Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tom Kamin

Oddaljeno krmiljenje centralnega ogrevanja

diplomsko delo

visokošolski strokovni študijski program prve stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Patricio Bulić

Ljubljana, 2012

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: 00305/2012
Datum: 15.04.2012

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

Kandidat: **TOM KAMIN**

Naslov: **ODDALJENO KRMILJENJE CENTRALNEGA OGREVANJA**
REMOTE CENTRAL HEATING CONTROL

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Načrtujte in izdelajte vgrajen sistem, ki bo omogočal oddaljenjo krmiljenje centralnega ogrevanja. Tako naj sistem omogoča uporabniku spremljanje in nastavljanje trenutne temperature oddaljene lokacije preko spleta. Sistem naj bo zgrajen na osnovi mikrokrmilnika ATmega2560. Za razvoj sistema uporabite razvojni kit Arduino Mega 2560. Spletni vmesnik implementirajte v okolju ASP.NET.

Mentor:
prof. dr. Patricio Bulić

Dekan:
prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Tom Kamin, z vpisno številko **63060103**, sem avtor diplomskega dela z naslovom:

Oddaljeno krmiljenje centralnega 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 8. oktobra 2012 Podpis avtorja:

Zahvaljujem se vsem, ki so me na poti do diplome potrpežljivo usmerjali, še posebej pa svoji mami Marijani, ki mi je s pridnim delom omogočila študij. Zahvaljujem se stari mami Majdi za pomoč in oporo. Zahvaljujem se tudi svojim bližnjim prijateljem Blažu, Tinetu, Pavletu, Dušanu in Dejanu ter prijateljicam Vesni, Tini in Almi za spodbudo in nasvete. Zahvalo za izjemnega pedagoga pa si zasluži mentor Patricio, predvsem za motivacijo in usmerjanje skozi študij.

Diplomsko delo posvečam očetu Venceslavu.

Kazalo vsebine

Kazalo slik

Kazalo izvlečkov kode

Seznam uporabljenih kratic in simbolov

AJAX	Asynchronous JavaScript and XML
ARM	Advanced RISC Machine - Atmelov mikroprocesor z arhitekturo RISC
ASCII	American Standard Code for Information Interchange - znakovni format
CLR	Common Language Runtime - virtualna mašina, na kateri se izvaja MSIL
DHCP	Dynamic Host Control Protocol - protokol za dinamično dodeljevanje naslova IP napravam
EEPROM	Electrically Erasable Programmable Read-Only Memory - obstojen pomnilnik, ki ob izpadu napajanja ohrani podatke, običajno namenjen shranjevanju strojne kode programa in ukaznega nabora
I^2C	Two Wire - serijski vmesnik in protokol "two-wire"
IP	Internet Protocol - internetni protokol
IIS	Internet Information Services - spletni strežnik
LCD	Liquid Crystal Display - zaslon s tekočimi kristali
LED	Light Emitting Diode - dioda, ki oddaja svetlobo
MD5	Message Digest - kriptografska zgoščevalna funkcija
MSIL	Microsoft Intermediate Language - vmesna koda, ki se izvaja na virtualni mašini
MVC	Model-View-Controller - koncept načrtovanja programske opreme
NPN	Negative-Positive-Negative - tip polprevodniškega spoja v tranzistorju
PCB	Printed Circuit Board - tiskano vezje
R/W	Read/Write
SCL	Serial CLock - serijsko urino vodilo
SD	Secure Digital - pomnilniška kartica tipa SD
SDA	Serial Data - serijsko podatkovno vodilo

SRAM Static Random Access Memory - neobstojen pomnilnik zgrajen iz hitrih, pomnilniških celic, običajno namenjen predpomnjenju

SSL Secure Socket Layer - kriptografski protokol

STL Standard Template Library - programska knjižnica za delo s podatkovnimi strukturami

TCP Transmission Control Protocol - protokol za namensko povezavo

UDP User Datagram Protocol - datagramski protokol brez namenske povezave

V/I Vhod/Izhod

Povzetek

Sodobna mikrokrmilniška tehnologija nam omogoča enostavno komunikacijo z oddaljeno lokacijo preko spleta. V okviru diplomske naloge smo zasnovali vgrajen sistem, ki uporabniku omogoča oddaljeno krmiljenje centralnega ogrevanja preko spletnega vmesnika. Vgrajen sistem sestavljajo mikrokrmilniški sistem Arduino Mega 2560, omrežni modul na osnovi mikrokrmilnika Wiznet W5100 in digitalni temperaturni senzor DS1621. Uporabniški vmesnik sestavljajo matrična tipkovnica in znakovni zaslon LCD. Za vgrajen sistem smo razvili programsko opremo, ki uporabniku omogoča nastavljanje in prikaz trenutne temperature prostora na zaslonu LCD. Centralno ogrevanje krmilimo s pomočjo močnostnega releja. Razvili smo komplementarni informacijski sistem v obliki spletnega vmesnika, ki uporabniku omogoča spremljanje in nastavljanje temperature prostora ter spremljanje različnih statusnih podatkov vgrajenega sistema iz oddaljene lokacije. Spletni vmesnik smo razvili na platformi ASP.NET in mu dodali še administratorske strani, ki omogočajo delo s podatki o vgrajenem sistemu in uporabniku. Na koncu smo v programskem okolju CadSoft EAGLE PCB načrtali še tiskano vezje našega vgrajenega sistema, katerega vezje še ni bilo izjedkano.

Ključne besede

mikrokrmilnik, vgrajen sistem, Arduino, centralno ogrevanje, oddaljeno krmiljenje, termostat

Abstract

Modern microcontroller technology allows for easy communication with a remote location over the Internet. In our thesis, we designed an embedded system, which allows the user to remotely control the central heating system through a web interface. The embedded system consists of a microcontroller development board Arduino Mega 2560, a network-based microcontroller module Wiznet W5100 and a DS1621 digital temperature sensor. The user interface consists of a matrix keyboard and a LCD character display. For the embedded system, we have developed software that allows the user to control and display the current ambient temperature on the LCD screen. Central heating is controlled by a power relay. We have developed a complementary information system in the form of a web interface that allows the user to monitor and adjust the ambient temperature and also monitor various embedded system's status data from a remote location. The web interface has been developed on ASP.NET platform. Administration pages have been added for the purpose of user and embedded system's data manipulation and storing. As a conclusion, we have designed a printed circuit board of our embedded system, but the circuit has not yet been etched. For the purposes of designing the printed circuit, CadSoft EAGLE PCB development tool has been used.

Keywords

microcontroller, embedded system, Arduino, central heating, remote controlling, thermostat

Poglavje 1

Uvod

Internetna komunikacija je postala izjemno razširjena, s tem se odprejo možnosti krmiljenja elektronskih naprav iz oddaljene lokacije, saj so tudi mrežni krmilniki postali cenovno dostopni in enostavni za uporabo. Za učinkovito krmiljenje elektronskih naprav pa potrebujemo mikrokrmilniške sisteme, ki nam omogočajo povezovanje perifernih naprav v koherentno celoto. Zastavili bomo enega izmed možnih načinov, ki nam omogoča oddaljeno krmiljenje elektronskih naprav.

V diplomskem delu si bomo najprej na kratko ogledali mikrokrmilniški sistem Arduino, ki je v zadnjih letih postal popularen, predvsem zaradi odprtokodne narave in s tem povezane obsežne skupnosti. Za tem si bomo na kratko ogledali razvojna okolja, ki smo jih v diplomskem delu uporabili za razvoj spletnega vmesnika in načrtovanje vgrajenega sistema.

V drugem delu diplomskega dela bomo podrobneje opisali razvoj vgrajenega sistema za oddaljeno krmiljenje centralnega ogrevanja. Podrobneje bomo opisali povezovanje posameznih strojnih komponent z mikrokrmilniškim sistemom. Opozorili bomo tudi na nekatere težave, ki se lahko pojavijo pri uporabi izbranih strojnih komponent in predlagali možne rešitve.

Tretji del diplomskega dela je namenjen razvoju spletnega vmesnika za oddaljeno krmiljenje centralnega ogrevanja. Omenili bomo tehnologije, ki smo jih uporabili za implementacijo oddaljene komunikacije z vgrajenim sistemom. Predstavili bomo enega izmed načinov zasnove takega informacijskega sistema, katerega cilj je predvsem uporabniško prijazen dostop do vgrajenega sistema.

Zadnji del bomo namenili opisu načrtovanja tiskanega vezja, ki sicer še ni bilo izdelano. Pri načrtovanju tiskanih vezij je potrebno paziti na nekatere detajle, zato bomo nekaj od

teh omenili in predlagali pristope za reševanje teh problemov.

Za konec bomo ocenili izbiro elektronskih komponent in podali možne alternative. V sklepno misel bomo tudi strnili, kaj bi bilo še nujno potrebno razviti in izboljšati za zagotavljanje stabilnega in varnega delovanja vgrajenega sistema.

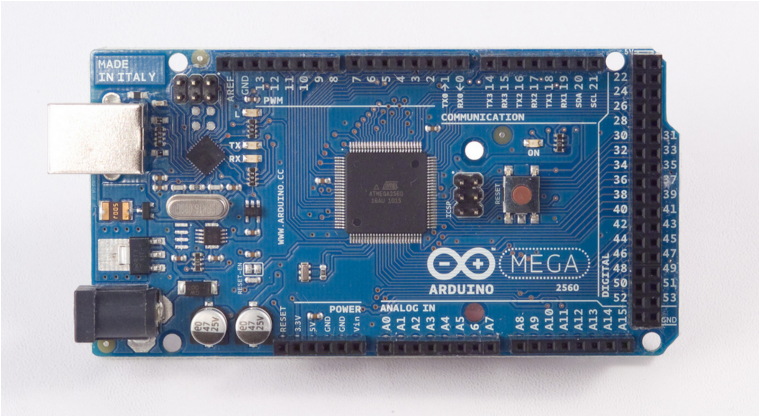
Poglavje 2

Mikrokrmilniški sistem Arduino

Arduino je odprtokodna platforma s širokim naborom mikrokrmilniških sistemov. Za izvedbo diplomskega dela smo izbrali mikrokrmilniški sistem Arduino Mega 2560 [3], predvsem zato, ker ima v primerjavi z drugimi razvojnimi ploščicami Arduino večje število vhodov in izhodov. Poleg tega ima večjo kapaciteto pomnilnika. Platforma je bila izbrana predvsem zaradi obsežne skupnosti razvijalcev in s tem tudi obsežne podpore, ne samo za mikrokrmilnik, temveč tudi za obsežen nabor popularnih strojnih komponent. S tem je platforma Arduino idealna za razvoj domačih, tako imenovanih projektov za navdušence DIY (*Do It Yourself*).

2.1 Specifikacije

Mikrokrmilniški sistem Arduino Mega 2560 (slika 2.1) temelji na Atmelovem mikrokrmilniku ATmega2560 [4]. Pomnilnika ima sicer na voljo precej malo, vendar za večino manjših projektov zadostuje. Frekvenca mikroprocesorja je relativno nizka, mikrokrmilnik tako ni primeren za izvedbo zahtevnejših projektov. Odlikuje pa ga relativno visoko število vhodov in izhodov, zaradi česar je idealen za krmiljenje večjega števila perifernih komponent.



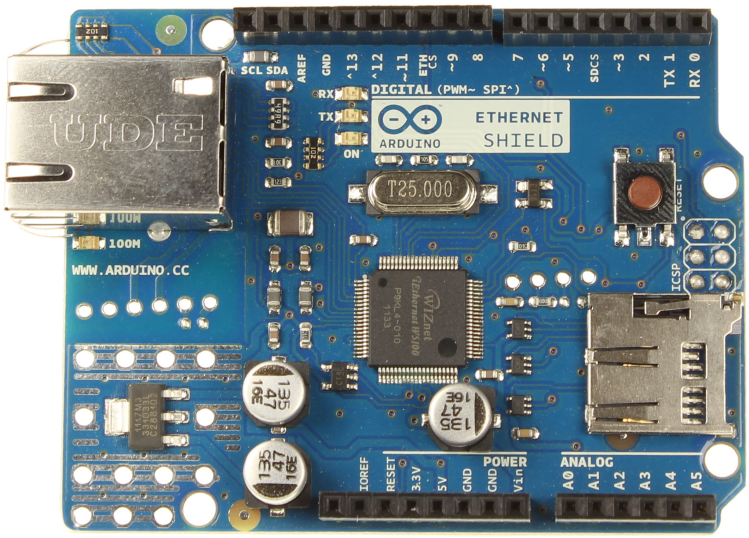
Slika 2.1: Mikrokrmilniški sistem Arduino Mega 2560, revizija 2. [3]

Specifikacije [3]:

- Obratovalna napetost: 5 V
- Št. digitalnih vhodov/izhodov: 54 (od tega 15 z možnostjo pulzno-širinsko moduliranega izhoda)
- Št. analognih vhodov: 16
- Maks. direktni tok na vhod/izhod: 40 mA
- Kapaciteta pomnilnika flash: 256 KB (od tega 8KB namenjenih primarnemu nalagalniku)
- Kapaciteta SRAM: 8KB
- Kapaciteta EEPROM: 4KB
- Frekvenca mikroprocesorja: 16 MHz

2.2 Mrežni krmilnik

Za komuniciranje preko interneta je potreben razširitveni mrežni modul, tako imenovani Ethernet "ščit" s krmilnikom WIZnet W5100 (slika 2.2). Razširitveni modul ima vtič RJ45 za priklop na omrežje, možnost priključitve razširitvene kartice SD (Secure Digital) za shranjevanje podatkov in 16 KB medpomnilnika [5] za shranjevanje podatkov, prejetih preko omrežja.



Slika 2.2: Ethernet ščit, revizija 3. [5]

2.3 Razvojno okolje

Za pisanje krmilnih programov se na platformi Arduino uporablja derivat programskega jezika C, tako da je uporabniku na voljo tudi omejena uporaba funkcij iz standardne knjižnice C. Pisanje programov je seveda možno tudi v zbirnem jeziku, vendar je to potrebno le ob potrebi po optimizaciji.

V času pisanja tega dela je bilo na voljo razvojno okolje Arduino 1.0.1. Programi se v strojno kodo prevajajo s prevajalnikom avrdude, ki je namenjen prevajanju izvorne kode za mikroprocesorje iz skupine AVR proizvajalca Atmel, h katerim spada tudi ATmega2560 [4].

Razvojno okolje je v času pisanja še precej nedodelano, zato bi bile slike odveč. Konfiguriranje prevajalnika in komunikacije z mikrokrmilniškim sistemom je delno podprto, velika pomanjkljivost je neprijaznost do razvijalca, saj okolje ne podpira niti osnovnih funkcij urejanja projekta ali izvorne kode. Prav tako je izpisovanje spremenljivk na serijski izhod mikrokrmilniškega sistema edini način razhroščevanja, ki ga ponuja okolje.

Kljub temu ima platforma Arduino ogromno podporo skupnosti, na voljo je obilo knjižnic, tudi za najosnovnejša opravila, kot je recimo krmiljenje perifernih vhodov in izhodov. Platformo tako odlikuje izjemno hiter razvoj.

Poglavje 3

Razvojno okolje

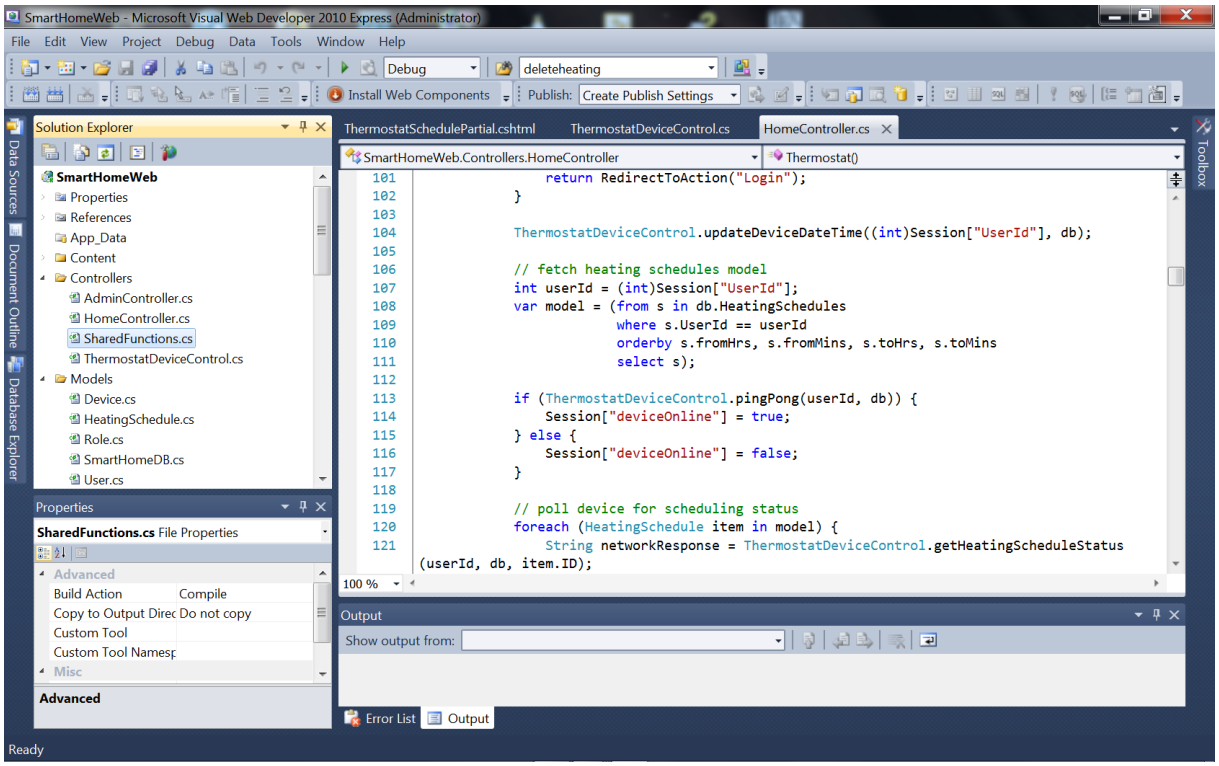
Informacijski sistem za oddaljeno upravljanje z vgrajenim sistemom smo razvili z uporabo platforme Microsoft .NET [6]. Tiskano vezje smo načrtali z uporabo razvojnega okolja CadSoft EAGLE PCB [7].

3.1 Microsoft .NET

Razvojna platforma .NET je produkt podjetja Microsoft. Odlikuje ga izjemno obsežna podpora in podpira veliko število ciljnih platform, saj se izvorna koda prevede v vmesno kodo MSIL (Microsoft Intermediate Language), ki se nato izvaja na virtualni mašini CLR (Common Language Runtime) [8]. S prevajanjem v vmesno kodo se ponudi tudi možnost pisanja izvorne kode v vseh jezikih, ki jih podpira platforma .NET. S platformo je možno razvijati vse od spletnih aplikacij, do samostojnih aplikacij in računalniških iger. Zato smo se informacijski sistem odločili implementirati s pomočjo te platforme, za programski jezik pa smo si izbrali C#.

MVC (Model-View-Controller) je razvojni konstrukt za hitrejši razvoj aplikacij. Ideja temelji na ločevanju podatkov, uporabniškega vmesnika in krmilne funkcionalnosti aplikacije, ki se razvija, kar je nujno potrebno za večje projekte, saj drugače izvorna koda postane neobvladljiva. Prav tako okolje podpira integrirano delo s podatkovnimi bazami.

Za postavitev aplikacije smo uporabili spletni strežnik IIS (Internet Information Services), ki je prav tako produkt podjetja Microsoft [9]. Razvoj aplikacij poteka v okolju Microsoft Visual Studio 2010, ki je sicer komercialni produkt, z registracijo pa si je možno brezplačno pridobiti ekvivalentni, vendar okrnjeni produkt Visual Web Developer Express [10] (slika 3.1).

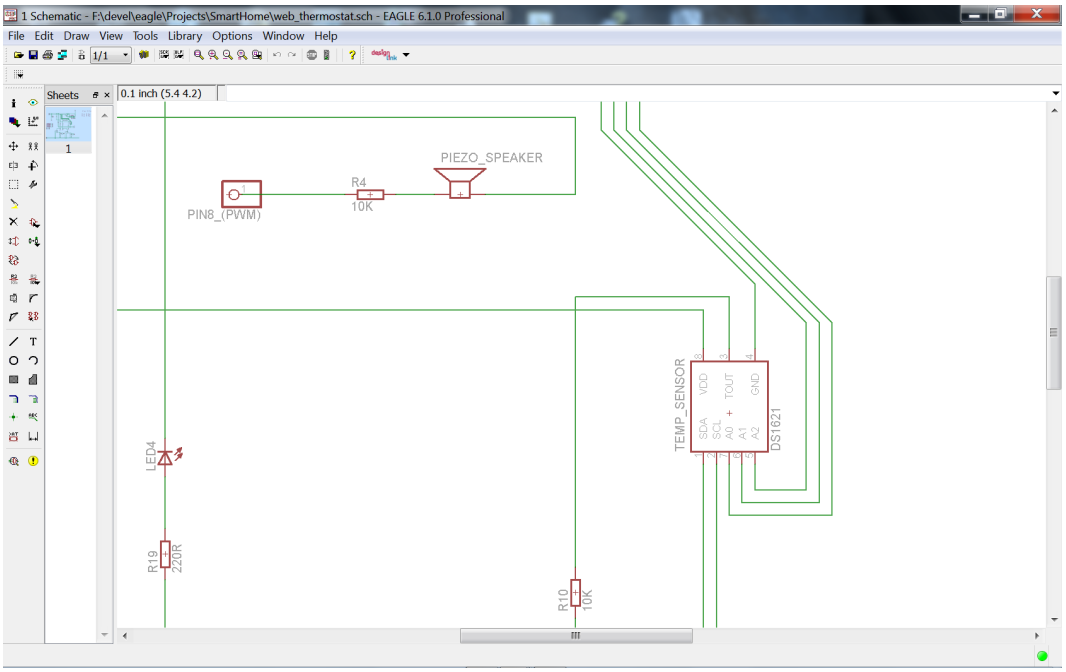


Slika 3.1: Visual Web Developer Express 2010.

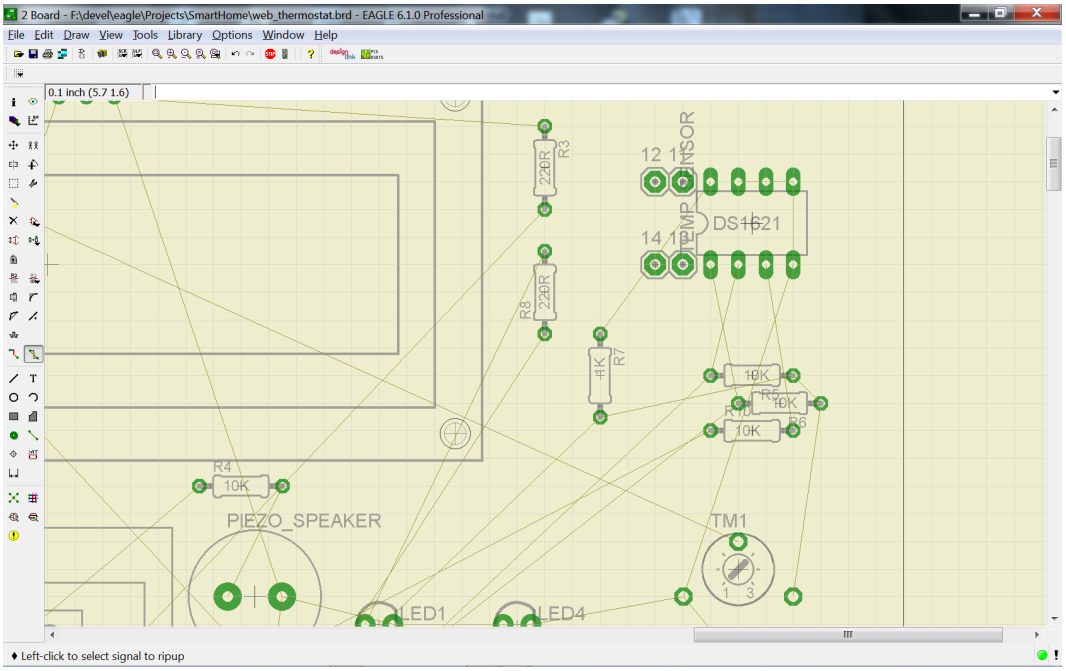
3.2 CadSoft EAGLE PCB

Programje EAGLE PCB [7], ki je produkt podjetja CadSoft, je namenjeno načrtovanju tiskanih vezij. Z njim lahko načrtujemo shemo našega vezja in z narisano shemo nato tudi generiramo tiskano vezje. Razvojno okolje si lahko ogledamo na sliki 3.2 in sliki 3.3.

Okolje je primerno tudi za načrtovanje kompleksnejših tiskanih vezij, idealno pa za načrtovanje manjših, domačih projektov. Posamezne elektronske komponente lahko narišemo tudi sami, v pomoč pa nam je ogromna podpora skupnosti. Za naš projekt sta nam bili v pomoč knjižnici skupnosti Adafruit [11] in SparkFun [12].



Slika 3.2: Risanje sheme v razvojnem okolju EAGLE.



Slika 3.3: Risanje tiskanega vezja v razvojnem okolju EAGLE.

Poglavje 4

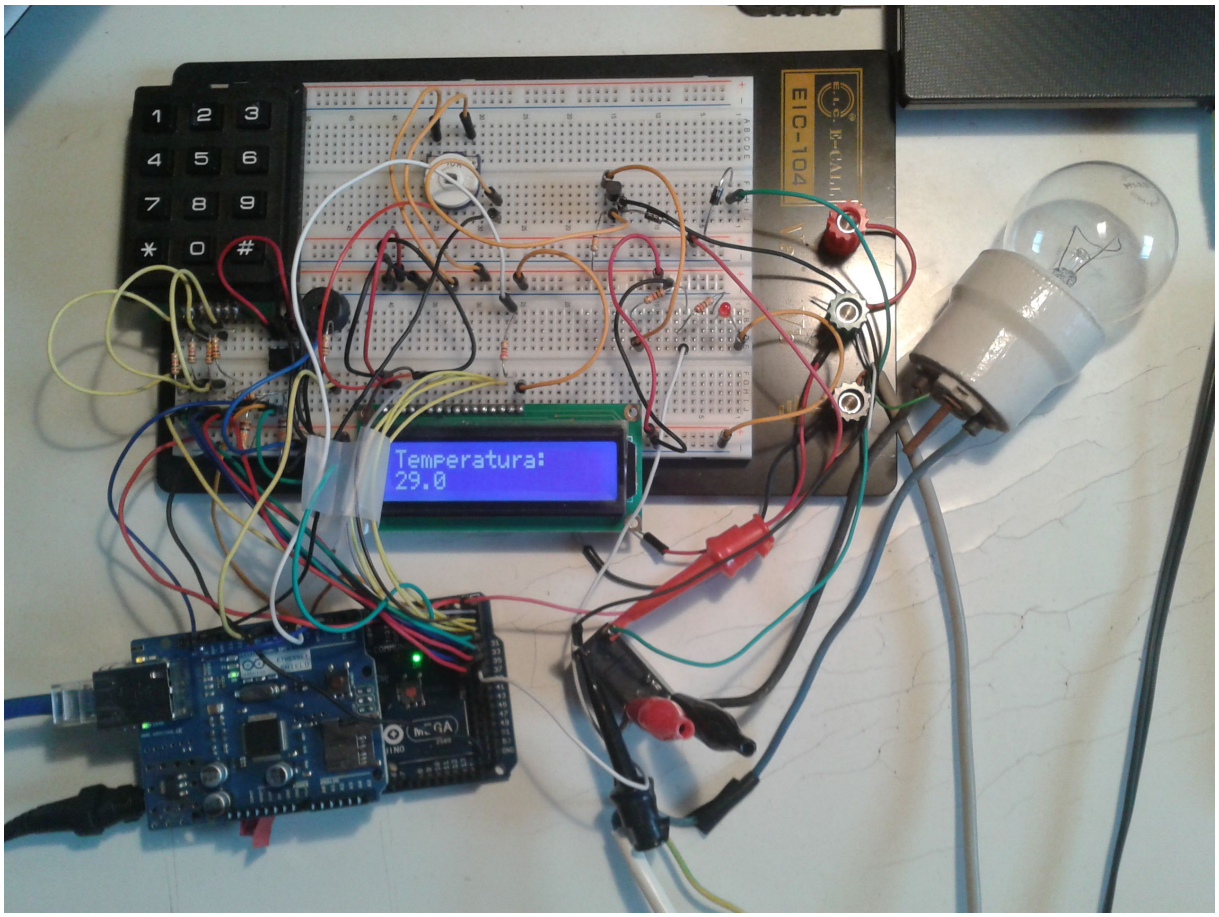
Vgrajen sistem

Za krmiljenje centralnega ogrevanja smo izdelali vgrajen sistem, baziran na mikrokrmilniškem sistemu Arduino Mega 2560 [3]. Vgrajen sistem je odgovoren tudi za komunikacijo s komplementarnim informacijskim sistemom, s katerim komunicira preko protokola IP. Sistemu smo dodali še uporabniški vmesnik, ki je v grobem zgrajen iz znakovnega zaslona, tipkovnice in indikatorskih lučk. Končni izdelek si lahko ogledamo na sliki 4.1.

4.1 Funkcije vgrajenega sistema

Namen vgrajenega sistema je upravljanje s centralnim ogrevanjem, zato sistem potrebuje funkcionalnost običajnega termostata. Sistem mora tako uporabniku omogočati spremljanje trenutne temperature prostora, v katerem se sistem nahaja. Prav tako mora sistem omogočati nastavljanje referenčne temperature, ki jo mora sistem ohranjati s krmiljenjem centralnega ogrevanja. Zaradi tehničnih razlogov omogoča sistem še spremljanje trenutnega časa v obliki digitalne ure in spremljanje trenutno dodeljenega naslova IP (Internet Protocol).

Preko komplementarnega informacijskega sistema lahko dodatno nastavimo še urnik delovanja termostata. Nastavimo lahko tudi periodično delovanje, tako da se termostat vklopi vsak dan v skladu z urnikom.



Slika 4.1: Končni izdelek na prototipni plošči.

Podprte funkcije:

- Spremljanje trenutne temperature prostora.
- Nastavljanje referenčne temperature.
- Spremljanje trenutnega časa in trenutno dodeljenega naslova IP.
- Upravljanje z urnikom delovanja preko informacijskega sistema.

4.2 Strojne komponente

Za izvedbo vgrajenega sistema smo uporabili nekaj perifernih komponent. Za interakcijo uporabnika s sistemom smo morali na sistem povezati matrično tipkovnico in znakovni zaslon LCD, za lažje spremljanje statusa termostata pa smo dodali še nekaj indikatorskih,

svetlobnih diod LED. Funkcije termostata smo dodali s pomočjo digitalnega temperaturnega senzorja DS1621 [13]. Komuniciranje z informacijskim sistemom smo omogočili z uporabo mrežnega krmilnika, tako imenovanega Arudino Ethernet ščita. Vklop in izklop centralnega ogrevanja pa smo implementirali s pomočjo močnostnega releja.

4.3 Ethernet ščit

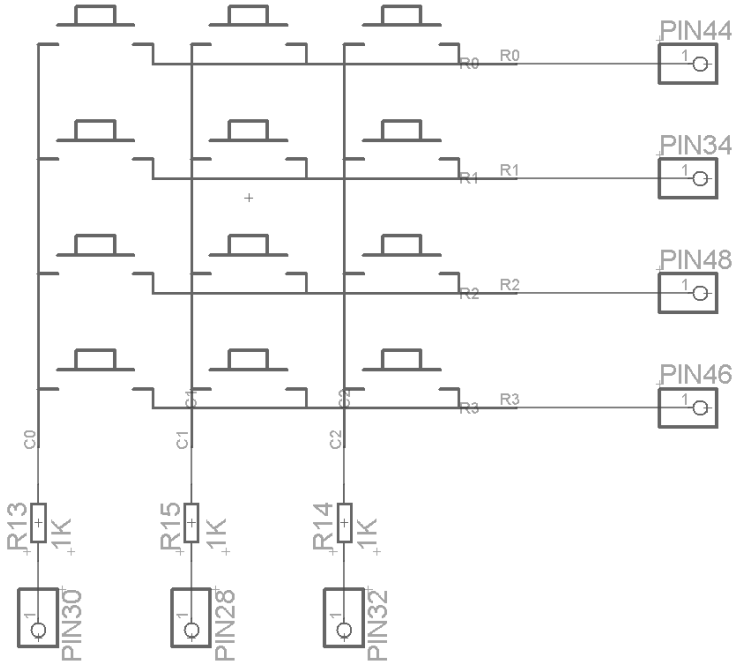
Mrežni krmilnik, tako imenovani Ethernet ščit [5], je odgovoren za prejemanje paketov IP (Internet Protocol). Podpira oba standardna usmerjevalna protokola, to sta TCP (Transmission Control Protocol) in UDP (User Datagram Protocol). Prejete pakete IP shranjuje v medpomnilnik, do katerega lahko dostopamo po principu povpraševanja (ang. polling). Modul je izdelan tako, da ga preprosto nataknemo na mikrokrmilniški sistem Arduino.

Eden izmed problemov ščita Ethernet je relativno visoka poraba toka, modul namreč porabi blizu $150\,mA$ električnega toka [14]. To je eden izmed razlogov, da se mikroprocesor ščita Ethernet prekomerno greje. Zaradi tega je potrebno poskrbeti za ustrezno lociranje temperaturnega senzorja na tiskanem vezju. Več o tem si lahko preberemo v razdelku 6.1. Poleg tega se zaradi visoke porabe električnega toka pojavijo tudi problemi s pregrevanjem regulatorja napetosti na mikrokrmilniškem sistemu Arduino, o čemer je govora v razdelku 4.9.

4.4 Matrična tipkovnica

Nastavljanje referenčne temperature in druga interakcija s sistemom, poteka preko matrične tipkovnice. Tipkovnica ima 3 stolpce in 4 vrstice. Shemo vezja priključene tipkovnice si lahko ogledamo na sliki 4.2. Iz slike je razvidno, da je tipkovnica zgrajena s preprostimi, fizičnimi stikali. Zaznavanje pritisnjene tipke običajno poteka tako, da na vsak stolpec posamezno, v iteraciji postavimo napetost $5\,V$. Če je katera tipka pritisnjena, se bo na ustrezni vrstici pojavila napetost $5\,V$.

Vrstice tipkovnice so povezane z vhodnimi vrati mikrokrmilnika. Vhodna vrata mikrokrmilnika Atmel ATmega2560 imajo sicer visoko upornost, tj. $100\,M\Omega$ [15], zato dodatni upor ni potreben. Problem pa se pojavi, če pritisnemo sosednje ležeči tipki v isti vrstici. S tem povzročimo kratek stik med stolpcem, ki trenutno oddaja signal z napetostjo $5\,V$ in med sosednjim stolpcem, ki ne oddaja signala in je vezan na ozemljitev (upornost iz-



Slika 4.2: Shema priključene matrične tipkovnice.

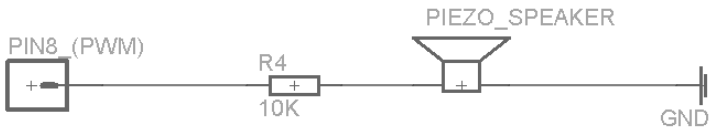
hoda oz. ozemljitve je blizu $0\,\Omega$). Po opravljeni meritvi z multimetrom ugotovimo, da ima posamezna povezava tipkovnice do $500\,\Omega$ upora. Zaradi previdnosti stolpce tipkovnice vežemo s serijskimi upori, vrednosti $1\,k\Omega$, s čimer dodatno zmanjšamo tok v primeru kratkega stika.

Za enostavnejšo implementacijo podpore tipkovnici smo uporabili knjižnico Arduino Keypad [16] (avtor knjižnice je v večji meri Mark Stanley), ki je namenjena delu z matričnimi tipkovnicami. Od verzije 3.0 naprej, knjižnica poskrbi za možnost pojavitve kratkega stika ob hkratnem pritisku več tipk. Stolpci so nastavljeni kot vhodna vrata (z visoko upornostjo), stolpec, ki trenutno oddaja signal je nastavljen kot izhodna vrata, vezana na ozemljitev. Vrstice so nastavljene kot vhodna vrata, ki so vezana z internim "pull-up" uporom (več o pull-up uporih si lahko preberemo v razdelku 4.7) vrednosti $20\,k\Omega$ [17]. Ko ustrezni stolpec, vezan na ozemljitev, sklenemo z neko vrstico, "potegnemo" oz. nastavimo vrednost vhodnih vrat ustrezne vrstice na logično ničlo. Povezava deluje po principu odprtega ponora. Dodatni upori tako niso potrebni.

Zaradi kompatibilnosti s starejšimi vrezijami knjižnice Keypad, pustimo serijske upore vrednosti $1\,k\Omega$.

4.5 Piezo piskač

Piezo piskač je kristal, ki s pomočjo piezoelektrične lastnosti generira zvočno valovanje. Zvok piezo piskača generiramo s pulzno širinsko modulacijo, tj. generirati moramo pravokotni signal. Za zmanjševanje jakosti zvoka, piskač vežemo s serijskim uporom vrednosti $10\,k\Omega$. Piskač sicer uporabljamo le za oddajanje tona s približno frekvenco $500\,Hz$, ko uporabnik pritisne na tipkovnico. Shemo vezja si lahko ogledamo na sliki 4.3.



Slika 4.3: Shema priključenega piezo piskača.

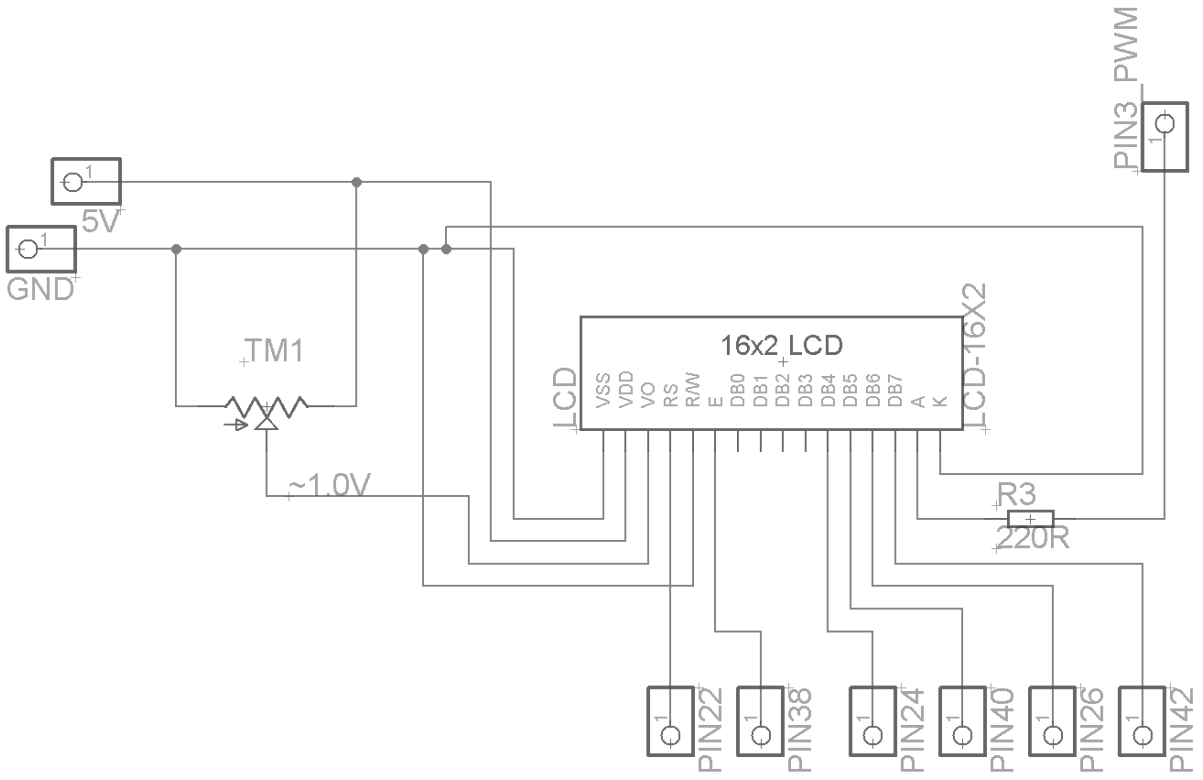
4.6 Znakovni zaslon LCD

V vgrajen sistem smo povezali znakovni zaslon LCD (Liquid Crystal Display) 16x2, to je 16 stolpcev in 2 vrstici osnovnih znakov ASCII (American Standard Code for Information Interchange). Zaslon je produkt proizvajalca Raystar Optronics, model RC1602BB in uporablja popularni gonilnik Hitachi HD44780 za znakovne zaslone [18]. Zaslon si lahko ogledamo na sliki 4.4.

Slika 4.4: Znakovni zaslon LCD 16x2.

Za povezovanje znakovnega zaslona z mikrokrmilniškim sistemom smo uporabili knjižnico LiquidCrystal [19]. Knjižnica omogoča tudi komunikacijo z modulom LCD preko štirih

pinov, namesto osem. S tem porabimo manj V/I (vhodno-izhodnih) vrat na mikrokrmilniku, vendar za to žrtvujemo hitrost osveževanja zaslona, kar pa za naš sistem ni ključnega pomena, zato smo se odločili za priključitev na manjše število V/I vrat. Osvetlitev zaslona krmilimo s pulzno-širinsko modulacijo, s tem lahko implementiramo pojemajočo intenziteto osvetlitve ob izklopu zaslona. Signal R/W (Read/Write) pa vežemo kar na ozemljitev, saj na zaslon le pišemo in nikoli ne beremo iz zaslona. Shemo, ki vključuje priklop zaslona si lahko ogledamo na sliki 4.5.



Slika 4.5: Shema priključenega modula LCD.

Na shemi opazimo, da smo za nastavljanje kontrasta uporabili potenciometer (ang. potenciometer, trimmer). Ker ima vsak modul LCD malo drugačno nastavitev kontrasta, smo morali le tega določiti empirično. Ustrezni kontrast smo nastavili na približno $1.0\,V$. Sicer bi lahko uporabili tudi napetostni delilnik, vendar bi nastavitev s tem ostala fiksna.

Na shemi opazimo tudi, da smo za povezavo osvetlitve zaslona, ki jo oddaja dioda LED (Light Emitting Diode), uporabili relativno visok upor, tj. $220\,\Omega$. Upornik prepušča približno $7\,mA$ toka (4.2). Tovarniška listina modula LCD sicer navaja minimalni tok za krmiljenje diode LED, ki naj bi bil $28.8\,mA$ [19], vendar je zaslon zadovoljivo osvetljen z

le $7\,mA$. Zaradi manjše porabe električnega toka, pustimo višji upor od predpisanega.

Izračunajmo porabljeni tok diode LED na shemi iz slike 4.5. Tovarniška listina navaja padec napetosti čez LED, ki tipično znaša $U_{led} = 3.5\,V$. Vhodna napetost znaša $U_0 = 5.0\,V$. Po Ohmovem zakonu:

$$I = \frac{U}{R} \tag{4.1}$$

$$= \frac{U_0 - U_{led}}{R} \tag{4.2}$$

$$= \frac{5.0V - 3.5V}{220\Omega}$$

$$= 6.8mA$$

, kjer je:

U - padec napetosti na uporu v Voltih $[V]$,

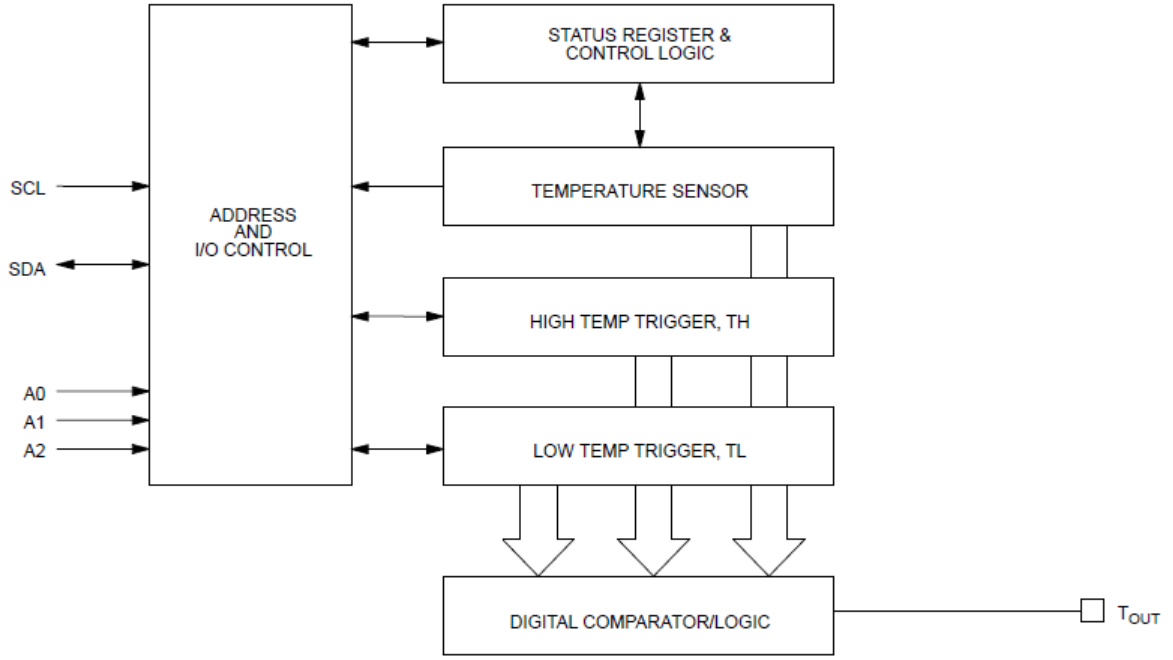
R - upornost v Ohmih $[\Omega]$,

I - tok, ki ga prepušča upornik, v Amperih $[A]$

4.7 Digitalni temperaturni senzor

Za zaznavanje temperature prostora smo uporabili digitalni temperaturni senzor Dallas DS1621 [13]. Senzor sporoča temperaturo z 9 biti in je sposoben brati temperaturo na pol stopinje celzija natančno v hitrem načinu ali na 2 decimalki stopinje celzija natančno v enojnem načinu (odčitek na 2 decimalki stopinje celzija natančno ni zanesljiv). Temperaturo je senzor zmožen zaznavati v območju od $-55\,°\mathrm{C}$ do $+125\,°\mathrm{C}$. Do vseh registrov čipa dostopamo po protokolu I^2C (ang. two-wire) [20]. Protokol je razvilo podjetje Philips Semiconductors in je namenjen za relativno počasno, serijsko komunikacijo s periferijo. Digitalni senzor, poleg konverzije temperature iz analognega odčitka v digitalno obliko, omogoča še običajne funkcije termostata. To je, omogoča nastavljanje referenčne temperature in histereze in temu ustrezno postavlja izhodni pin v obliki alarma, kadar presežemo ali se spustimo pod nastavljeno temperaturo. Podprte funkcije tega senzorja si lažje ponazorimo s sliko 4.6, ki jo lahko najdemo v tovarniški listini produkta [13].

Shemo priključenega senzorja si lahko ogledamo na sliki 4.7.

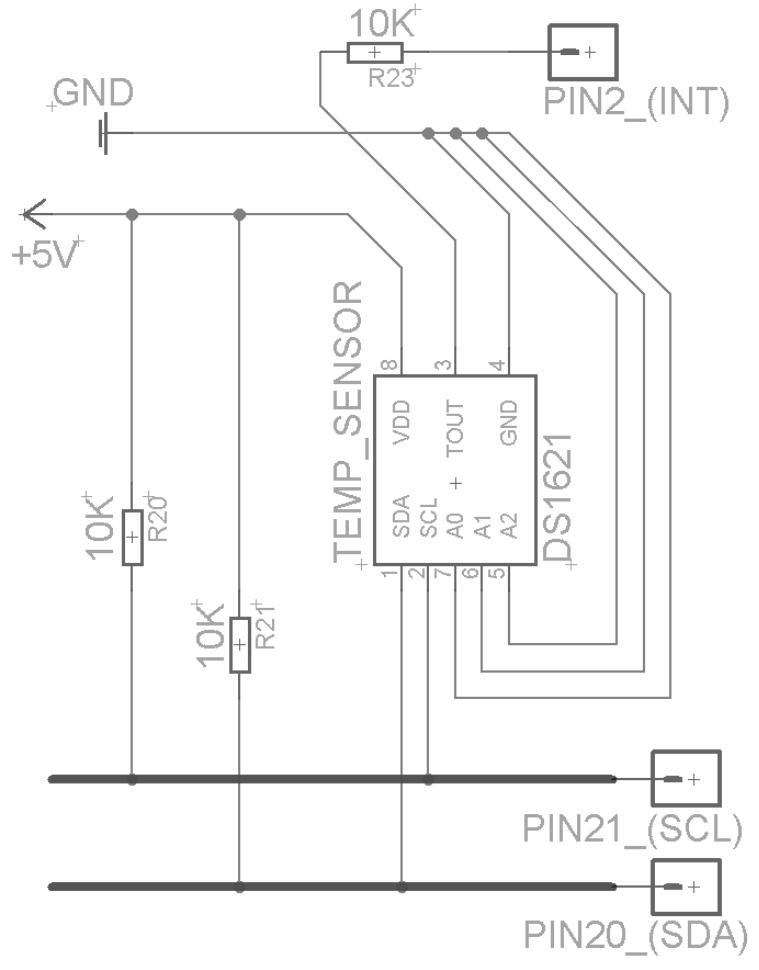


Slika 4.6: Funkcionalni diagram senzorja DS1621.

Po protokolu I^2C komuniciramo preko skupnega vodila, naprave si tako delijo eno vodilo in se odzovejo na prejete ukaze, ko so naslovljene. Iz sheme na sliki 4.7 lahko opazimo, da ima naprava DS1621 tri signale, namenjene določanju naslova naprave, to so signali A0, A1 in A2. S tem imamo na voljo 2^3 oz. 8 možnih naslovov za naprave DS1621 na istem vodilu. Signale našega senzorja, ki določajo naslov naprave na vodilu, smo vezali na ozemljitev, to pomeni, da se naprava nahaja na naslovu "000", napravo pa naslovimo preko protokola I^2C, kar si lahko podrobneje preberemo v tovarniški listini čipa [13].

Vodili SDA (Serial Data line) in SCL (Serial CLock) vežemo s tako imenovanimi "pull-up" upori. To so običajno upori višjih vrednosti (vrednosti 10 ali 20 $k\Omega$), ki jih vežemo na izhodno napetost mikrokrmilniškega sistema (v našem primeru $5V$). S tem preprečimo, da bi pini naprav obviseli v nedoločenem stanju, ko niso sklenjeni v vezje. Vodili SDA in SCL protokola I^2C sta tipa odprti ponor ali zbiralnik (ang. open-drain) in za svoje delovanje potrebujeta vezavo s pull-up uporom.

DS1621 prav tako podpira običajne funkcije termostata. Izhod Tout na shemi iz slike 4.7 deluje kot alarm, ki se postavi v visoko stanje, ko temperatura prostora preseže nastavljeno referenčno temperaturo termostata in v nizko stanje, ko temperatura prostora pade pod referenčno temperaturo. Pri tem digitalni temperaturni senzor upošteva



Slika 4.7: Shema digitalnega temp. senzorja DS1621.

nastavljeno histerezo. Princip delovanja histereze si lahko ogledamo na sliki 4.8. Iz slike je razvidno, da je histereza pri ogrevalnih in hladilnih sistemih zelo pomembna, saj bi v nasprotnem primeru naprava skušala vzdrževati točno določeno temperaturo in bi se v kratkih časovnih intervalih vklapljala in izklapljala. To je za napravo zelo slabo, tako iz vidika mehanske obrabe naprave kot tudi iz vidika porabe energije. Histerezo bi sicer morali implementirati programsko, vendar za to že poskrbi temperaturni senzor in ustrezno postavlja vrednost izhoda Tout. Zato izhod Tout pripeljemo v prekinitveni vhod mikrokontrolerja, ki se sproži ob pozitivni ali negativni fronti, t.j. ob spremembi signala, kar nam daje ukaz za vklop ali izklop centralne kurjave.

Za delo s temperaturnim senzorjem smo uporabili odprtokodno knjižnico Arduino DS1621. Knjižnica podpira vse funkcionalnosti, ki jih omogoča integrirani čip DS1621,

razen zapisovanja in branja temperature na 0.5 °C natančno, kar smo morali implementirati sami. Knjižnica je sicer podpirala le nastavljanje temperature v celoštevilčnih stopinjah, prav tako je bilo branje temperature prostora možno le v celih številih [21]. Ker čip omogoča konverzijo temperature v hitrem načinu na 0.5 °C natančno, smo se odločili dopolniti knjižnico in napisati ustrezno podporo. Izvleček primera branja temperature na 0.5 °C natančno, si lahko ogledamo v izpisu kode 4.1 in izpisu kode 4.2. Podporo smo napisali s pomočjo knjižnice Wire [22], ki je del razvojne platforme Arduino in implementira komunikacijo po protokolu I^2C.

Poleg tega je potrebno zaradi oddajanja toplote mikrokrmilniškega sistema Arduino in mrežnega mikroprocesorja W5100 [14] poskrbeti tudi za ustrezno lociranje temperaturnega senzorja na tiskanem vezju. Več o tem si lahko preberemo v razdelku 6.1.

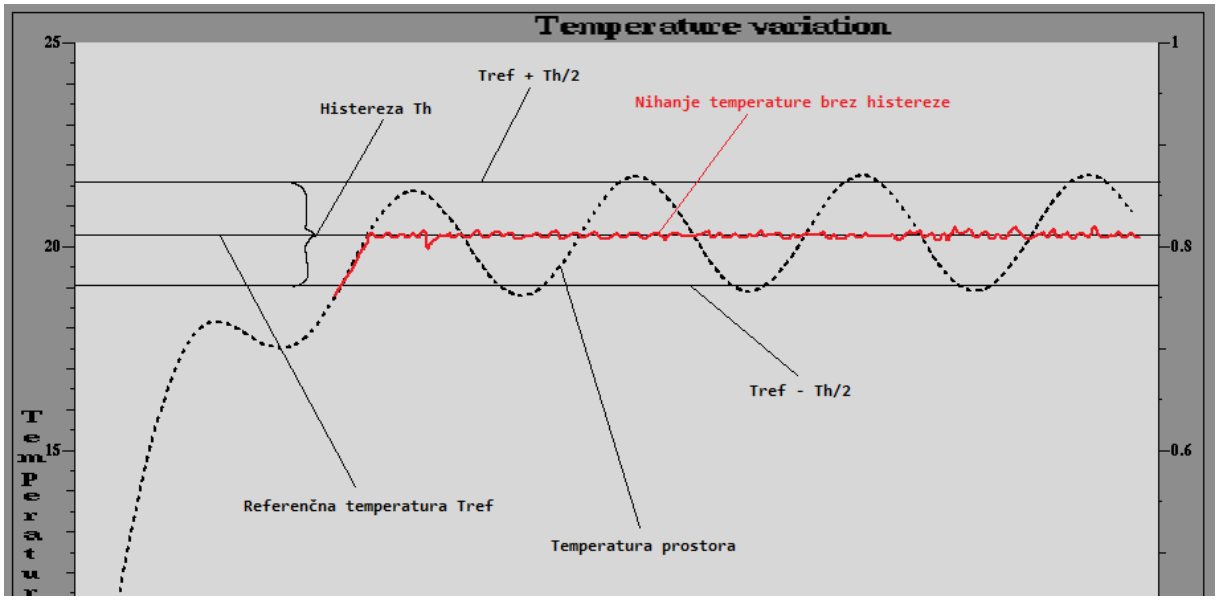
Izvleček kode 4.1: Branje dveh bajtov iz registra čipa DS1621.

```
1  // Read a DS1621 register, 2 bytes precision (needed for .5 temp reading)
2  uint16_t DS1621::getReg2(uint8_t reg) {
3    Wire.beginTransmission(SLAVE_ID);
4    Wire.write(reg);        // set register to read
5    Wire.endTransmission();
6
7    Wire.requestFrom(SLAVE_ID, 2);   // read 2 bytes
8
9    uint8_t regVal1 = Wire.read();   // read 1st byte
10   uint8_t regVal2 = 0;       // second byte
11   if (Wire.available()) {
12    regVal2 = Wire.read();      // read 2nd byte
13   }
14
15   // concat bytes
16   uint16_t regVal = regVal1;
17   regVal = regVal << 8;
18   regVal += regVal2;
19
20   return regVal;
21 }
```

Izvleček kode 4.2: Branje temperature na 0.5 °C natančno iz registra čipa DS1621.

```
1  // Reads temperature or threshold
2  // -- half degrees C included in second byte
3  // -- works only with RD_TEMP, ACCESS_TL, and ACCESS_TH
4  float DS1621::getTemp2(uint8_t reg) {
5    float temp = 0.0f;          // temperature to be read
6
7    if (reg == RD_TEMP || reg == ACCESS_TL || reg == ACCESS_TH) {
8      uint16_t tVal = getReg2(reg);
```

```
9
10    // if 9th bit is 1, then add 0.5 (see manual)
11    if ((tVal << 8) > 0) {
12      temp = (float)((tVal >> 8) + 0.5);
13    } else {
14      temp = (float)(tVal >> 8);
15    }
16
17    return temp;
18  }
19
20  return 0;              // bad reg, return 0
21 }
```



Slika 4.8: Princip ogrevanja s histerezo in nihanje temperature prostora s časom. Simulacija je bila sicer izvedena za hipotetično klimatsko napravo, ki je sposobna tako ogrevati kot hladiti. Klimatska naprava prav tako prilagaja obratovalno moč glede na trenutno razliko v referenčni temperaturi in temperaturi prostora.

4.8 Močnostni rele

Funkcijo vklapljanja in izklapljanja centralne peči realiziramo s pomočjo elektromehanskega (ang. reed), močnostnega releja (slika 4.9).

Za naš projekt potrebujemo rele, ki je zmožen preklopiti breme z napetostjo 220V in 16 do 20 A električnega toka. Poleg tega mora biti možno tuljavo upravljati z napetostjo

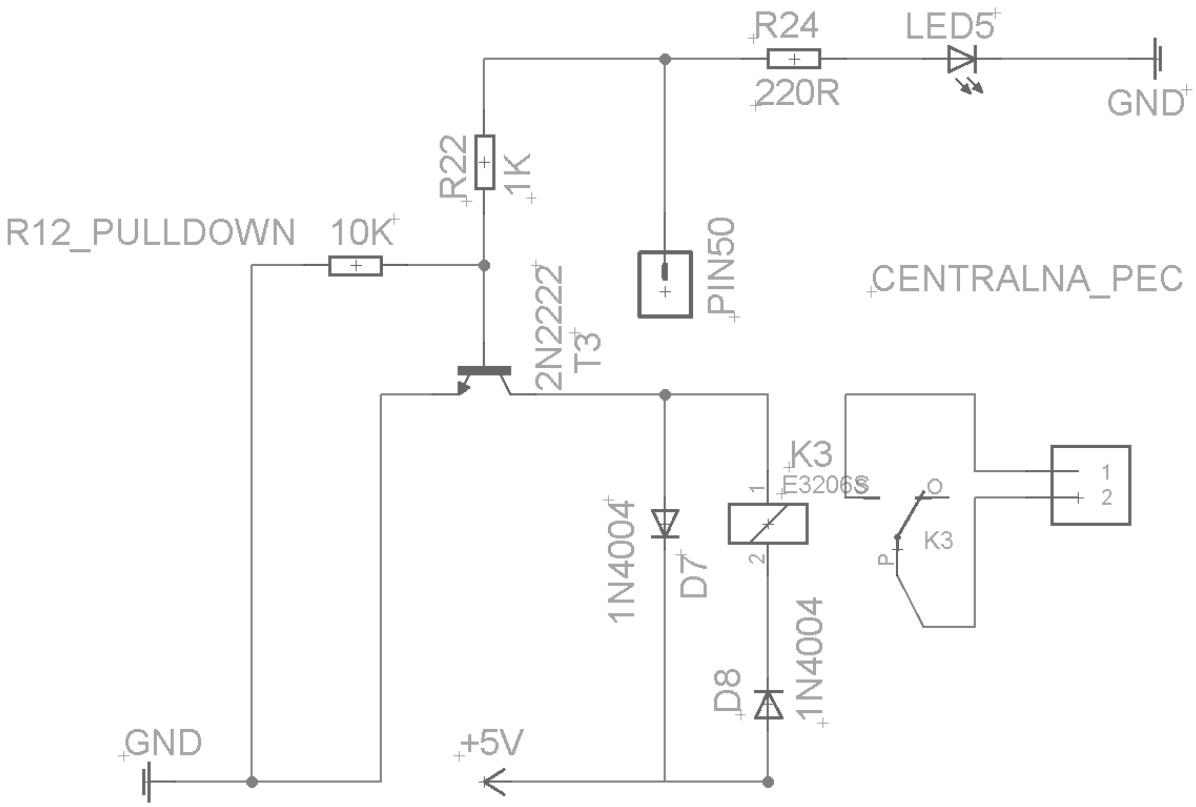
Slika 4.9: Reed rele.

$5\,V$, saj ne želimo uporabiti dodatnega napajanja. Zaenkrat na tiskanem vezju ostaja rele z možnostjo preklapljanja $10\,A$, kar ni dovolj za upravljanje centralne peči in ga bo potrebno zamenjati z zmogljivejšim relejem. Trenutni rele je zmožen upravljati s hišnimi napravami, kot so npr. žarnice (ki običajno porabijo 0.2 do $0.3\,A$ električnega toka), ventilatorji, manjše klimatske naprave itd.

Tuljava, ki jo je moč upravljati s $5\,V$ napetosti, porabi približno $50\,mA$ električnega toka pri vezavi z uporniško diodo oz. dobrih $100\,mA$ pri vezavi brez upornika [23], kar je relativno veliko. Zaradi tega se pojavijo problemi s pregrevanjem regulatorja napetosti na mikrokrmilniškem sistemu Arduino, kar je potrebno ustrezno nasloviti. Več o tem bomo omenili v naslednjem razdelku (4.9).

Shemo priključenega releja si lahko ogledamo na sliki 4.10. Močnostni rele krmilimo z digitalnimi V/I vrati 35 mikrokrmilnika. Na shemi lahko desno nad digitalnim izhodom vidimo priključeno indikatorsko svetlobno diodo LED, ki je namenjena obveščanju uporabnika o stanju centralnega ogrevanja. V razdelku 2.1 smo navedli, da posamezni digitalni izhod mikrokrmilniškega sistema lahko dostavi največ $40\,mA$ toka [3], zato moramo za krmiljenje releja uporabiti tranzistor (na shemi 4.10 ima komponenta oznako "T3"), s katerim preklapljamo napjanje direktno iz izhoda V_{cc}. Tranzistor je tipa NPN, kar pomeni, da so vrata odprta, ko na bazo spustimo električni tok (t.j. na V/I vrata zapišemo logično enico). "Pull-down" upor je v naši shemi nepotreben, saj se ob zapisu logične

ničle na bazo tranzistorja tok sprazni skozi svetlobno diodo LED. Zaradi korektnosti smo vseeno dodali redundantni upor.



Slika 4.10: Shema priključenega releja.

Močnostna dioda z oznako "D7" (na shemi 4.10 ležeča levo od releja) je namenjena varovanju tako tuljave releja kot tudi tranzistorja. Mehanski releji namreč ob izklopu tuljave zaradi pospeška stikala inducirajo vzvratni tok z visoko napetostjo, tudi do nekaj $100\,V$, kar na dolgi rok poškoduje tranzistor in tuljavo releja. Vhod releja zato vežemo z diodo na V_{cc}, tako da ob napetostni špici dioda tok prepušča le v smeri napajanja in v tranzistor udari nižji tok.

Druga močnostna dioda, z oznako "D8" (na shemi 4.10 ležeča pod relejem) je namenjena zmanjševanju električnega toka, s katerim krmilimo rele. Močnostne diode lahko namreč uporabimo kot nizek upor, to je kot upornik z nelinearno upornostjo. S tem v našem primeru zmanjšamo porabo električnega toka in znižamo vhodno napetost za približno $0.5\,V$.

4.9 Napajanje

Mikrokrmilniški sistem Arduino ima vgrajen linearni regulator napetosti, ki vhodno napetost zniža na $5\,V$, teoretično lahko uporabimo vhodno napjanje z napetostjo tudi do $20\,V$ [24]. Vendar višja kot je razlika med vhodno in izhodno napetostjo, višja je izguba energije v obliki toplote. Brez dodatnih hladilnih reber je regulator napetosti serije LM1117 zmožen odvesti do $1.4\,W$ toplotnega toka, preden se začne pregrevati. Idealno moramo toplotni tok znižati na maksimalno vrednost $1.2\,W$. Ena izmed očitnih rešitev je, da uporabimo napajalnik z nizko napetostjo, vendar ne nižje kot $7.0\,V$, saj se del napetosti izgubi pri regulaciji (ang. dropout voltage). Za regulator serije LM1117 ta padec napetosti znaša približno $1.1\,V$. Toplotni tok, ki ga oddaja regulator lahko izračunamo po enačbi 4.3, ki jo lahko najdemo v tovarniški listini linearnega regulatorja napetosti serije LM1117 [25].

$$\begin{aligned} I &= I_b + I_z \\ P &= (V_{vhod} - V_{izhod}) \cdot I_b + V_{vhod} \cdot I_z \end{aligned} \tag{4.3}$$

, kjer je:

I - - vhodni tok v Amperih $[A]$, I_b - tok bremena, I_z - tok na ozemljitev,

V - - razlika napetosti v Voltih $[V]$, V_{vhod} - vhodna napetost, V_{izhod} - izhodna napetost

Izračunajmo toplotni tok, ki ga oddaja regulator, če uporabimo vhodno napajanje z napetostjo $12\,V$ in porabimo $300\,mA$ električnega toka. Tok na zemljo zanemarimo.

$$\begin{aligned} P &= (V_{vhod} - V_{izhod}) \cdot I_b \\ &= (12\,V - 5\,V) \cdot 0.3\,A \\ &= 2.1\,W \end{aligned}$$

Vidimo, da toplotni tok, ki ga oddaja linearni regulator, krepko presega zmogljivosti odvajanja toplote, brez dodatnega hladilnega rebra. Če uporabimo vhodno napetost $7.5\,V$, sicer zmanjšamo oddajanje toplote regulatorja, vendar vrednost toplotnega toka vseeno znaša $0.75\,W$, kar je že na meji. Naš vgrajen sistem pa ob maksimalni obremenitvi porabi tudi do $400\,mA$ električnega toka. Samo zmanjšanje napetosti napajalnega

adapterja očitno ni dovolj in je potrebno poiskati dodatno rešitev. Najbolje je, da se regulatorju napetosti povsem izognemo, pri čemer pa nastopijo dodatni problemi, saj moramo poskrbeti za ustrezno napetost napajanja mikrokrmilniškega sistema in paziti na možne preobremenitve. Ta problem še ni povsem rešen zaradi časovnih omejitev.

4.10 Oddaljeno krmiljenje

Oddaljeno krmiljenje vgrajenega sistema omogoča mrežni krmilnik ali Ethernet "ščit". Sistem v te namene postavi strežnik, ki čaka na prejete ukaze oz. posluša na vratih 42322. Ukazi se sprejemajo po protokolu IP (Internet Protocol), vzpostavljena pa je povezava tipa TCP (Transmission Control Protocol). Naprava pridobi naslov IP z dinamičnim dodeljevanjem naslovov, to je po protokolu DHCP (Dynamic Host Control Protocol).

Sistem sprejema ukaze v formatu nizov ASCII, konec ukaza pa predstavlja znak za novo vrstico, to je znak '\n'. Sicer bi z implementacijo ukazov v binarni obliki lahko precej zmanjšali obremenitev komunikacijskega kanala, vendar smo se za implementacijo z nizi odločili zaradi večje robustnosti izvorne kode. Poleg tega ni predvideno, da bi bil sistem visoko obremenjen, zato je skrb zaradi nekaj odvečnih bajtov podatkov odveč. Izbrani mikrokrmilniški sistem tudi ni primeren za dosti obsežnejšo obremenitev, tako da tak format ukazov ni primarna skrb za skalabilnost sistema.

Več o naboru ukazov in oddaljenem krmiljenju si lahko ogledamo v razdelku 5.2.

4.11 Krmilni program

Krmilni program mora poleg upravljanja s perifernimi napravami skrbeti še za logiko sistema. Logika je v osnovi preprosta, krmilni program v zanki čaka na sprejem ukaza preko tipkovnice ali preko omrežja. Ko sprejmemo uporabnikov vnos, ga najprej obdelamo in glede na prepoznavnost ukaza kličemo ustrezne funkcije za delo s periferijo.

Delo s perifernimi napravami smo že opisali v predhodnih razdelkih, zato se osredotočimo na prepoznavanje ukazov, prejetih preko mrežnega priključka. Za te namene smo napisali funkcijo *processNetRequest*, ki sprejme zahtevo preko parametra v funkcijo, zahtevo obdela in kliče ustrezne funkcije. Celotne funkcije zaradi obsega ne bomo izpisovali, izvleček pa si lahko ogledamo na izpisu izvorne kode 4.3.

Izvleček kode 4.3: Procesiranje zahteve prejete preko omrežnega priključka.

```
1  /** process incoming network request **/
2  void processNetRequest(EthernetClient client, String recvRequest) {
3    char tempString[10];
4
5    if (recvRequest == DEVICE_COMMAND_GET_TEMP) {
6      client.print(dtostrf(tempCurr, 0, 1, tempString));
7
8      recvRequest = "";
9    } else if (recvRequest == DEVICE_COMMAND_REF_TEMP_INC) {
10     incRefTemp();
11     updateLCD();
12
13     client.print(DEVICE_RESPONSE_OK);
14
15     recvRequest = "";
16   } else if (recvRequest.substring(0, DEVICE_COMMAND_SCHEDULE_HEATING.length()) ==
         DEVICE_COMMAND_SCHEDULE_HEATING) {
17     char* tokenCurr;
18     // just for conversion between string and char*
19     char* recvRequestCpy = (char*)malloc(sizeof(char)*(recvRequest.length()+1));
20
21     // schedule time data
22     int fromHrs = -1;
23     int fromMins = -1;
24     int toHrs = -1;
25     int toMins = -1;
26     int scheduleId = -1;
27     // true if scheduled heating is periodic (repeat every day)
28     boolean heatingPeriodic = false;
29
30     recvRequest.toCharArray(recvRequestCpy, recvRequest.length() + 1);
31
32     // remove first token, which is DEVICE_COMMAND_SCHEDULE_HEATING
33     strtok(recvRequestCpy, DEVICE_PARAM_SEPARATOR);
34
35     scheduleId = atoi(strtok(NULL, DEVICE_PARAM_SEPARATOR));  // extract scheduleId
36     fromHrs = atoi(strtok(NULL, DEVICE_PARAM_SEPARATOR));   // extract ftomHrs
37     fromMins = atoi(strtok(NULL, DEVICE_PARAM_SEPARATOR));    // extract fromMins
38     toHrs = atoi(strtok(NULL, DEVICE_PARAM_SEPARATOR));     // extract toHrs
39     toMins = atoi(strtok(NULL, DEVICE_PARAM_SEPARATOR));      // extract toMins
40
41     // extract optional parameter DEVICE_SCHEDULE_PERIODIC
42     tokenCurr = strtok(NULL, DEVICE_PARAM_SEPARATOR);
43     if (tokenCurr != NULL && DEVICE_SCHEDULE_PERIODIC.equals(tokenCurr)) {
44       heatingPeriodic = true;
45     } else {
46       heatingPeriodic = false;
47     }
48
49     // schedule heating
50     addHeatingSchedule(scheduleId, fromHrs, fromMins, toHrs, toMins, heatingPeriodic);
```

```
51
52      client.print(DEVICE_RESPONSE_OK);
53
54      recvRequest = "";
55      free(recvRequestCpy);
56    } else {
57       client.print(DEVICE_RESPONSE_NOT_OK);
58
59      recvRequest = "";
60    }
61  }
```

Na tem mestu moramo omeniti še nastavljanje urnika preko spletnega vmesnika. Spletni vmesnik omogoča nastavljanje terminov ogrevanja. Ker nismo želeli omejevati števila terminov, ki jih uporabnik lahko nastavi, smo za namene shranjevanja urnika implementirali podatkovno strukturo s povezanim seznamom. Za mikroprocesorje modela AVR sicer obstaja modulirana knjižnica STL (Standard Template Library) jezika C, vendar v prevedeni obliki zavzema nekaj dodatnih KB pomnilnika. Ker je kapaciteta pomnilnika mikrokrmilniškega sistema Arduino močno omejena, smo povezani seznam implementirali sami.

Omeniti je še potrebno dejstvo, da smo procesiranje prejetih ukazov implementirali v obliki funkcije, namesto v obliki knjižnice za krmiljenje termostata preko omrežja. To je iz informacijskega stališča morda res nekoliko okrnjeno, vendar pa je krmiljenje vgrajenega sistema tako močno vezano na prejete ukaze, da bi ločevanje povzročilo nekaj dodatnega dela in težav. Zato smo se procesiranje ukazov odločili implementirati bolj pragmatično.

Poglavje 5

Informacijski sistem

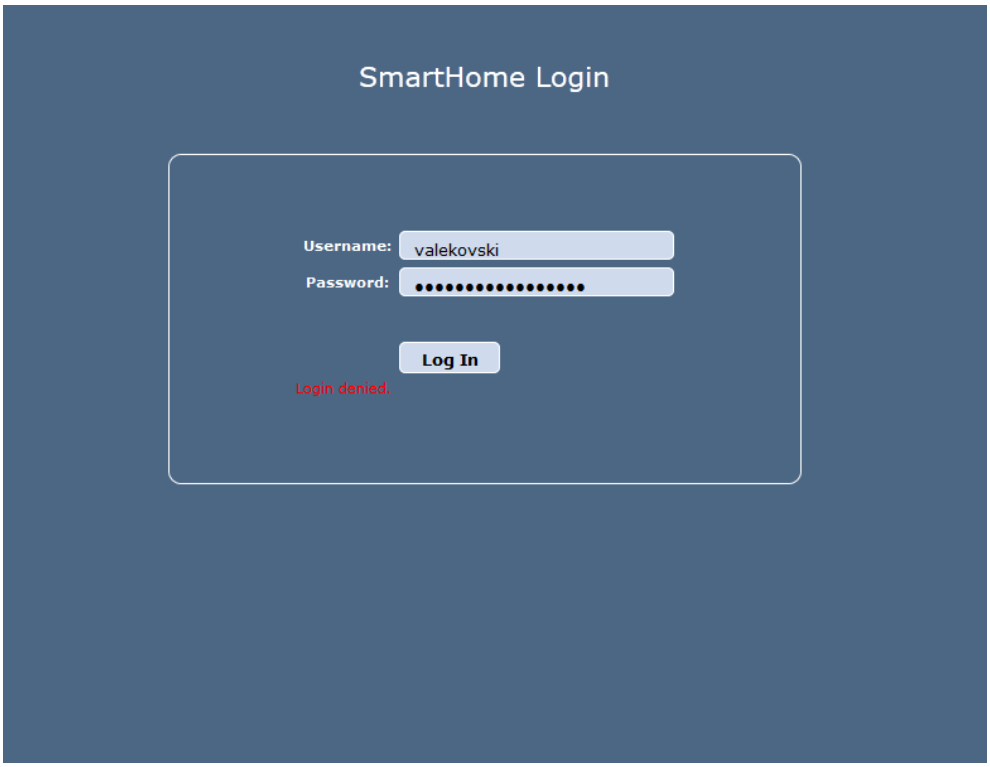
Komplementarni informacijski sistem vgrajenemu sistemu, ki smo ga opisali v prejšnjem poglavju, omogoča oddaljeno krmiljenje centralnega ogrevanja preko spletnega uporabniškega vmesnika. Za te namene smo izdelali administratorske strani, ki omogočajo dodajanje uporabnikov in urejanje podatkov o le teh. Poleg administratorskih strani smo izdelali še spletno mesto, v katerega se mora uporabnik prijaviti z uporabniškim imenom in geslom, kjer lahko krmili centralno ogrevanje. Sistem je bil razvit v okolju ASP.NET MVC 3 [2], ki je produkt podjetja Microsoft.

5.1 Administratorske strani

Administratorske strani omogočajo dodajanje in urejanje uporabnikov in podatkov o le teh. Samostojna registracija novih uporabnikov je namerno onemogočena, tako da ima pravico dodajanja novega uporabnika le uporabnik z administratorskimi pravicami. Sistem omogoča tudi shranjevanje podatkov o napravi, to so podatki o naslovu IP naprave (mikrokrmilniškega sistema). Podatki se shranjujejo v podatkovno bazo strežnika Microsoft IISExpress [9]. Za enostavnejšo predstavo delovanja si uporabniški vmesnik administratorskih strani lahko ogledamo na sliki 5.1 in sliki 5.2.

5.2 Oddaljeno krmiljenje vgrajenega sistema

Spletni vmesnik za oddaljeno krmiljenje vgrajenega sistema prav tako zahteva prijavo v informacijski sistem iz očitnih potreb po varnosti. Uporabniško geslo je zavarovano s pomočjo zgoščevalnega algoritma MD5 in je v taki obliki tudi shranjeno v podatkovni bazi.



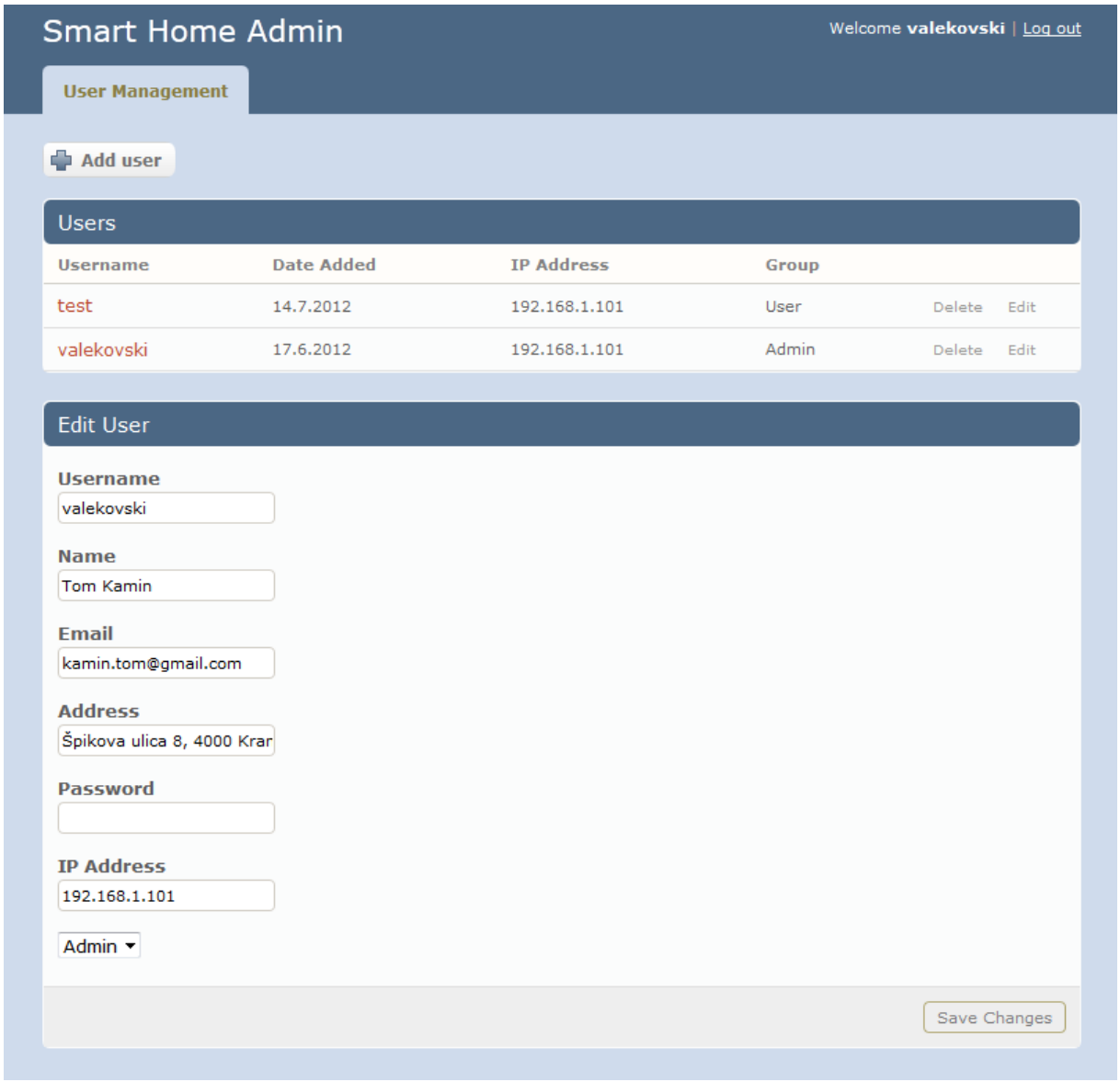
Slika 5.1: Prijava v administratorske strani.

Podprte funkcije oddaljenega krmiljenja:

- Oddaljeno vklapljanje in izklapljanje termostata.
- Spremljanje trenutne temperature prostora in statusa ogrevanja.
- Nastavljanje referenčne temperature.
- Upravljanje in spremljanje urnika delovanja termostata.

Spletni vmesnik za oddaljeno krmiljenje vgrajenega sistema si lahko ogledamo na sliki 5.3.

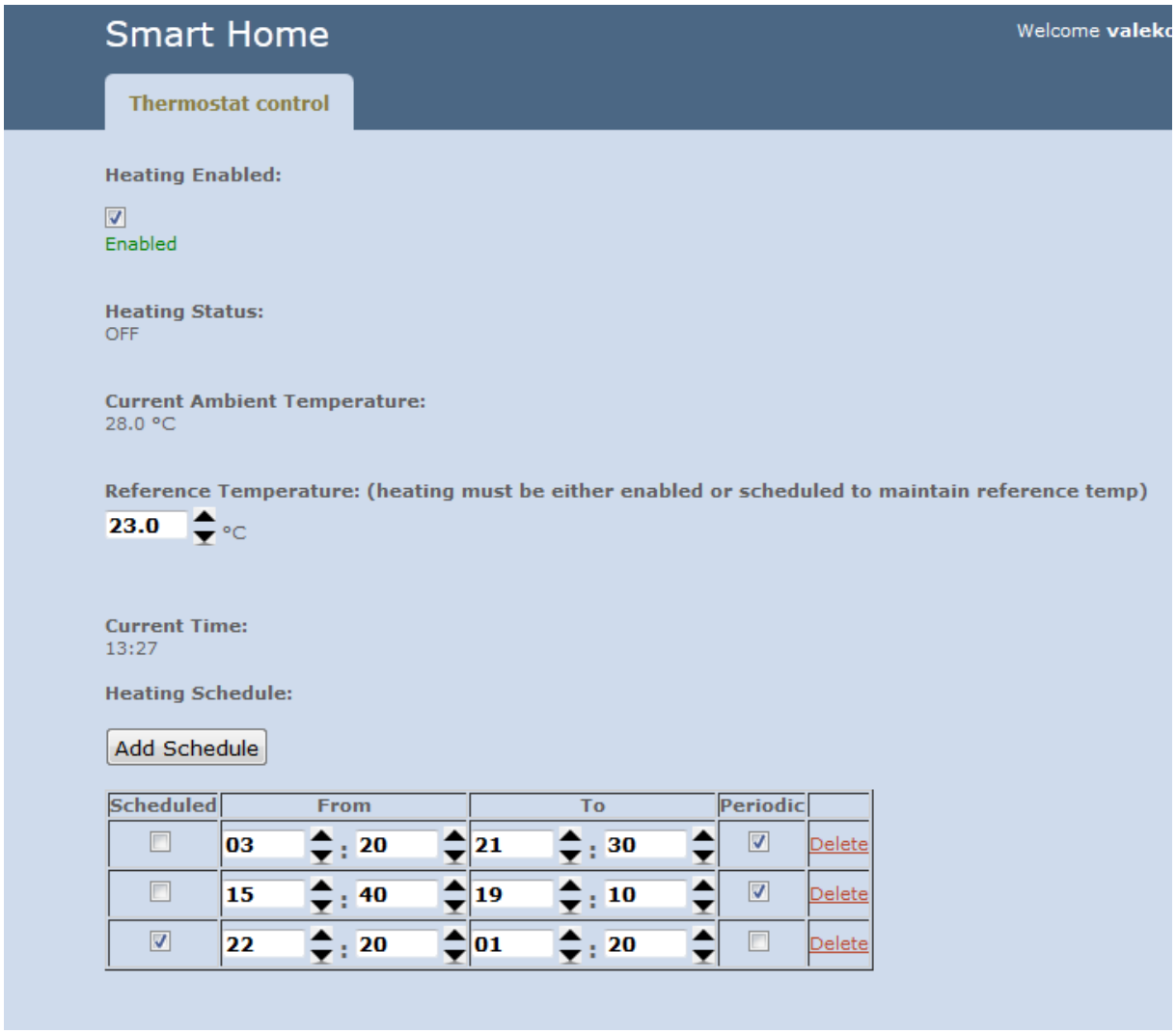
Osveževanje spletnega vmesnika s podatki iz vgrajenega sistema (trenutna temperatura temperatura, status ogrevanja, urnik ogrevanja itd.) smo implementirali s pomočjo knjižnice jQuery [26], ki implementira tehnologijo AJAX [27] (Asynchronous JavaScript and XML). Tehnologija AJAX omogoča asinhrono komunikacijo s spletnim strežnikom, strežnik pa posreduje le zahtevane podatke, namesto celotne spletne strani. Tako osveževanje



Slika 5.2: Administratorske strani.

strani ni potrebno. Osveževanje spletnega vmesnika s podatki iz vgrajenega sistema se zahteva periodično s pomočjo jezika Javascript.

Klici AJAX sprožijo kontrolno funkcijo sistema .NET na spletnem strežniku, ki je napisana v programskem jeziku C# . Kontrolna funkcija je odgovorna za posredovanje zahtevanih podatkov spletnemu brskalniku, podatke iz vgrajenega sistema pa pridobi s pomočjo knjižnice *ThermostatDeviceControl*, ki smo jo napisali v obliki razreda. Knjižnica je odgovorna za vso komunikacijo z vgrajenim sistemom. Vsaka zahteva, poslana vgrajenemu sistemu, gre skozi funkcijo *queryDevice*, ki si jo lahko ogledamo na izpisu 5.1.



Slika 5.3: Spletni vmesnik za krmiljenje vgrajenega sistema.

Funkcija najprej vzpostavi povezavo TCP, nato pošlje zahtevo vgrajenemu sistemu, na katero se mora vgrajen sistem ustrezno odzvati. Če vgrajen sistem ne prepozna posredovane zahteve ali se iz drugih razlogov ne odzove v zadanem času, se povezava prekine, o čemer uporabnika obvestimo preko spletnega vmesnika.

Izvleček kode 5.1: Komunikacija spletnega strežnika z vgrajenim sistemom.

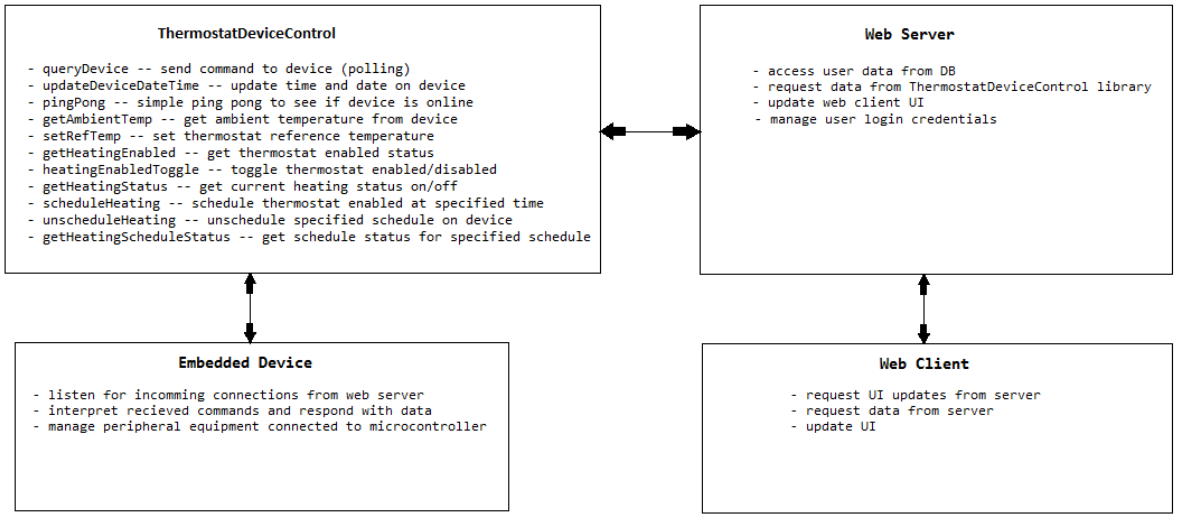
```
1 /** queryDevice -- send command to device with hostname deviceHost and return response **/
2 public static string queryDevice(string deviceHost, string command) {
3   NetworkStream networkStream = null;              // socket stream with device
4   String networkResponse = DEVICE_RESPONSE_NOT_OK;      // response from device
5   // request socket with the device and open stream
6   using (TcpClient tcpClient = new TcpClient()) {
7     tcpClient.ReceiveTimeout = 50;
```

```
8     tcpClient.SendTimeout = 50;
9
10    IAsyncResult asyncReq = tcpClient.BeginConnect(deviceHost, DEVICE_PORT, null, null);
11    System.Threading.WaitHandle waitHandle = asyncReq.AsyncWaitHandle;
12    try {
13      if (!asyncReq.AsyncWaitHandle.WaitOne(TimeSpan.FromSeconds(
        DEVICE_SOCKET_TIMEOUT), false)) {
14        tcpClient.Close();
15
16        throw new TimeoutException();
17      }
18
19      tcpClient.EndConnect(asyncReq);
20
21      networkStream = tcpClient.GetStream();
22      StreamWriter streamWriter = new StreamWriter(networkStream);
23      streamWriter.AutoFlush = true;
24      StreamReader streamReader = new StreamReader(networkStream);
25
26      // send command to device
27      streamWriter.Write(command);
28
29      // wait for response and fetch
30      try {
31        networkResponse = streamReader.ReadLine();
32      } catch (Exception ex) {
33        Console.WriteLine(ex.StackTrace);   // does nothing
34      }
35
36      streamWriter.Close();
37      streamReader.Close();
38      networkStream.Close();
39      tcpClient.Close();
40    } catch {
41      if (networkStream != null) networkStream.Close();
42      if (tcpClient != null) tcpClient.Close();
43    } finally {
44      waitHandle.Close();
45    }
46  }
47
48  return networkResponse;
49 }
```

Za lažjo predstavo poteka komunikacije med spletnim brskalnikom, spletnim strežnikom in vgrajenim sistemom si lahko ogledamo diagram na sliki 5.4. Iz diagrama lahko razberemo tudi podprte funkcionalnosti spletnega vmesnika. Vidimo lahko tudi, da gredo vse zahteve po podatkih iz vgrajenega sistema preko knjižnice *ThermostatDeviceControl*.

Ključna pomanjkljivost glede varnosti sistema je nezavarovan komunikacijski kanal med spletnim strežnikom in vgrajenim sistemom. Podatki in zahteve se namreč med na-

pravama izmenjujejo v nezavarovani, tekstovni obliki. Mikrokrmilniški sistem Arduino žal ne ponuja podpore za implementacijo postopka SSL (Secure Socket Layer), protokol pa je preobsežen, da bi ga za namene diplomskega dela implementirali sami. Komunikacijski kanal bi bilo potrebno ustrezno zavarovati, saj je trenutno dovzeten za tako imenovani napad "mož v sredini" (ang. Man in the middle). Komunikacijo med strežnikom in vgrajenim sistemom lahko namreč prestrežemo brez večjih težav, če imamo seveda ustrezno tehnično znanje. S tem napadom bi zmogli razbrati format ukazov in podatkov, s čimer je varnost sistema ogrožena. Zato bi bilo nujno potrebno implementirati neko vrsto kriptiranja podatkov. Omenjeni problem zaenkrat ostaja nerešen.



Slika 5.4: Potek komunikacije spletnega vmesnika in spletnega strežnika z vgrajenim sistemom.

Poglavje 6

Tiskano vezje

Izdelava tiskanega vezja se prične s kreiranjem prototipa na preizkusni plošči. Končni izdelek si lahko ogledamo na sliki 4.1. Zaradi časovnih omejitev tiskanega vezja še nismo izdelali. Poleg tega ostajajo nerešeni problemi (ki sicer niso kritičnega pomena) z napajanjem in s tipom uporabljenega močnostnega releja, zato smo izdelavo tiskanega vezja zaenkrat odložili. Izdelava vezja ostaja v fazi načrtovanja.

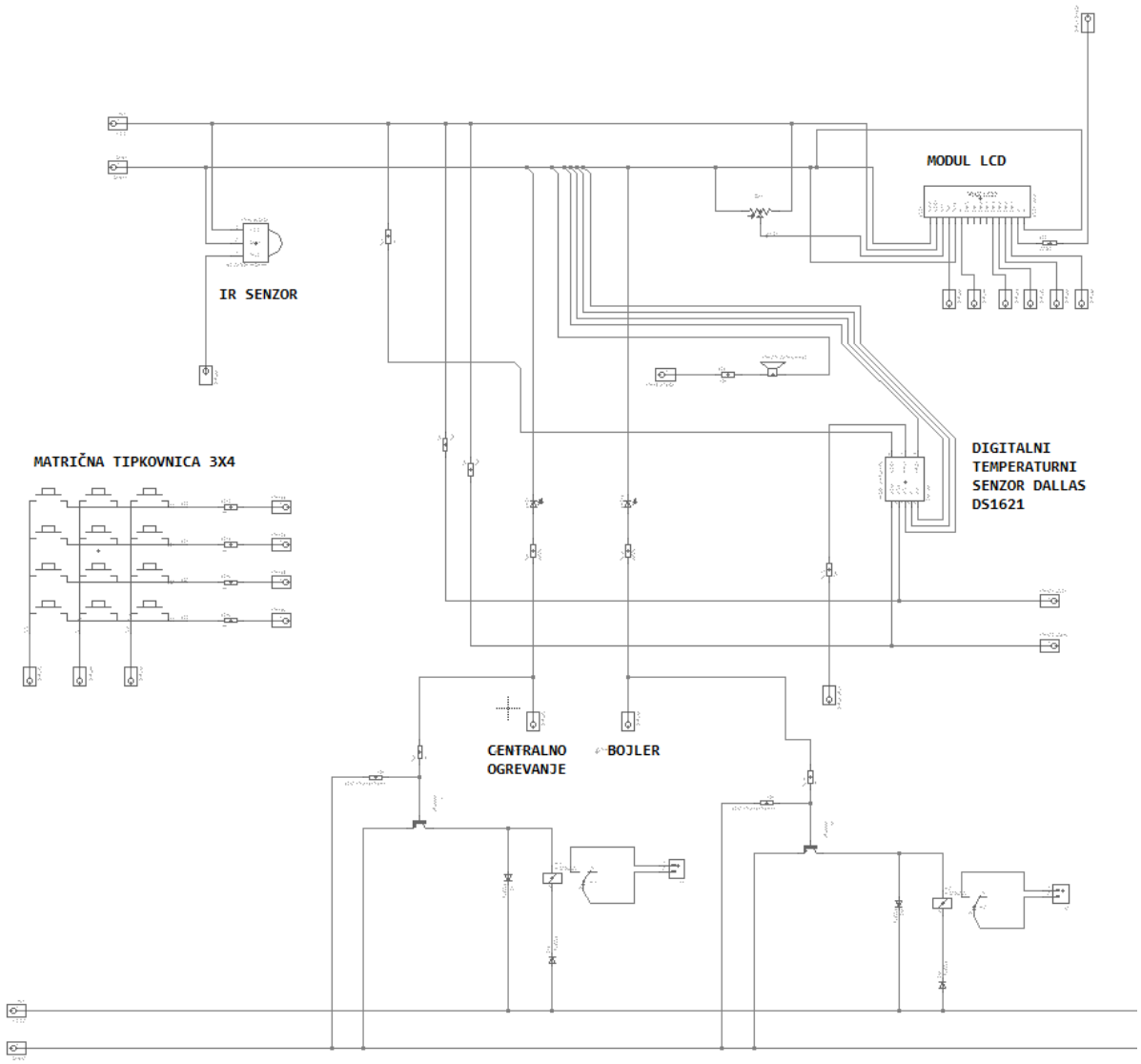
Povezovanje komponent smo že opisali v poglavju 4. Zato se bomo v tem poglavju osredotočili na načrtovanje tiskanine iz narisane sheme vezja.

6.1 Načrtovanje

Shemo vezja smo načrtali v programskem okolju CadSoft EAGLE PCB [7]. Pri načrtovanju smo uporabili še dodatne knjižnice skupnosti Adafruit [11] in skupnosti Sparkfun [12]. Celotno shemo končnega vezja si lahko ogledamo na sliki 6.1.

Številke V/I vrat so drugačne od tistih na prototipni plošči. K diplomskemu delu je priložena shema vezja in krmilni program, prilagojen tiskanemu vezju. Priložen je tudi krmilni program, ki deluje na prototipni plošči. Pri načrtovanju vezja smo upoštevali tudi predvidene razširitve vgrajenega sistema, ki še niso bile programsko izvedene. Zato smo na shemo dodali še infrardeči senzor, ki bo uporabljen za prejemanje ukazov preko daljinskega upravljalnika. Dodali smo tudi močnostni rele, ki bo uporabljen za krmiljenje ogrevanja sanitarne vode.

Ko imamo načrtano shemo moramo narisati še tiskano vezje, ki določa, kje na vezju bodo ležale komponente in kje bodo potekale bakrene povezave. Omenili smo že, da je posebno treba paziti na lociranje temperaturnega senzorja na tiskanem vezju, saj bi

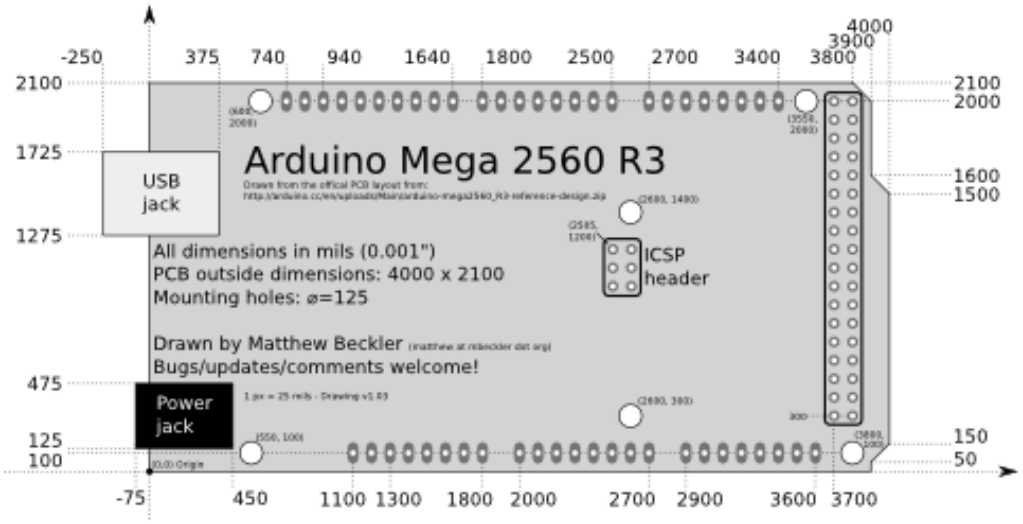


Slika 6.1: Shema končnega vezja.

oddana toplota mikrokrmilniškega sistema lahko vplivala na odčitek temperature. Poleg tega je potrebno natančno paziti na razdalje med posameznimi pini mikrokrmilniškega sistema.

Pri upoštevanju dimenzij, nam je bila v veliko pomoč risba mikrokrmilniškega sistema iz slike 6.2 [28].

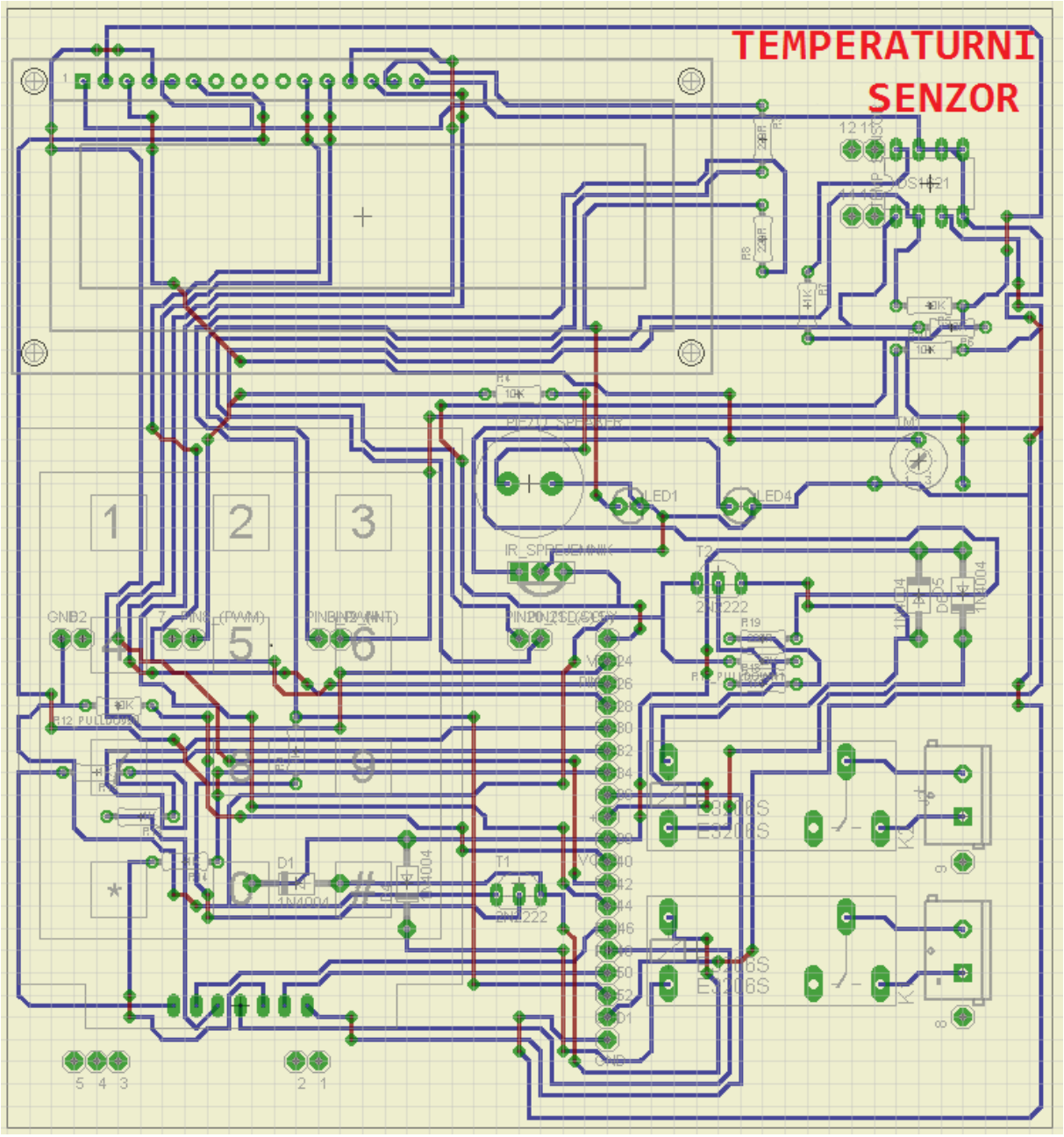
Na sliki 6.3 je lokacija temperaturnega senzorja označena v zgornjem desnem kotu tiskanega vezja. Spodnji levi del vezja vsebuje pine za letvice, s katerimi vezje povežemo z mikrokrmilniškim sistemom. Povezave so morale biti na novo urejene in niso identične



Slika 6.2: Dimenzije mikrokrmilniškega sistema Arduino Mega 2560 (avtor risbe je Davide Gomba). [28]

povezavam na prototipni plošči. Uporabili smo enojne letvice, saj s tem omogočimo več prostora za izris bakrenih sledi.

Razdalje med komponentami, postavitev komponent in ostale dimenzije so ključnega pomena za uspešno izdelavo vezja. Narisano vezje nima nujno odpravljenih vseh napak, saj je pred izdelavo vezja in brez obsežnega testiranja upoštevanje vseh detajlov nehvaležno delo. Paziti je potrebno tudi na širino povezav med relejem in sponko (ang. terminal) (desno spodaj na sliki), saj bo skozi te povezave speljan električni tok s približno vrednostjo $15\,A$ in z napetostjo $220\,V$. Sicer bi zahtevano širino bakrene sledi lahko izračunali, vendar priporočamo, da se za to povezavo uporabi izolirani, enožilni vodnik in se bakreno sled preprosto odstrani. Potrebno je še omeniti, da zaradi prostorske stiske, obligatorne postavitve nekaterih komponent in števila komponent oz. povezav med njimi, enostransko tiskano vezje ni mogoče načrtati. Zaradi pomanjkanja izkušenj z načrtovanjem tiskanih vezij, smo za dokončni izris bakrenih povezav uporabili kar "autorouting" funkcijo programskega okolja EAGLE. Nekatere povezave se nevarno približajo drugim povezavam, kar lahko povzroči potencialne probleme pri jedkanju, saj obstaja nevarnost nastanka kratkega stika. Ta problem bi morali odpraviti z ročnim popravkom povezav, vendar zaradi časovnih omejitev vezje ostaja kot je.



Slika 6.3: Načrtano tiskano vezje s postavitvami komponent.

Poglavje 7

Sklep

Daleč največ težav pri izvedbi projekta smo imeli z elektrotehniškimi problemi, kar je tudi razlog, da smo se odločili za tak projekt. Znanje iz elektrotehnike in načrtovanja tiskanih vezij je bilo namreč pomanjkljivo, zato se je pojavila želja, da se pobližje spopademo s težavami, ki se pojavijo pri načrtovanju vgrajenih sistemov. Znanje seveda ostaja okrnjeno, uspeli pa smo pridobiti pregled nad področjem. Med samim izvajanjem projekta smo tudi pridobili znanja iz višjenivojskih tehnologij, kot je platforma .NET, saj se pred tem z omenjeno tehnologijo nismo bližje srečali. Znanje iz nižjenivojskega programskega razvoja nam med izvedbo ni povzročalo večjih težav. Vseeno nas je malo presenetila količina vloženega dela, ki je bila višja od pričakovane, kar je morda posledica dejstva, da se pred začetkom projekta z nekaterimi uporabljenimi tehnologijami še nismo srečali, vključno s platformo Arduino. Predvsem pa nam je obsežen okvir dela predstavljal razvoj spletnega vmesnika.

Sprva smo nameravali za izvedbo projekta uporabiti mikroprocesor ARM (Advanced RISC Machine). Za izvedbo je bilo potrebno na mikroprocesor naložiti jedro operacijskega sistema Linux, kar nam ni povzročalo težav. Tudi postavitev razvojnega okolja in prevajanje programov nam ni povzročalo večjih težav. Vendar pa se je kaj hitro pokazalo, da bi bil razvoj zaradi pomanjkanja podpore skupnosti dolgotrajen. Zato smo se odločili za platformo Arduino. Problemi s platformo Arduino pa so se pokazali šele ob zaključevanju projekta, saj mikrokrmilniški sistem ni primeren za izvedbo zahtevnejših projektov. Razočaranje nad mikrokrmilniškim sistemom je bilo predvsem posledica pregrevanja regulatorja napetosti in mrežnega krmilnika. Prav tako je frekvenca mikroprocesorja prenizka za računsko kompleksnejša opravila. Nizka pa je tudi perioda vzorčenja na analognih vhodih, ki jih sicer za naš projekt nismo potrebovali, je pa moteča karakteristika

mikrokrmilnika. Za izvedbo našega projekta bi bil morda bolj primeren mikrokrmilniški sistem STM32F4 Discovery [29], razvojna ploščica proizvajalca STMicroelectronics, ki temelji na mikrokrmilniku ARM Cortex M4 [30]. To razvojno platformo odlikuje predvsem ogromna podpora skupnosti, za razvoj programske opreme pa obstajajo tudi dovršena orodja, ki so povečini plačljiva, brezplačno pa si lahko pridobimo okrnjeno različico okolja Atollic TrueSTUDIO for ARM [31]. Mikrokrmilnik ARM Cortex M4 je neprimerljivo zmogljivejši od mikrokrmilnika ATmega2560, ki smo ga uporabili za izvedbo diplomskega dela. Poleg tega je mikrokrmilniški sistem STM32F4 Discovery cenovno nekajkrat ugodnejši od mikrokrmilniškega sistema Arduino Mega 2560. Podporo mrežne komunikacije bi lahko plaformi STM32F4 Discovery dodali s pomočjo razširitvenega modula DP83848 PHY [32], za katerega že obstaja odprtokodna programska podpora [33]. Prav tako je za razvojno platformo že napisana podpora za delo z znakovnimi zasloni LCD in matričnimi tipkovnicami.

Tako je mikrokrmilniški sistem Arduino primeren predvsem za začetniške projekte in učenje, kar je bil tudi namen diplomskega dela. Naučili smo se osnov povezovanja perifernih naprav s poljubnim mikrokrmilnikom in spoznali težave, s katerimi se pri takem projektu srečujemo.

Za izvedbo projekta smo se morali spoznati s principi povezovanja višjenivojskih tehnologij z nižjim nivojem. Za izvedbo spletnega vmesnika smo morali obsežno uporabiti programski jezik Javascript in komplementarno tehnologijo AJAX. Srečali smo se tudi z razvojem na platformi .NET in programskim jezikom C#. Za načrtovanje tiskanega vezja smo morali uporabiti orodje CadSoft EAGLE. Za izvedbo vgrajenega sistema pa smo se morali spoznati z mikrokrmilniškim sistemom Arduino in njegovo platformo, pri tem pa smo si v manjšem obsegu morali umazati roke tudi s spajkanjem in elektrotehniškimi detajli, ki so nujno potrebni za uspešno povezovanje perifernih naprav z mikrokrmilnikom. Pridobili smo pregled nad področjem in se soočili s konkretnimi težavami, s katerimi se srečujemo pri načrtovanju takih sistemov.

Pri načrtovanju spletnega vmesnika smo vsaj delno morali paziti na ustaljeno prakso načrtovanja uporabniških vmesnikov. Potrebno je bilo paziti na principe komuniciranja s podatkovno bazo, zagotavljanje osnovne varnosti uporabniških podatkov z implementacijo prijavljanja v sistem in vsaj v mali meri tudi na prijaznost vmesnika do uporabnika. Kjer je čas dopuščal, smo posebej pazili tudi na informacijsko korekten način programiranja.

Kjer je bilo mogoče, smo za izvedbo projekta uporabili že napisano podporo, saj je bil zaželen hiter razvoj. Kljub temu se je pojavila potreba po dopolnitvi nekaterih

podprtih funkcionalnosti. Tako smo dopolnili odprtokodno knjižnico za delo z digitalnim temperaturnim senzorjem DS1621. Predvsem pa je bil poudarek na logiki sistema in povezovanju komponent v koherentno celoto.

Zanimivo je tudi dejstvo, da so potrebne komponente za izvedbo projekta postale cenovno zelo dosegljive. Če komponente naročimo iz tujine pa lahko stroške nakupa strojnih komponent še zmanjšamo. Nekaj težav nam je povzročala tudi šibka dobavna zmožnost slovenskih podjetij, saj so slovenske trgovine z elektroniko zaradi majhnosti trga slabše založene.

Projekt je namen dokončati v omenjenem obsegu za domačo uporabo. Ostaja še nekaj nerešenih problemov, kot je napajanje vgrajenega sistema in pa varnost komunikacijskega kanala med spletnim strežnikom in vgrajenim sistemom. Potencialne razširitve pa so bile že predvidene pri načrtovanju tiskanega vezja in bi bilo potrebno napisati še ustrezno krmilno podporo, vendar pa bo verjetno potreben nakup še nekaterih komponent, ki nam jih še ni uspelo priskrbeti.

Literatura

[1] Arduino “Arduino“. Dostopno na: http://www.arduino.cc/

[2] Microsoft “ASP.NET MVC 3”. Dostopno na: http://www.asp.net/mvc/mvc3

[3] Arduino “Arduino Mega 2560”. Dostopno na: http://arduino.cc/it/Main/ArduinoBoardMega2560

[4] Atmel “ATmega2560”. Dostopno na: http://www.atmel.com/devices/atmega2560.aspx

[5] Arduino “Arduino Ethernet Shield”. Dostopno na: http://arduino.cc/en/Main/ArduinoEthernetShield

[6] Microsoft “Microsoft .NET”. Dostopno na: http://www.microsoft.com/net

[7] CadSoft “CadSoft EAGLE PCB Design Software”. Dostopno na: http://www.cadsoftusa.com/

[8] Microsoft “Compiling to MSIL”. Dostopno na: http://msdn.microsoft.com/en-us/library/c5tkafs1%28v=vs.71%29.aspx

[9] Microsoft “IIS”. Dosotpno na: http://www.iis.net/

[10] “Visual Web Developer 2010 Express”. Dostopno na: http://www.microsoft.com/visualstudio/en-us/products/2010-editions/visual-web-developer-express

[11] Adafruit “Adafruit Eagle Library”. Dostopno na: https://github.com/adafruit/Adafruit-Eagle-Library

[12] SparkFun “SparkFun Eagle Library”. Dostopno na: https://github.com/sparkfun/SparkFun-Eagle-Library

[13] Dallas Semiconductor “DS1621 Digital Thermometer and Thermostat”. Dostopno na: http://www.produktinfo.conrad.com/datenblaetter/175000-199999/176141-da-01-en-DS_1621_1621S.pdf

[14] WIZnet (2010) “W5100 Datasheet Version 1.2.2”, str. 4-5 in str. 64. Dostopno na: http://www.wiznet.co.kr/UpLoad_Files/ReferenceFiles/W5100_Datasheet_v1.2.2.pdf

[15] Atmel “ATmega640/1280/1281/2560/2561“, maj 2012, pogl. 31, str. 377. Dostopno na: http://www.atmel.com/Images/doc2549.pdf

[16] Arduino “KeypadLibrary”. Dostopno na: http://www.arduino.cc/playground/code/Keypad

[17] Arduino “Digital Pins”. Dostopno na: http://arduino.cc/en/Tutorial/DigitalPins

[18] Raystar Optronics “RC1602B-BIW-ESX”. Dostopno na: http://www.tme.eu/en/Document/53481d6ef60900b7bb1011c164e7f0a0/rc1602b-biw-esx.pdf

[19] Arduino “LiquidCrystal Library”. Dostopno na: http://arduino.cc/en/Reference/LiquidCrystal

[20] I2C Bus “I2C-Bus: What’s that?”. Dostopno na: http://www.i2c-bus.org/

[21] Martin Hans “DS1621-temperature-proble-library-for-Arduino”. Dostopno na: https://github.com/martinhansdk/DS1621-temperature-proble-library-for-Arduino

[22] Arduino “Wire Library”. Dostopno na: http://www.arduino.cc/en/Reference/Wire

[23] Panasonic “Compact PC Board Power Relay - JW Relays”. Dostopno na: http://pewa.panasonic.com/assets/pcsd/catalog/jw-catalog.pdf

[24] Arduino “Arduino: What Adapter?”. Dostopno na: http://arduino.cc/playground/Learning/WhatAdapter

[25] National Semiconductor “LM1117/LM1117I, 800mA Low-Dropout Linear Regulator”. Dostopno na: http://simplemachines.it/Datasheets_mizar/LM1117.pdf

[26] The Jquery Foundation “jQuery”. Dostopno na: http://jquery.com/

[27] Javascript Kit “Ajax”. Dostopno na: http://www.javascriptkit.com/jsref/ajax.shtml

[28] Davide Gomba “Nice drawings of the Arduino UNO and Mega 2560”. Dostopno na: http://arduino.cc/blog/2011/01/05/nice-drawings-of-the-arduino-uno-and-mega-2560/

[29] STMicroelectronics “Discovery kit for STM32 F4 series - with STM32F407 MCU”. Dostopno na: http://www.st.com/internet/evalboard/product/252419.jsp

[30] ARM “Cortex-M4 Processor”. Dostopno na: http://www.arm.com/products/processors/cortex-m/cortex-m4-processor.php

[31] Atollic “Atollic TrueSTUDIO”. Dostopno na: http://www.atollic.com/index.php/truestudio

[32] Texas Instruments “DP83848C”. Dostopno na: http://www.ti.com/product/dp83848c

[33] TKJ Electronics “Ethernet on STM32F4DISCOVERY using external PHY”. Dostopno na: http://blog.tkjelectronics.dk/2012/08/ethernet-on-stm32f4discovery-using-external-phy/