanja ali izklapljanja in s tem tudi do okvare naprave. Zato ima vsak rele polje pogojev, ki jih postavljajo različna opravila. S tem enostavno omogočimo tudi ročni nadzor releja. Vedno moramo predvideti tudi situacijo kdaj se rele izključi. Vsak pogoj ki vključi rele ga mora tudi izključiti, kar na prvi pogled ni samoumevno. Če je vsaj en pogoj v polju postavljen, vključimo rele, v nasprotnem ga izključimo. Podobno moramo poskrbeti za sistemsko opravilo, da se ustvari samo enkrat. Primer: prvi termostat se vključi in ustvari se sistemsko opravilo. Po petih minutah se vključi drugi termostat. Ker je sistemsko opravilo že ustvarjeno, se mora normalno izvajati naprej vse dokler nista oba termostata izključena.

Sistem smo testirali v dveh posodah kot je prikazano na sliki 3.8. V eni



Slika 3.8: Testiranje sistema.

je bil led v drugi vrela voda.

Poglavje 4

Sklepne ugotovitve

Pri izvedbi projekta smo imeli veliko težav zaradi pomanjkanja znanja na področju elektrotehnike. Najprej smo morali osvojiti osnove, da smo lahko priklopili periferijo in razumeli možne napake, ki lahko nastanejo. Šele nato smo lahko začeli z programiranjem. Skozi izvedbo projekta smo imeli izkušnjo kot s pravim naročnikom. Načrt izvedbe smo morali večkrat prekonstruirati, saj so se pojavljale vedno novejše ideje. Stvari, ki so naročniku bile samoumevne, so se izkazale kot velike dodatne komplikacije v programiranju. Srečali smo se tudi s preprostim načrtovanjem uporabniškega vmesnika, ki bo uporabniku dovolj preprost, jasen in razumljiv za uporabo. Z avtomatizacijo ogrevanja zagotovimo uporabniku prijaznejši in enostavnejši način ogrevanja in boljše izkoristimo vire energije. Z večjo varčnostjo in izkoriščenostjo damo, tudi z vidika ekologije, naravi bolj prijazno rešitev.

Sistem bi lahko uporabili tudi kjer koli drugje, najbolj primerno pa bilo na področjih kjer je pozimi veliko burje. Spremeniti bi morali le sistemsko opravilo glede na potrebe uporabnika, vse ostalo bi lahko ostalo enako vključno z strojno opremo.

Zastavljene cilje diplomske naloge smo uspešno uresničili, saj smo se spoznali z delovanjem vgrajenih sistemov, pisanjem programov za mikrokrmilnike, ki za delovanje uporabljajo zaslon na dotik in temperaturne senzorje. Uspeli smo narediti program in prototip sistema, ki bo pripomogel k znižanju

stroškov ogrevanja. Za lažjo postavitev in priklop vseh zunanjih naprav na `stm32f4-discovery`, bi bilo smiselno naredili tiskano vezje in omarico v katero bi montirali elektroniko.

Sistemu bi lahko naredili veliko izboljšav. Lahko bi dodali temperaturne senzorje po hiši, kjer so trenutno termostati in bi le-te odstranili. S tem bi še bolj centralizirali nadzor ogrevanja. Dodali bi lahko možnost različnih programov kot so nočni, dnevni, počitniški, podaljšani in drugi, katere bi uporabnik izbiral na zaslonu. Na proste pine bi lahko priključili ethernet modul in dodali možnost oddaljenega spremljanja in nadzora ogrevanja prek spleta. Za to bi lahko razvili tudi aplikacije za mobilne naprave in tablične računalnike. Na zunanji pomnilnik ali server bi lahko tudi intervalno shranjevali temperaturo senzorjev in čas vklopljenih porabnikov. S tem bi imeli pregled nad zgodovino delovanja z izrisom ustreznih grafov. Nato bi lahko implementirali inteligentni sistem, ki bi glede na porabljen čas do katerega potrebuje, da segreje prostor in zunanje dejavnike, točneje vklapljal naprave v trenutnem programu ogrevanja (strojno učenje). V tem primeru bi bilo smiselno tudi dodati dodatne senzorje za vlažnost in veter, za boljše napovedovanje.

Literatura

[1] ARM, Čortex-M4 Processor"Dostopno na:
http://www.arm.com/products/processors/cortex-m/cortex-m4-processor.php 10

[2] P. Bulić, Prosojnice za predmet Vgrajeni sistemi, 2013 Dostopno na:
https://ucilnica.fri.uni-lj.si/course/view.php?id=39 9

[3] P. Bulić (2013), knjižnica za LCD Dostopno na:
https://github.com/dcrystalj/heating-automation 17

[4] P. Bulić (2013), knjižnica za panel na dotik Dostopno na:
https://github.com/dcrystalj/heating-automation 17

[5] ChibiOS/RT in brief, Dostopno na:
http://www.chibios.org/dokuwiki/doku.php 11

[6] CooCox (2011), "Free/Open ARM Cortex MCU Development Tools"Dostopno na:
http://www.coocox.org/index.html 11

[7] dcrystalj (2013), štm32f4-discovery-Eclipse"Dostopno na:
https://github.com/dcrystalj/stm32f4-discovery-Eclipse/blob/master/Nastavitev%20razvojnega%20orodja%20eclipse%20za%20STM32F4%20discovery.pdf?raw=true 10

[8] DealExtreme, "8-Channel 5V Relay Module Shield for Arduino"Dostopno na:

http://dx.com/p/8-channel-5v-relay-module-shield-for-arduino-148811 15

[9] Eclipse, "Eclipse IDE for C/C++ Developers"Dostopno na: http://www.eclipse.org/downloads/packages/eclipse-ide-cc-developers/heliossr2 10

[10] Jinyun-ye (2013), "GNU Tools for ARM Embedded Processors"Dostopno na: https://launchpad.net/gcc-arm-embedded 10

[11] Real Time Engineers Ltd, zaglavje FreeRTOSConfig.h Dostopno na: http://www.freertos.org/a00110.html 20

[12] Real Time Engineers Ltd, "About FreeRTOS"Dostopno na: http://www.freertos.org/RTOS.html 10

[13] FIRŠT, "regulacije ogrevanja, UNI TEMPGF3"Dostopno na: http://www.first.si/2012/www.first2012/PDF/KATALOG2012/Katalog2012-slo-v3-print_70.pdf 4

[14] Giorgos Lazaridis (2010), "How Relays Work"Dostopno na: http://www.pcbheaven.com/wikipages/How_Relays_Work/ 15

[15] I2C Bus "I2C-Bus: What's that?"Dostopno na: http://www.i2c-bus.org/

[16] IAR Systems, "IAR Embedded Workbench for ARM"Dostopno na: http://www.iar.com/Products/IAR-Embedded-Workbench/ARM/ 10

[17] Maxim Integrated, "DS18B20 Programmable Resolution 1-Wire Digital Thermometer"Dostopno na: http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf 12, 15, 17

[18] Maxim Integrated, "Guidelines for Reliable Long Line 1-Wire® Networks"(2008) Dostopno na: http://www.maximintegrated.com/app-notes/index.mvp/id/148 14

[19] Nathan Chantrell (2011), Building a graphical display for OpenEnergyMonitor Dostopno na:
http://nathan.chantrell.net/20111015/building-a-graphical-display-for-openenergymonitor/ 18

[20] National Control Devices, LLC, "Relay Logic"Dostopno na:
http://www.controlanything.com/Relay/Device/RELAY_LOGIC 5

[21] Natural Resources Canada's Office of Energy Efficiency, "Heating and Cooling With a Heat Pump"Dostopno na:
http://oee.nrcan.gc.ca/sites/oee.nrcan.gc.ca/files/pdf/publications/infosource/pub/home/heating-heat-pump/booklet.pdf 5, 7

[22] SELTRON, ogrevanje Dostopno na:
http://www.seltron.si/ogrevanje.html 4

[23] STMicroelectronics "Discovery kit for STM32 F4 series - with STM32F407VG MCU"Dostopno na:
http://www.st.com/web/catalog/tools/FM116/SC959/SS1532/PF252419 9, 10

[24] Waveshare Electronics, "3.2inch 320x240 Touch LCD (B)"Dostopno na:
http://www.wvshare.com/product/3.2inch-320x240-Touch-LCD-B.htm 11

[25] Waveshare Electronics, "Open407V-D Standard"Dostopno na:
http://www.wvshare.com/product/STM32F101VD.htm 11

[26] Vojtěch Vigner(vigisson) (2013), "OneWire repository"Dostopno na:
https://github.com/vigisson/OneWire 18