UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Miha Mežik

Razvoj modula za varovanje objekta s pomočjo mikrokrmilnika

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

Ljubljana, 2014

UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Miha Mežik

Razvoj modula za varovanje objekta s pomočjo mikrokrmilnika

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: izr. prof. dr. Branko Šter

Ljubljana, 2014

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Zasnujte in realizirajte varnostni sistem za varovanje objektov. Kot platformo uporabite mikrokrmilnik. Opišite funkcionalnosti varnostnega sistema, uporabljena orodja pri razvoju sistema in način delovanja.

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Miha Mežik, z vpisno številko **63080324**, sem avtor diplomskega dela z naslovom:

Razvoj modula za varovanje objekta s pomočjo mikrokrmilnika

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom izr. prof. dr. Branka Štera,
- 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 na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 24. junija 2014 Podpis avtorja:

V prvi vrsti bi se rad zahvalil mentorju izr. prof. dr. Branku Šteru za vložen trud in pomoč, ki mi jo je nudil med izdelavo diplomske naloge. Prav tako bi se rad zahvalil svoji družini, punci, in prijateljem za finančno in moralno podporo, katere sem bil deležen v času študija.

Kazalo

Povzetek

Že pračlovek je veliko časa namenil iskanju in varovanju zatočišča, ki mu je nudilo varnost pred sovražniki in vremenskimi vplivi. Z razvojem tehnološko naprednih varnostnih sistemov za uporabo v vojaških in letalskih sistemih se je močno dvignila tudi stopnja varovanja objektov. Trenutno imamo na trgu široko paleto sistemov za varovanje objektov, od najbolj osnovnih, kot so razne prepreke, ograje in protivlomna vrata, pa vse do najbolj izpopolnjenih varnostnih sistemov, ki temeljijo na hitrem odzivu, učinkovitosti in preprostosti njihove uporabe.

V diplomski nalogi je osrednja tema razvoj modula za varovanje objekta s pomočjo mikrokrmilnika. Namen je razviti ustrezen modul, ki bo sposoben zaznavati spremembe v prostoru in se nanje tudi ustrezno odzvati. Za osnovo modula smo uporabili razvojno platformo Arduino. Dokupili smo tudi nekaj specifičnih modulov, ki smo jih uporabili za določene funkcionalnosti. Za prikazovanje izhodnih informacij smo uporabili LCD zaslon, na katerem se prikazujejo aktivni alarmi, sicer pa ura z datumom. Glavni namen diplomske naloge je bil dokazati, da je razvoj tovrstnega modula lahko zelo enostaven, hkrati tudi poučen, sem ter tja pa lahko najdemo tudi kakšen trši oreh, ki pa ga ni moč streti v doglednem času.

Ključne besede: varnostni sistem, pametna hiša, alarm, senzor, mikrokrmilnik, programiranje, platforma Arduino

Abstract

A prehistoric man used to spend a lot of time for finding and protecting his shelter, which provided security against enemies and weather. The development of advanced technologic security systems for use in military and aeronautical systems has raised the level of protection. Current market provides a wide range of systems for the protection, some of them are the most basic such as barriers, fences and burglar-proof doors, but most of them are very sophisticated security systems, based on fast response, efficiency and simplicity of use.

The thesis central theme is development of module for the protection with a microcontroller. Target of thesis is to develop an appropriate module, which will be capable of detecting changes in space and appropriately responding. The heart of module is the Arduino development platform. We have bought some special elements also, that we have used for some specific functionalities. Output information is displayed on LCD screen, which is showing active alarms, otherwise clock with date. The main purpose of the thesis was to demonstrate that the development of such a module can be very easy, but also instructive, sometimes we can find some harder issues, which can't be solved in due time.

Keywords: security system, smart house, alarm, sensor, microchip, programming, Arduino platform

Poglavje 1 Uvod

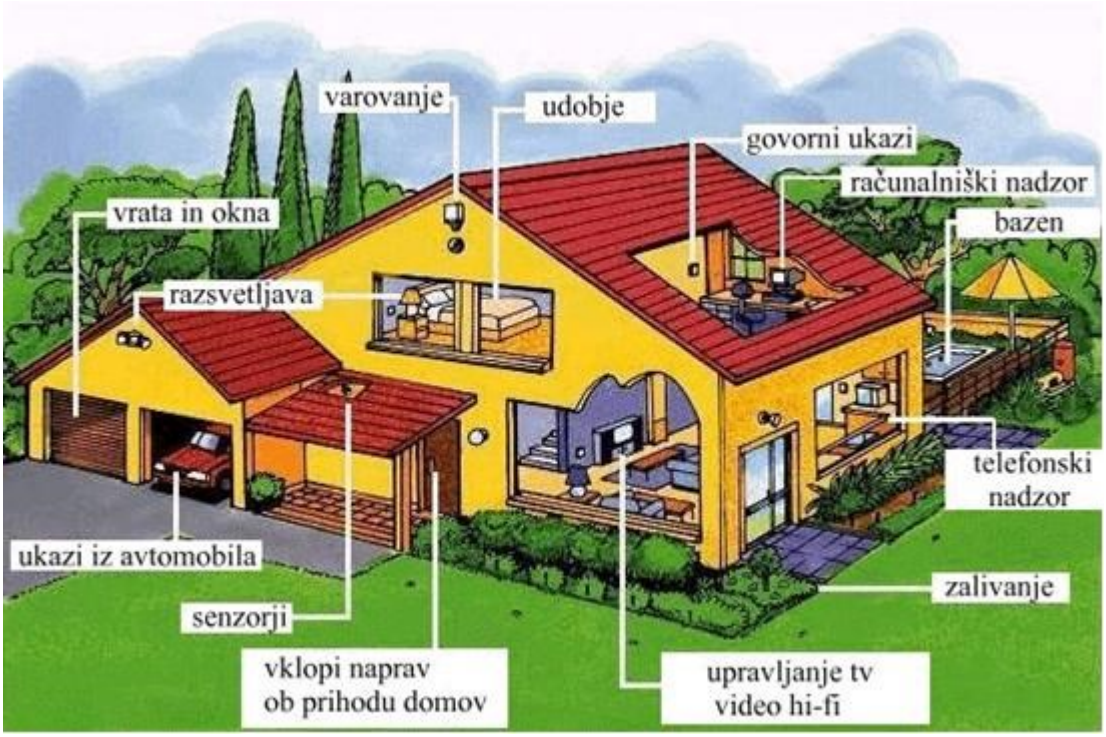
Že v času pračloveka je bilo jasno, da si mora vsako bitje poiskati zatočišče, ki mu bo nudilo varnost pred vsiljivci oz. sovražniki. Najbolj primitiven primer je jama, s pomočjo katere se je človek obvaroval pred naravnimi pojavi, kot so dež, vročina, veter oz. pred nepovabljenimi gosti. Za zadostitev tovrstnega varovanja je moral človek vložiti zelo veliko svojega znanja, truda in moči, ki bi jo sicer lahko porabil za ostala opravila.

Vse to lahko preslikamo v današnji čas, ko imamo ljudje objekte, za katere bi radi v vsakem trenutku vedeli za dogajanje v ali ob njem. Običajno pomislimo na najslabši scenarij, lahko je v njem nepovabljen gost, ki nam želi škodovati na različne načine, lahko pa pride tudi do nepričakovanih dogodkov, kot sta požar ali poplava. V tem primeru lahko namesto nas tovrstno delo opravlja nekaj, čemur pravimo varnostni sistem.

Z razvojem tehnološko naprednih sistemov, ki se uporabljajo v vojaške in letalske namene, kjer je varnost na prvem mestu, se je stopnja varovanja objektov bistveno povečala, hkrati pa je varovanje postalo bolj učinkovito in v smislu porabe energije ter onesnaževanja tudi bolj ekološko. Človek se s pomočjo tovrstnih sistemov počuti bolj varnega in udobnega. Varnostne sisteme je možno vgraditi tako v nepremičnine, kot tudi v premičnine, tako smo praktično obdani z njimi na vsakem koraku. Na sprehodu po mestu nas spremljajo varnostne kamere, prav tako ob vstopu v trgovino, kjer so kamere nameščene predvsem z namenom varovanja blaga, največkrat pa so poleg kamer nameščeni tudi senzorji gibanja, s katerimi postane sistem za varovanje precej bolj učinkovit in avtomatiziran.

Na trgu imamo veliko različic varnostnih sistemov, ki so prilagojeni zahtevam in potrebam uporabnika. Tako je možno izbirati med različnimi senzorji, kot so senzor gibanja, senzor plina, senzor zvoka, temperaturni senzor (termometer) in senzor dima. Podobno je s komunikacijsko opremo, ki služi za posredovanje sporočil okolici oz. ključnim osebam. Skoraj vedno je prisotna alarmna naprava, ki ob sprožitvi prične oddajati človeku zelo moteč zvočni signal, s katerim opozarja na določeno, nepričakovano spremembo v prostoru. Za rokovanje z varnostnim sistemom je največkrat prisotna tipkovnica, s pomočjo katere lahko vklopimo ali izklopimo varnostni sistem. Modernejša različica pa omogoča rokovanje z daljinskim upravljalnikom, ki služi podobnim nalogam.

Varnostni sistem se trenutno lahko pojavlja v kombinaciji s sistemom za upravljanje pametnih hiš (Slika 1.1). Takšni sistemi služijo poleg varovanja tudi za udobnejše bivanje in večjo učinkovitost vgrajenih naprav [6], saj omogočajo nastavljanje različnih parametrov za delovanje naprav. Ti parametri lahko predstavljajo temperaturo izbranega prostora, postavitev senčnih elementov, svetilnost žarometov, usmerjenost žarometov ali glasnost predvajane glasbe. Zanimanje za tovrstne objekte je v zadnjih letih zelo naraslo in je še vedno v fazi naraščanja. Prednosti, ki jih ponuja pametna hiša, so udobje, učinkovitost, ekološka usmerjenost in varnost.



Slika 1.1: Primer pametne hiše

Udobje je pojem, ki je največkrat odvisen od finančne situacije posameznika. V našem primeru je primer udobja, da lahko iz kavča s pomočjo daljinskega upravljalnika oz. preko pametnega telefona nastavimo pozicijo senčnih elementov, ki nam v danem trenutku najbolj ustreza. Enako velja za svetilnost in usmerjenost žarometov.

Učinkovitost je glavni razlog za sprejem odločitve o gradnji pametne hiše. Tukaj mislimo predvsem na varčevanje z električno energijo, kar dosežemo s pametno nastavljenimi parametri za vgrajene naprave in s časovnimi omejitvami delovanja naprav (osvetlitev ali TV). Hkrati lahko z nastavljanjem pozicije senčnih elementov varčujemo pri porabi toplotne energije. Učinkovitost in ekologija sta povezani z roko v roki, brez učinkovitosti ni možnosti za ekologijo, ki pa je v trenutnih razmerah faktor izrednega pomena.

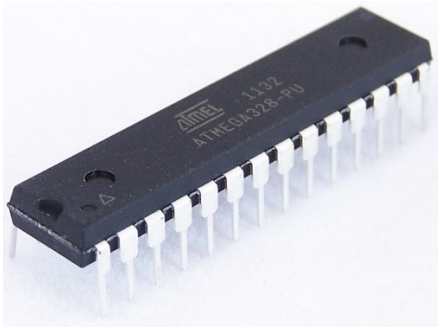
Varnost je glavnega pomena na vsakem koraku posameznika. Brez varnosti, bi bilo življenje veliko manj kompliciranо, saj bi človek lahko pozabil na marsikatero zakonitost, ki ga trenutno varuje pred nevarnostmi. Tako lahko zatrdimo, da je varnost resnično ključnega pomena pri doseganju kateregakoli cilja.

V diplomski nalogi sem se osredotočil predvsem na varnostni sistem za varovanje objektov, v katerem sem dobil idejo o razvoju lastnega varnostnega sistema. Odločil sem se, da pripravim modul za varovanje objektov s pomočjo mikrokrmilnika, ki bo zaznaval spremembe v okolici.

Glede na to, da me področje razvoja sistemov z mikrokrmilniki zelo veseli in zanima, želim s z razvojem omenjenega modula raziskati in pokazati probleme, s katerimi se lahko sreča vsak, ki se želi spopasti z avtomatizacijo kateregakoli sistema. To je sicer zelo široko področje, ki se uporablja tudi na področju robotike, toda v našem primeru, se bomo osredotočili le na ozek sklop razvoja s pomočjo vnaprej pripravljene razvojne plošče Arduino UNO in njenimi dodatki, kot so senzorji in sprejemniki.

1.1 Mikrokrmilniki

Mikrokrmilnik (Slika 1.2) je enota, ki služi za izvajanje ukazov. Sestavljena je iz pomnilnika, procesne enote in različnih vmesnikov. S tujko mu pravimo tudi »microchip«, sicer pa ima razen vhodno-izhodnih enot, skoraj vse lastnosti mikroračunalnika [4]. Običajno je to majhno vezje s stranskimi nožicami, ki so namenjene komunikaciji med vhodno-izhodnimi enotami in mikrokrmilnikom. Nožice predstavljajo pine, ki jih delimo na vhode in izhode. Preko njih se prenašajo podatki o informacijah, glede na izbran pin. Mikrokrmilnike srečamo v večini elektronskih naprav, ki jih uporabljamo vsakodnevno. Glede na vgrajen tip procesorja, se mikrokrmilniki delijo v družine. Glavni razliki med mikrokrmilniki sta hitrost delovanja oz. takt in število bitov, ki jih lahko obdeluje hkrati. Med bolj razširjene sodijo mikrokrmilniki ATMEL, ki jih običajno uporabljamo v šolske namene (okolje ARM).

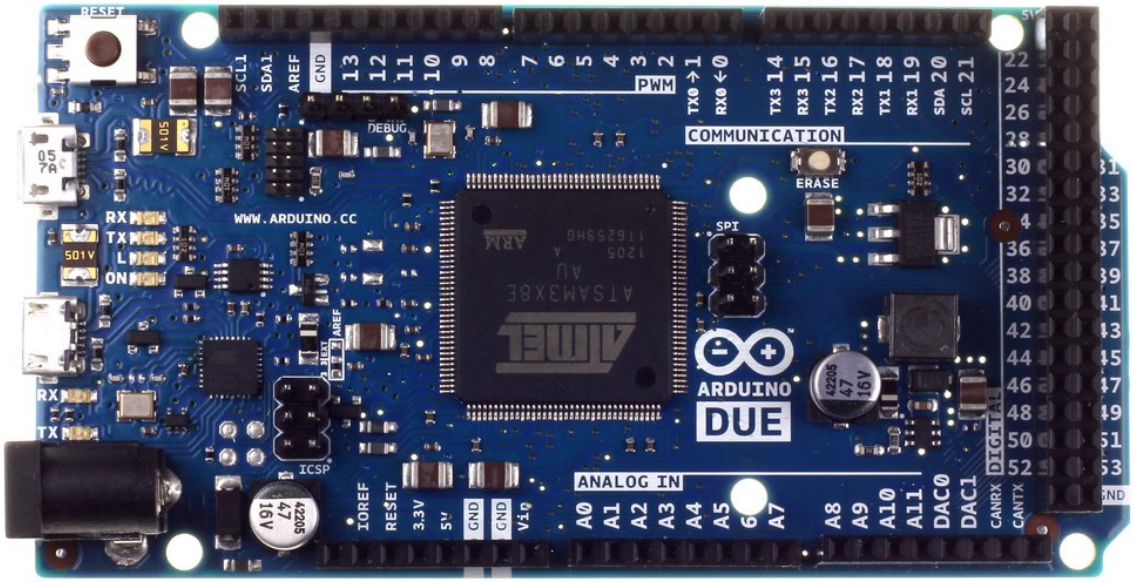


Slika 1.2: Mikrokrmilnik ATMEGA 328-PU

1.2 Mikrokrmilniki na ploščah

Mikrokrmilnik v osnovi nima vgrajenih vhodno-izhodnih enot, kot so gumbi, prikazovalniki, diode LED, tako so brez dodatnih elementov praktično neuporabni. V ta namen so nastale matične plošče, na katerih so sprva združevali le mikrokrmilnik in enote, potrebne za praktično uporabo, trenutno pa imamo na ploščah vgrajenih tudi po več deset mikrokrmilnikov. Sicer so to vnaprej pripravljena tiskana vezja s točno določenimi specifikacijami, ki jih lahko uporabimo kot osnovo za razvoj avtomatiziranega sistema. S pomočjo priloženih informacij, lahko ugotovimo s kakšno hitrostjo deluje vgrajen mikrokrmilnik, kakšne tipe vhodov in izhodov podpira, na kakšen način je možno programirati mikrokrmilnik, kje in na kakšen način je shranjen program in še številne druge informacije.

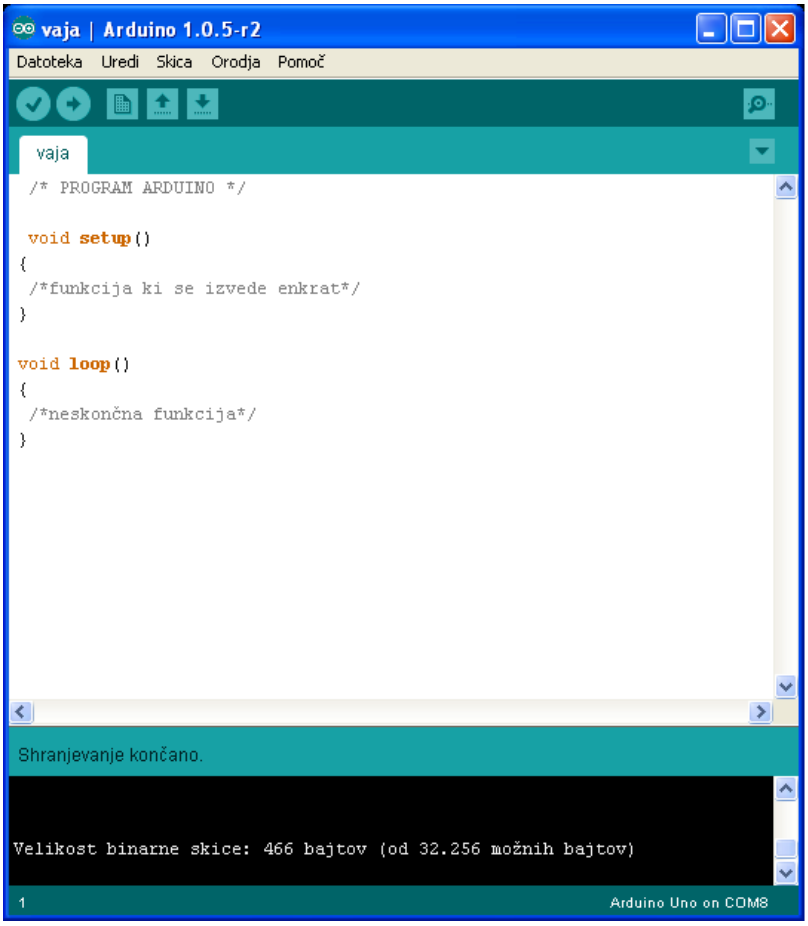
V diplomski nalogi smo se na podlagi ciljev odločili, da bomo za razvoj modula uporabili razvojno ploščo Arduino (Slika 1.3) s pripadajočim razvojnim okoljem Arduino IDE (3.3.1), ki je trenutno med ljubitelji elektrotehnike najbolj priljubljeno.



Slika 1.3: Plošča Arduino DUE

1.3 Programiranje mikrokrmilnikov

Zaradi prostorske stiske lahko posamezen pin uporabljamo tudi za več različnih operacij in na različne načine. Na kakšen način in kako bo uporabljen pin, določimo s programom, ki ga naložimo v poseben pomnilnik matične plošče. Ta program se ob vsakem zagonu oz. priklopu začne izvajati v samem mikrokrmilniku. Programiranje mikrokrmilnikov s pomočjo okolja Arduino je izredno enostavno, saj ima priloženo vso potrebno programsko opremo (Slika 1.4), ki jo je potrebno namestiti na računalnik. Tako ima uporabnik vse pogoje za razvoj prvega programa za mikrokrmilnik. Programska oprema Arduino je sestavljena iz prevajalnika, zagonskega nalagalnika in programskega jezika, ki temelji na programskem jeziku C.



Slika 1.4: Razvojno okolje Arduino

Poglavje 2 Varnostni sistem

Potreba po varnosti je ena izmed osnovnih potreb vsakega človeka. Posameznik, ki ima v lasti kakršnokoli nepremičnino, si močno prizadeva, da bi bila le-ta čimbolj varna za bivanje, v primeru poslovne nepremičnine pa za zaposlene in stranke. Pri tem je ključnega pomena, da je upravljanje varnosti karseda enostavno. Z vgradnjo varnostnega sistema skušamo zmanjšati vpliv človeških dejavnikov pri varovanju.

Varnostni sistemi, ki so namenjeni tovrstnemu varovanju, morajo delovati brezhibno. Pri izbiri varnostnega sistema cena ne bi smela odigrati pomembne vloge, saj v tem primeru največkrat velja, da cena sorazmerno narašča s kompleksnostjo, veličino in zanesljivostjo sistema. Po drugi strani pa so tovrstni sistemi postali precej bolj dostopnejši, tako običajnim uporabnikom nudijo zadovoljivo varnost in enostavno uporabo, pri tem pa jim ni potrebno odšteti velikih zneskov. S samo vgradnjo varnostnega sistema pa privarčujejo tudi pri zavarovanju nepremičnine, saj zavarovalnice upoštevajo faktor varnosti.

2.1 Zaščita

Za zaščito pred vsiljivci imamo na voljo široko paleto možnosti, s katerimi se lahko borimo proti nezaželenim dogodkom. Tako lahko vgradimo mehansko zaščito [3], ki služi le obveščanju o varovanem območju in preprečitvi vstopa nepooblaščenim osebam. Za mehansko varovanje lahko uporabimo elemente, kot so ključavnice, različne ograje, protivlomna vrata in druge ovire. Nadgradnjo mehanske zaščite lahko opravimo z elektronsko zaščito, ki jo delimo na notranjo in zunanjo, saj se pri zunanji uporabljajo posebni materiali, ki so odporni na zunanje vremenske vplive. V primeru, da se vsiljivec uspešno izogne mehanski zaščiti, se v objektu sproži elektronsko varovanje, ki temelji na detektorjih gibanja. Elektronsko varovanje se na takšen dogodek ustrezno odzove, in sicer tako, da aktivira svetlobno opozarjanje, obveščanje preko komunikacijskih naprav in zvočno opozarjanje. V večini primerov se uporablja obe vrsti varovanja, mehansko in elektronsko, saj si s tem zagotovimo višjo stopnjo varnosti objekta.

2.2 Varnostni sistem

Vsi elektronski varnostni sistemi so sestavljeni iz centralne enote (Slika 2.1) in ostalih varnostnih naprav [1]. Centralna enota skrbi za nadzor signalov, ki jih sporočajo ostale naprave, namenjene zaznavanju sprememb v prostoru in komuniciranju z okolico. Centralna enota za svoje delovanje potrebuje omrežno napajanje, v primeru izpada pa se napaja s pomočjo transformatorske postaje in akumulatorja.

Prav tako vsaka centrala vključuje komunikator, ki omogoča povezavo z uporabnikom ali varnostno službo v nekaj sekundah. Parametri komunikacije so nastavljeni glede na želje uporabnika. Centrala ima običajno tudi možnost shranjevanja zgodovine dogodkov, ki jih lahko uporabnik pregleduje s pomočjo tipkovnice in prikazovalnika.

Slika 2.1: Centralna enota s transformatorjem in akumulatorjem

2.2.1 Spomin

Glavni pomen spomina v alarmni centrali je hranjenje dogodkov iz preteklosti, ki lahko pomagajo definirati potek dogodkov, s pomočjo katerega uporabnik in pooblaščene osebe skušajo raziskati primer neljubega dogodka. Večji alarmni sistemi omogočajo vpogled v preteklost, tudi za 3000 preteklih dogodkov.

2.2.2 Funkcijske tipke

Funkcijske tipke omogočajo lažjo interakcijo med uporabnikom in varnostnim sistemom, lahko služijo za pregled dogodkov ali pa za vpis gesel pri izklapljanju in vklapljanju varnostnega sistema.

2.2.3 Geslo

Geslo omogoča vklop ali izklop alarmne naprave. S pomočjo gesel lahko nastavimo različne profile uporabnikov, ki jim dovolimo gibanje samo po omejenem prostoru. Tako lahko z enim geslom vklopi alarmno napravo samo za določen del objekta, tako lahko uporabniku omejimo gibanje v prostoru. Daljše je geslo, višja je stopnja varnosti, saj ga je posledično težje zlorabiti. Zaželeno je, da imamo poleg gesel tudi alternativno rešitev, saj obstaja možnost, da uporabnik pozabi svoje geslo.

2.2.4 Avtomatski vklop

Ljudje smo željni udobja, zato je možnost časovnega nastavljanja zelo dobrodošla funkcija varnostnega sistema. Tako lahko uporabnik nastavi točen čas, kdaj naj se varnostni sistem vključi oz. prične delovati.

2.2.5 Senzor

Senzorji so elementi, ki jim drugače pravimo tudi tipala, saj na svoj način »tipajo« spremembe v prostoru. Za različne spremembe se uporabljajo različni tipi senzorjev. Vsi senzorji so povezani s centralno enoto, ki sprejema njihove signale in jih ustrezno obravnava.

Glede na vrsto senzorja, obstajajo tudi specifične lastnosti oz. omejitve le-tega. Tako ima senzor gibanja omejene možnosti namestitve, saj je zelo pomembno, da pokriva zadosten del prostora in da ni nameščen v bližini naprave, ki oddaja elektromagnetno valovanje, ki bi ga pri samem delovanju lahko zmotilo. Tovrstni senzorji so občutljivi tudi na odbojne površine, ki bi lahko preusmerile signal, pri tem pa je pomembna tudi višina namestitve, običajno med 180cm in 230cm. Težava, ki bi lahko ogrozila delovanje senzorja za gibanje je tudi povišana vlažnost v prostoru.

Za detekcijo sprememb poznamo več vrst senzorjev, ki se med seboj razlikujejo glede na tehnologijo zaznavanja.

2.2.5.1 Infrardeči senzor gibanja

IR senzorji (Slika 2.2) so običajno uporabljeni za varovanje prostorov. Omogočajo enostavno vgradnjo in veliko polje pokrivanja. Že samo ime pove, da senzorji uporabljajo tehnologijo infrardeče svetlobe [7]. V primeru zaznanega gibanja objekta, senzor pošlje ustrezen signal centralni enoti. IR senzorje ločimo na notranje in zunanje, ki so sestavljeni iz različnih materialov. Poznamo stenske senzorje, ki imajo običajen kot pokrivanja okoli 180° in stropne, ki pokrivajo kot 360°. Poleg tega ločimo tudi senzorje, ki razlikujejo gibanje človeka od gibanja preostalih bitij, s tem se zmanjša možnost lažnih alarmov.

Slika 2.2: IR senzor gibanja

2.2.5.2 Požarni senzorji

Senzorji za detekcijo požara oz. dima (Slika 2.3) so v sedanjih časih nepogrešljiv del varnostnih sistemov. Za boljšo zaščito nepremičnine je potrebno senzor vgraditi v vsako etažo nepremičnine [7]. Najbolj priporočljiva vgradnja je v prostorih, kjer se nahajajo ogrevalne naprave oz. kurišča. Z vgradnjo požarnega senzorja pridobimo tudi popust pri zavarovanju nepremičnine.

Slika 2.3: Požarni senzor

2.2.5.3 Senzorji loma stekla

V zadnjem času se je povečalo število vlomov v objekte s pomočjo razbijanja stekel, še posebno v objektih, ki imajo pretežno steklene površine. Zato so ponudniki varnostnih sistemov razvili senzor za detekcijo loma stekla (Slika 2.4). Senzor reagira na točno določene frekvence [7], ki se sprostijo ob lomljenju stekla. Vgradnja senzorja nima večjih posebnosti, le usmerjen mora biti proti stekleni površini. Prednost tovrstnih senzorjev je, delovanje v nočnem režimu, ko je vidnost zelo slaba.

Slika 2.4: Senzor loma stekla

2.2.5.4 Bariere

Bariere so sestavljene iz sprejemnika in oddajnika (Slika 2.5), med katerimi potujejo IR žarki [7]. Prekinitev žarkov sproži alarmni signal. Uporabljajo se lahko tako v zunanjih kot v notranjih prostorih.

Slika 2.5: Oddajnik in sprejemnik za bariere

2.2.5.5 Magnet

Za zaščito vrat in oken se uporabljajo posebni senzorji, ki za osnovo uporabljajo magnet (Slika 2.6). Takšni senzorji so vgrajeni v sistem zapiral, ki ob odpiranju okna oz. vrat ustrezno reagirajo. Velika prednost je predvsem cena, saj so na tržišču med najcenejšimi senzorji [7], enostavna pa sta tudi uporaba in vgradnja. Zaradi kompliciranе napeljave kablov, so razvili tudi brezžično različico senzorja.

Slika 2.6: Magnetni senzor

Poglavje 3 Orodja za razvoj modula

Za razvoj modula smo morali najprej izbrati ustrezno platformo in razvojno okolje, odločili smo se za platformo Arduino. Začetki Arduina segajo v leto 2005, ko so ga predstavili na srečanju »Interaction Design Institute Ivrea« v Italiji. K razvoju Arduina so veliko pripomogli kar študenti sami.

Trenutno se na trgu pojavlja več različic plošč Arduino:

- Arduino Diecimala
- Arduino Duemilanove
- Arduino UNO
- Arduino Leonardo
- Arduino Mega
- Arduino Nano
- Arduino DUE

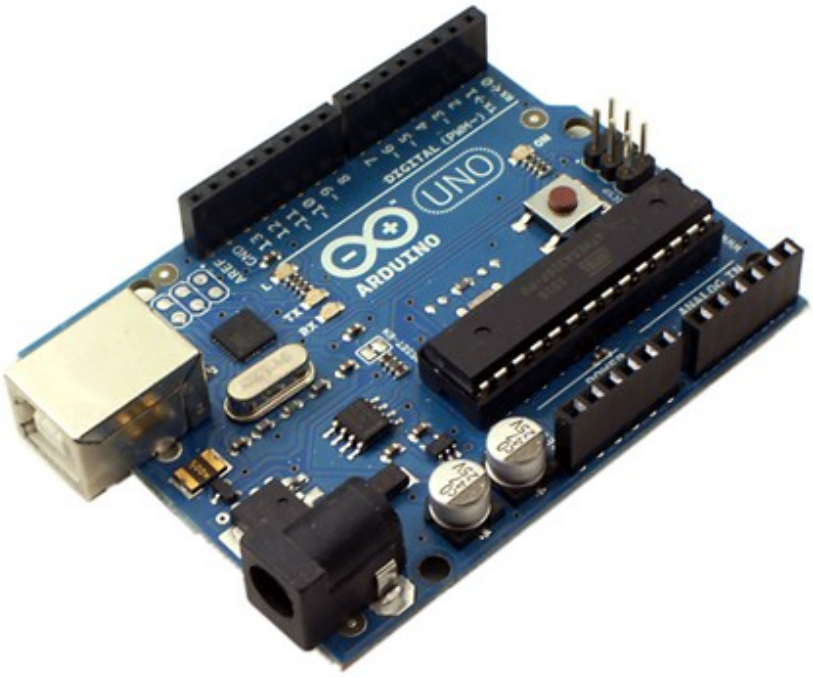
3.1 Arduino UNO

Na podlagi seznama potreb in ciljev, ki sem si jih zadal v diplomski nalogi, sem izbral ploščo Arduino UNO (Slika 3.1), ki bo služila kot osnova za razvoj modula za varovanje objektov. Zaradi manjšega stroška nakupa plošče, sem na koncu izbral kitajsko različico, imenovano Funduino UNO, ki pa je prav tako kompatibilna z razvojnim okoljem Arduino. V osnovi je praktično identična plošči Arduino UNO, saj uporablja enak mikrokrmilnik in ima enak nabor funkcionalnosti, le cena je primernejša. Ploščo sem kupil skupaj v paketu z nekaterimi uporabnimi vzhodno-izhodnimi elementi, dokupil pa sem še vse potrebne zunanje module, ki sem jih potreboval za detekcijo sprememb oz. komunikacijo med okolico in modulom.

Plošča Arduino UNO [2] temelji na 8-bitnem mikrokrmilniku Atmel. Napajanje plošče je omogočeno preko USB priklopa ali zunanjega napajanja. Zunanje napajanje lahko vzpostavimo s pomočjo adapterja ali baterije.

Platforma Arduino je sicer odprto kodna rešitev, ki je zasnovana na enostavni razvojni plošči z vzhodno-izhodnimi pini in uporabniku prijaznem razvojnem okolju. Programska oprema je

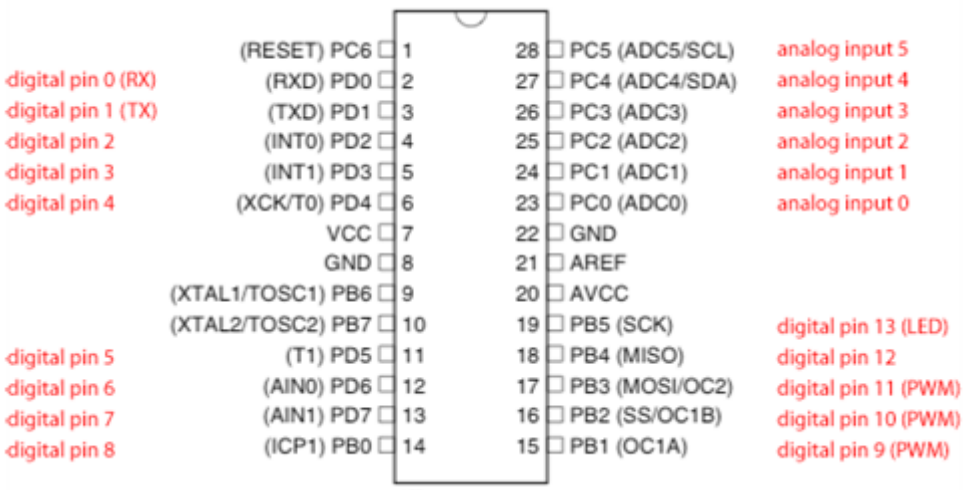
povsem brezplačna, zato je priljubljenost toliko večja. Problemi, na katere naletiš med samim razvojem, so skoraj zagotovo opisani že na uradnem forumu, preko katerega poteka komunikacija med razvijalci.



Slika 3.1: Razvojna plošča Arduino UNO

Specifikacije razvojne plošče Arduino UNO [2]:

- Mikrokrmilnik: *ATmega328* (Slika 3.2)
- Delovna napetost: *5V*
- Vhodna napetost (priporočena): *7-12V*
- Vhodna napetost (omejena): *6-20V*
- Digitalni vhodne/izhodne nožice : *14*
- Analogni vhodi: *6*
- Tok na vhodno/izhodnih nožicah : *40mA*
- Tok na nožicah z napetostjo 3.3V: *50mA*
- Pomnilnik FLASH: *32kB*
- Pomnilnik SRAM: *2kB*
- Pomnilnik EEPROM: *1kB*
- Frekvenca delovanja: *16MHz*



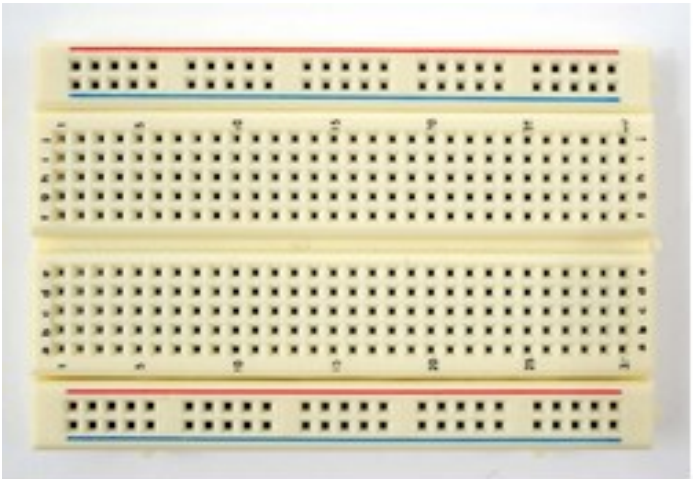
Slika 3.2: Shema mikrokrmilnika ATmega328

3.2 Ostala orodja

Za samo izdelavo varnostnega sistema, poleg mikrokrmilnika potrebujemo tudi druge komponente. Nekatere izmed njih so razvijalcu v veliko pomoč, druge imajo točno določene vloge.

3.2.1 Protoboard

Plošča za povezovanje pinov (Slika 3.3) je osnovni del za izdelavo prototipov. S svojo mrežo povezav je razvijalcu v veliko pomoč. Običajno se uporabljajo za gradnje vezij brez spajkanja. Tako lahko elemente ponovno uporabimo v prihajajočih projektih. V našem primeru je bila plošča za povezovanje že vključena v paketu z razvojno ploščo.



Slika 3.3: Protoboard

3.2.2 Žice

Za povezovanje komponent s ploščo potrebujemo tudi žice (Slika 3.4), ki so za lažjo uporabo največkrat že vnaprej pripravljene in tako, kot v našem primeru priložene v paketu.

Slika 3.4: Žice za povezovanje elementov

3.2.3 Zvočnik

Za zvočno opozarjanje smo se odločili, da uporabimo priložen zvočnik (Slika 3.5), ki je zmožen oddajati zvočne signale na podlagi različnih frekvenc. Tako bomo lahko oddajali človeku neprijeten zvočni signal, ki ga bo zaznalo vsako zdravo uho.

Slika 3.5: Zvočnik (beeper)

3.2.4 Senzor gibanja

Za detekcijo gibanja v prostoru bomo uporabili senzor gibanja ali senzor PIR (Slika 3.6), ki deluje s pomočjo infrardeče svetlobe. Za nemoteno delovanje je potrebno povezati tri nožice:

- GND – ozemljitev
- OUT – izhod (komunikacija z mikrokrmilnikom)
- VCC – napetost (5V)

Slika 3.6: IR senzor gibanja

3.2.5 LED dioda

LED diode (Slika 3.7) bomo uporabili za nazornejši prikaz statusa alarma. Krajšo nožico je vedno potrebno povezati z ozemljitvijo (GND). V našem primeru bomo uporabili le dve diodi, in sicer rdečo in zeleno.

Slika 3.7: LED dioda

3.2.6 LCD zaslon

Za prikaz informacij uporabniku se običajno uporabljajo zasloni, na katere lahko izpisujemo pomembne informacije. V našem primeru smo uporabili LCD zaslon (Slika 3.8), ki je bil vključen v paketu z razvojno ploščo Arduino UNO. LCD zaslon uporablja tehnologijo tekočih kristalov. Zaslon ima za komunikacijo z mikrokrmilnikom pripravljenih kar 16 pinov (Tabela 3.1). Naša izvedba omogoča prikaz teksta v dveh vrsticah po 16 znakov. Vsako polje sestavlja 40 točk (5x8).

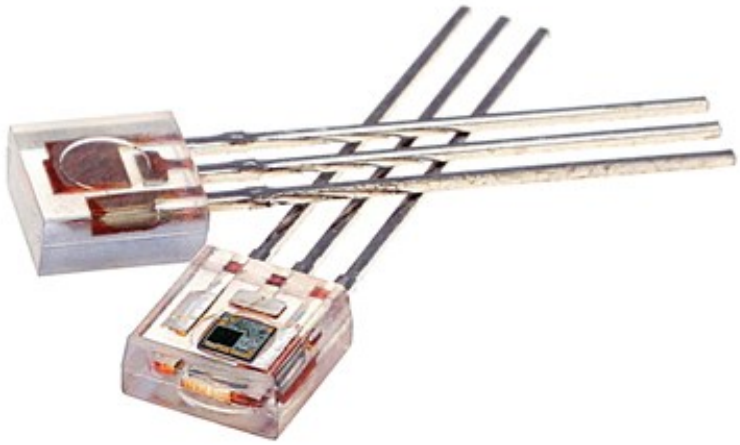
Slika 3.8: LCD zaslon

PIN	OPIS
VSS	Negativna napetost
VDD	Pozitivna napetost (5V)
VO	Kontrast
RS	Register
RW	Branje / pisanje
E	Omogočiti
D0	Prenos podatkov
D1	Prenos podatkov
D2	Prenos podatkov
D3	Prenos podatkov
D4	Prenos podatkov
D5	Prenos podatkov
D5	Prenos podatkov
D7	Prenos podatkov
A	Negativni priključek osvetlitve
K	Pozitivni priključek osvetlitve

Tabela 3.1: Opis nožic LCD zaslona

3.2.7 IR sprejemnik

Za upravljanje varnostnega sistema bomo uporabili tudi daljinski upravljalnik, ki bo komuniciral s pomočjo IR sprejemnika (Slika 3.9), nameščenega v modulu. Daljinski upravljalnik oddaja infrardeče signale, sprejemnik pa skuša signale prestreči in dekodirane poslati mikrokrmilniku, kjer jim bomo programsko določili pomen.



Slika 3.9: IR sprejemnik

3.2.8 Daljinski upravljalnik

S pomočjo priloženega daljinskega upravljalnika bomo realizirali izklop posameznega alarma. Za posamezen alarm bomo izbrali svojo tipko, in sicer na naslednji način:

- 1 - izklopi alarm, aktiviran s strani laserske bariere
- 2 - izklopi alarm, aktiviran s strani senzorja za gibanje (PIR)
- 3 - izklopi alarm, aktiviran s strani senzorja za dim (MQ135)

Za detekcijo posamezne tipke smo morali najprej poiskati kode, ki se pošiljajo ob sprejemu IR signala le-te. Te kode bomo kasneje v programu uporabili za določitev pomena tipke. Na podlagi testiranja smo prišli do naslednjih rezultatov:

- Tipka s številko '1' : *0x9716BE3F*
- Tipka s številko '2': *0x3D9AE3F7*
- Tipka s številko '3': *0x6182021B*

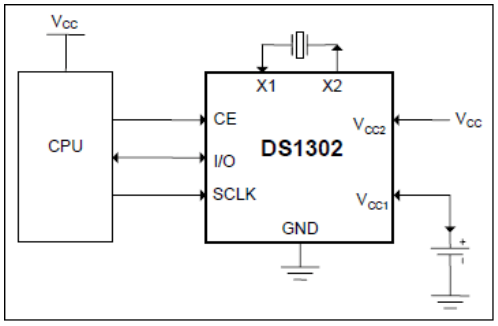
3.2.9 Senzor MQ135

Za detekcijo požara bomo uporabili modul MQ135 (Slika 3.10), ki smo ga dokupili naknadno. Senzor s pomočjo zaznavanja različnih plinov sporoča signal z različnimi vrednostmi. Na podlagi prebrane vrednosti lahko po specifikaciji določimo točno določen plin, ki je trenutno prisoten v bližini senzorja. V našem primeru se bomo osredotočili na plin CO_2, ki pomeni, da je v prostoru najverjetneje prišlo do požara.

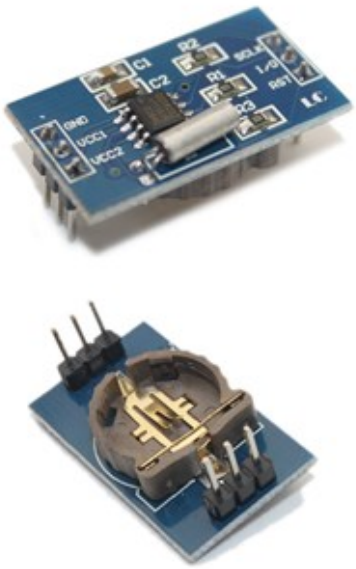
Slika 3.10: Senzor za detekcijo plinov

3.2.10 Časovni modul RTC DS1302

Za potrebe prikaza časa smo dokupili časovni modul RTC DS1302 (Slika 3.12), ki s pomočjo točno določene frekvence oddaja signal za štetje realnega časa. Modul je sestavljen iz časovnika in statičnega RAM-a velikosti 31 bajtov. Komunikacija z mikrokrmilnikom poteka preko serijskega vmesnika. Za lažjo uporabo pri razvoju modula smo uporabili eno izmed že obstoječih knjižnic. Modul ima podpora tako za 24-urni, kot tudi za 12-urni format. Za komunikacijo mikrokrmilnika z uro potrebujemo tudi baterijsko napajanje, ki je vidno na spodnji shemi (Slika 3.11).



Slika 3.11: Shema časovnega modula RTC DS1302



Slika 3.12: Časovni modul RTC DS1302

3.2.11 Drugi elementi

Za spreminjanje kontrasta na LCD zaslonu smo uporabili priložen potenciometer. Prav tako smo vgradili tudi fotocelico, ki zaznava spremembo svetlobe in pošilja vrednosti mikrokrmilniku, ki nato preveri mejni prag in ob ustreznih vrednostih programsko aktivira alarm. Za izključitev vseh alarmov hkrati lahko uporabnik pritisne na gumb, ki pošlje signal mikrokrmilniku, ta pa programsko izključi vse alarme. Za pravilno povezovanje med moduli smo morali uporabiti izbrane elektrotehnične elemente, med drugimi tudi upore.

3.3 Programska oprema

Za razvoj modula smo potrebovali različno programsko opremo. Tako smo za programiranje mikrokrmilnika uporabljali razvojno okolje Arduino IDE in za načrtovanje sheme vezja ter električnega načrta programsko opremo EAGLE.

3.3.1 Arduino IDE

Razvojno okolje za pisanje programov za plošče Arduino se imenuje Arduino IDE (Slika 1.4), ki je napisan v jeziku Java. Zasnovan je tako, da ga lahko uporabljajo tako začetniki, kot tudi programerji z nekaj več kilometrine. Okolje vsebuje prevajalnik za prevajanje kode in nalagalnik za enostavno nalaganje programa na ploščo.

Sama uporaba okolja Arduino IDE je izredno enostavna, saj nima nobenih kompliciranih menujev, niti postopkov, ki bi bili odvratni za uporabnika. Tako pisanje programov postane resnično zabavno tudi na nivoju ljubiteljskega programiranja. Za pisanje kode je potrebno le nekaj znanja o jeziku C oz. C++, saj programiranje v okolju Arduino IDE temelji na omenjenih jezikih.

Paket programske opreme za Arduino vsebuje tudi prednaložene knjižnice, ki jih lahko kadarkoli uporabimo v svojem projektu. Za povezavo z Arduino ploščo je potrebno nastaviti tudi ustrezna vrata serijskega vmesnika, v našem primeru je bil to COM8. Programi, napisani v okolju Arduino IDE temeljijo na osnovi dveh funkcij:

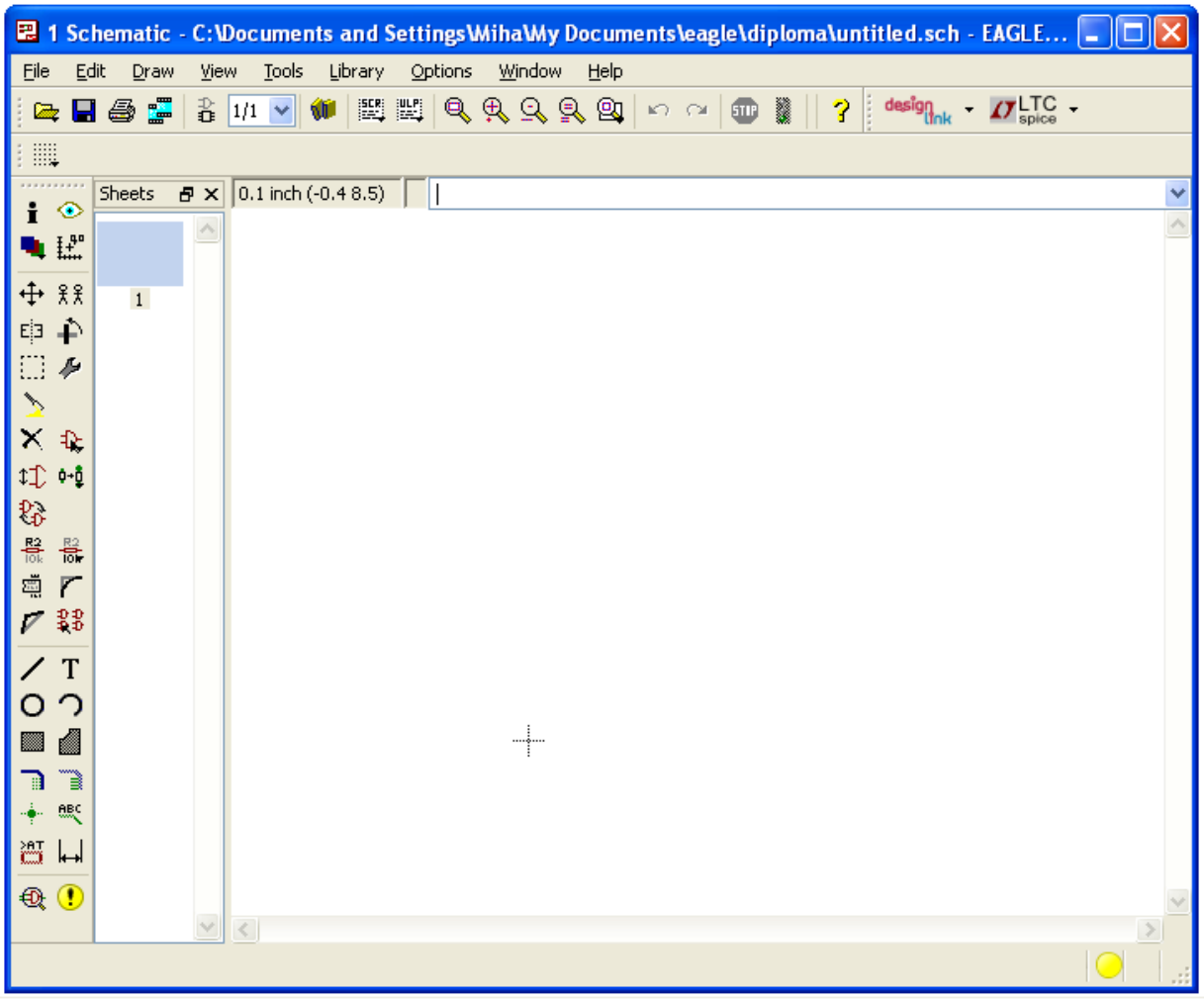
- *void setup()* – funkcija se kliče le enkrat, in sicer pred prvim klicem funkcije *loop()*. Na tem mestu je priporočljivo nastaviti vse parametre, ki so potrebni za komunikacijo med moduli in mikrokrmilnikom.
- *void loop()* – funkcija se izvaja v neskončni zanki. Zadnji klic funkcije je določen s trenutkom izklopa iz napajanja oz. s pritiskom na gumb 'reset'.

3.3.2 EAGLE

Programsko orodje EAGLE (*Easily Applicable Graphical Layout Editor*) je namenjeno risanju shem oz. načrtovanju tiskanih vezij. Z njim lahko načrtujemo tiskana vezja, ki ustrezajo vsem tehnološkim standardom, zato je orodje priljubljeno tudi pri profesionalnih razvijalcih.

Program vključuje tudi knjižnice, s katerimi je moč najti specifične elemente, ki jih potrebujemo za izdelavo vezja. Uporabnik ima možnost nadgradnje knjižnice, in tako lahko oblikuje svoje elemente, ki jih želi uporabiti v svojem načrtu. Tako orodje pridobiva na dinamičnosti in širokem spektru uporabnikov.

Orodje daje uporabniku možnost, da iz pripravljene električne sheme, kjer lahko izbira tudi ohišja elementov, z enim klikom generira shemo tiskanega vezja. Pri tem nam orodje omogoča, da električno shemo izvozimo kot sliko, ki jo kasneje lahko uporabimo pri pisanju dokumentacije. Po zaključku načrtovanja električne sheme nam orodje avtomatsko generira listo elementov, ki smo jih uporabili v sami shemi. V listi se nahajajo tudi posamezne povezave do elementov in njihovi opisi. V primeru, da ne želimo avtomatsko generirane sheme za tiskano vezje, moramo v naslednjem koraku povezati elemente v tiskano vezje. To shemo lahko prav tako izvozimo kot sliko. Torej s programsko opremo EAGLE (Slika 3.13) lahko pripravimo tako električno shemo, kot tudi shemo za tiskano vezje.

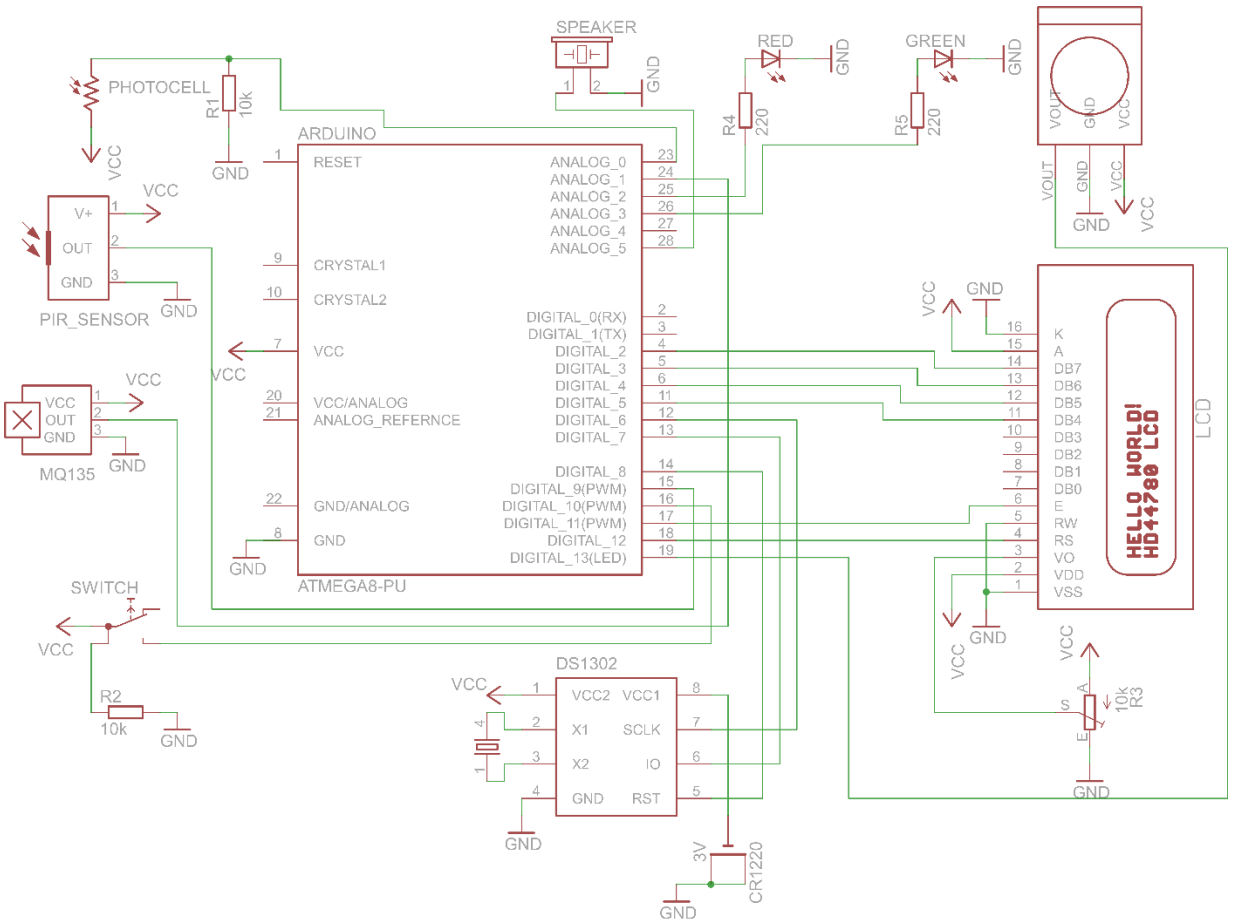


Slika 3.13: Programsko okolje EAGLE (električno vezje)

Poglavje 4 Razvoj modula

Razvoj modula je potekal v več iteracijah z naslednjimi koraki:

1. Risanje sheme vezja (Slika 4.2)
2. Realizacija vezja (strojni del modula)
3. Priprava programa za delovanje modula (programski del modula)
4. Testiranje modula
5. Nadgradnja modula



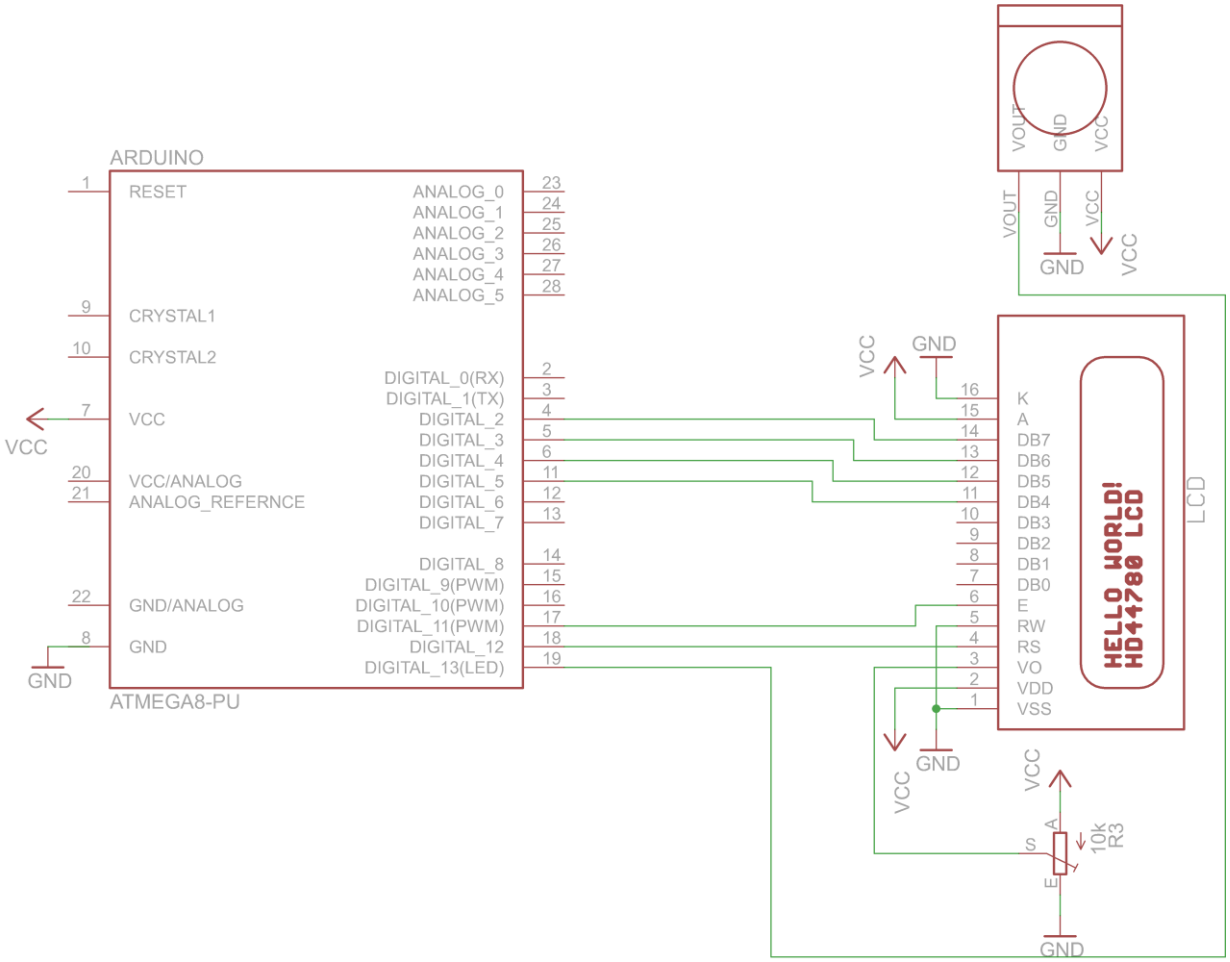
Slika 4.1: Končna shema vezja

4.1 LCD zaslon in IR sprejemnik

Najprej smo želeli povezati LCD zaslon in IR sprejemnik s ploščo Arduino UNO. S tem smo pridobili možnost prikaza informacij in upravljanje z daljinskim upravljalnikom.

Elementi, ki smo jih potrebovali za povezovanje:

- IR sprejemnik
- Arduino UNO
- LCD zaslon
- potenciometer 10k
- žice



Slika 4.2: Shema vezja (LCD zaslon in IR sprejemnik)

Nato smo napisali program, s katerim smo določili odzivne dogodke na določene signale. Komunikacija med daljinskim upravljalnikom in IR sprejemnikom poteka na osnovi kodiranih signalov, vsaka tipka na daljinskem upravljalniku ima namreč svojo kodo. Za naš projekt smo si izbrali prve tri tipke številčnice, in sicer '1', '2' in '3', za katere smo pridobili kode signalov in jih definirali kot konstante.

```
#define RECEIVER_CODE_1 0x9716BE3F
#define RECEIVER_CODE_2 0x3D9AE3F7
#define RECEIVER_CODE_3 0x6182021B
```

Za nemoteno delovanje LCD zaslona in IR senzorja sta potrebni tudi specifični knjižnici, ki ju najdemo v paketu knjižnic, vključenim v programskem orodju Arduino IDE.

```
#include <LiquidCrystal.h>
#include <IRremote.h>
```

Za vsakega izmed uporabljenih elementov potrebujemo spremenljivko, ki nam kasneje služi za lažjo komunikacijo med mikrokrmilnikom in elementom, v našem primeru LCD zaslonom. Običajno moramo pri inicializaciji spremenljivke podati ustrezne parametre, ki opisujejo način priklopa in pine, preko katerih poteka komunikacija z mikrokrmilnikom.

```
const int pin_RECEIVER = 13;
LiquidCrystal pin_LCD(12, 11, 5, 4, 3, 2);

IRrecv irrecv(pin_RECEIVER);
```

V glavni funkciji *setup()* omogočimo delovanje IR sprejemnika, pri LCD zaslonu pa podamo število vrstic in stolpcev, ki so na voljo. Za komunikacijo računalnika z razvojno ploščo smo uporabili serijski vmesnik, za katerega smo določili hitrost 9600 bitov na sekundo.

```
void setup()
{
  irrecv.enableIRIn();
  pin_LCD.begin(16, 2);
  Serial.begin(9600);
}
```

V ponavljajoči funkciji *loop()* določimo potek programa, in sicer vrstni red operacij. Tako smo pripravili svojo funkcijo za preverjanje poslanih signalov na IR sprejemnik, in ločeno funkcijo za izpisovanje informacij na LCD zaslon. Omenjeni funkciji kličemo v neskončni funkciji, s tem pa zagotovimo, da bomo na IR sprejemniku skoraj zagotovo prestregli vse signale, na LCD zaslonu pa zagotovili prikaz aktualnih informacij.

```
void loop()
{
  checkReceiver();
  textLCD();
}
```

Funkcija *checkReceiver()* ima nalogo, da preverja IR sprejemnik, če je morda zaznal za nas zanimiv signal iz daljinskega upravljalnika. V primeru, da prejeta koda ustreza eni izmed kod, definiranih na začetku, mu programsko določimo funkcijo, v našem primeru izklop določenega tipa alarma. Preverjanje prejete kode smo realizirali s pomočjo stavka za vejitve. Za izklop alarma kličemo funkcijo *resetAlarm()*, ki onemogoči alarm.

```
void checkReceiver()
{
  decode_results results;
  if (irrecv.decode(&results)) {
    Serial.println(results.value, HEX);
    switch(results.value)
    {
      //button '1'
      case RECEIVER_CODE_1:
        resetAlarm(ALARM_TYPE_LASER);
        break;
      //button '2'
      case RECEIVER_CODE_2:
        resetAlarm(ALARM_TYPE_MOVING);
        break;
      //button '3'
      case RECEIVER_CODE_3:
        resetAlarm(ALARM_TYPE_GAS);
        break;
      default:
        //nothing to do
        break;
    }
    irrecv.resume(); // Receive the next value
  }
}
```

Statuse za posamezen tip alarma hranimo s pomočjo bitnih zastavic, kar pomeni, da nastavljamo posamezen bit v binarnem številu na '0' ali '1'. V našem primeru za to skrbi funkcija *switchAlarmStatus()*, ki spremeni zastavico glede na parameter *newStatus*.

```
void resetAlarm(int alarmType)
{
  switchAlarmStatus(alarmType, false);
}
```

Za nastavljanje bitnih zastavic smo uporabili vgrajeni funkciji *bitSet()* in *bitClear()*, s katerima lahko spreminjamo vrednosti bitov v spremenljivki *alarmStatus*. Za izbiro pravega bita si pomagamo s konstantami, ki smo jih določili za posamezen alarm.

```
void switchAlarmStatus(int alarmType, boolean newStatus)
{
  if (newStatus) {
    bitSet(alarmStatus, alarmType);
  } else {
    bitClear(alarmStatus, alarmType);
  }
}
```

Funkcija *textLCD()* je namenjena izpisovanju informacij na zaslon. Na zaslonu prikazujemo dve vrsti informacij, in sicer v času, ko ni posebnih dogodkov prikazujemo čas z datumom, v trenutku alarma pa so izpisani tipi alarmov (Slika 4.3), ki so trenutno aktivirani. Tako funkcija glede na stanje alarmov lahko pokliče funkcijo *writeTextToLCD()*, kjer kot parameter pošlje trenutno aktivirane alarme, združene v nizu znakov. Za izpis ure z datumom skrbi funkcija *writeClockToLCD()*, ki jo bomo opisali kasneje (4.3).

```
void textLCD()
{
  if (alarmStatus != 0) {
    writeTextToLCD(getAlarmTypeString() +"!");
  } else {
    writeClockToLCD();

  }
}

void writeTextToLCD(String text)
{
  pin_LCD.setCursor(0,0);
  pin_LCD.clear();
  pin_LCD.print(" ALARM : ");
  pin_LCD.setCursor(0,1);
  pin_LCD.print(text);
}
```

Slika 4.3: Rezultat izpisa aktivnih alarmov

Za pregled aktiviranih alarmov skrbi funkcija *getAlarmTypeString()*, ki nazive trenutno aktiviranih alarmov združi v niz znakov. Nazive alarmov hranimo v seznamu *alarmNames*, kjer so nazivi razvrščeni glede na konstante, ki jih bomo določili v naslednjem koraku (4.2). Vsak alarm bo namreč moral imeti svojo številčno konstanto. Za preverjanje statusa alarma uporabljamo funkcijo *bitRead()*, ki deluje na podoben način kot *bitSet()* ali *bitClear()*.

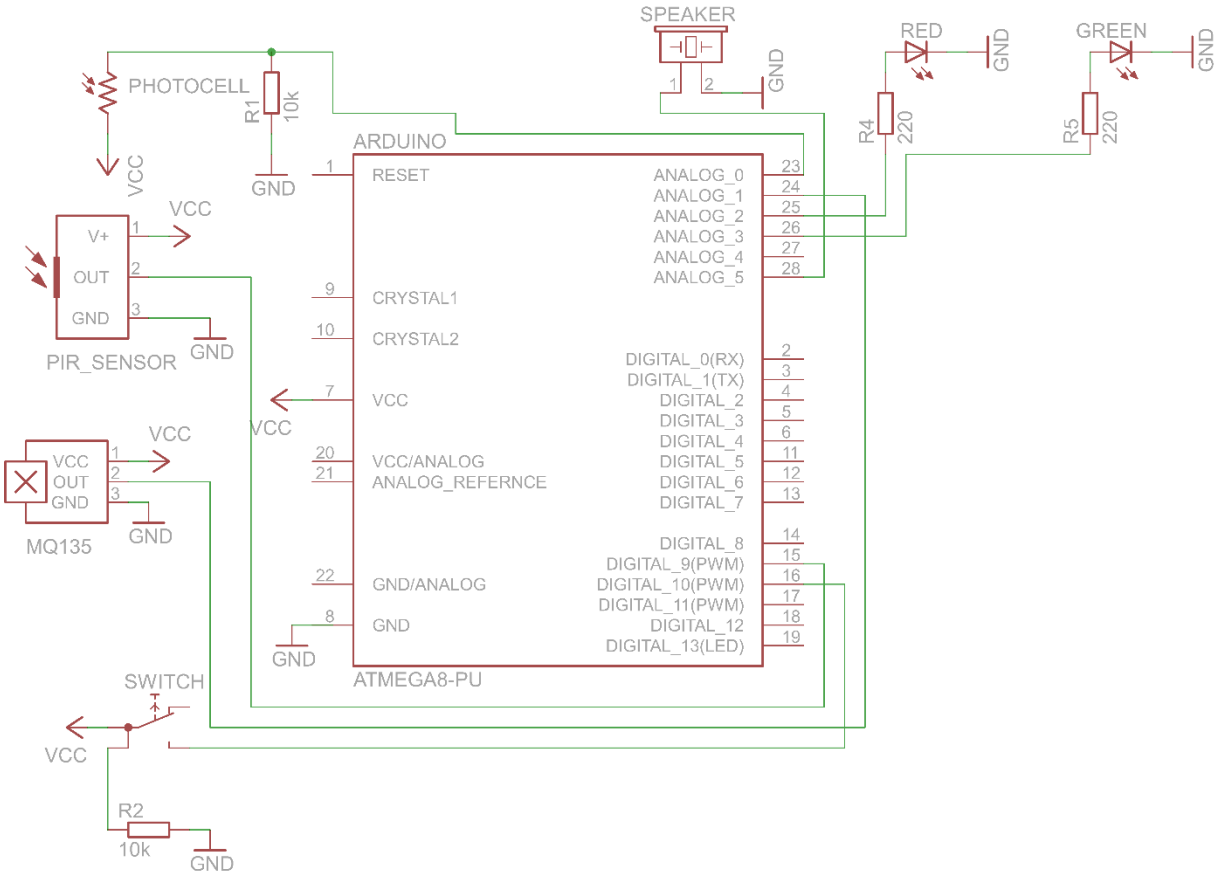
```
String getAlarmTypeString()
{
  String resultStr = "";
  if (bitRead(alarmStatus, ALARM_TYPE_LASER) == 1)
  {
    resultStr += " " + alarmNames[ALARM_TYPE_LASER];
  }
  if (bitRead(alarmStatus, ALARM_TYPE_MOVING) == 1)
  {
    resultStr += " " + alarmNames[ALARM_TYPE_MOVING];
  }
  if (bitRead(alarmStatus, ALARM_TYPE_GAS) == 1)
  {
    resultStr += " " + alarmNames[ALARM_TYPE_GAS];
  }
  return resultStr;
}
```

4.2 Senzor gibanja, fotocelica in senzor plinov

V nadaljevanju razvoja smo se odločili, da trenutne zmožnosti modula nadgradimo s senzorjem gibanja, fotocelico in senzorjem plinov. Tako bo modul postal odziven na gibe, ki se bodo zgodili v njegovi okolici, s pomočjo fotocelice bomo postavili lasersko bariero, ki bo ob prekinitvi laserskega žarka aktivirala alarm. S senzorjem plinov pa bomo skušali realizirati požarni senzor, ki bo aktiviral alarm na podlagi zaznanih plinov.

Elementi, ki smo jih vgradili v modul:

- senzor plinov (MQ135)
- senzor gibanja (PIR)
- fotocelica
- 2x upor 10k
- 2x upor 220
- LED diodi (rdeča in zelena)
- zvočnik (beeper)
- stikalo (gumb)
- žice



Slika 4.4: Shema vezja (senzor PIR, fotocelica in senzor MQ135)

Za vgrajene elemente smo napisali programsko kodo, s katero smo jim dodelili vloge in naloge, ter napisali primerne odzive na možne dogodke. Glede na to, da smo vgradili kar tri senzorje, smo morali določiti tudi konstante s katerimi bomo kasneje lahko ločevali med posameznimi tipi alarmov.

```
#define ALARM_TYPE_LASER    1
#define ALARM_TYPE_MOVING   2
#define ALARM_TYPE_GAS      3
```

Za vsak element smo določili konstanto, v kateri bomo hranili oznako pina, preko katerega bo potekala komunikacija dotičnega senzorja z mikrokrmilnikom. Vsi pini, katerih oznaka se prične s črko 'A', so analogni, preostali so digitalni. Poleg konstant smo določili še nekaj spremenljivk, ki nam bodo v pomoč za hranjenje zadnjih stanj. Tako smo spremenljivko *lastButtonState* določili za hranjenje zadnjega stanja gumba, ki omogoča izklop vseh alarmov hkrati. Spremenljivko za hranjenje seznama nazivov alarmov, ki smo jo omenili že pri izpisu aktiviranih alarmov na zaslon (4.1), smo poimenovali *alarmNames*. Prav tako smo potrebovali

ločeno spremenljivko, v kateri smo s pomočjo bitnih zastavic hranili statuse posameznih alarmov, poimenovali smo jo *alarmStatus*.

```
const int pin_SOUND = A5;
const int pin_LED_RED = A2;
const int pin_LED_GREEN = A3;
const int pin_GAS = A1;
const int pin_PHOTO = A0;
const int pin_PIR = 9;
const int pin_BUTTON = 10;

int lastButtonState = LOW;

String alarmNames[] = {"UNKNOWN", "LASER", "MOVING", "GAS"};

byte alarmStatus = 0;
```

V začetni funkciji *setup()* smo morali tokrat nastaviti način uporabe pinov, in sicer glede na smer pretoka podatkov, nekatere smo nastavili kot izhode, druge kot vhode, gledano s strani mikrokrmilnika.

```
void setup()
{
  pinMode(pin_LED_RED, OUTPUT);
  pinMode(pin_LED_GREEN, OUTPUT);
  pinMode(pin_SOUND, OUTPUT);
  pinMode(pin_GAS, INPUT);
  pinMode(pin_PIR, INPUT);
  pinMode(pin_BUTTON, INPUT);
}
```

V glavni funkciji *loop()* smo tokrat poklicali še nekaj dodatnih funkcij, in sicer smo za vsak dodan senzor napisali funkcijo ki preveri informacije na njem in odloči, ali se bo določen alarm aktiviral. Tako npr. *checkMovingAlarm()* preveri, če je bil morda na senzorju zaznan gib, ki bi pomenil aktiviranje alarma za gibanje. Enako velja tudi za funkciji *checkLaserAlarm()* in *checkGasAlarm()*. Pred samim preverjanjem statusov na senzorjih, moramo opraviti tudi naslednje operacije, kot je preverjanje pozicije gumba, če je bil medtem časom morda pritisnjen, zato smo napisali funkcijo *checkResetButton()*. Za proženje alarma z zvočnim signalom smo napisali funkcijo *beep()*, za prižiganje in ugašanje LED diod pa funkcijo *LED()*.

```
void loop()
{

  checkResetButton();
  beep();
  LED();

  checkLaserAlarm();
  checkMovingAlarm();
  checkGasAlarm();
}
```

Za preverjanje pozicije gumba smo napisali funkcijo *checkResetButton()*, ki prebere trenutno stanje gumba in ga primerja s prejšnjim, v primeru, da je gumb trenutno v položaju '*high*' in je bil pred tem v položaju '*low*' pomeni, da je bil ravnokar pritisnjen. Tako so izpolnjeni vsi pogoji za izklop vseh alarmov, ki so bili trenutno v aktivnem stanju. Za to poskrbi funkcija *resetAllAlarms()*.

```
void checkResetButton()
{
  int buttonState = digitalRead(pin_BUTTON);
  if ((buttonState == HIGH) && (lastButtonState == LOW)) {
    resetAllAlarms();
  }
  lastButtonState = buttonState;
}
```

Za izklop vseh alarmov smo za vsakega izmed obstoječih tipov alarmov poklicali funkcijo *resetAlarm()*, ki ne stori nič drugega, kot samo to, da kliče funkcijo *switchAlarmStatus()*, tokrat s parametrom *'false*', kar pomeni izklop izbranega alarma.

```
void resetAllAlarms()
{
  resetAlarm(ALARM_TYPE_LASER);
  resetAlarm(ALARM_TYPE_MOVING);
  resetAlarm(ALARM_TYPE_GAS);
}


void resetAlarm(int alarmType)
{
  switchAlarmStatus(alarmType, false);
}
```

Za preverjanje gibanja v okolici modula smo uporabili digitalni vhod, kar pomeni, da sta na tem vhodu lahko samo dve stanji, in sicer logična '*0*' ali logična '*1*'. V primeru, da je na vhodu pina logična '*1*', ki pomeni visoko stanje, torej je pogoj za zaznavanje gibanja v okolici modula izpolnjen. Tako s funkcijo *activateAlarm()* aktiviramo alarm za gibanje.

```
void checkMovingAlarm()
{
  if (digitalRead(pin_PIR) == HIGH) {
    activateAlarm(ALARM_TYPE_MOVING);
  }
}
```

Podobno smo napravili za senzor plinov, le da je v tem primeru v uporabi analogni pin, kar pomeni, da bomo kot rezultat branja pina lahko dobili več različnih številskih stanj. Ta stanja pomenijo trenutno koncentracijo plina v okolici senzorja, ki pa je odvisna tudi od temperature zraka. Za potrebe razvoja modula smo za določitev spodnje meje nastavili vrednost na 200, kar se je po testiranjih prisotnosti CO_2 izkazalo za korektno.

```
void checkGasAlarm()
{
  if(analogRead(pin_GAS) > 200) {
    activateAlarm(ALARM_TYPE_GAS);
  }
}
```

Za postavitev bariere smo prav tako uporabili analogen pin, preko katerega nam fotocelica pošilja informacije o trenutni svetlobi. Glede na to, da ima laserski žarek značilno valovno dolžino in specifične lastnosti svetlobe, je ob prekinitvi žarka na fotocelici opazna velika razlika prebrane vrednosti na pinu, kar lahko izkoristimo za aktiviranje alarma.

```
void checkLaserAlarm()
{
  if(analogRead(pin_PHOTO) < 500) {
    activateAlarm(ALARM_TYPE_LASER);
  }
}
```

V trenutku, ko so izpolnjeni pogoji za aktivacijo alarma, se kliče funkcija *activateAlarm(),* ki ne naredi nič posebnega, kot le-to, da s pomočjo klica funkcije *switchAlarmStatus()*, spremeni status izbranega alarma. Glede na parameter *newStatus* se odloči, ali gre za brisanje bitne zastavice, ali za nastavljanje le-te.

```
void activateAlarm(int alarmType)
{
  switchAlarmStatus(alarmType, true);
}


void switchAlarmStatus(int alarmType, boolean newStatus)
{
  if (newStatus) {
    bitSet(alarmStatus, alarmType);
  } else {
    bitClear(alarmStatus, alarmType);
  }
}
```

Po dogodku aktivacije alarma, je potrebno poskrbeti tudi za oddajanje zvočnega signala. V ta namen smo pripravili funkcijo *beep()*, ki v zamiku delčka sekunde zamenja status na digitalnem izhodu iz visokega stanja v nizko. Ob naslednji iteraciji neskončne funkcije *loop()* se menjava statusa ponovi, tako da se na koncu oglašanje zvočnika izkaže kot utripajoč zvočni signal.

```
void beep()
{
  if (alarmStatus != 0)
  {
    digitalWrite(pin_SOUND, HIGH);
    delay(250);
    digitalWrite(pin_SOUND, LOW);
    delay(250);
  }
}
```

Poleg oddajanja zvočnih signalov, smo pripravili tudi svetlobni signal, ki opozarja na aktiven alarm. Princip je podoben kot pri zvočnem signalu, le da glede na statuse alarmov izberemo pravilen izhodni pin, vezan na diodo in mu nastavimo visoko stanje. Tako zelena dioda sveti v primeru neaktiviranih alarmov, rdeča pa v primeru aktiviranega alarma.

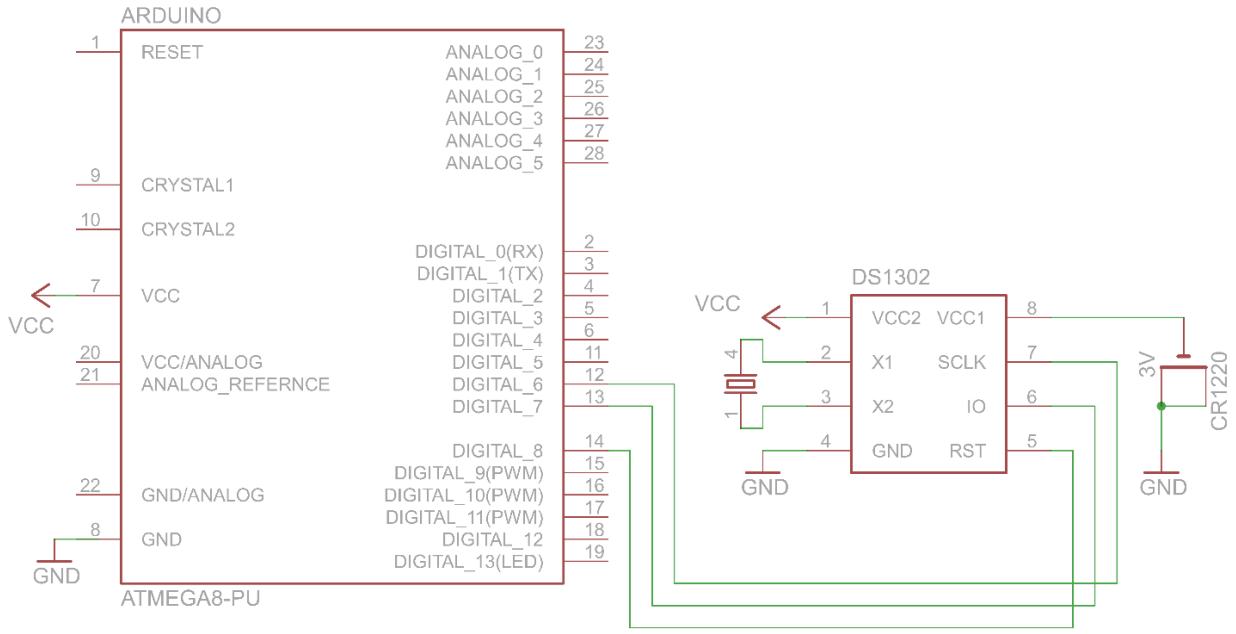
```
void LED()
{
  if (alarmStatus != 0) {
    digitalWrite(pin_LED_RED, HIGH); // pin_LED_RED on
    digitalWrite(pin_LED_GREEN, LOW); // pin_LED_GREEN off
  } else {
    digitalWrite(pin_LED_GREEN, HIGH); // pin_LED_GREEN on
    digitalWrite(pin_LED_RED, LOW); // pin_LED_RED off

  }
}
```

4.3 Časovni modul

Za merjenje časa, ki ga prikazujemo na zaslonu med neaktivnostjo alarmov, smo se odločili, da uporabimo časovni modul RTC DS1302. Modul smo s pomočjo specifikacije povezali z mikrokrmilnikom, mu vgradili baterijsko napajanje in raziskali možnosti same integracije v programu.

Elementi, ki smo jih potrebovali za integracijo časovnega modula:

- časovni modul RTC DS1302
- baterija v obliki gumba (3V)
- žice



Slika 4.5: Shema vezja (časovni modul RTC DS1302)

Za lažjo komunikacijo s časovnim modulom smo našli in uporabili knjižnico *'virtuabotixRTC'*, s pomočjo katere je uporaba modula precej bolj enostavna, saj se ni potrebno spuščati na nivo registrov [5]. Vse kar je potrebno storiti, je to, da najprej uvozimo omenjeno knjižnico, sporočimo, na katerih pinih bo potekala komunikacija s časovnim modulom in mu nastavimo začetni čas, ki bo ob zagonu osnova za nadaljevanje štetja časa. Pripravili smo tudi spremenljivko *clockBuffer*, ki jo bomo uporabili kasneje za sestavljanje niza podatkov,

namenjenega za prikaz na zaslonu. V funkciji *setup()* smo nastavili začetni čas in datum, v našem primeru je bila to nedelja, 15.6.2014 ob 10:00:00.

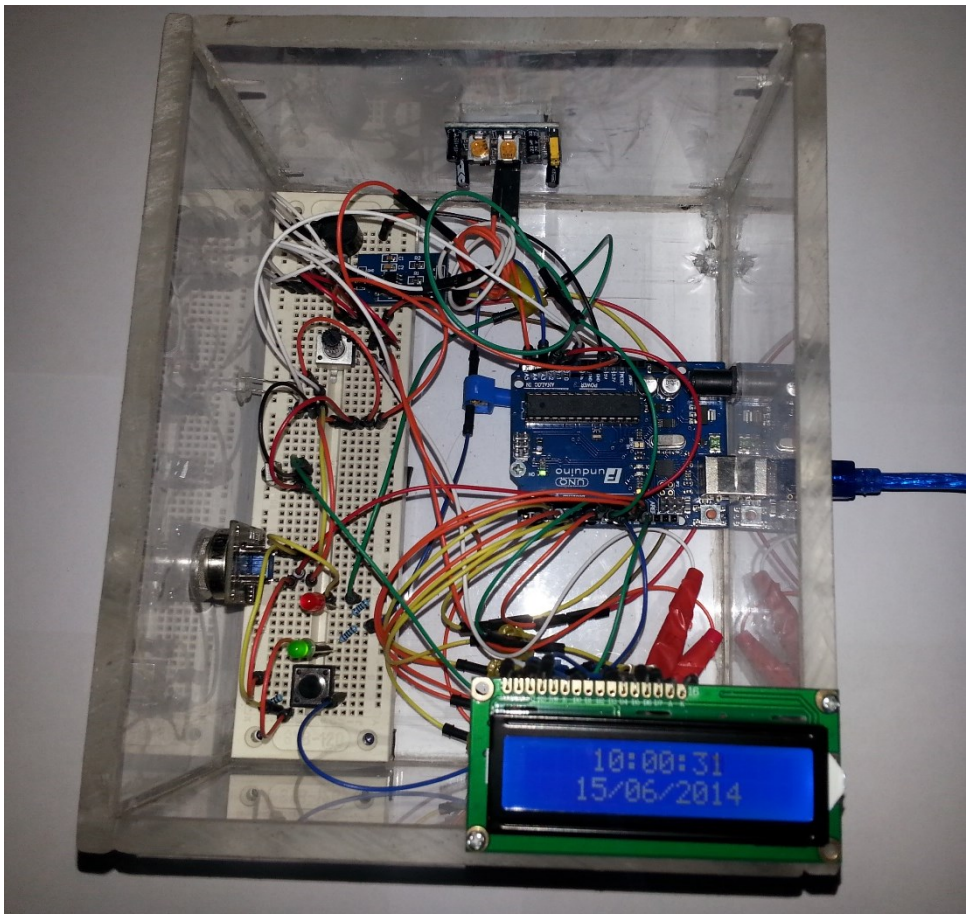
```
#include <virtuabotixRTC.h>
virtuabotixRTC pin_RTC(6, 7, 8);
char clockBuffer [50];

void setup()
{
  pin_RTC.setDS1302Time(0, 0, 10, 7, 15, 6, 2014);
}
```

Sedaj imamo vse pripravljeno za komunikacijo časovnega modula z mikrokrmilnikom, na koncu smo napisali še funkcijo *writeClockToLCD()*, s katero bomo izpisovali čas in datum na zaslon (Slika 4.6). Za osvežitev podatkov v uporabljeni knjižnici, je potrebno poklicati funkcijo *updateTime()*, ki nam zagotovi pravilne vrednosti spremenljivk v knjižnici. Za oblikovanje izpisa smo uporabili funkcijo '*sprintf*', kateri smo podali spremenljivko za začasno shranjevanje rezultata, format izpisa in trenutne vrednosti časovnega modula. V prvi vrstici zaslona smo izpisovali uro, v spodnji vrstici pa trenutni datum. Določitev pozicije izpisa smo izvedli s pomočjo funkcije *setCursor()*, ki je vmesnika za komunikacijo z zaslonom LCD.

```
void writeClockToLCD()
{
  pin_RTC.updateTime();
  sprintf(clockBuffer, "%02d:%02d:%02d", pin_RTC.hours, pin_RTC.minutes, pin_RTC.seconds);
  pin_LCD.clear();
  pin_LCD.setCursor(4,0);
  pin_LCD.print(clockBuffer);

  sprintf(clockBuffer, "%02d/%02d/%4d", pin_RTC.dayofmonth, pin_RTC.month, pin_RTC.year);
  pin_LCD.setCursor(3,1);
  pin_LCD.print(clockBuffer);

  delay(1000);
}
```

Slika 4.6: Prikaz časa in datuma na LCD zaslonu

Na koncu smo preverili delovanje modula in skušali identificirati možne nadgradnje oz. izboljšave. Modul smo razvili kot prototip in ni primeren za komercialno uporabo. Zaradi lažjega transporta, smo vezje pritrdili v škatlo iz akrilnega stekla, ki smo jo pripravili posebej v ta namen. Tako je končni izdelek videti precej bolj realen, s tem pa tudi bolj uporaben. Imeli smo tudi težavo pri realizaciji laserske bariere, kjer smo za potrebe delovanja v modul vgradili fotocelico za sprejem svetlobe, na drugi strani pa smo potrebovali oddajnik laserskega žarka. Po krajšem razmisleku smo se odločili, da lahko za oddajnik uporabimo laserski točkovnik, ki ga običajno vidimo pri predstavitvah oz. prezentacijah. Za lažjo demonstracijo smo pripravili še posebno stojalo, s pomočjo katerega smo stacionirali položaj laserskega točkovnika, in sicer tako, da smo ga pritrdili v stojalo in ga usmerili proti fotocelici. Tako bo ob prekinitvi laserskega žarka fotocelica zaznala spremembo in aktivirala ustrezen alarm. Končni izdelek, ki smo ga razvili iz radovednosti in z namenom ugotavljanja ključnih težav pri tovrstnem razvoju, je s ptičje perspektive videti tako, kot je prikazano na Slika 4.7.



Slika 4.7: Modul za varovanje objekta s pomočjo mikrokrmilnika

Poglavje 5 Zaključek

Varnost je bila in vedno bo ključnega pomena, saj zagotavlja mir, brez katerega si je težko predstavljati življenje na zemeljski polobli. Obstaja ogromno sistemov za varovanje, prav tako tudi različnih pristopov. Nekateri si lahko privoščijo celo svojega osebnega varnostnika, tukaj mislimo predvsem na zvezdnike in vplivne ljudi. Varnostni sistemi se bodo nadgrajevali v skladu s potrebami in povpraševanjem družbe, ki bo tako narekovala tempo razvoja in realizacijo novih idej.

Z našim prototipom modula za varovanje nismo posegali po novih metodah ali pristopih, želeli smo le raziskati, koliko truda in vložka je potrebnega, da pridemo do takšne rešitve. Prav tako smo želeli identificirati kakšne težave nas čakajo pri tem in kako se jim je mogoče izogniti.

Opazil sem, da sem bil pred začetkom samega razvoja veliko bolj skeptičen, kot pa potem, ko sem napravil prve korake, ki so me na koncu uspešno pripeljali do cilja. Prav tako sem imel določene pomisleke, glede nakupa kitajske različice plošče Arduino, ki pa so se na koncu na mojo srečo izkazale za brezpredmetne. Ob testiranju modula, sem sicer našel še nekaj pomanjkljivosti, ki bi jih za kakšno resno rešitev najverjetneje moral odpraviti, a v našem primeru niso imele posebnega vpliva.

Za razvoj modula sem izbral metodo brez spajkanja, to pomeni, da bom lahko vse gradnike ponovno uporabil v namen uresničitve mojih prihajajočih načrtov in ciljev.

Ob koncu čutim svoje zadoščenje in veselje, saj sem končno uspel uresničiti svojo dolgoletno željo in pri tem uspešno združil prijetno s koristnim, poleg tega pa tudi neizmerno užival v samem razvoju modula. Verjamem, da sem se pri projektu marsikaj naučil in nabral veliko prepotrebnih izkušenj, ki mi bodo koristile v moji nadaljnji karieri.

Literatura

[1] Alarmne centrale. (2014). *Spletna stran podjetja Mobicom d.o.o.* [Online]. Dosegljivo: http://www.mobicom.si/alarmne-centrale/69.html.

[2] Arduino UNO. (2014) *Spletna stran skupnosti Arduino* [Online]. Dosegljivo: http://arduino.cc/en/Main/arduinoBoardUno.

[3] F. Lipuš. Varnostna opozorilna naprava z mikrokrmilnikom. (junij 2008). *Digitalna knjižnica Univerze v Mariboru* [Online]. Dosegljivo: http://dkum.ukm.si/Dokument.php?id=6585.

[4] Mikrokrmilniki. (december 2013). *Wikipedia* [Online]. Dosegljivo: http://sl.wikipedia.org/wiki/Mikrokrmilnik.

[5] Modul RTC DS1302. (2014) *Spletna stran skupnosti Arduino* [Online]. Dosegljivo: http://playground.arduino.cc/Main/DS1302.

[6] Pametne hiše. (2014) *Spletna stran podjetja PS PROMIS d.o.o.* [Online]. Dosegljivo: http://www.ps-promis.si/sl/pametne-hise.

[7] Senzorji. (2014) *Spletna stran podjetja Mobicom d.o.o.* [Online]. Dosegljivo: http://www.mobicom.si/senzorji/70.html.