Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Bine Gorjanc

Nadzor hiše na daljavo

diplomsko delo na univerzitetnem študiju

Mentor: prof. dr. Dušan Kodek

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: 01795/2012
Datum: 15.01.2012

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

Kandidat: **BINE GORJANC**

Naslov: **NADZOR HIŠE NA DALJAVO**
REMOTE HOME MONITORING

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Pomemben del lastnosti tako imenovanih "pametnih hiš" je sposobnost za oddaljeni nadzor nad stanjem v prostorih in krmiljenje naprav v njih. Tu je vključeno tudi opozarjanje pred nevarnimi stanji. V ta namen zasnujte in izdelajte rešitev, ki omogoča nadzor in krmiljenje več naprav v hiši preko spletnega vmesnika in preko SMS sporočil. Uporabite Arduino platformo in izdelajte vso potrebno programsko in strojno opremo. Pravilnost delovanja preverite in podajte oceno delovanja ter možne načine za izboljšave.

Mentor:

prof. dr. Dušan Kodek

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

Dekan:

prof. dr. Nikolaj Zimic

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Bine Gorjanc, z vpisno številko **63030119**, sem avtor diplomskega dela z naslovom:

Nadzor hiše na daljavo

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Dušana Kodeka,
- 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 12. aprila 2012 Podpis avtorja:

Zahvaljujem se vsem, ki so mi na kakršenkoli način pomagali pri dosegu te diplome. Še posebej pa mojim staršem, punci in prijateljem za njihovo podporo in zaupanje, ter mentorju Dušanu Kodeku za pomoč.

Kazalo

Povzetek

Vsak med nami, tudi tisti manj tehnološko osveščen, nima druge izbire, kot da sprejme tehnologijo v svoje vsakdanje življenje. To velja tudi za naš dom, kjer se vse več govori o konceptu "pametne hiše", torej hiše, ki ima vdelano elektroniko, namenjeno avtomatizaciji in nadzoru določenih situacij.

Za cilj te diplomske naloge sem si tako zadal izdelati sistem za oddaljeni nadzor in upravljanje z določenimi sistemi v hiši. Predvsem mi je bilo v interesu upravljanje s svetili in opozarjanje pred nevarnimi situacijami (vlom, zmrzovanje, puščanje plina). S tem si lahko zagotovimo prihranek elektrike in pa takojšnje obvestilo ob kritični situaciji. Za doseg tega cilja sem razvil spletno aplikacijo, prek katere sistem nadzorujemo, krmilno aplikacijo, ki nadzoruje mikrokrmilnik, in pa knjižnico, ki skrbi za komunikacijo med spletnim strežnikom in mikrokrmilnikom.

Ključne besede

Pametna hiša, avtomatizacija, nadzor svetil, splet, obveščanje v kriznih situacijah

Abstract

Every one of us, even the less technologically aware, has no choice but to accept technology into his day to day life. This also includes our home, where the term ”smart house” is more and more frequently heard. It means a house which has built in electronics in order to automate and control certain situations.

My goal for this thesis was to create a system for remote control and manipulation of certain systems in a house. Lighting control and warnings on dangerous situations were my first priorities. Lighting control is a potential electricity saver, while environment control ensures immediate notification in critical situations. In order to achieve this goal I have developed a web application which is used to control the system, control application which controls the microcontroller and a library which takes care of communication between the web server and the microcontroller.

Keywords

Smart house, automatization, web, lighting control, critical situations notifications

Poglavje 1

Uvod

Kdorkoli ima v lasti objekt, od njegovega doma oddaljen več kot nekaj kilometrov, se je zagotovo že srečal s problemom nadzora in upravljanja. Pozimi je potrebno preveriti, če vodovodne cevi ne zmrzujejo in vedno je dobro vedeti, da plinska napeljava ne pušča, ali pa da v objektu nimamo nezaželenega gosta.

V ta namen sem si zamislil sistem, kjer lahko prek spletnega vmesnika ali SMS sporočila dobimo stanje objekta, obenem pa ima sistem vgrajene dovolj avtomatike, da se zna tudi sam odzvati na določene primere. Tako recimo ob prekoračenih mejnih vrednostih dobimo obvestilo v obliki opozorilnega SMSa, ob nastavljenem avtomatskem delovanju luči pa zna ob vstopu človeka v prostor (če je ta prostor dovolj temen) samodejno prižgati luči zanj.

Spletni vmesnik je napisan v okolju .NET, teče pa na spletnem strežniku IIS, kar pomeni, da mora biti zraven mikrokrmilnika priključen tudi PC, na katerem strežnik teče. Prvotno sem sicer spletni strežnik imel namen implementirati kar na mikrokrmilniku, a sem ta načrt opustil zaradi prevelike časovne zahtevnosti. Dodatna pomankljivost je tudi ta, da je kapaciteta spletnega strežnika, ki bi tekel na mikrokrmilniku, zelo omejena. Mikrokrmilnik ima nato nalogo upravljati z vhodi in izhodi glede na prebrane vrednosti senzorjev, ter ukaze s spletnega vmesnika.

Na trgu sicer obstaja že kar nekaj podobnih izdelkov, vendar je bil zame

to tudi svojevrsten izziv, kako z dokaj cenenimi komponentami izdelati funkcionalen sistem, ki bo v pomoč in dejansko namenjen vsakodnevni uporabi. To je razvidno tudi v izbiri komponent, kjer je bil najvišji posamezen strošek nakup Arduino mikrokrmilnika.

Poglavje 2

Mikrokrmilnik Arduino Mega 2560

Za osnovo diplomske naloge je bil izbran mikrokrmilnik *Arduino Mega 2560*, predvsem zaradi, v primerjavi z ostalimi Arduino mikrokrmilniki, velikega števila vhodov in izhodov in večje količine delovnega spomina.

2.1 Lastnosti

Arduino Mega 2560 temelji na *ATmega2560* procesorju in ima:

- 54 digitalnih vhodov/izhodov
- 16 analognih vhodov
- 4 UART priključke
- 16 MHz hitrost ure
- 8 KB SRAM pomnilnika
- 256 KB Flash pomnilnika
- 4 KB EEPROM pomnilnika

Vsaka V/I nožica zmore dobavit ali prejeti 40 mA toka, nožica, ki zagotavlja napajalno napetost 3.3V pa 50 mA toka [1]. To sicer ni dovolj za večje porabnike, kot so npr. razni motorčki in podobno, je pa dovolj, da take večje porabnike krmilimo z releji, priklopljenimi na mikrokrmilnik.

2.2 Razvojno okolje

Krmilna aplikacija za Arduino je razvita v razvojnem okolju Arduino 1.0, ki je izšel ravno malo pred začetkom pisanja diplomske naloge. Prej so bile na voljo "beta"verzije tega razvojnega okolja.

2.2.1 Programski jezik

Programi za Arduino mikrokrmilnik so napisani v jeziku C, tako da lahko uporabljamo tudi ukaze, kot so *malloc* in *strcmp*, kar pomeni, da lahko nekdo, ki že pozna ta jezik, hitro piše tovrstne programe. Ima pa poleg standardnih ukazov jezika C seveda tudi funkcije za nadzor Arduina [4]. Npr.:

- pinMode() - določimo način delovanja neke nožice (vhod/izhod)
- analogRead() - branje vhoda v analognem načinu, kar pomeni, da vrne vrednost od 0 do 1023
- digitalRead() - branje vhoda kot logična 0 ali 1
- millis() - trenutni čas izvajanja programa, izražen v milisekundah

2.3 Krmilni program

Krmilni program (skica oz. "sketch") je razdeljen v več modulov:

- Osnovni del, ki pravzaprav vsebuje samo komentar z opisom programa.
- Glavni del, ki vsebuje inicializacijsko metodo *setup* in zanko *loop*.

- GSM modul, ki skrbi za pošiljanje SMS sporočil.
- Modul, ki skrbi za shranjevanje in priklic nastavitev.
- Pomožni modul, kjer so implementirane funkcije nadzora, npr. klicanje funkcij glede na prejeti ukaz.
- Modul, ker so prek DEFINE ukaza definirane konstante.
- Senzorski modul, kjer so zbrane vse funkcije, ki upravljajo s senzorji.
- Komplementarni modul, ki vsebuje pomožne funkcije, nepovezane s funkcionalnostjo. To je npr. logiranje napak.

Delitev je v največji meri namenjena lažjemu nadzoru nad izvorno kodo programa, saj bi drugače kmalu postala neobvladljiva. Izvedena pa je s pomočjo več datotek, ki so na voljo v eni skici.

2.3.1 Glavni modul

Glavni modul vsebuje inicializacijo vseh potrebnih objektov znotraj metode *setup*, kot na primer inicializacija serijskega vmesnika, krmilnika SD kartice in pa ostalih modulov. V glavni zanki *loop* pa nadzorujemo vhodne parametre, ki prihajajo ali iz senzorjev, ali pa s spletne strani. To počnemo s klicem funkcij, ki se nahajajo v pomožnem modulu.

2.3.2 GSM modul

Modul je sposoben pošiljati opozorilna kratka sporočila na številko, ki je shranjena kot kontaktna številka. Zaenkrat je predvidena uporaba samo ene številke, je pa to zelo enostavno spremeniti. Več informacij je na voljo v poglavju 4.

2.3.3 Modul za nastavitve

Nastavitve se shranjujejo v tekstovne datoteke, ki se nahajajo na SD kartici. Pod nastavitve spadajo:

- Sobe, ki jih obvladujemo, ta imena pa so potem uporabljena na grafičnem vmesniku spletne strani. Tudi urejamo jih lahko prek spletne strani.
- Kontaktna GSM številka, ki je uporabljena, ko je potrebno poslati SMS z obvestilom.
- Lokacije senzorjev. Ta imena so potem ravno tako uporabljena pri pregledu senzorjev.

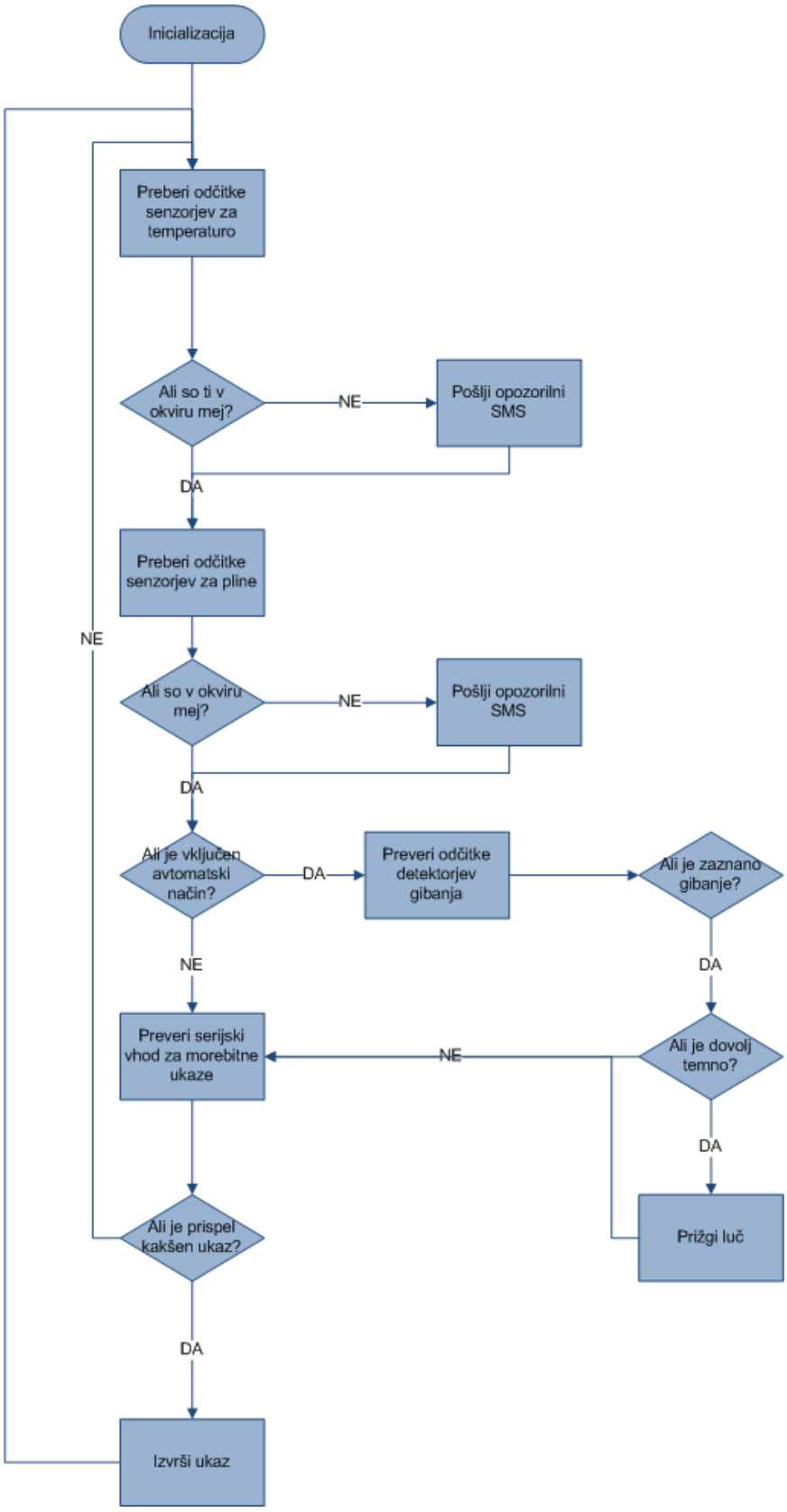
Za zapisovanje nastavitev na SD kartico je bila uporabljena knjižnica SD.

2.3.4 Pomožni modul

Pomožni modul vsebuje funkcije, ki so klicane iz glavne zanke programa. Te namreč procesirajo vhodne parametre in se na njihovi podlagi potem odločajo, katere akcije je potrebno izvesti. Od tam se tako kličejo funkcije za pošiljanje SMS sporočil, avtomatski prižig luči, procesiranje sprejetega ukaza, itd.

2.4 Ethernet ščit W5100

Poleg Arduina Mega sem kupil še Ethernet "ščit", ki je nataknjen na mikrokrmilnik in mu prek njegovih vhodov omogoča interakcijo prek Ethernet priključka, ima pa tudi vgrajen čitalec SD kartic. Na začetku sem mislil s tem ščitom realizirati spletni strežnik, a sem si premislil, zato je ta ščit uporabljen izključno zaradi čitalca SD kartic, kamor se shranjujejo nastavitve.



Slika 2.1: Osnovni potek krmilnega programa

Poglavje 3

Spletna aplikacija

Okolje .NET je razvil Microsoft, različica 1.0 pa je bila izdana leta 2002. Teče v okolju CLR (Common Language Runtime), kar omogoča razvoj projektov v različnih programskih jezikih, saj je to okolje skupno vsem jezikom okolja .NET [3]. Ne ravno zanemarljiva prednost okolja .NET pa je tudi podpora razhroščevanju, ki je v okolju Arduino zelo oteženo, saj imamo na voljo samo izpis stanja spremenljivk prek serijskega vhoda. Za mojo aplikacijo sem si izbral jezik C#.

Spletna aplikacija, s katero nadzorujemo in krmilimo sistem, je napisana z uporabo tehnologije MVC3 v Microsoft .NET okolju. To nam omogoča hiter razvoj zmogljive aplikacije, ki teče na strežniku IIS.

Poglavitna prednost, ki jo nam prinese tovrstna aplikacija je predvsem to, da so nekateri mehanizmi, ki bi jih drugače morali posebej razviti, že avtomatsko vključeni. To so recimo prijava uporabnika v sistem, branje GET in POST spremenljivk, kodiranje in dekodiranje znakov, pa še mnogo več. Vse to se seveda da implementirati tudi v Arduinu, ampak bi zahtevalo preveč časa.

3.1 Inicializacija aplikacije

Ob zagonu aplikacija najprej prebere nastavitve serijskega vhoda, ki so shranjene v konfiguracijski datoteki *Web.config*, s temi pa kreira novo instanco knjižnice, ki skrbi za komunikacije spletne aplikacije z mikrokrmilnikom.

3.2 Delo z aplikacijo

Uporabnik se mora za začetek dela najprej prijaviti v aplikacijo, kar stori na začetnem zaslonu. Če hoče brez tega dostopati do vsebine, ga aplikacija samodejno preusmeri na zaslon za prijavo. Iz očitnih razlogov seveda ne moremo dopustiti neprijavljenim uporabnikom, da upravljajo z aplikacijo.

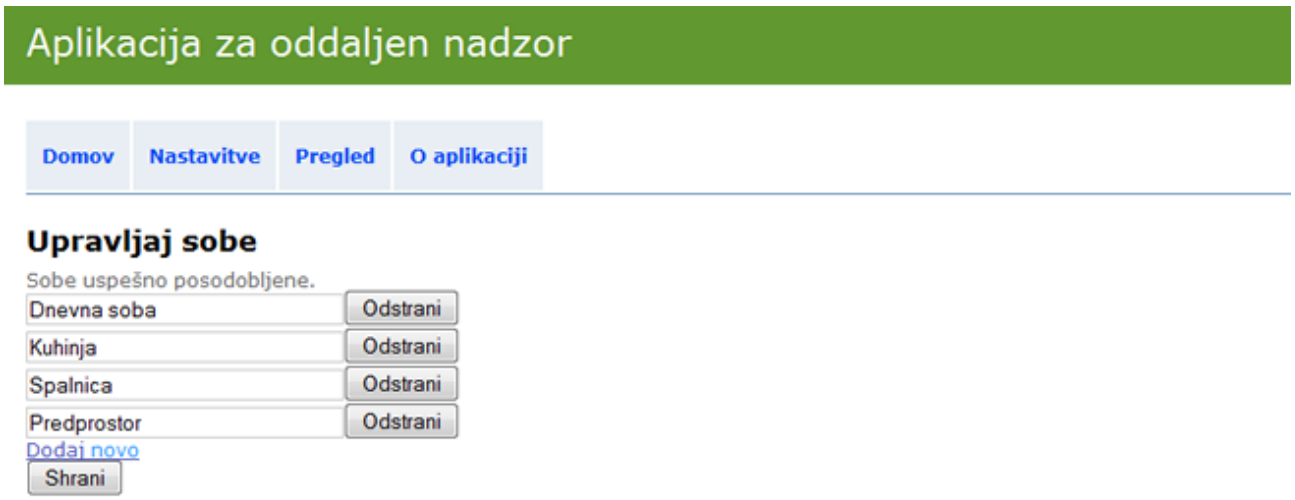
Aplikacija nato vsebuje več mask, preko katerih lahko upravljamo in pregledujemo podatke z mikrokrmilnika.

3.2.1 Upravljanje s sobami

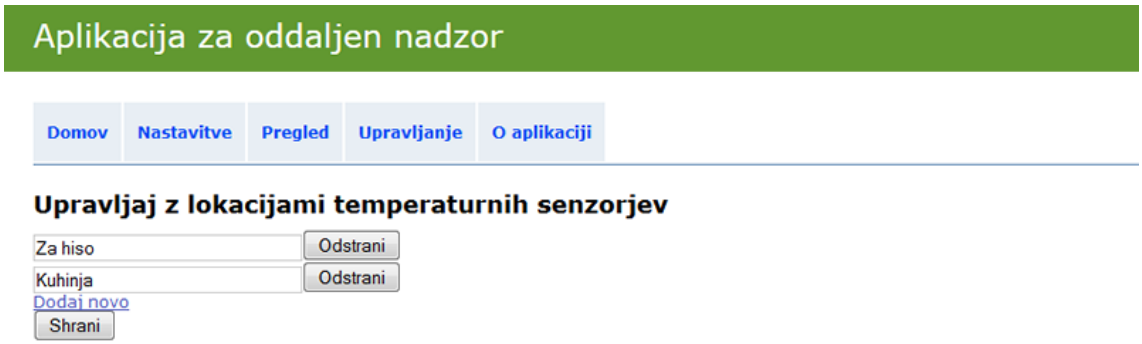
Na tej maski lahko dodajamo, brišemo in urejamo sobe, ki jih nadzorujemo. Ti podatki so potem uporabljeni pri preverjanju razsvetljave in pri zaslonu za pregled stanja detektorjev gibanja. Mikrokrmilnik namreč preverja toliko sob, kot jih je navedenih, seveda do prej določene meje. Ta preprečuje, da bi priključkov za priklop senzorjev zmanjkalo, saj so ti od neke meje naprej namenjeni drugim senzorjem.

3.2.2 Upravljanje s senzorji

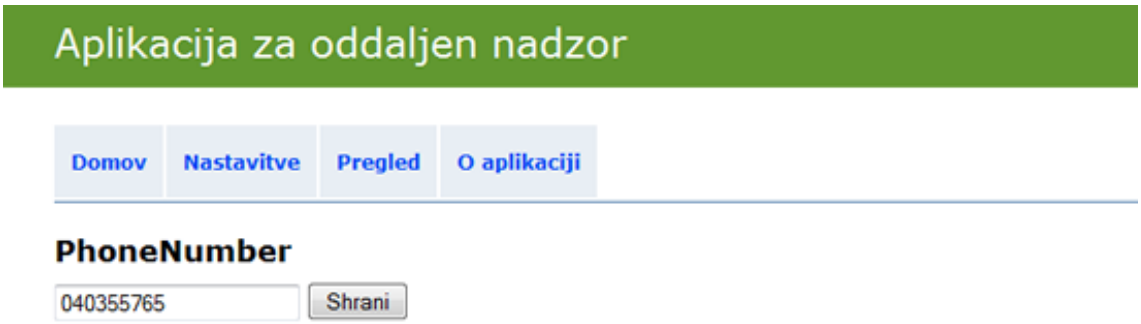
Upravljanje s temperaturnimi in senzorji za plin je zelo podobno, ravno tako kot z upravljanjem s sobami. Ob vnašanju se namreč preverja, če število vnešenih senzorjev ni večje od števila priključkov, namenjenih zanje.



Slika 3.1: Zaslonska maska za upravljanje s sobami



Slika 3.2: Zaslonska maska za upravljanje s temperaturnimi senzorji



Slika 3.3: Upravljanje s kontaktno številko

3.2.3 Upravljanje kontaktne telefonske številke

Tu definiramo številko, ki je uporabljena za pošiljanje SMSa v primeru, da pride do izrednega stanja. Če številka ni shranjena in pride do tega stanja, se SMS seveda ne pošlje.

3.2.4 Pregled odčitkov senzorjev

Tu so zbrani odčitki temperaturnih senzorjev, senzorjev za plin in pa senzorjev za gibanje. Prikaz je razdeljen v sekcije, stanje pa lahko z osvežimo z gumbom, namenjenim za to.

3.2.5 Pregled posebnih dogodkov, ki jih je zabeležil mikrokrmilnik

Posebni dogodki, kot so razne napake, se zabeležijo v posebno, za to namenjeno, datoteko na SD kartici. Prek te maske lahko dostopamo do teh informacij.

Slika 3.4: Zaslonska maska, kjer lahko pregledamo trenutne odčitke senzorjev

3.3 Knjižnica za komuniciranje med spletno aplikacijo in Arduinom

V knjižnici imamo metode, namenjene krmiljenju in zahtevanjem podatkov od mikrokrmilnika, izpostavljene na tako imenovanem "fasadnem"objektu, ki potem kliče implementacije v ozadju.

Knjižnica je zasnovana tako, da preprečuje, da bi zahteve, ki pridejo istočasno, motile ena drugo pri izvajanju. To sem dosegel tako, da vsak klic doda zahtevo v statično čakalno vrsto. Za obdelavo posamezne zahteve pa se najprej zaklene kritični del kode, ki odpira/zapira serijski vhod in obdeluje zahtevo, po dokončanem delu pa se zaklep spet sprosti. Zaklep se izvaja nad nekim statičnim objektom s ključno besedo *lock*. Omenjeno sinhroniziranje zahtev je seveda kritično za primer, ko bi aplikacijo uporabljalo več uporabnikov hkrati.

Pri komuniciranju sem imel tudi težave v primeru, da sem za hitrost prenosa nastavil 115200bps. Takrat je Arduino prek serijskega vhoda dobil poleg željenih še začetne 4 dodatne znake, za kar nisem našel rešitve, razen če bi prve 4 znake preprosto odrezal. Ko sem kot hitrost nastavil 9600bps je sprejem znakov deloval brezhibno.

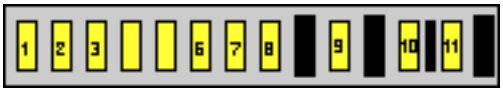
Poglavje 4

GSM vmesnik

Ker sem hotel, da lahko sistem tudi obvešča v primeru napak ali nujnih situacij, je bil sistemu dodan GSM vmesnik za pošiljanje SMS sporočil. Ta je zgrajen okoli mobilnega telefona Sony Ericsson K700i, ki je z Arduinom povezan prek serijskega vmesnika s pomočjo podatkovnega kabla, predelanega v ta namen.

Lahko bi sicer uporabil kakšnega od namenskih GSM modulov, a so ti neprimerno dražji. Aparat, ki ga tu uporabljam, me je namreč stal 3€, medtem ko cene modulov hitro dosežejo 70€ ali več. Poleg tega mislim, da kupovanje takšnih namenskih modulov tudi ni namen diplomske naloge.

Slika 4.1: Uporabljen Sony Ericsson K700i



Slika 4.2: Vezava priključkov za podatkovni kabel.

4.1 Vezava

Podatkovni kabel, s katerim je mobilni telefon povezan na Arduino, je bil prvotno sicer mišljen za USB priklop, a je bil za to nalogo nekoliko predelan. To pa tako, da je bil USB priključek odščipnjen, povezave za serijski prenos podatkov speljane na Arduinove podatkovne priključke, poleg tega pa je bil še GND signal speljan na GND Arduina. Kot izvor napajanja za mobilni telefon služi star polnilec za Sony Ericsson telefone z odščipnjenim konektorjem in speljano žico za napetost na ustrezni priključek na podatkovnem konektorju. Delovanje mobilnega telefona je tako zagotovljeno.

Arduino Mega podpira 4 serijske vhode/izhode, tako da sem lahko mobilnik priključil na serijski vhod 1, hkrati pa še vedno ohranil serijski vhod 0 za komunikacijo s PCjem.

4.2 Programska implementacija

Prek serijskega vmesnika lahko mobilnemu telefonu pošiljamo ukaze, ki simulirajo praktično kakršnokoli uporabniško interakcijo, prav tako pa tudi sistemske ukaze, kakršen je recimo ukaz za pošiljanje sporočil.

Tu se na žalost stvar nekoliko zakomplicira, saj moramo ukaz in tekst sporočila posredovati v formatu PDU, ki zahteva specifičen format podatkov, potrebnih za pošiljanje [9] [10]. Na srečo pa obstaja že nekaj primerov uporabe, s katerimi sem si pomagal [11].

PDU format je okrajšava za "Protocol Data Unit" in se uporablja, kjer mobilnik ne podpira tekstovnega načina pošiljanja SMSov. S tem formatom

Priključek	Ime	Smer	Opis
1	ATMS	<	Avdio za mobilnik
2	AFMS/RTS	>	Avdio iz mobilnika
3	CTS/ONREQ	-	CTS/Zahteva za vklop
4	data in	<	Podatki za mobilnik (Rx)
5	data out	>	Podatki od mobilnika (Tx)
6	ACC in	<	Nadzor pripomočkov za mobilnik
7	ACC out	>	Nadzor pripomočkov od mobilnika/zaznava prostoročne naprave
8	AGND	-	Ozemljitev avdio signala in 0V referenca
9	flash	-	Voltaža flash pomnilnika in servisni priključek
10	DGND	-	Digitalna ozemljitev
11	Vcc	-	DC + za polnjenje baterije in napajanje zunanjih pripomočkov

Tabela 4.1: Opis priključkov na podatkovnem kablu, ki povezuje Arduino mikrokrmilnik in mobilni telefon

lahko pošiljamo tudi vsebino, ki je s tekstovnim načinom ne moremo, npr. binarne podatke [12].

Pri pretvorbi sporočila v PDU format moramo podati tudi še nekatere druge parametre. Rezultat pretvorbe je niz heksadecimalnih znakov, s katerimi mobilniku sporočamo dolžino kontrolnega zaporedja, telefonsko številko naslovnika, tekst sporočila in še nekaj drugih ukazov.

V tabeli 4.2 je PDU niz za sporočilo s tekstom "hellohello", poslano na številko +386 12 345 678. Ta niz je tudi razbit na posamezne dele za razlago posameznih komponent ukaznega niza.

Za pretvorbo teksta v PDU format moramo najprej vsak znak teksta pretvoriti v decimalno obliko, tega pa potem v 7 bitno binarno obliko. Te potem pretvorimo v bajte (8 bitne binarne znake) tako, da jemljemo bite z desne strani naslednjega znaka, ki jih pripnemo trenutnemu. Ker zdaj temu naslednjemu znaku ostane samo še 6 bitov, to pomeni, da še naslednjemu znaku vzamemo 2 bita, še naslednjemu 3 in tako naprej... V spodnji tabeli ponazarjajo odebeljeni biti tiste bite, ki jih določenemu znaku odvzamemo, da lahko z njimi dopolnimo prejšnji znak. Ko pridemo do konca pa nastale binarne znake pretvorimo v heksadecimalno obliko. Če na koncu ostane manj kot 8 bitov, namesto njih na levi strani vstavimo ničle.

V programu je to rešeno s pomikanjem bitov.

4.3 Uporaba v aplikaciji

Pošiljanje SMS sporočil je izvedeno preprosto, saj je potrebno le poklicati ustrezno funkcijo z želenim tekstom, ostalo pa se uredi samo. Telefonska številka je namreč shranjena v tekstovni datoteki, tako da jo je potrebno le prevesti v ustrezen format in uporabiti.

Pri pošiljanju sporočil sem se tudi hotel zavarovati pred poplavo poslanih sporočil, če bi vrednost senzorjev večkrat v kratkem času prestopila mejno vrednost. V ta namen v programu obstajajo spremenljivke, ki vsebujejo zadnje čase, ko je bilo poslano sporočilo zaradi prekoračene mejne vrednosti

Bajti	Opis
00	Dolžina SMSC informacije. Če je 0, to pomeni, da naj bo uporabljen SMSC, shranjen v telefonu.
11	Prvi bajt SMSC-SUBMIT
00	Referenca sporočila. Pri 0 je referenciranje prepuščeno telefonu.
0B	Dolžina telefonske številke naslovnika
91	Tip telefonske številke. 91 pomeni mednarodni format.
8316325476F8	Telefonska številka naslovnika. Generirana je tako, da pri vsakemu paru številk v telefonski številki obrnemo številki (npr. pri 1234 dobimo 2143). Ker je dolžina številke liha, ji na koncu dodamo še znak F.
00	Identifikator protokola TP-PID.
00	Format kodiranja podatkov TP-DCS. 00 pomeni, da je uporabljena 7 bitna abeceda.
AA	Veljavnost sporočila. Če sporočilo ne more biti dostavljeno pred tem rokom, je zavrženo. AA tu pomeni 4 dni veljavnosti.
0A	Dolžina teksta v sporočilu. Pri nas je to 0A (10 v decimalnem zapisu).
E8329BFD4697D9EC37	Zakodiran tekst sporočila. Več informacij o pretvorbi je spodaj.

Tabela 4.2: Primer PDU ukaznega zaporedja, razbitega na komponente

določenega senzorja. Ob inicializaciji pa so elementi postavljeni na neko zelo visoko negativno vrednost. Če je bilo zadnje sporočilo za nek senzor poslano v okviru prednastavljene vrednosti (trenutno 6 ur), se sporočilo še ne pošlje.

ASCII	Decimalno	Binarno
h	104	1101000
e	101	1100101
l	108	1101100
l	108	1101100
o	111	1101111
h	104	1101000
e	101	1100101
l	108	1101100
l	108	1101100
o	111	1101111

Tabela 4.3: Pretvorba znakov v binarni zapis.

Septet	Bajt	Heksadecimalno
1101000	11101000	E8
110010**1**	00110010	32
1101**100**	10011011	9B
110**1100**	11111101	FD
11**0001111**	01000110	46
1**1001000**	10010111	97
1100101	11011001	D9
1101100	/	/
1101100	11101100	EC
110111**1**	110111	37

Tabela 4.4: Tvorba bajtov iz septetov znakov teksta.

Poglavje 5

Strojna izvedba

V tem poglavju je predstavljeno, kako so strojne komponente povezane med seboj.

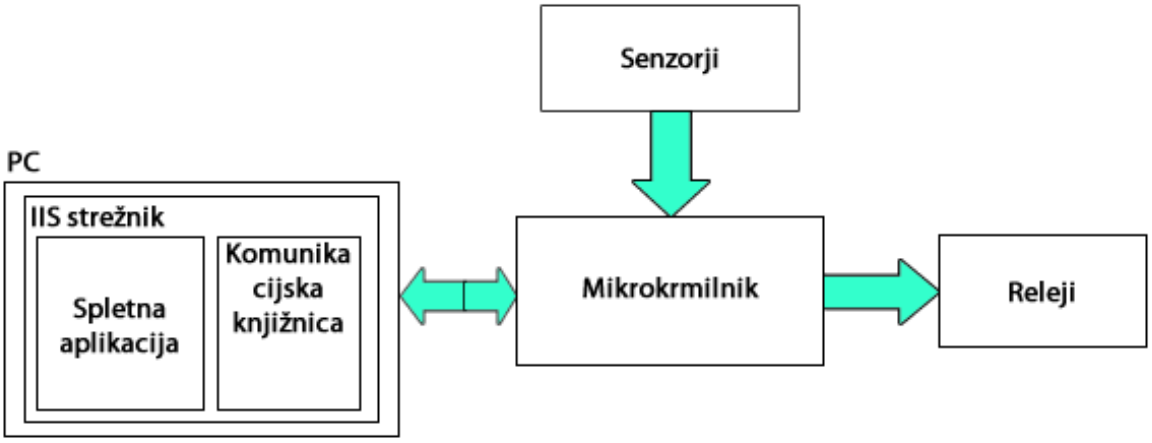
5.1 Pregled celotnega sistema

Osnovni gradnik sistem je seveda mikrokrmilnik Arduino Mega 2560, opisan v poglavju 2. Z njim komunicira spletna aplikacija prek priključka USB, ki teče na IIS strežniku. Z njim si tako izmenjujeta ukaze in podatke. Poleg tega nadzoruje senzorje in glede na ukaze in podatke s senzorjev pošilja signale na releje, ki krmilijo sistem.

5.1.1 Preklop med ročnim in avtomatskim načinom

Pri načrtovanju nadzora sem naletel na problem, kako med seboj uskladiti ročne in avtomatizirane ukaze. V ta namen sem dodal še gumb za preklop med avtomatskim in ročnim načinom. V avtomatskem sistem spoštuje ukaze z mikrokrmilnika (torej tudi spletnega vmesnika), v ročnem pa ukaze, podane prek stikal na kraju samem.

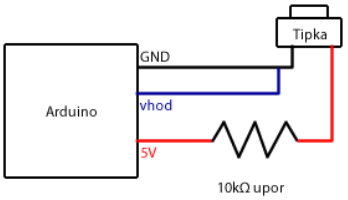
Mikrokrmilnik ima za spremljanje trenutnega načina delovanja interno boolean spremenljivko, zato lahko stanje spremenimo tudi prek spletnega



Slika 5.1: Pregled vezave osnovnih komponent

vmesnika. Ker je potrebno za vstop v spletni vmesnik vnesti uporabniško ime in geslo ni bojazni, da bi stanje spreminjal nekdo brez naše vednosti.

Spreminjanje načina prek gumba poteka tako, da gumb držimo pritisnjen vsaj 2 sekundi. Ko mikrokrmilnik zazna preklop stanja preveri še trajanje pritiska in stanje preklopi. Sama detekcija pritiska je izvedena prek dveh internih spremenljivk, od katerih ena hrani staro stanje gumba, ena pa trenutnega. Ko zazna preklop se shrani stanje trenutnega časa (funkcija *millis*), ko pa zazna še, da uporabnik drži gumb dovolj časa, se stanje preklopi. Obenem se shrani še boolean spremenljivka, ki označuje, da se je stanje že preklopilo in naj več ne upošteva gumba, dokler uporabnik gumba ne spusti.



Slika 5.2: Vezava gumba za preklop med ročnim in avtomatskim načinom

5.2 Uporabljeni senzorji

5.2.1 Senzor svetlobe

Senzor svetlobe je senzor, s katerim zaznavamo nivo svetlobe. To je v bistvu fotoupor, ki glede na nivo svetlobe spreminja svojo upornost. Je poceni in zanesljiv, ni pa ravno zelo natančen, saj se lahko odčitki različnih primerkov enakih senzorjev med seboj zelo razlikujejo (tudi do 50%). Za trenutni namen je pa to povsem dovolj, saj nas zanima samo, če je okolje dovolj temno, da se luč prižge.

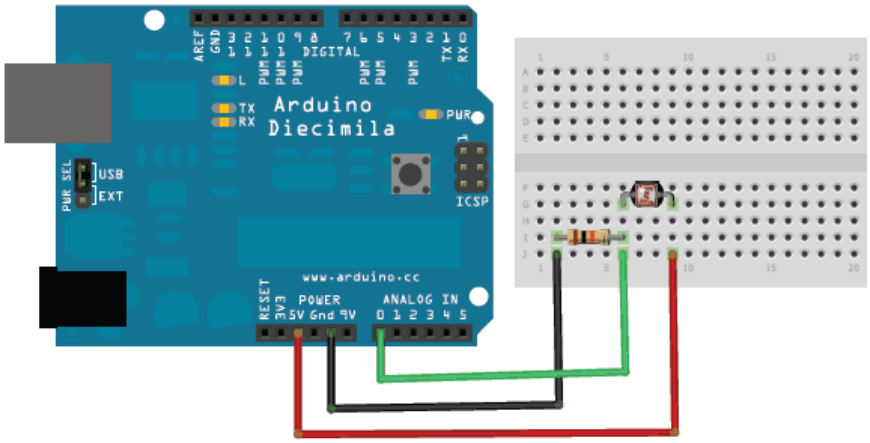
Slika 5.3: Senzor svetlobe oz. fotoupor

Vezava

Senzor na eni strani vežemo na napetost +5V, na drugi pa na ozemljitev prek 1kΩ upora. Z nogice, kamor je vezan na upor, peljemo signal na analogni vhod, kjer lahko določimo padec napetosti in s tem nivo svetlobe [7].

5.2.2 Senzor gibanja

Za zaznavanje gibanja je uporabljen PIR detektor gibanja (*Passive Infrared*), ki zaznava spremembo stanja infrardečega sevanja v prostoru in s tem tudi gibanje. Zaradi načina delovanja se mora senzor na začetku kalibrirati, da si ustvari sliko normalnega stanja okolice. Ko se to stanje spremeni, odda visok signal, ki označuje gibanje v prostoru. Kalibracija naj bi trajala nekje od 10 do 60 sekund. Sam sem za trajanje kalibracije nastavil čas 30 sekund. Doseg senzorja pa je približno 7 metrov [8].



Slika 5.4: Vezava fotoupora.

Slika 5.5: Senzor gibanja

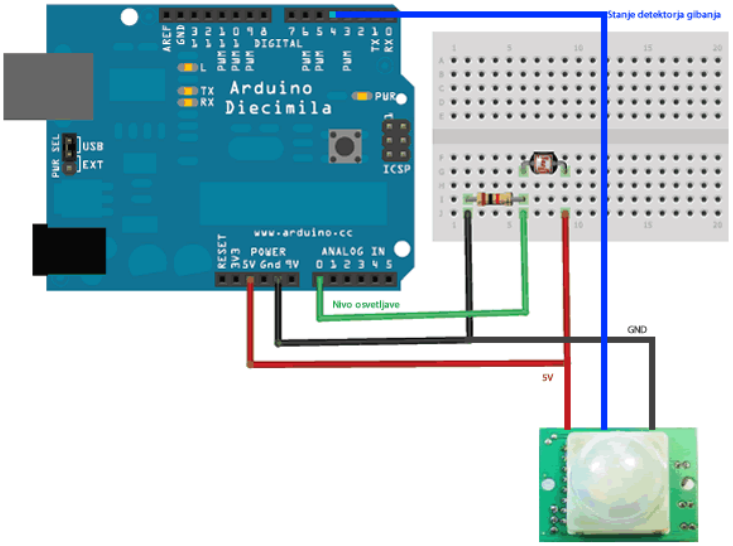
Vezava

Senzor ima Vcc, GND in OUT nožico, ki preide v visoko stanje, ko zazna gibanje. Tam vztraja še nekaj časa po tem, ko gibanja ni več (trajanje je nastavljivo), potem pa signal preide nazaj v nizko stanje.

5.3 Nadzor svetil

Nadzor je implementiran z uporabo senzorjev svetlobe in senzorjev gibanja. Vezava senzorjev pa je precej preprosta. Ko namreč senzor gibanja zazna pre-

mik se najprej preveri še izhod senzorja svetlobe. Če ta ne presega določene meje, se prek izhodnega priključka Arduina sproži rele, ki prižge luč. Najprej sem sicer hotel vsaki sobi nameniti en senzor svetlobe, ampak sem po razmisleku ugotovil, da je dovolj, če uporabim enega, ki meri svetlobo zunaj objekta.

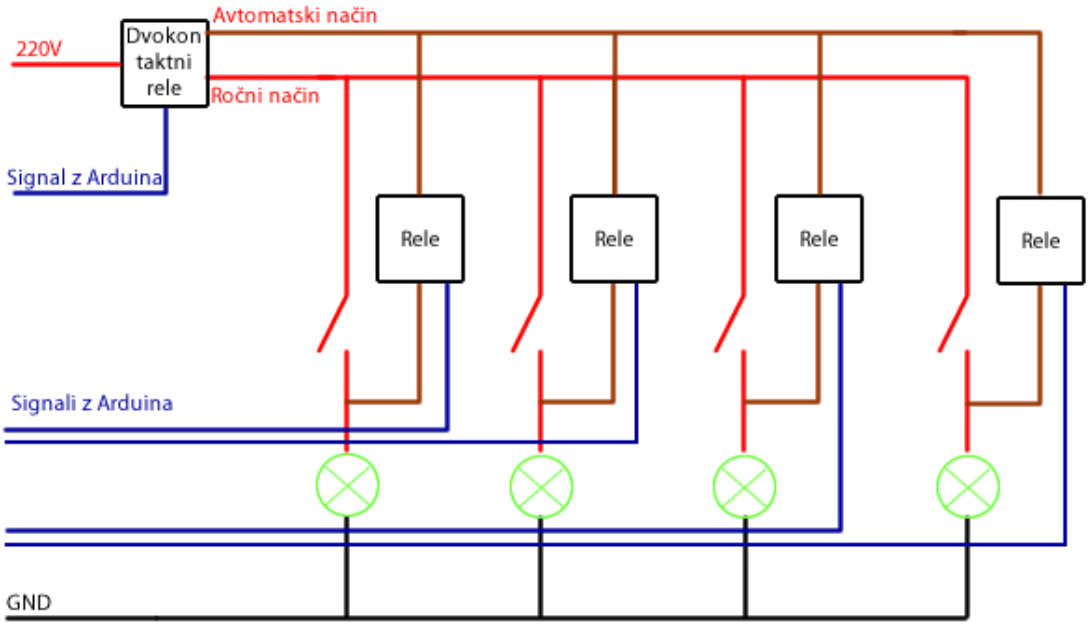


Slika 5.6: Vezava fotoupora skupaj z detektorjem gibanja.

Preklapljanje med ročnim in avtomatskim načinom je izvedeno prek ločenih vezav, ki jih nadzoruje dvokontaktni rele. Glede na signal z mikrokrmilnika tako tok teče po ustrezni veji. Če je nastavljen ročni način, luči upravljamo normalno prek stenskih stikal, če pa je aktiven avtomatski način, pa se luči vklapljajo glede na signale z Arduina, ki krmilijo releje luči.

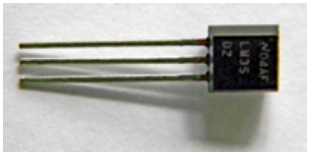
5.4 Opozarjanje na nevarne koncentracije plinov

Moja želja pri dizajniranju sistema je bila tudi, da bi sistem znal reagirati na previsoko koncentracijo nevarnih plinov. V ta namen je koncentracija mer-



Slika 5.7: Okvirna shema preklapljanja med ročnim in avtomatskim nadzorom luči.

Slika 5.8: Senzor za merjenje koncentracije plinov.



Slika 5.9: Senzor za temperaturo.

jena prek senzorja MQ6, ki meri koncentracijo propana, butana, ter kuhinjskega in zemeljskega plina, ter razne vnetljive pline, kot je metan. Zmožen je meriti koncentracijo plinov od 200 do 10000 delcev na milijon. Da vzpostavi pravilno delovno okolje, pa se mora najprej segreti na delovno temperaturo [13]. Vezava je preprosta. Ena nožica se veže na 5V, ena na zemljo, ena pa na analogni vhod. Mejo za obveščanje je sicer težko nastaviti, sem pa jo sam nastavil na 800 delcev na milijon [14].

5.5 Opozarjanje na nevarnost zmrzovanja

Za zaznavanje možnosti zmrzovanja so uporabljeni temperaturni senzorji LM35, ki so dokaj natančni, temperaturo namreč zaznavajo na 0,5 stopnje Celzija. Njihovo območje delovanja pa sega od -55 do +150 stopinj Celzija, kar je povsem dovolj za naše potrebe [5]. Tudi priklop je enostaven. Ena nožica se veže na 5V, druga na zemljo, tretja pa na vhodni priključek mikrokrmilnika. Kot opozorilna meja je nastavljena temperatura 4°C.

Poglavje 6

Sklepne ugotovitve

Sama izvedba projekta je bila zame izziv, saj je bilo potrebno izdelati rešitev, ki sega od nizkonivojskega nadzora senzorjev in krmiljenja sistema prek mikrokrmilnika, do njihovega prikaza na spletni aplikaciji. Medtem ko se v svetu spletnih rešitev precej dobro znajdem, je jezik C, v katerem je Arduino programiran, zame bil večja neznanka. Zato sem marsikdaj naletel pri navidez banalnih problemih na precejšnje probleme pri njihovi izvedbi v kodi. V veliko pomoč so mi bili tudi že izvedeni projekti na osnovi Arduina, saj je skupnost zaradi popularnosti tega mikrokrmilnika precej velika.

Prostora za izboljšave je definitivno precej, saj mislim, da sem s tem projektom postavil praktično samo ogrodje, ki se potem lahko razvija naprej. Za začetek bi lahko dodal več funkcionalnosti, kot recimo upravljanje zaves glede na željeno temperaturo v prostoru, saj tako lahko izkoristimo sonce za gretje, oz. ob poletnih dneh preprečimo še dodatno gretje sobe.

Glede same izvedbe pa mislim, da je precejšnjih izboljšav potrebno v samem krmilnem programu mikrokrmilnika, še posebej kar se tiče upravljanja s pomnilnikom in pa tudi optimizacije delovanja. Razlog za to pa je v največji meri moje nepoznavanje jezika C.

Projekt je dejansko namenjen vsakodnevni uporabi na nekem objektu, ampak je trenutno še v fazi testiranja in optimizacije. Kljub temu pa mislim, da mi je prvotni cilj precej dobro uspel, saj sem sestavil funkcionalen sistem

za nadzor za precej majhen proračun, kar tudi pokaže, kako zmogljive in dostopne so postale tovrstne komponente. V nadaljnje pa nameravam projekt tudi še razširiti.

Literatura

[1] Arduino (2011) "Arduino - ArduinoBoardMega2560". Dostopno na: http://www.arduino.cc/en/Main/ArduinoBoardMega2560

[2] Arduino (2011) "Arduino - ArduinoEthernetShield". Dostopno na: http://arduino.cc/en/Main/ArduinoEthernetShield

[3] Wikipedia ".NET Framework". Dostopno na: http://en.wikipedia.org/wiki/.NET_Framework

[4] Arduino "Arduino - Reference". Dostopno na: http://www.arduino.cc/en/Reference/HomePage

[5] Texas Instruments "LM35 Precision Centigrade Temperature Sensors". Dostopno na: http://www.ti.com/lit/ds/symlink/lm35.pdf

[6] ladyada.net "How to use photocells, LDRs, CdS cells, photoresistors!". Dostopno na: http://www.ladyada.net/learn/sensors/cds.html

[7] Parallax "PIR Sensor". Dostopno na: http://www.parallax.com/dl/docs/prod/audiovis/PIRSensor-V1.1.pdf

[8] Wikipedia "Passive infrared sensor". Dostopno na: http://en.wikipedia.org/wiki/Passive_infrared_sensor

[9] Tim Zaman "Sony Ericsson PDU". Dostopno na: http://hollandshoogte.wordpress.com/2010/07/27/t68i-sms-working/

[10] Dreamfabric "SMS messages and the PDU format". Dostopno na: http://www.dreamfabric.com/sms/

[11] Tim Zaman "[SMS] Sony Ericsson PDU". Dostopno na: http://www.timzaman.nl/?p=47&lang=en

[12] Sony Ericsson "Gx4x App note Construction of SMS PDUs". Dostopno na: http://archive.sierrawireless.com/resources/AirPrime/GT47-GT48/Application_Notes/Gx4x%20App%20note%20Construction%20of%20SMS%20PD

[13] emartee.com "MQ6 Gas Sensor Brick -Arduino Compatible - emartee.com". Dostopno na: http://www.emartee.com/product/41945/MQ6

[14] Luc Brunet - Rural Environment Engineer/OMAFRA "Hazardous Gases". Dostopno na: http://www.omafra.gov.on.ca/english/engineer/facts/04-087.htm