UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

David Mavec

KRMILJENJE RAZSVETLJAVE Z UPORABO BREZŽIČNO-KOMUNIKACIJSKEGA MODULA XBEE

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

Ljubljana, 2012

UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

David Mavec

KRMILJENJE RAZSVETLJAVE Z UPORABO BREZŽIČNO-KOMUNIKACIJSKEGA MODULA XBEE

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

Mentor: doc. dr. Patricio Bulić

Somentor: viš. pred. dr. Robert Rozman

Ljubljana, 2012

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: 00194/2012
Datum: 01.02.2012

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

Kandidat: **DAVID MAVEC**

Naslov: **KRMILJENJE RAZSVETLJAVE Z UPORABO BREZŽIČNO-KOMUNIKACIJSKEGA MODULA XBEE**

HOME LIGHTING CONTROL USING AN XBEE WIRELESS-COMMUNICATION MODULE

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Razvijte sistem za daljinsko brezžično krmiljenje razsvetljave z uporabo brezžičnega komunikacijskega modula XBee. Sistem naj bo zasnovan okrog Atmelovega mikrokrmilnika ATMega 32 ter modula XBee XB24-Z7WIT-004. Sistem naj omogoča simulacijo prisotnosti na domu s funkcijo vnaprej nastavljenega prižiganja in ugašanja luči.

Mentor:
doc. dr. Patricio Bulić

Dekan:
prof. dr. Nikolaj Zimic

Somentor:
viš. pred. dr. Robert Rozman

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani/-a David Mavec,

z vpisno številko 63070319,

sem avtor/-ica diplomskega dela z naslovom:

BREZŽIČNO KRMILJENJE LUČI

Z UPORABO BREZŽIČNO-KOMUNIKACIJSKEGA MODULA XBEE

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom (naziv, ime in priimek) doc. dr. Patricio Bulić
- in somentorstvom (naziv, ime in priimek) viš. pred. dr. Robert Rozman
- 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 20. marec 2012 Podpis avtorja/-ice: _____

Zahvala

Najprej se želim zahvaliti mentorju in predavatelju na Fakulteti za računalništvo in informatiko v Ljubljani doc. dr. Patriciu Buliću, ki mi je pomagal z nasveti in idejami pri izdelavi te diplomske naloge. Enako gre zahvala tudi somentorju viš. pred. dr. Robertu Rozmanu, ki je posodil vse elemente in pomagal z nasveti.

Hvala tudi delodajalcu Boštjanu Gomilšku za prosti čas ob delu, ki sem ga lahko namenil izdelavi diplomske naloge.

Posebna zahvala pa gre staršema, ki sta me podpirala v celotnem času študija.

Kazalo vsebine

Kazalo slik

Kazalo kode

Kazalo tabel

Seznam uporabljenih kratic

- Arduino Razvojne ploščice (različnih tipov) za programiranje mikrokrmilnikov
- XBee Modul za brezžično komunikacijo
- SMA Standardni priključek za koaksialne vodnike
- USB Univerzalno serijsko vodilo
- UART Univerzalni asinhroni sprejemnik/oddajnik
- bps bitov na sekundo
- LCD prikazovalnik na tekoče kristale
- PWM impulzno-širinska modulacija

Povzetek

Namen diplomske naloge je izvedba preproste rešitve, s katero je možno delno zavarovati hišo pred tatovi. Ta rešitev bi, tudi če ni nikogar doma, dajala občutek, kot da hiša vseeno ni prazna. Do ideje smo prišli z mentorjem po pogovoru o temi diplomske naloge, saj smo sprva mislili izdelati alarmni sistem za stanovanje, a ker je takšnih produktov že kar nekaj, smo se odločili za izdelavo modula za avtomatsko brezžično krmiljenje luči s funkcijo vnaprej nastavljenega prižiganja in ugašanja luči. Do končnega izdelka diplomske naloge smo prišli postopoma. Na začetku smo se naučili uporabljati programator Arduino in programsko okolje Arduino 1.0, nato pa smo dograjevali in dodajali komponente, ki smo jih uporabili pri končni rešitvi. Na programatorju se uporablja mikrokrmilnik družine ATMega, in sicer ATMega32. Za brezžično komunikacijo smo uporabili module XBee. Ko so enkrat nastavljeni za pravilno delovanje, sta delo in komunikacija med moduli enostavna. Izdelek smo sestavljali na testni ploščici, tako da spreminjanje in dodajanje elementov h končni rešitvi ni bilo zapleteno.

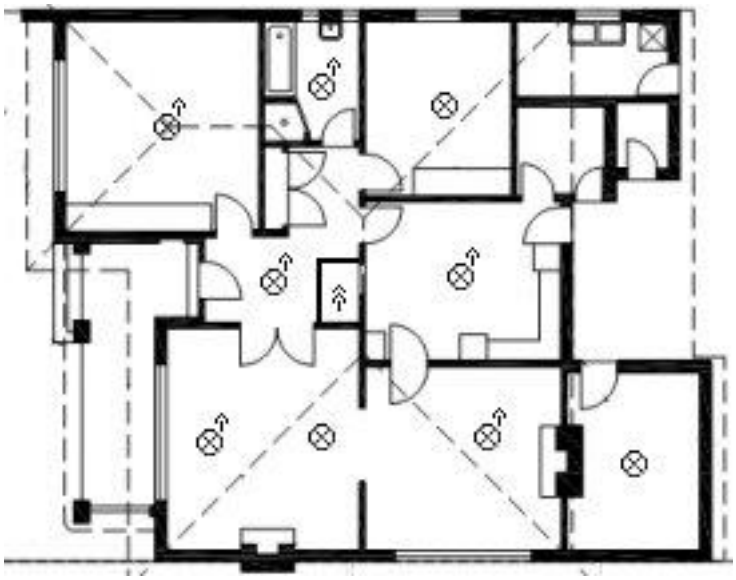
Abstract

The subject of this diploma thesis is the implementation of a simple anti-burglary solution to protect homes. The solution would creates the illusion that a house is inhabited even when its owners are not at home. We came up with this idea after a discussion about the thesis. Initially, we intended to focus on a home alarm system, but since there already are a multitude of similar products on the market, we decided to focus on a wireless automatic lighting control module that allows the user to set up in advance the order in which the lights in a house will automatically turn themselves on and off. The creation of the final product that is the subject of this thesis was a gradual process. First, we mastered the use of the Arduino programmer and of the Arduino 1.0 programming environment, after which we started upgrading and adding the components which were used in the final solution. The programmer uses a microcontroller from the ATMega family, namely ATMega32. For wireless communication, we used XBee modules. Once configured for proper operation, communication between the modules is quite straightforward. We assembled the product on a testing board, which allowed simpler incremental development of elements to the final solution.

1 Uvod

S to diplomsko nalogo smo skušali razviti rešitev za varnejši dom, ko nas ni doma. Ob prebiranju in raziskovanju, kakšne hiše so zanimive za tatove, smo prišli do naslednjega zaključka. Za hišo, ki stoji na samem ali pa je videti, da se v njej že nekaj dni nič ne dogaja, obstajajo povečane možnosti, da jo bodo tatovi oropali. S tem izdelkom brezžičnega krmiljenja luči pa bi bilo videti, kot da je vseeno nekdo doma. Poseben izziv brezžičnega nadzora luči je bil ravno ta, da bi centralna krmilna procesorska enota in modul za prižiganje in ugašanje luči komunicirala brezžično. To pa je tudi ena izmed prednosti, saj bi s tem dosegli lažjo namestitev in vgradnjo v hišo. Če ni potrebe po spremembi hišne napeljave, to vsekakor pripomore k odločitvi kupca za nakup brezžične rešitve.

Sprva nismo imeli ne ideje in ne znanja, kako bo potekala brezžična komunikacija. A je s svojo pomočjo pristopil dr. Rozman in ponudil module, ki so jih že uporabljali v njegovem laboratoriju. Uporabljali so jih na raznih robotskih tekmovanjih, delavnicah itd. Poleg teh modulov za brezžično komunikacijo nam je posodil tudi vse ostale elemente (mikrokrmilnika, programatorja in druga tipala). Nato smo morali le še določiti, kako bosta potekala celotna komunikacija in protokol delovanja. Začrtali smo grobo rešitev, ki jo sestavlja glavna centralna krmilna procesorska enota in pa moduli za prižiganje in ugašanje luči. Po načrtovanju je sledilo programiranje in testiranje. Na koncu pa smo vse to še sestavili v praktičen primer in preizkusili, kako rešitev deluje.



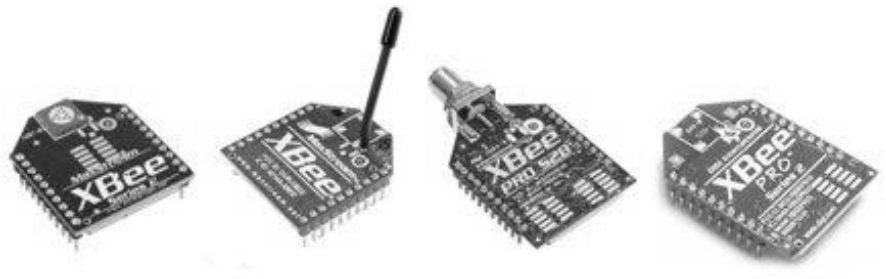
Slika 1: Primer krmiljenja razsvetljave v hiši

2 XBee

2.1 Kaj je XBee?

Xbee [1] je modul za brezžično komunikacijo, ki se uporablja na različnih področjih (modelarstvo, elektrotehnika …). Modul je bil sprva proizvod podjetja Maxstream, nato pa je proizvodnjo in razvoj modula prevzelo podjetje Digi. Tako se zaenkrat na tržišču še pojavljajo proizvodi podjetja Maxstream in tudi podjetja Digi, zaradi česar prihaja pri iskanju prave dokumentacije za določen tip modula do rahle zmede, sploh če gre za starejši izvor.

Tudi moduli za brezžično komunikacijo podjetja Digi so razvrščeni v različne kategorije. Tako si lahko natančno izberemo model, ki bo najbolj ustrezal pogojem, v katerih bo deloval. Na voljo imamo osnovne modele, ki zadostujejo za sisteme, ki komunicirajo med seboj znotraj enega prostora. Ti osnovni modeli po navajanju proizvajalca delujejo na razdalji do 120 m. Izbiramo pa lahko tudi med modeli višje kategorije, ki komunicirajo na razdalji do nekaj 10 km. Razlika med modeli je v tipu antene, po kateri oddajamo ali sprejemamo podatke. Obstajajo klasične antene, čip antene, ali pa je na modulu le priključek, na katerega priklopimo zunanjo anteno za daljši domet. Za priklop antene se uporablja standardni priključek tipa SMA.



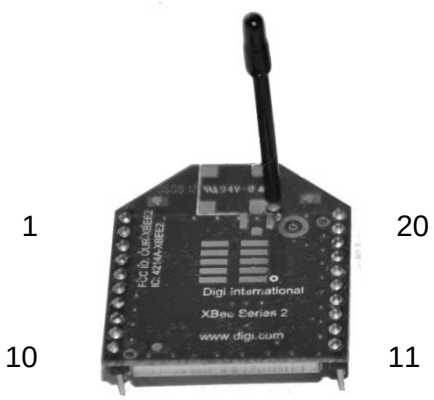
Slika 2: Različni tipi modulov

Zgoraj (Slika 2) so predstavljeni različni tipi modulov. Prvi in drugi sta z vgrajeno anteno (prvi ima »čip« anteno[1], drugi pa navadno (žično) anteno[2]), medtem ko tretji in četrti modul na sliki potrebujeta zunanjo anteno[3][4].

[1] XB24-Z7CIT-004
[2] XB24-Z7WIT-004
[3] XBP24BZ7UIT-004
[4] XBP24BZ7SIT-004

2.2 Modul XBee XB24-Z7WIT-004

XB24-Z7WIT-004 [2] je naziv brezžično-komunikacijskega modula, ki smo ga dobili za uporabo in s katerim smo izdelali tudi praktični primer za to diplomsko nalogo. Iz oznake lahko razberemo, da gre za navadni modul (moduli PRO imajo oznako XBP24) in da ima ta modul anteno, ki je že pritrjena na samem modulu. Ta brezžično-komunikacijski modul ima nazivni domet do 120 m, če smo v odprtem prostoru, če pa smo notri, se zmanjša na 40 m. Kot ostali brezžično-komunikacijski moduli XBee, ima tudi ta prav tako 20 nožic z razmikom 2 mm. S tem smo imeli nekaj težav, saj tak razmik ne ustreza standardnemu priklopu na testno ploščico, a več o tem sledi v poglavju 2.3.



Slika 3: Brezžično-komunikacijski modul XBee, ki smo ga uporabljali

Na zgornji sliki je prikazano, kako se številčijo [3] nožice na modulu. Glavne nožice, ki smo jih uporabili, so 1, 2, 3 in 10. Če gledamo na sliki zgoraj (Slika 3), se pričnejo nožice številčiti zgoraj levo.

Nožica	**Oznaka**	**Opis**
1	VCC	Napajanje
2	Data out	Izhod podatkov
3	Data in	Vhod podatkov
10	GND	

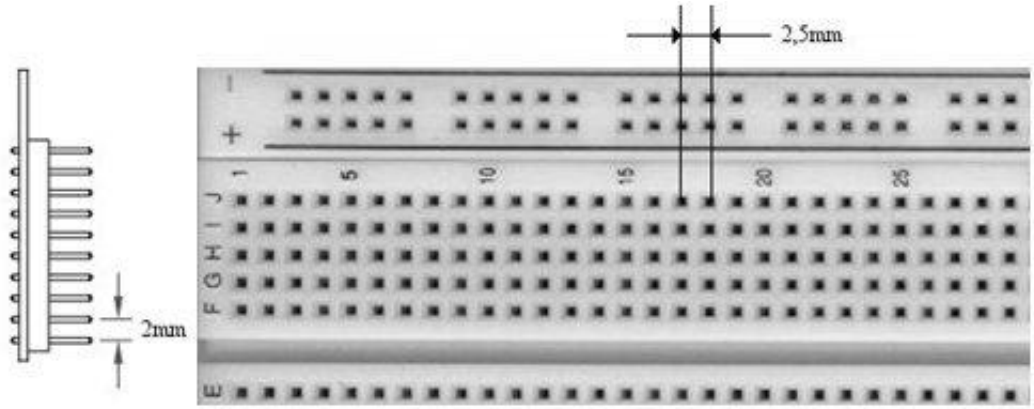
Tabela 1: Pomembne povezave

Za uporabo se priključi napajalna napetost VCC na +3,3 V; podatkovni izhod povežemo z vhodom na mikrokrmilniku, izhod mikrokrmilnika (ta je označen z oznako Tx) pa na podatkovni vhod brezžično-komunikacijskega modula XBee. Veliko težav in nejasnosti nam je sprva povzročala sama povezava z mikrokrmilnikom v načinu konfiguracije, saj se v tem

primeru shema povezave nekoliko spremeni, a več o tem v nadaljevanju (2.4), ko bomo opisali, kako se upravlja z nastavitvami brezžično-komunikacijskega modula. Na brezžično-komunikacijski modul lahko priklopimo senzorje oz. tipala, ki oddajajo analogne signale, saj ima brezžično-komunikacijski modul XBee med drugimi tudi analogne vhode, in sicer na priključnih nožicah 17, 18, 19, 20 (to so zgornje štiri nožice na desni strani, če gledamo po zgornji sliki (Slika 3)).

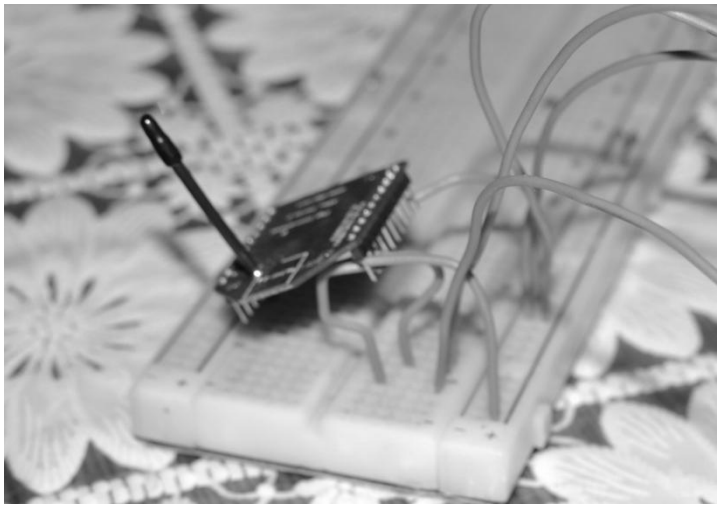
2.3 Neskladni priklop na testno ploščico

Kot smo omenili že v opisu brezžičnega-komunikacijskega modula, ki smo ga uporabljali, smo se že na začetku srečali s težavo. Brezžično-komunikacijski modul ima razmik med nožicami 2 mm, običajne testne ploščice pa so narejene z razmikom 2,5 mm med vrsticami in stolpci, kot prikazuje tudi spodnja slika (Slika 4).



Slika 4: Primerjava razmika nožic brezžično-komunikacijskega modula in testne ploščice

Preden smo sploh lahko preverili delovanje brezžično-komunikacijskega modula, smo morali narediti neko improvizirano povezavo med modulom in testno ploščico.



Slika 5: Povezava modula na testno ploščico

Kot prikazuje zgornja slika (Slika 5), smo si pomagali tako, da smo s pomočjo štirih žičk povezali modul na testno ploščico, testno ploščico pa na mikrokrmilnik. Treba je bilo zelo paziti, saj bi kakršen koli premik lahko povzročil, da povezava ne bi bila dobra in ne bi delovala komunikacija med mikrokrmilnikom in modulom.

Pozneje smo na internetu našli neke vrste vmesnik (Slika 6), na katerega priklopimo brezžično-komunikacijski modul XBee, vmesnik pa priklopimo v testno ploščico. V Sloveniji ga zaenkrat še nihče ne prodaja, zato smo ga morali naročiti iz tujine, kar je trajalo mesec dni.

Slika 6: Vmesnik za priklop na testno ploščico

Za uporabo brezžično-komunikacijskega modula s programatorjem Arduino pa nam proizvajalec ponuja svoj vmesnik[5], ki se imenuje XBee shield [4] (Slika 7). Ena izmed prednosti uporabe tega vmesnika je, da ga zgolj položimo na programator in ne potrebujemo posebnih povezav ali kakršnih koli drugih elementov.

[5] Proizvajalec Arduino ponuja še druge vmesnike: *ethernet shield*, *bluetooth shield*, *motor shield* …

Slika 7: XBee shield

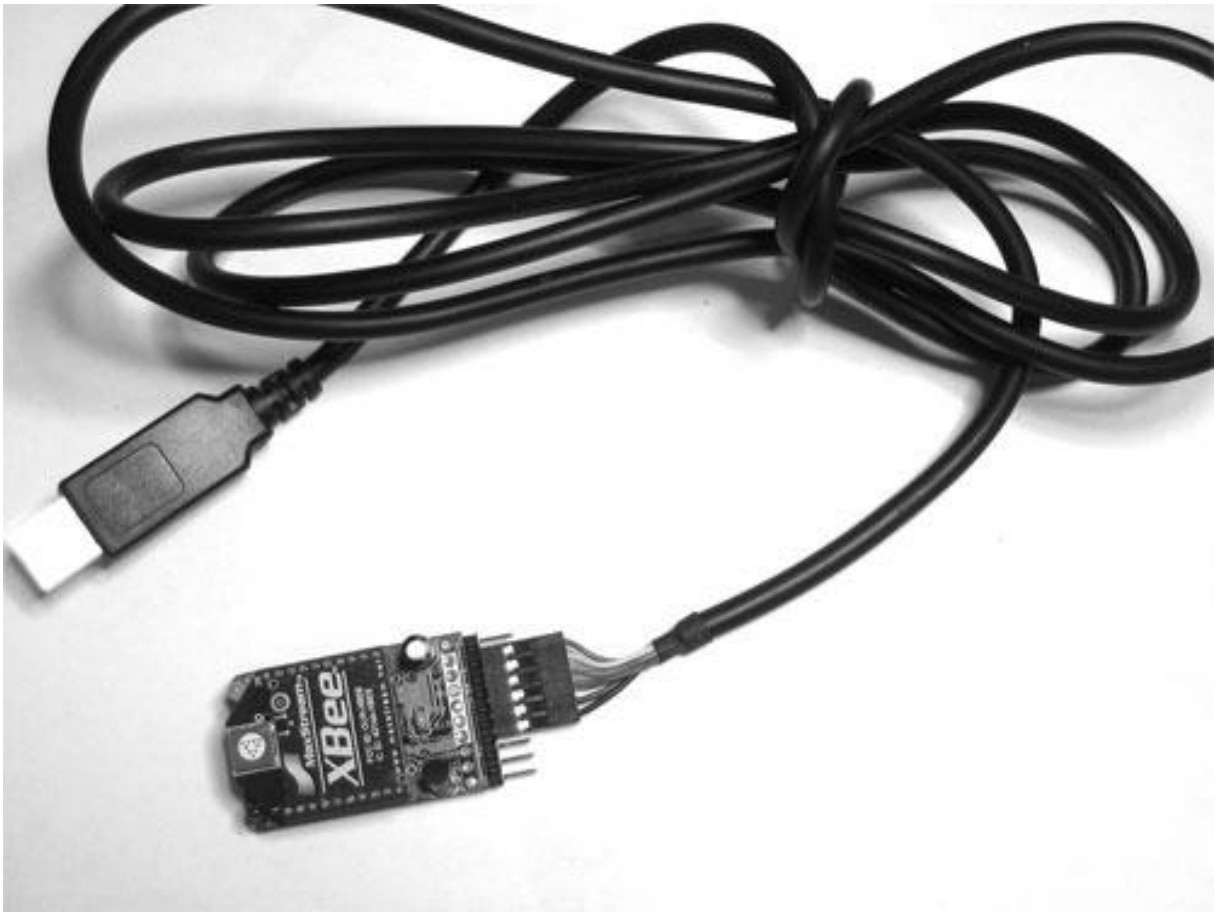
Je zelo priročen glede na priklop in odklop in ne zahteva dodatnih povezav in prostora.

Slabost pa je, da potem ne moremo po želji nastavljati, na katerih priklopih bomo uporabljali modul oz. lahko uporabljamo samo en modul na en programator. Druga slaba stran pa je tudi višja cena v primerjavi z vmesnikom, ki smo ga uporabljali mi.

2.4 Povezava z X-CTU

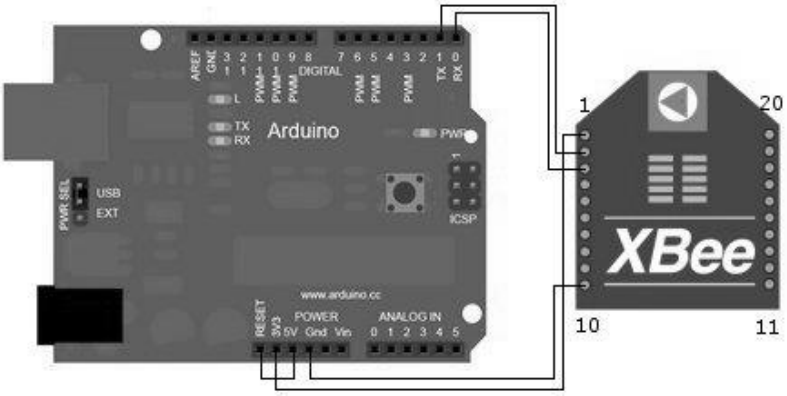
X-CTU [5] je program, s katerim lahko prek grafičnega vmesnika dostopamo do nastavitev brezžično-komunikacijskega modula.

Vsak brezžično-komunikacijski modul je treba pred uporabo še nastaviti. S tem si zagotovimo, da so v istem omrežju in da delujejo na istih kanalih. Nastavimo še vlogo brezžično-komunikacijskega modula. Poleg XBee shielda obstaja tudi poseben vmesnik za konfiguriranje brezžično-komunikacijskih modulov, ki ga priključimo na vrata USB [6].



Slika 8: Priklop USB

Če ne želite kupiti zgoraj navedenega vmesnika za priklop brezžično-komunikacijskega modula na vrata USB, si lahko pomagate s programatorjem Arduino, kot smo to storili tudi mi. Na naslednji sliki (Slika 9) je prikazana shema povezave, v nadaljevanju pa bomo še pokomentirali ključne stvari, na katere je treba paziti.

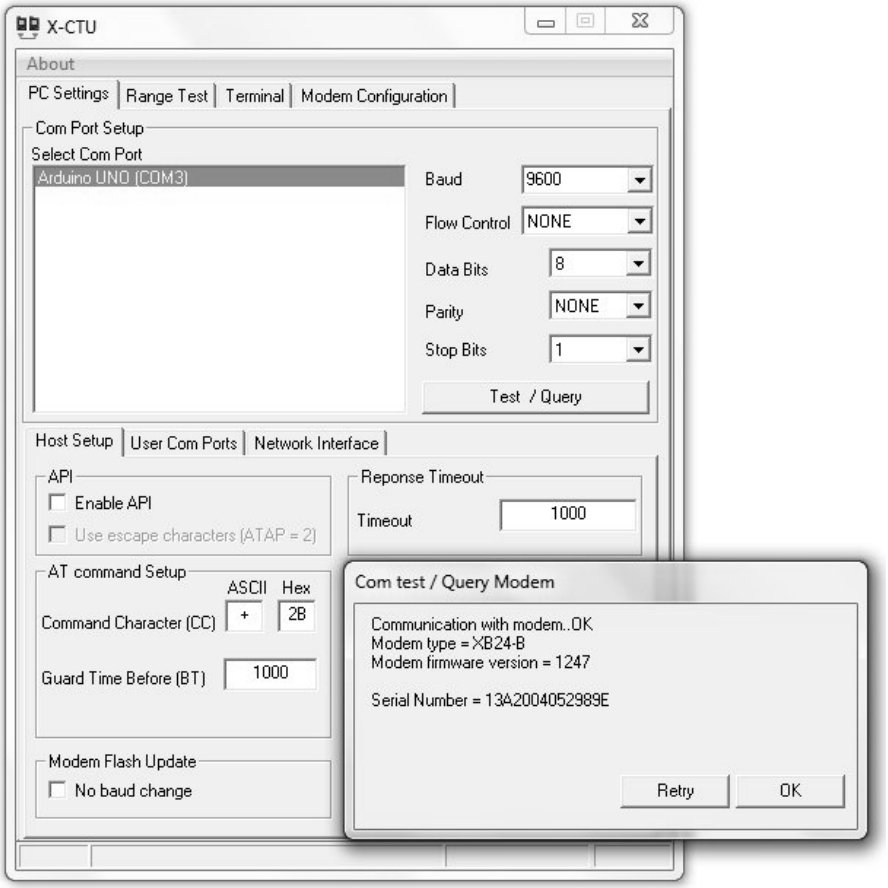


Slika 9: Shema za konfiguracijo

Če želite, da bi komunicirali neposredno z brezžično-komunikacijskim modulom, mora programator Arduino delovati v načinu *reset*. To naredite tako, da na programatorju povežete med saboj priključka RESET in GND. To je identično, kot če bi ves čas držali tipko *reset*. Tako bodo šli vaši ukazi iz vrat USB neposredno na brezžično-komunikacijski modul. Pomembno pa je tudi, da pravilno povežete oddajne (Tx) in sprejemne (Rx) nožice na programatorju z modulom. To storite po naslednjem razporedu: povežete vhod Rx na programatorju z nožico 3[6] in podatkovni izhod Tx programatorja povežete na 2[7].

Zdaj lahko zaženete program X-CTU, ki se lahko uporablja za neposredno komunikacijo z brezžično-komunikacijskim modulom in ima tudi uporabniku prijazen vmesnik. Lahko komunicirate tudi z ukazi prek terminalskega okna, vendar je mnogo lažje prek uporabniškega vmesnika.

[6] Podatkovni vhod brezžično-komunikacijskega modula.
[7] Podatkovni izhod brezžično-komunikacijskega modula.



Slika 10: X-CTU

Da bi preverili, ali povezava z modulom deluje, lahko kliknete Test/Query v X-CTU in izpisati bi se morali podatki o modulu, kot prikazuje slika zgoraj (Slika 10). Če izpiše napako, se prepričajte, da so povezave pravilno vzpostavljene (napajanje za modul, oddajana in sprejemna povezava, reset programatorja).

Ko uspešno vzpostavite povezavo, lahko pod zavihkom »Modem Configuration« vidite vse nastavitve, ki so trenutno nastavljene na modulu. Da vam izpiše nastavitve, morate še prej klikniti gumb »Read«.

Z X-CTU lahko pa tudi dostopate do modula s terminalskim načinom, torej prek ukazov v ukazni vrstici (to je v zavihku »Terminal«).

3 Nastavitve

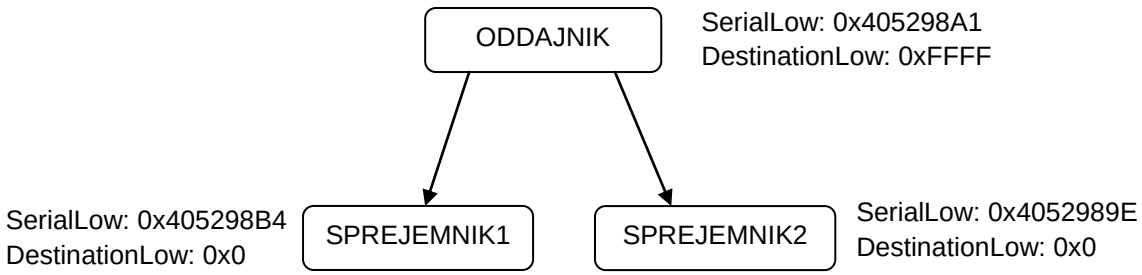
Brezžično-komunikacijski modul XBee omogoča različne tipe topologij omrežij [7]. So običajna dvotočkovna omrežja (*point to point*), ki omogočajo komunikacijo zgolj dveh modulov. Nekoliko bolj napredna omrežja so, ko več krajiščnih točk komunicira z eno centralno (*point to multipoint*). Eden izmed tipov omrežja pa je tudi mešano omrežje (*mesh network*), ko vsi moduli komunicirajo med seboj.

Za večje projekte, kjer mora biti organizacija omrežja striktno določena, se po pravilih uporabi omrežje, v katerem imajo brezžično-komunikacijski moduli različne vloge (*coordinator, router, end device*), a o tem več pozneje v poglavju 3.2. Najprej bomo predstavili, kako smo se lotili nastavitev modulov.

3.1 Nastavitve našega omrežja

Module smo nastavili po najbolj enostavni metodi, tako da je lahko komunikacija stekla. Ker smo si zastavili idejo tako, da bomo imeli eno centralno enoto, ostali moduli pa bodo le končni uporabniki, katerim je dovolj, da sprejemajo signale, smo se odločili za topologijo *point to multipoint* [8].

Da lahko moduli komunicirajo med seboj, morajo biti vsi nastavljeni tako, da delujejo na istem PAN[8] ID-ju. Nato smo se lotili nastavitev, in sicer, kot smo omenili že zgoraj, za omrežje *point to multipoint*. Na voljo smo imeli tri brezžično-komunikacijske module, zato smo enega uporabili za centralno enoto, ki bo oddajnik, ostala dva pa kot sprejemnika v modulih za prižiganje in ugašanje luči.



Slika 11: Shema omrežja

[8] Personal Area Network je oznaka omrežja, v katerem komunicirajo moduli.

Ker ima oddajnik na DL[9] nastavljeno vrednost 0xFFFF, to pomeni, da bo oddajal na vse module, ki so v istem omrežju, torej imajo nastavljen isti PAN ID in delujejo na istem kanalu kot oddajnik.

3.2 Nastavljanje kompleksnejšega omrežja

Kot smo že omenili, je prav, da pri večjih projektih, bolj zapletenih in kompleksnejših omrežjih bolj upoštevamo pravila tvorjenja omrežja, ki jih priporoča proizvajalec brezžično-komunikacijskih modulov. To pomeni, da podatkov ne pošiljamo kar vsem po omrežju, ampak jih usmerjamo do točno določene končne točke. Tako je omrežje zelo podobno računalniškemu omrežju, kjer uporabljamo usmerjevalnike in stikala.

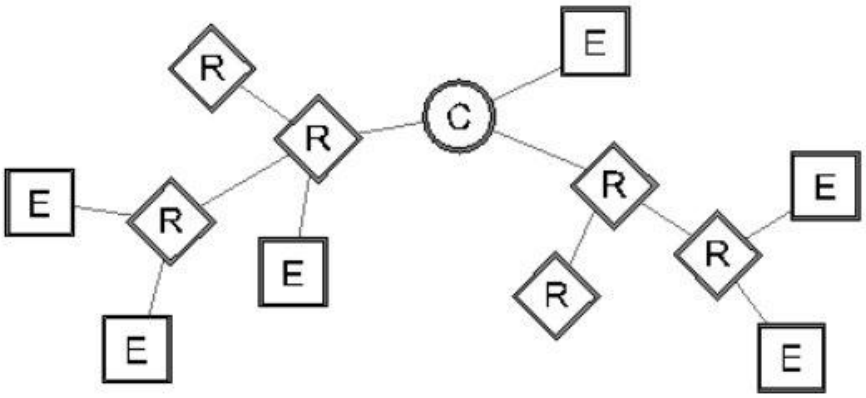
Če sestavljamo omrežje na tak način, potem se boste srečali z različnimi pojmi nastavitev vloge brezžično-komunikacijskega modula, in sicer *coordinator, router in end device*.

Koordinator (*coordinator*) je prvi, ki začne z grajenjem omrežja. Začne tako, da nastavi kanal, na katerem bo potekala komunikacija, in PAN ID. Koordinator odloča, kdo se lahko priklopi na omrežje, lahko pa tudi usmerja podatke. Ker ima nenehno delo, mora biti zagotovljeno neprestano napajanje, torej ni namenjen za napajanje z baterijo.

Usmerjevalnik (*router*) se mora najprej povezati v omrežje, preden sploh kaj oddaja, sprejema ali usmerja podatke. Šele potem, ko se uspešno priklopi v omrežje, lahko omogoči drugim usmerjevalnikom ali končnim napravam, da se povežejo nanj in ko je povezan v omrežje, lahko opravlja svojo nalogo (sprejemanje, oddajanje, usmerjanje podatkov naslednikom v omrežju).

Končna naprava (*end device*) se mora prav tako najprej povezati v omrežje na usmerjevalnik ali na koordinatorja, preden lahko sprejema oz. pošilja podatke. Na končno napravo se ne more povezati nobena naprava več, torej je ta modul zadnji člen v omrežju. Sprejemanje in oddajanje podatkov poteka vedno po hierarhiji, kar pomeni, da pošilja podatke samo prek nadrejenega, na katerega je povezan, in prav tako lahko sprejema podatke le od nadrejenega. Ker je odvisen od nadrejenega, ima poseben način delovanja, tj. metoda spanja (*sleep mode*). To pomeni, da porabi zelo malo energije, saj je v fazi delovanja le takrat, ko nadrejeni modul tako hoče oz. ima podatke zanj ali pa jih hoče oddati. Ker porablja malo energije, ga je možno napajati kar iz baterije.

[9] Destination Address Low.



Slika 12: Primer omrežja [7]

Oznaka	**Opis**
C	Koordinator
R	Usmerjevalnik
E	Končna naprava

Tabela 2: Oznake na Sliki 12

Na zgornji sliki (Slika 12) lahko vidite pravilno obliko organizacije omrežja. Na končnih točkah so uporabljene končne naprave, do katerih usmerjajo podatke usmerjevalniki. Glavni gradnik, ki pa povezuje omrežje, je koordinator.

4 Načini delovanja

Brezžično-komunikacijski moduli lahko delujejo oz. so lahko v različnih stanjih [7], medtem ko čakajo na podatke, sprejem ali oddajanje in ukazni način.

4.1 Idle

Idle je način delovanja, ko modul ne sprejema ali ne oddaja podatkov in tudi ni v načinu nastavljanja. V tem načinu delovanja preverja podatke in če so na voljo, preklopi v način sprejemanja. Če pa mora podatke poslati, gre v način pošiljanja podatkov.

4.2 Spanje (*sleep*)

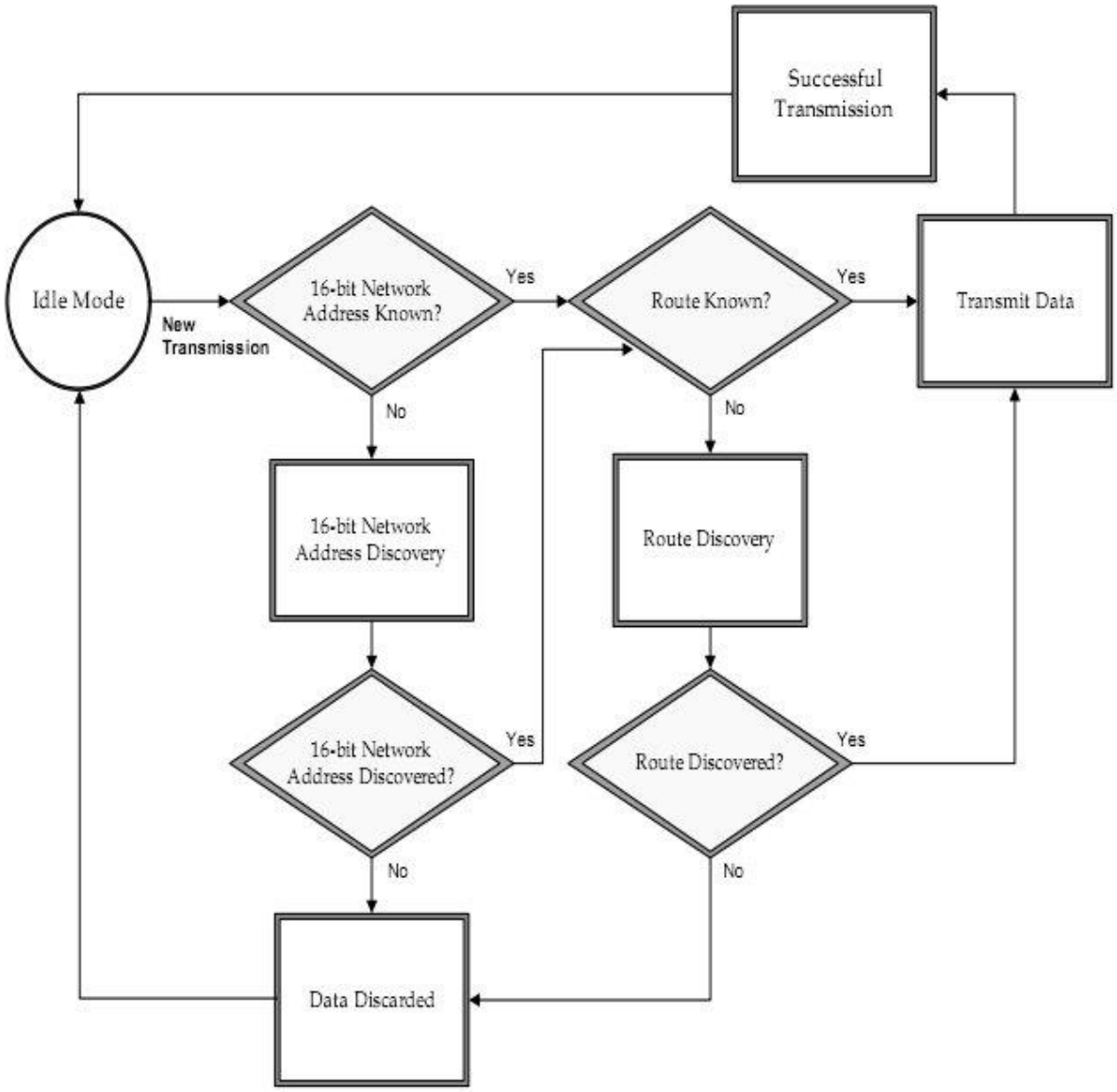
Končne naprave, za katere smo omenili, da porabijo malo energije, lahko preidejo v stanje spanja. Brezžično-komunikacijski modul lahko preide v način spanja prek dveh poti. Prvi način spanja omogočimo s signalom na nožici 9 (SLEEP_RQ). Drugi pa je fiksni način spanja, ko modul spi nek določeni čas in preveri stanja in če ni podatkov za sprejem, se vrne za isti določeni čas v spanje.

4.3 Branje (*receive*)

Ko so na voljo podatki za sprejem, se modul iz načina *idle* preklopi v način za sprejem in podatke sprejme ter shrani v pomnilnik pred izhodom, nato pa jih lahko mikrokrmilnik (ali katera izmed drugih naprav) prebere na podatkovnem izhodu brezžično-komunikacijskega modula (DOUT, nožica 2).

4.4 Pošiljanje (*transmit*)

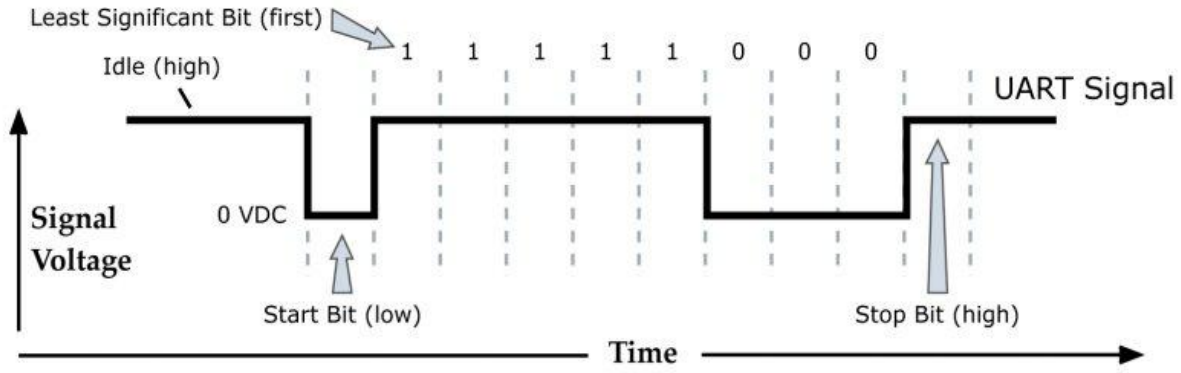
V način pošiljanja modul preklopi iz načina *idle*, ko se pojavijo podatki za pošiljanje. Ko so podatki pripravljeni za pošiljanje, se s pomočjo ciljnega naslova poizkuša vzpostaviti pot do naslovnika. Če je pot uspešno vzpostavljena, se paket pošlje, drugače pa je paket zavržen. Ko so podatki poslani, se čaka na potrditev prejema podatkov. Če potrditev ne pride do pošiljatelja, ta poizkuša poslati podatek še enkrat. To prikazuje diagram poteka na naslednji sliki (Slika 13).



Slika 13: Diagram poteka pri načinu pošiljanja [1]

4.4.1 Kako poteka pošiljanje podatkov

Ko pošljemo podatek za oddajanje na podatkovni vhod modula, se ta podatek odda prek modula UART. Za vsak bajt se na začetku generira nizki bit (znak za začetek oddajanja), potem je 8 bitov, kar predstavlja podatek, in na koncu še visoki bit kot znak za konec oddajanja. Na spodnji sliki je prikazan primer pošiljanja številke 31 (0b11111 dvojiško).



Slika 14: Primer pošiljanja števila 31 [1]

4.5 Ukazni način

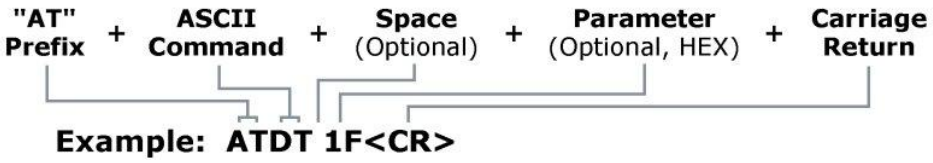
Modul preklopi v ukazni način, ko dobi na vhod posebno določeno kombinacijo znakov. To je znakovni niz +++. Če uspešno zazna ta niz znakov, potem vrne na podatkovni izhod podatek OK in nato lahko pričnemo z vpisovanjem ukazov za nastavitve. Pozor: če imamo hitrost prenosa podatkov nastavljeno na 9600 bps, potem imamo na voljo tri sekunde časa, da vtipkamo komando. Po treh sekundah se modul samodejno vrne v način *idle*, ne da bi kar koli vrnil na izhod, le ukazov ne bo več zaznaval.

Ko smo v komandnem načinu in želimo zamenjati PAN ID, storimo sledeče:

ATID1234	s tem nastavimo PAN ID na 1234
ATID	vrne 1234
ARWR	zapišemo, kar smo nastavljali, ATRE pa pobriše nastavitve

4.5.1 Zgradba ukaza

Za primer pa si lahko ogledamo še, kako je zgrajen ukaz v ukazni vrstici:



Slika 15: Zgradba ukaza za konfiguracijo [1]

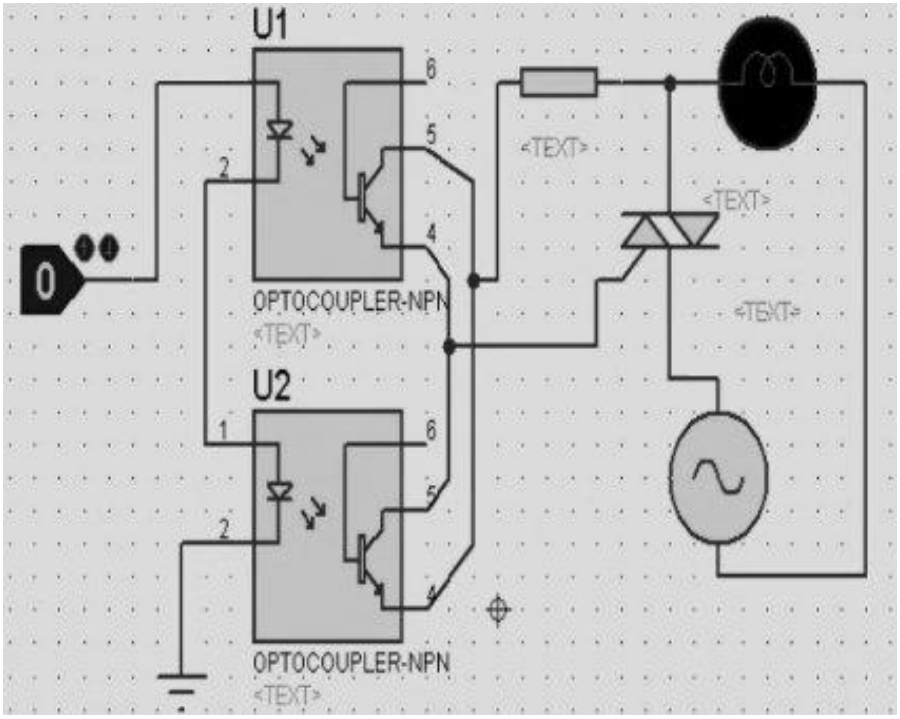
Ukaz, kot vidimo zgoraj, sestavlja standardna predpona AT, ki ji sledi ukaz (ID, WR, RE, DT, BD, MY …). Če vpišemo samo ukaz brez vrednosti, nam bo modul vrnil vrednost, ki je trenutno nastavljena. Če pa dodamo poleg ukaza še vrednost, bo to novo nastavljena vrednost na modulu.

5 Praktični primer

Diplomsko delo, ki predstavlja delo z mikrokrmilniki in brezžičnimi komunikacijskimi moduli, je narejeno po ideji pametne hiše oz. varovanja. Sprva je bila ideja diplomske naloge alarmni sistem, vendar je ponudba teh produktov velika in lahko izbiramo med preprostejšimi rešitvami kot tudi med produkti za varovanje večjih objektov. Tako da smo z mentorjem prestavili idejo korak nazaj na preventivno varovanje objekta. To pomeni, da če nam alarmna naprava koristi, tako da nas oz. okolico obvesti, ko kdo vlomi v objekt, bi produkt brezžičnega krmiljenja luči, ki je nastajal ob tem diplomskem delu, opravljal svojo nalogo varovanja pred vlomom.

Primer uporabe, ki nam je najprej prišel na misel, je bil iz filma Sam doma. Ko družina odide na počitnice, za sabo pusti prazno hišo, ki je lahko zelo privlačna za nepridiprave. Tako bi s to rešitvijo vseeno, tudi ko nas ni doma, avtomatizirali in na nek način oživili hišo. To pomeni, da bi na zunaj delovala, kot da se v njej vseeno nekaj dogaja. To v osnovi predstavlja avtomatizirano prižiganje in ugašanje luči, ki je tekom diplomskega dela uspelo, sicer zgolj indikacijsko, a to je le korak stran od realizacije. Indikacijsko pomeni, da če bi v končni fazi gorela luč v dnevni sobi, se bo v trenutnem primeru prižgala dioda LED na sprejemniku v dnevni sobi.

Do končne rešitve je tako le korak, saj je treba narediti zgolj še to, da z mikrokrmilnikom na sprejemniku prižgemo luč. Nekaj smo že razmišljali v tej smeri in prišli do rešitve s triakom, ki bi bil krmiljen z mikrokrmilnikom na sprejemniku.



Slika 16: Shema za krmiljenje 220 V luči z mikrokrmilnikom [9]

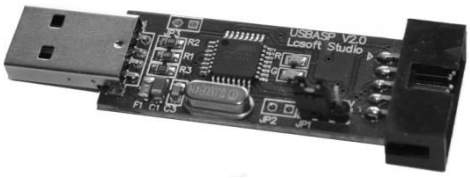
Na prejšnji sliki (Slika 16) je prikazana shema, kako lahko s pomočjo mikrokrmilnika, ki ima lastno napajanje (5 V), krmilimo elemente priključene na 220 V omrežno napetost. Shemo sestavljajo izhod mikrokrmilnika, dva optična sklopnika, upor, triak, luč in 220 V omrežna napetost.

Naj najprej obrazložimo vlogo optičnega sklopnika v shemi. Njegov namen je, da ločimo tokokrog nizke napetosti od tokokroga visoke napetosti. Kot vemo, mikrokrmilniki delujejo v območju nizke napetosti (3 V, 5 V), luči v hišni napeljavi pa so priključene v 220 V izmenično omrežje. Kot je tudi prikazano na shemi (Slika 16), sestavljata optični sklopnik dioda LED in svetlobno tipalo; na tak način ločimo dve strani. Dva optična sklopnika pa morata biti zaradi izmenične napetosti. Samo delovanje je dobro predstavljeno v posnetku [9].

Triak je gradnik, ki je zelo podoben tranzistorju, le da je namenjen za višje napetosti. Razlikuje se po oznakah; oznake nožic triaka so namreč MT1, MT2 in G, pri čemer MT predstavlja Main Terminal, G pa Gate oz. krmilni vhod. Torej, ko pride napetost na G, se triak odpre in prevaja [9].

5.1 Postopek načrtovanja izdelka

Sam postopek izdelave izdelka za diplomsko nalogo se je pričel s spoznavanjem razvojnega okolja za mikrokrmilnike. Sprva smo ravno tako delali z mikrokrmilniki proizvajalca Atmel, z modelom ATtiny 2313. Programiranje je preprosto, saj se programator USBASP dobi ugodno, vendar pa se nekoliko bolj zakomplicira pri samem programskem okolju. Vsaj nam je povzročalo kar nekaj težav s prevajalnikom in nalaganjem prevedene kode na mikrokrmilnik.



Slika 17: Programator USBASP

Mentor nam je po posvetovanju predlagal uporabo razvijalskega okolja Arduino in nam tudi priskrbel programatorje, prikazovalnike LCD in pa tudi nekaj tipal. Nato smo se lahko začeli spoznavati in učiti programiranja v okolju Arduino. Tukaj pa je stvar zares preprosta, saj so na ploščici jasno označeni vhodi in izhodi. Pri samem programu, v katerem razvijamo in nalagamo kodo na mikrokrmilnik, pa so priložene knjižnice, ki se jih največ uporablja.

Sprva smo izdelovali zelo preproste stvari, kako se postavlja stanje na izhode, nato še uporabo zakasnitev. Preizkušali smo tudi analogne vhode s svetlobnim tipalom, kar je tudi zelo preprosto. Namreč, za vse to ima Arduino napisane metode, ki jih uporabljamo, npr. *delay*, *digitalRead*, *digitalWrite* … Ko smo želeli uporabiti brezžični komunikacijski modul, smo videli, da bomo morali najprej še spoznati knjižnico *Serial*, ki omogoča serijsko komuniciranje ali z USB vrati ali pa s serijskim podatkovnim vhodom/izhodom 1, 2.

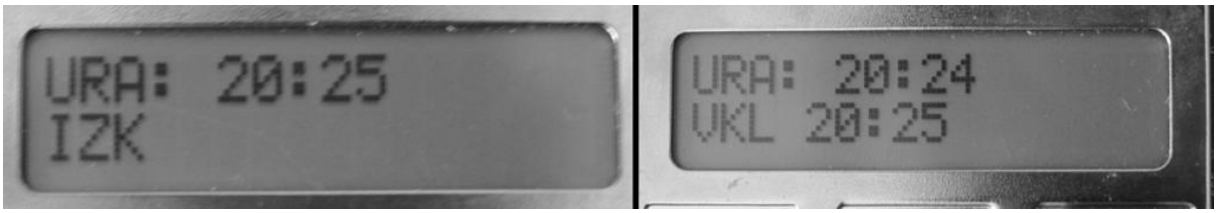
5.2 Uporaba tipk

V končnem produktu se za upravljanje z mikrokrmilnikom ter nastavitvami uporabljajo štirje gumbi, tako da je najprej treba določiti, na katere vhode bomo te gumbe priklopili.

```
pinMode(12,INPUT);
pinMode(13,INPUT);
pinMode(11, INPUT);
pinMode(10, INPUT);
```

Koda 1: Definiranje vhodov

Gumb na vhodu 12 se uporablja kot glavno stikalo za vklop ali izklop samega postopka prižiganja in ugašanja luči. Na zaslonu se prikazuje status, ali je postopek prižiganja in ugašanja vklopljen ali ne.

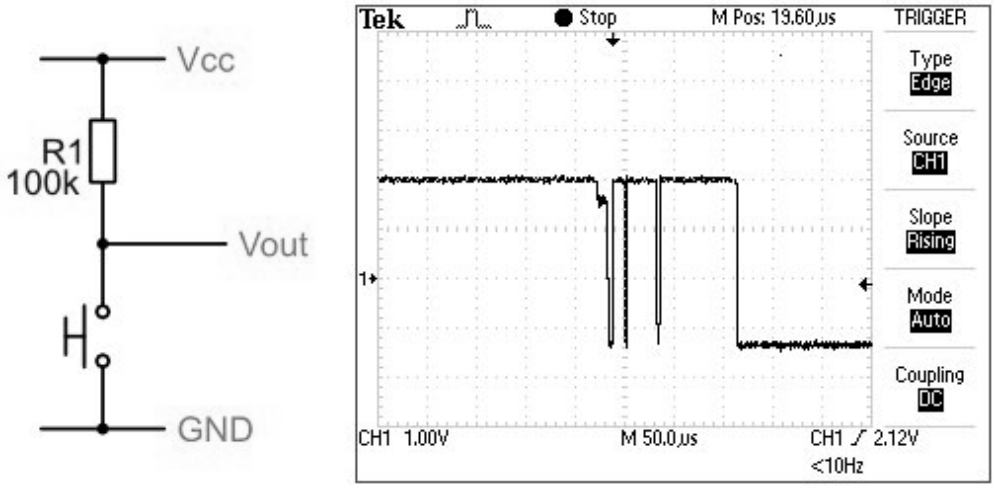


Slika 18: Status

Kakor prikazuje zgornja slika (Slika 18), je na desni sliki prikazan status, ko je postopek prižiganja in ugašanja izklopljen. V tem primeru kažemo le trenutni čas. Na levi sliki pa je prikazan status na zaslonu, ko je postopek krmiljenja luči vklopljen, a še ni čas za začetek. Tako le prikazujemo trenutni čas in pa čas, kdaj se krmiljenje prične. Ura se trenutno programsko nastavi na 20:24. Več o tem sledi v poglavju 5.3.

5.2.1 Stabiliziranje branja stikal - Debounce

Situacija, ki jo je še treba omeniti in ki nam je sprva povzročala nekaj težav, je uporaba stikal oz. gumbov v vezju. Saj, če preverjamo status gumba samo z metodo digitalRead(), bomo dobili zelo nestabilen podatek. Tako moramo uporabiti metodo, ki se imenuje *debounce* [10]. To pa pomeni, da stabiliziramo branje stikala. Če bi branje tipke gledali skozi osciloskop, bi dobili tak izris na zaslonu osciloskopa, kot prikazuje naslednja slika (Slika 19).



Slika 19: Branje izhoda tipke z osciloskopom

Da bi si zgornjo sliko lažje predstavljali, opišimo dogajanje pri pritisku tipke bolj podrobno. Če gledamo kontakte v tipki v trenutku, ko pritisnemo nanjo, ugotovimo, da ne moremo zagotoviti, da bi vedno pritisnili tako, da bi iz enega stanja preklopili v drugega brez vmesnih motenj. Zato lahko v vezju uporabimo kondenzator, prek katerega stabiliziramo stanje. V primeru mikrokrmilnika pa lahko to rešimo programsko, s tem da beremo stanje na tipki dvakrat s krajšim razmikom med branjem stanja na tipki. Takšno programsko rešitev smo uporabili tudi mi, prikazana pa je v spodnji kodi.

```
boolean tipka(int pin) {
  int state = digitalRead(pin);
  boolean r=false;
  if(state==HIGH) {
    delay(50);
    if(state==digitalRead(pin)) r=true;
    while(digitalRead(pin)==HIGH) {
    }
  }
  return r;
}
```

Koda 2: Programska rešitev

5.3 Čas na mikrokrmilniku

Nekaj težav nam je povzročal čas na mikrokrmilniku. Prva težava je bila, ker se podatki ob ponovnem zagonu oz. odklopu iz napajanja naložijo, kakor so določeni programsko. Ravno tako je z uro, ki je zdaj nastavljena na 20:24 in se ob vsakem ponovnem zagonu mikrokrmilnika nastavi na ta čas. Druga težava, ki jo velja omeniti tudi tukaj, so vse spremenljivke (statusi sob, koliko časa gorijo luči, kdaj se prične postopek krmiljenja), ki se ravno tako nastavijo na vrednosti, določene v kodi.

To drugo težavo bi lahko rešili z uporabo pomnilnika EEPROM na samem mikrokrmilniku (ATmega32 ima 1024 B prostora za shranjevanje podatkov v EEPROM [11]) in bi te nastavitve ob koncu nastavljanja (izhod iz nastavitev) prepisali v EEPROM [12]. Tako bi te vrednosti ostale zapisane tam in bi se ob ponovnem zagonu mikrokrmilnika prebrale iz EEPROM-a. Ure pa verjetno ni smiselno zapisovati v EEPROM, ampak jo je boljše ob vsakem ponovnem zagonu sinhronizirati z nekim zunanjim virom (prek modula GPS ali s povezavo na računalnik). Primer sinhronizacije je tudi naveden na spletni strani [13], kjer smo našli vse potrebne informacije za delo s časom.

Naslednja težava, na katero smo naleteli, pa je spreminjanje časa začetka. Za čas začetka se ravno tako uporablja spremenljivka *time*, težava pa je, da je ne moremo kar tako spreminjati, temveč jo moramo ponovno definirati, v tem primeru pa zgubimo informacijo o trenutnem času. Kako poteka spreminjanje ure začetka, je prikazano v spodnji kodi.

```
if(tipka(10)) {
       trenutniCas=now();
       millis_od=millis();
       setTime(hour(zacniOb), minute(zacniOb)+1,0,0,0,0);
       zacniOb=now();
       setTime(hour(trenutniCas)+hour(millis()-millis_od), minute(trenutniCas)+minute(millis()-
millis_od), second(trenutniCas)+second(millis()-millis_od),0,0,0);
     }
     else if(tipka(11)) {
       trenutniCas=now();
       millis_od=millis();
       setTime(hour(zacniOb)+1, minute(zacniOb),0,0,0,0);
       zacniOb=now();
       setTime(hour(trenutniCas)+hour(millis()-millis_od), minute(trenutniCas)+minute(millis()-
millis_od), second(trenutniCas)+second(millis()-millis_od),0,0,0);
     }
```

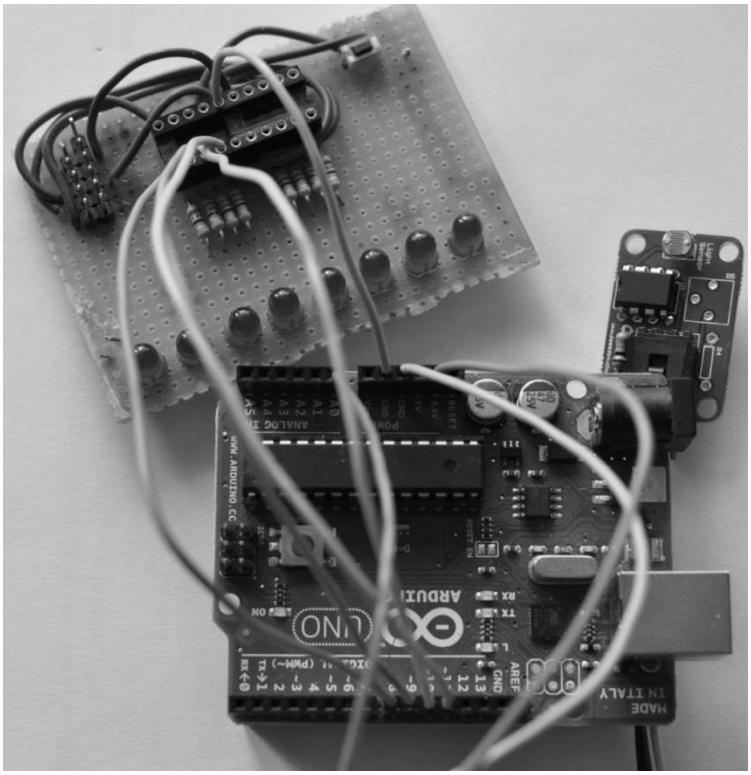
Koda 3: Spreminjanje časa začetka

V zgornji kodi je predstavljeno, kako s tipkama T2 in T3 povečujemo ure in minute. Če je tipka(10) (to pomeni tipka, ki je priklopljena na podatkovni vhod 10) aktivna, potem

shranimo trenutni čas v spremenljivko in zapišemo, kdaj se je pričelo spreminjanje ure. Nato spremenimo uro, povečamo minute za 1 in nastavimo v spremenljivko zacniOb. Na koncu pa moramo še popraviti trenutni čas za to vrednost, ki je minil od pritiska na tipko(10) pa do takrat, ko je bila tipka spuščena. Če si zamislimo primer, da bi bili v nastavitvah in držali tipko eno minuto in če potem ne bi prišteli tega časa, ki je minil, bi trenutni čas zaostajal za eno minuto.

5.4 Svetlobno tipalo

Eden izmed teh primerov, ki vsebuje vse osnovne metode za programiranje, je identifikator svetlobe. S štirimi diodami LED ponazorimo, koliko svetlobe pada na svetlobno tipalo: bolj je temno, več diod bo svetilo. Obenem pa sporoča vrednost na svetlobnem tipalu preko serijske komunikacije, tako da vsebuje ta primer vse, kar smo pozneje potrebovali za končni projekt, z izjemo analognega branja vrednosti. Spodaj pa je priložena tudi koda (Koda 4) za celotni postopek svetlobnega tipala.



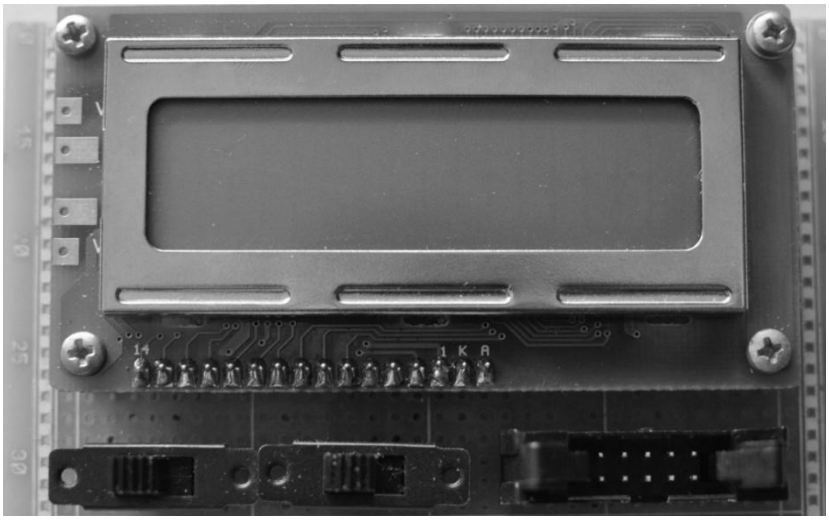
Slika 20: Svetlobno tipalo

```
float light;
void setup() {
  delay(3000);

  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(11, OUTPUT);

  light=0;
  Serial.begin(9600);
}

void loop() {
  light=analogRead(0);

  if(light<300) digitalWrite(8, HIGH);
  else digitalWrite(8, LOW);
  if(light<225) digitalWrite(9, HIGH);
  else digitalWrite(9, LOW);
  if(light<150) digitalWrite(10, HIGH);
  else digitalWrite(10, LOW);
  if(light<75) digitalWrite(11, HIGH);
  else digitalWrite(11, LOW);

  Serial.println(light);
  delay(1000);
}
```

Koda 4: Primer svetlobnega tipala

5.5 Prikazovalnik LCD

Za prikaz nastavitev in stanja na centralni enoti smo uporabili prikazovalnik LCD (Slika 21), a pred samo uporabo smo se morali še spoznati s tem. Na voljo smo imeli standardni prikazovalnik LCD, na katerem lahko prikažemo 2 vrstici po 16 znakov. Ker smo dobili vse elemente na fakulteti, vključno s prikazovalnikom LCD, nismo spreminjali ničesar, tako da na samem LCD-ju nismo dodajali povezav za osvetlitev.



Slika 21: prikazovalnik LCD 16x2

Na sami ploščici je poleg prikazovalnika tudi priključek[10] za povezavo prikazovalnika na mikrokrmilnik in še dve stikali. Eno stikalo se uporablja za vklop in izklop prikazovalnika, drugo stikalo pa je prosto in ga je možno uporabiti po lastni volji.

Po nasvetih različnih člankov smo se odločili za uporabo knjižnice *LiquidCrystal* [14], ki je tudi že priložena v namestitvenem paketu, ko si nameščamo programsko orodje Arduino.

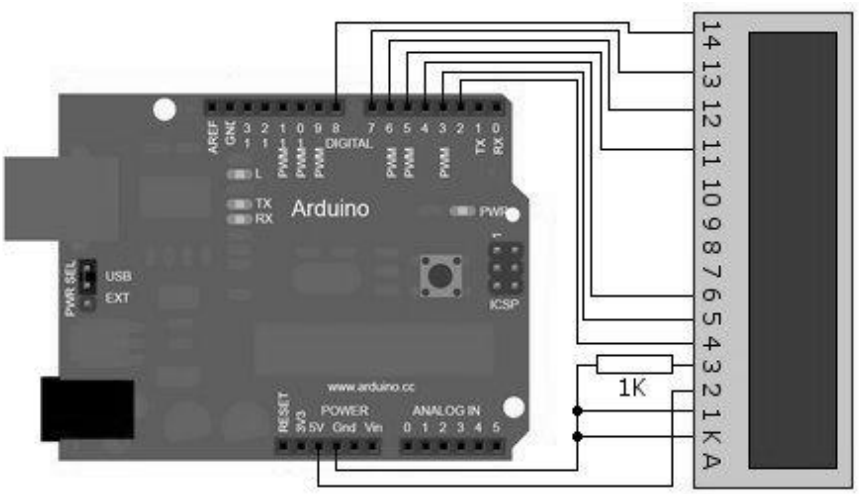
```
LiquidCrystal lcd(2,3,4,5,6,7,8);
```

Koda 3: Nova instanca prikazovalnika LCD

Zgoraj vidimo prikaz, kako se definira novo instanco, prek katere komuniciramo s prikazovalnikom LCD. Sprva smo imeli težave, ko smo želeli uporabljati prikazovalnik, saj se povezave v našem primeru razlikujejo od teh, ki so navedene v primerih, katere smo videli, ko smo raziskovali, kako uporabiti prikazovalnik LCD na programatorju Arduino. Namreč, če v definiciji niso pravilno definirane nožice, kamor je priklopljen prikazovalnik, nam po zaslonu prikazuje čudne znake. Zato smo morali še enkrat preveriti povezave in tako smo prišli do pravilne kode.

[10] Priključek IDC

Na spodnji sliki (Slika 22) je prikazana shema povezav med programatorjem Arduino in prikazovalnikom LCD.



Slika 22: Shema priklopa prikazovalnika LCD 16x2

Za uporabo prikazovalnika moramo najprej v delu kode, kjer nastavljamo parametre, navesti, da bomo uporabljali LCD, obenem pa navedemo tudi, kakšnih dimenzij je naš prikazovalnik in nastavimo začetni položaj pisanja.

```
void setup() {
  lcd.begin(16,2);
  lcd.setCursor(0,0);
  lcd.print("Loading...");
}
```

Koda 5: Nastavitve za prikazovalnik

Kot vidimo napisano v zgornji kodi, z metodo begin() navedemo, kakšen prikazovalnik bomo uporabljali. Omenili smo že, da smo uporabljali 2-vrstičnega po 16 znakov. Z ukazom setCursor() postavimo začetni položaj za pisanje. 0, 0 predstavlja prvo vrstico, prvi znak na levi strani. In potem z ukazom print() preprosto izpisujemo znake na prikazovalnik.

Če izpisujemo znake in prekoračimo 16 znakov, prikazovalnik ne bo šel samodejno v novo vrstico, ampak se moramo z ukazom setCursor(0,1) prestaviti na začetek druge vrstice.

5.6 Serijska komunikacija (Serial)

Po uspešno opravljeni nastavitvi in priklopu brezžičnega-komunikacijskega modula smo se lotili še spoznavanja s komunikacijo med moduli, a ni bilo nič kaj novega, saj poteka komunikacija z modulom s pomočjo knjižnice Serial. Uporabljajo se funkcije, napisane v tej knjižnici, in sicer za pošiljanje Serial.print(), za branje pa Serial.available() in Serial.read(). Kot nam pove že ime funkcije available(), le-ta preveri, ali je na podatkovnem vhodu kakšen podatek, in če je, potem ta podatek preberemo z read(). Pomembno je, da ob začetku (v setup()) nastavimo hitrost serijske komunikacije, in sicer z ukazom Serial.begin(). Kot parameter podamo hitrost prenosa podatkov. Mora pa se ujemati z vrednostjo, ki je nastavljena na brezžično-komunikacijskem modulu, saj se v nasprotnem primeru podatki ne berejo pravilno. To pomeni, če pošljemo na serijsko komunikacijo podatek A s hitrostjo 9600, na brezžično komunikacijskem modulu pa beremo s 19200, bodo podatki nepravilni in se bodo razlikovali od poslanih.

5.7 Definiranje signalov

Edino, kar smo še morali storiti, je to, da smo določili, s kakšnimi signali bomo krmilili luči v sobah. To smo zaenkrat naredili tako, da je sestavljeno iz dveh števk, in sicer predstavlja prva števka sobo, v kateri naj se zgodi neka akcija, v drugi števki pa navedemo, kakšna naj bo ta akcija. Če npr. želimo prižgati luč v sobi 5, potem se izvede naslednja koda:

```
vkljuci(5);
…
void vkljuci(int i_soba) {
  soba[i_soba-1]=true;

  Serial.print((char)(i_soba*10+1));
}
```

Koda 6: Primer prižiganja luči v sobi 5

Če bi pogledali na serijska vrata, kamor je priklopljen brezžično-komunikacijski modul XBee, bi videli, da se pošlje signal 51. Ena izmed slabosti oz. kar bi lahko popravili, je to, da bi bil za eno sobo en sprejemnik, potem bi morali drugače pošiljati podatke. V takem primeru bi bil podatek sestavljen iz treh števk, kjer bi prva števka označevala, kateri sprejemnik naj sprejme podatek, drugi podatek bi povedal, za kateri izhod (katera luč) velja, in tretja števka še akcija, ali se luč prižge ali ugasne.

Ko smo znali komunicirati z moduloma, smo ju še nastavili tako, da sta bila v istem omrežju, kot smo že opisali zgoraj s postopkom nastavljanja z X-CTU. Nato smo se lotili še programiranja mikrokrmilnika.

5.8 Sprejemnik

Najenostavnejše je bilo narediti sprejemnik. Ker pa smo imeli samo dva programatorja Arduino in tri brezžično-komunikacijske module, smo na sprejemnik morali povezati dva modula. Težava se pojavi, saj ima Arduino zgolj ena vrata za serijsko komunikacijo. Vendar smo našli rešitev s knjižnico NewSoftSerial [15], ki omogoči uporabo še dodatnega vhodna in izhoda za serijsko komunikacijo.

```
#include <NewSoftSerial.h>
#define rxPin1 2
#define txPin1 4
NewSoftSerial mySerial(rxPin1, txPin1);
```

Koda 7: Nova instanca serijske komunikacije

Po definiciji zgornje kode smo lahko uporabljali dve serijski komunikaciji, s tem pa dva brezžično-komunikacijska modula na enem mikrokrmilniku. Enega na 0, 1 priključkih, to so standardna serijska vrata, drugi modul pa smo priklopili na 2, 4 in ga uporabljali prek knjižnice NewSoftSerial. Ta knjižnica ima iste funkcije kakor običajna knjižnica Serial. Priključka 3 nismo uporabili, ker je definiran kot priključek PWM, kar pa nam je pri serijski komunikaciji povzročalo težave, zato smo raje uporabili priključek 4.

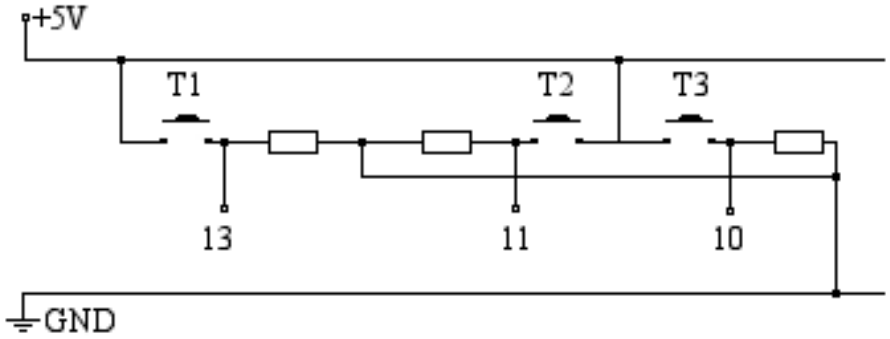
5.9 Oddajnik

V našem primeru je to centralo-krmilna enota, ki skrbi za vse nastavitve in pošiljanje podatkov modulom za prižiganje in ugašanje luči. Sprva smo se lotili osnovnega, statičnega delovanja. Vnaprej smo določili, koliko časa bo gorela luč in kdaj se bo vse začelo izvajati. Tako smo preverili delovanje brezžične komunikacije med centralno-krmilno enoto in moduli za prižiganje in ugašanje luči. Potem smo začeli izdelek avtomatiziranega brezžičnega krmiljenja luči razvijati še v tej smeri, da lahko čim več stvari nastavljamo poljubno, prek nastavitev. Najprej smo omogočili nastavljanje ure začetka postopka prižiganja in ugašanja, pri čemer smo morali v vezje vključiti dva dodatna gumba, s katerima spreminjamo vrednost ure in minute začetka. Nato smo dodali še, da se stanje luči in čas aktivne sobe zapisujeta v tabelo, tako da je možno spreminjati tudi vrednosti teh parametrov. Število sob je mogoče prilagoditi programsko s povečanjem tabele, koliko časa pa bo v sobi gorela luč, je možno nastavljati v nastavitvah. Tako smo za boljšo preglednost še dodali meni, v katerem to nastavljamo. Spodnja slika (Slika 23) prikazuje izpis na zaslonu ob nastavljanju, koliko časa bo gorela luč v sobi 3.

Slika 23: Nastavljanje, koliko časa gori luč v sobi 3

5.9.1 Upravljanje nastavitev na centralno-krmilni enoti

Tri tipke, ki so še poleg tipke za vklop in izklop, se uporabljajo za premikanje po nastavitvah in za spreminjanje vrednosti, ki jih nastavljamo. Te vrednosti so sprva ob zagonu nastavljene programsko v kodi. Če pogledamo na shemo (Slika 24), se uporablja tipka T1 za vstop v nastavitve. V nastavitve vstopimo tako, da zadržimo tipko T1 za 3 sekunde. Po nastavitvah se potem premikamo z gumbom, ki ga imamo za prižiganje in ugašanje.



Slika 24: Shema gumbov za nastavitve

Gumba T2 in T3 pa imata različne vloge. V primeru, ko nastavljamo uro s tipko T2, povečujemo minute, s tipko T3 pa uro pričetka krmiljenja luči, če je postopek vklopljen. Ko pa nastavljamo, koliko časa bo gorela luč, s tipko T2 povečujemo, s tipko T3 pa zmanjšujemo čas aktivne sobe. S tipko T1 pa se premikamo med sobami.

Spodaj (Koda 8) pa je še prikazan delček kode, s katero se spreminja vrednost, koliko časa bo gorela luč.

```
else if(i_izpisNastavitve==1) {
     if(tipka(10)) {
       casPrizgano[id_sobe]+=1;
     }
     else if(tipka(11)) {
       casPrizgano[id_sobe]-=1;
     }
     else if(tipka(13)) id_sobe=(id_sobe+1)%STEVILO_SOB;
   }
```

Koda 8: Spreminjanje nastavitev

6 Sklepne ugotovitve

Ob izdelavi avtomatiziranega brezžičnega prižiganja in ugašanja luči smo se spoznali s programiranjem mikrokrmilnikov z uporabo različnih tipal in uporabo ene izmed možnosti brezžične komunikacije v svetu mikrokrmilnikov. Naučili smo se, da lahko z uporabo mikrokrmilnikov rešimo enostavne primere. Dobra lastnost dela z mikrokrmilniki je tudi ta, da lahko razvijamo na enem tipu mikrokrmilnika in potem v končni fazi razvoj le prenesemo na strnjeno obliko vezja, v katerem uporabimo tak mikrokrmilnik, kakršen zadostuje za potrebe delovanja.

Težave, ki se pojavljajo ob delovanju postopka prižiganja in ugašanja luči in katere bi bilo dobro odpraviti, so v prvi vrsti težave z uro. To je najbolj moteče in bi bilo veliko bolje, da bi se ura sinhronizirala z zunanjim virom. Naslednja težava, ki je moteča, je tudi povezana s ponovnim zagonom centralno-krmilne enote. Namreč, vrednosti, ki smo jih nastavili v nastavitvah, se ne shranjujejo. Te in še ostale težave, ki bi se pojavile tekom nadaljnjega testiranja, bi morali odpraviti in tako bi ta produkt avtomatiziranega brezžičnega prižiganja in ugašanja luči postal ljudem zanimiv in dostopnejši za uporabo.

Produkt avtomatiziranega prižiganja in ugašanja luči, ki smo ga naredili ob tem diplomskem delu, bi po mojem mnenju lahko uporabili ne samo za razsvetljavo v hiši, ampak tudi za avtomatizirano prižiganje in ugašanje na drugih komponentah. Seveda je treba v sam produkt vložiti še mnogo razvoja in mu dodati še kakšno idejo, da bi bil izdelek še zanimivejši in uporabnejši. Še ena izmed idej, s katero bi lahko dopolnili in s tem izboljšali uporabnost avtomatiziranega krmiljenja luči, je, da bi izdelek vključili v internetno omrežje, tako da bi lahko statuse preverjali oddaljeno z računalnika ali pa bi dodali možnost povezave s telefonom.

Literatura

[1] (2011) Product Manual. Dostopno na: ftp://ftp1.digi.com/support/documentation/90000866_C.pdf

[2] (2011) XB24-Z7WIT-004. Dostopno na: http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/zigbee-mesh-module/xbee-zb-module#overview

[3] (2011) XBeePins. Dostopno na: http://code.google.com/p/xbee-api/wiki/XBeePins

[4] (2011) ArduinoXbeeShield. Dostopno na: http://arduino.cc/it/Main/ArduinoXbeeShield

[5] (2011) X-CTU. Dostopno na: http://www.digi.com/support/productdetail?pid=3352&osvid=57&type=utilities

[6] (2011) XBee adapter – Simple wireless communication. Dostopno na: http://www.ladyada.net/make/xbee/

[7] (2011) XBee™ Series 2 OEM RF Module. Dostopno na: ftp://ftp1.digi.com/support/documentation/90000866_A.pdf

[8] (2011) Setup a Zigbee Network. Dostopno na: https://sites.google.com/site/xbeetutorial/xbee-introduction/zigbee_setup

[9] (2011) Triac interface to Microcontrollers. Dostopno na: http://www.youtube.com/watch?v=n2f9u2xI624

[10] (2011) Switch Debounce. Dostopno na: http://www.labbookpages.co.uk/electronics/debounce.html

[11] (2011) ATmega32 datasheet. Dostopno na: http://www.atmel.com/Images/doc2503.pdf

[12] (2011) Arduino Playground – EEPROM. Dostopno na: http://arduino.cc/playground/Code/EEPROMWriteAnything

[13] (2011) Arduino Playground – Time. Dostopno na: http://www.arduino.cc/playground/Code/Time

[14] (2011) Arduino Playground – LiquidCrystal. Dostopno na: http://arduino.cc/en/Tutorial/LiquidCrystal

[15] (2011) NewSoftSerial | Arduiniana. Dostopno na: http://arduiniana.org/libraries/newsoftserial/