Nikša Maslović
0036422660




Fakultet elektrotehnike i računarstva, Sveučilišta u Zagrebu
Zavod za elektroničke sustave i obradbu informacija
Sveučilište u Zagrebu

- Tekst je namijenjen za sve studente i one koji žele više naučiti o praktičnom radu u razvojnom sustavu Arduino
- Potrebno je predznanje iz C programskog jezika i elektrotehnike
- Prikazano je kako spajati senzore i aktuatore u razvojni sustav i kako njima upravljati.

27. lipanj 2009

## Sažetak

U ovom tekstu se opisuje izvedba elektronskog dijela sustava za upravljanje DC motorima i senzorima temperature i osvjetljenja. Također je opisan i programski kod koji omogućuje upravljanje. Ovaj projekt je namijenjen za integraciju u sustav inteligentne kuće. Opisano je ručno, automatsko upravljanje motorima kao i upravljanje preko glavnog sustava kuće, što sustav čini vrlo praktičnim u svakodnevici. Zbog želje za lagodnijim životom, kojem svi težimo, razvijen je sustav koji omogućuje vlasnicima inteligentne kuće upravljanje roletama, prozorima, vratima iz udobnosti naslonjača ili automatsko upravljanje u ovisnosti o trenutnoj temperaturi i osvjetljenju.

## Sadržaj



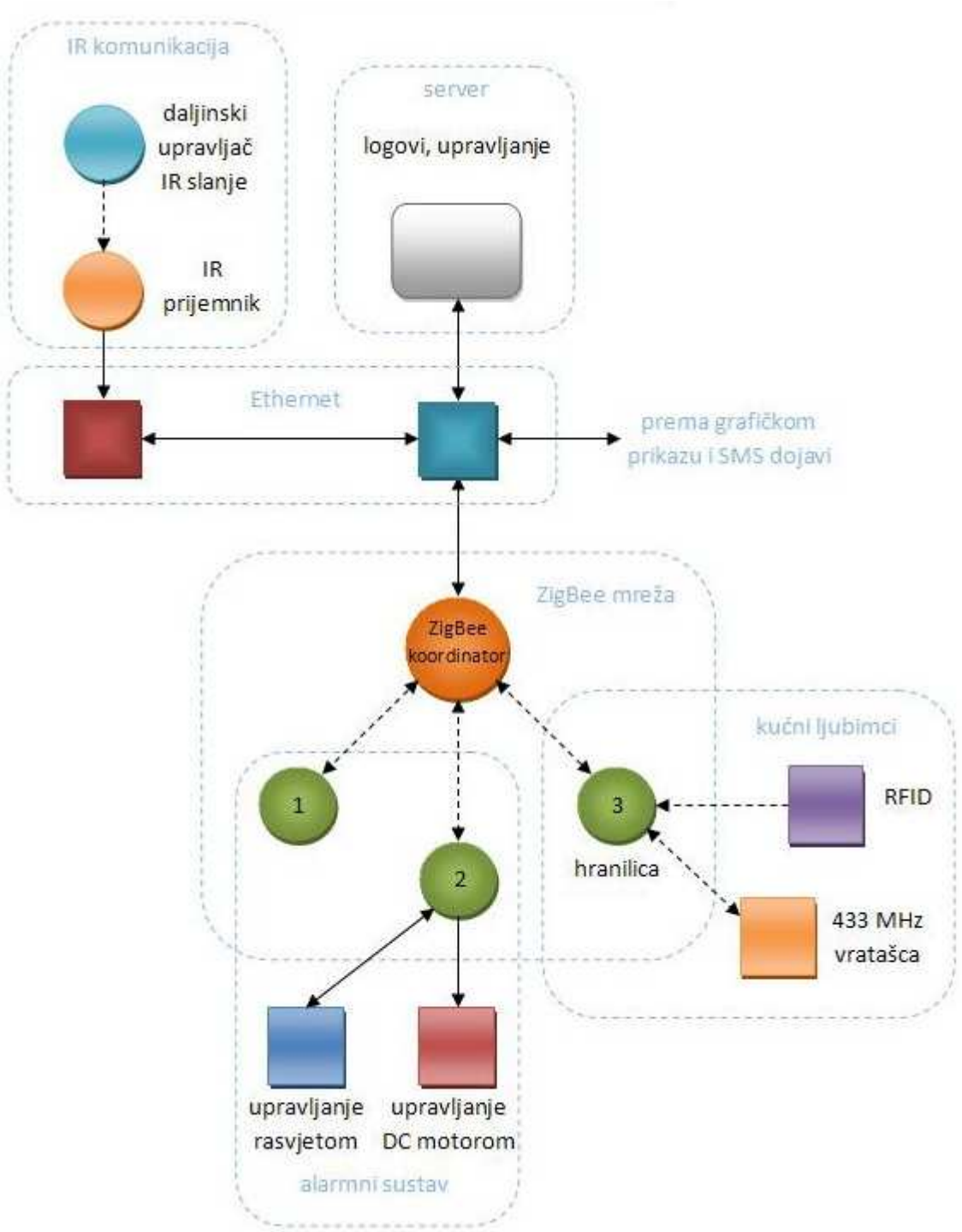
Ovaj seminarski rad je izrađen u okviru predmeta „Sustavi za pracenje i vođenje procesa“ na Zavodu za elektroničke sustave i obradbu informacija, Fakulteta elektrotehnike i računarstva, Sveučilišta u Zagrebu.

# 1. Uvod

Digitalni ulazi/izlazi mogu služiti u mnogim primjenama. Ako ih koristimo kao ulaze onda su oni početni dio nekog sustava i na njih se tada spajaju razni senzori koji prikupljaju diskretne informacije koje nam kasnije mogu poslužiti za upravljanje nekim sustavima. Ako se koriste kao izlazi tada su oni krajnji dijelovi sustava na koji se spajaju neki elementi (rasvjeta, rolete, električni ventil, vrata, prozore, rolete, rampe za invalidska kolica...) kojima želimo upravljati. Cilj ovog projekta je bio napravit sustav za upravljanje DC motorima, mjerenje temperature i osvjetljenja. Također je predviđeno da ovaj sustav bude dio velikog projekta inteligentne kuće i da upravlja motorima ovisno o informacijama koje dobije od ostalih dijelova velikog sustava inteligentne kuće.

Tema inteligentne kuće je vrlo aktualna i privlačna u današnje vrijeme. Cilj joj je olakšati život ljudima kod rada svakodnevnih poslova u vlastitom domu. Svaki čovjek se želi što ugodnije osjećati u svom domu što mu inteligentna kuća u velikoj mjeri i omogućuje. Konkretno, projekt *Digitalni ulazi-izlazi* omogućuje upravljanje DC motorima koji će podizati ili spuštati rolete na tipku ili automatski kada su temperatura i osvjetljenje u prostoriji preveliki/premali. Također se omogućuje upravljanje motorima ovisno o naredbama sustava inteligentne kuće npr. ako se preko daljinskog upravljača pošalje naredba o zatvaranju/otvaranju prozora, vrata, roleta...

## 2. Prikaz sustava inteligentne kuće




**Slika 1: Sustav inteligentne kuće**

Navedena shema predstavlja sustav inteligentne kuće kojoj pripada i projekt *Digitalni ulazi-izlazi*. Na slici 1 se vidi da je projekt *Digitalni ulazi-izlazi* dio jednog od dva čvora alarmnog sustava bežične ZigBee mreže u kojoj se još nalazi i čvor "kućni ljubimci". ZigBee mreža preko etherneta komunicira i sa ostalim dijelovima kuće. Projekt *Digitalni ulazi-izlazi* u sustav inteligentne kuće šalje podatke o temperaturi i osvjetljenju svake prostorije, te podatak o statusu DC motora (rolete podignute/spuštene, prozor otvoren/zatvoren). Od sustava inteligentne kuće prima se podatak koji upravlja radom DC motora (npr. podatak zatvori prozor) i koji se šalje sa daljinskog upravljača.

# 3. Spajanje senzora i DC motora na Arduino

Arduino ima 14 digitalnih ulazno/izlaznih pinova, od kojih se na 6 može dati PWM signal, i 6 analognih ulaza. Vrijednosti se sa analognih pinova čitaju pomoću funkcije *analogRead()*, dok se sa digitalnih pinova vrijednosti čitaju pomoću funkcije *digitalRead()*. Za upis vrijednosti na digitalne pinove se koriste funkcije *analogWrite()* i *digitalWrite()*.[1]

## 3.1. Analogni ulazi

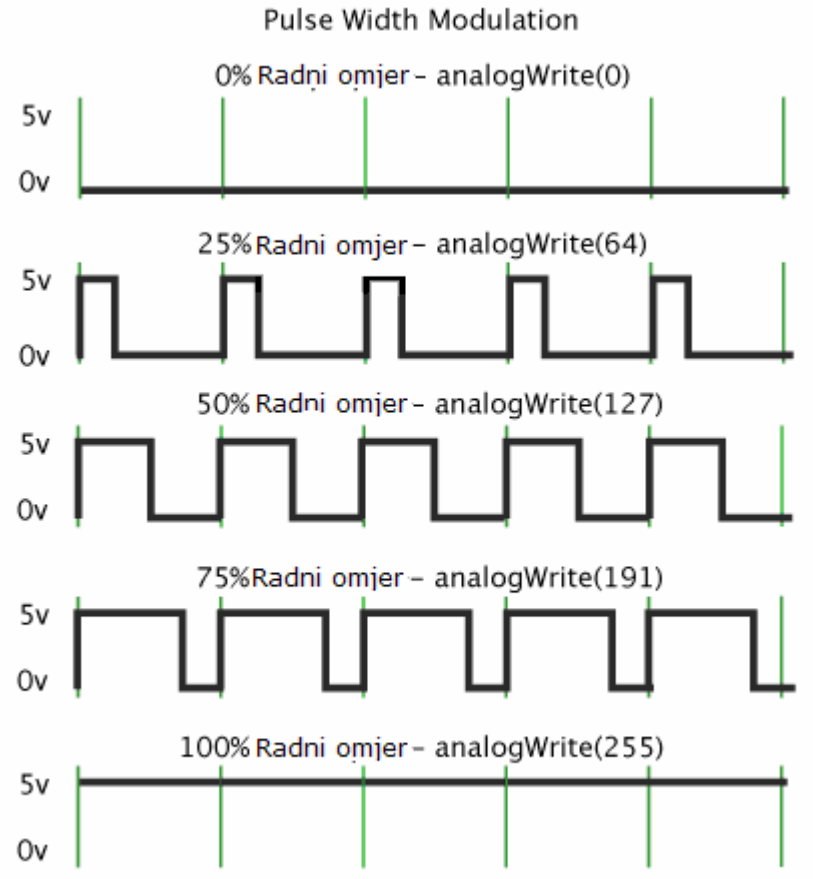
Arduino ima 6 kanalni 10-bitni AD pretvornik. To znači da će se, koristeći funkciju *analogRead()*, ulazni napon podijeliti između 0 i 5V u integer vrijednosti između 0 i 1023. Funkcija *analogRead()* prima broj pina sa kojega želimo čitati, a vraća integer vrijednosti između 0 i 1023. Ovime se dobije da je korak kvanzitacije: 5V / 1024 = 0.0049V (4.9 mV). Da bi se pročitalo sa analognog ulaza potrebno je 100us, pa je najviša brzina čitanja 10 000 puta u sekundi.

## 3.2. Digitalni ulazi/izlazi

Funkcija *digitalRead()* čita vrijednosti sa određenog pina i vraća HIGH ili LOW, dok funkcija *digitalWrite()* upisuje vrijednost HIGH ili LOW na odabrani pin. Funkcija *analogWrite()* daje analognu vrijednost (PWM signal) na PWM digitalne pinove. Može se koristiti za promjenu brzine motora ili za promjenu jačine osvjetljenja LED diode.

### 3.2.1. PWM modulacija

Pulsno širinska modulacija (Pulse Width Modulation ili PWM) je tehnika dobivanja analognih vrijednosti preko digitalnih oblika. Digitalno upravljanje se iskoristi za dobivanje pravokutnog signala koji se mijenja između "ima signala" i "nema signala" (signal on/off). Ovaj način rada može simulirati napon koji se mijenja između stanja kada signal ima najvišu vrijednost (5V) i stanja kada signal ima minimalnu vrijednost (0V) tako da se mijenja dio vremena u kojem ima signala i vremena u kojem nema signala. Trajanje vremena u kojem je signal upaljen naziva se širina pulsa. Dobivanje različitih analognih vrijednosti postiže se modulacijom širine pulsa. Ako se ponavlja ovaj on-off uzorak kroz npr. LED diodu, rezultat je isti kao da je signal stalni napon između 0V i 5V upravlja osvjetljenjem LED diode. Na slici 2 zelene linije predstavljaju stvarno vrijeme perioda. Ovaj period je inverz PWM frekvencije. Frekvencija PWM signala je približno 490Hz.

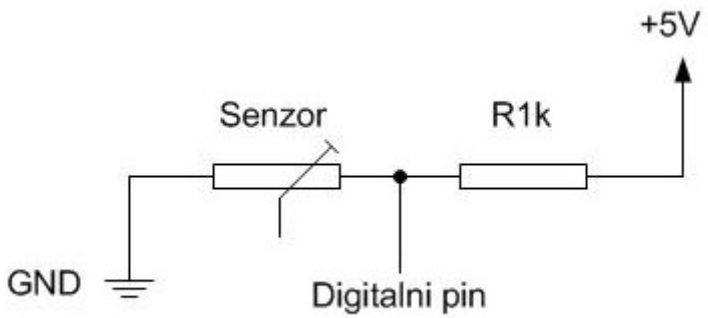



**Slika 2: Pulsno-širinska modulacija (PWM)[1]**

Nakon poziva *analogWrite()* proizvesti će se pravokutni signal sa određenim radnim omjerom (0-uvijek isključeno ili 255-uvijek uključeno) sve do sljedećeg poziva *analogWrite()* (ili poziva *digitalRead()* ili *digitalWrite()* na isti pin). Ako je pozvana naredba *analogWrite(255)* radni omjer će biti 100% (uvijek 5V), dok će poziv analogWrite(127) rezultirati radnim omjerom od 50% (5V će biti 50% vremena). Najviša struja na svakom pinu je 40mA.

## *3.3. Spajanje senzora*

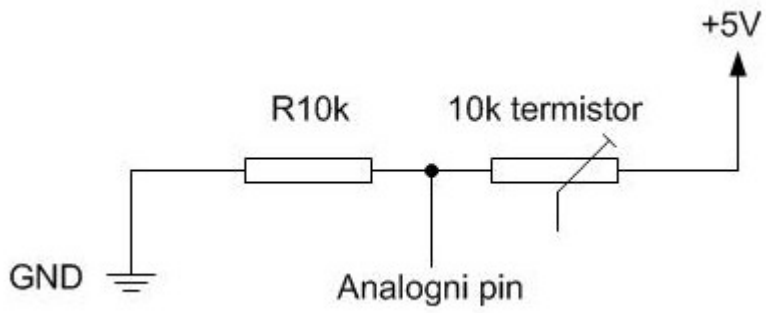
Ovaj projekt sadrži dva senzora, senzor temperature i senzor osvjetljenja. Senzor osvjetljenja je LDR otpornik (light dependent resistor) kojemu se otpor mijenja ovisno o svjetlosti koja ga obasjava. Otpor LDR senzora je vrlo velik, ponekad i do 1 MΩ, ali kad se obasja svjetlošću otpor se uvelike smanji. LDR senzor se na Arduno spaja prema slici 3:




**Slika 3: Spajanje LDR senzora na Arduino[2]**

Kada se obasja svjetlošću otpor će mu se smanjiti, pa će se na analognom ulazu Arduina pročitati manja vrijednost (funkcija *analogRead()*).

Temperatura se mjeri termistorom. Termistor je otpornik kojemu se otpor mijenja ovisno o temperaturi. Temperaturni koeficijent mu može biti pozitivan (PTC) ili negativan (NTC), što znači da će kod PTC termistora otpor rasti sa porastom temperature, dok će kod NTC-a otpor padati. Kod mjerenja temerature najčešće se koristi NTC termistor dok se PTC termistor koriste kao element za zaštitu. Način spajanja termistora prikazan je na slici 4:




**Slika 4: Spajanje termistora na Arduino[3]**

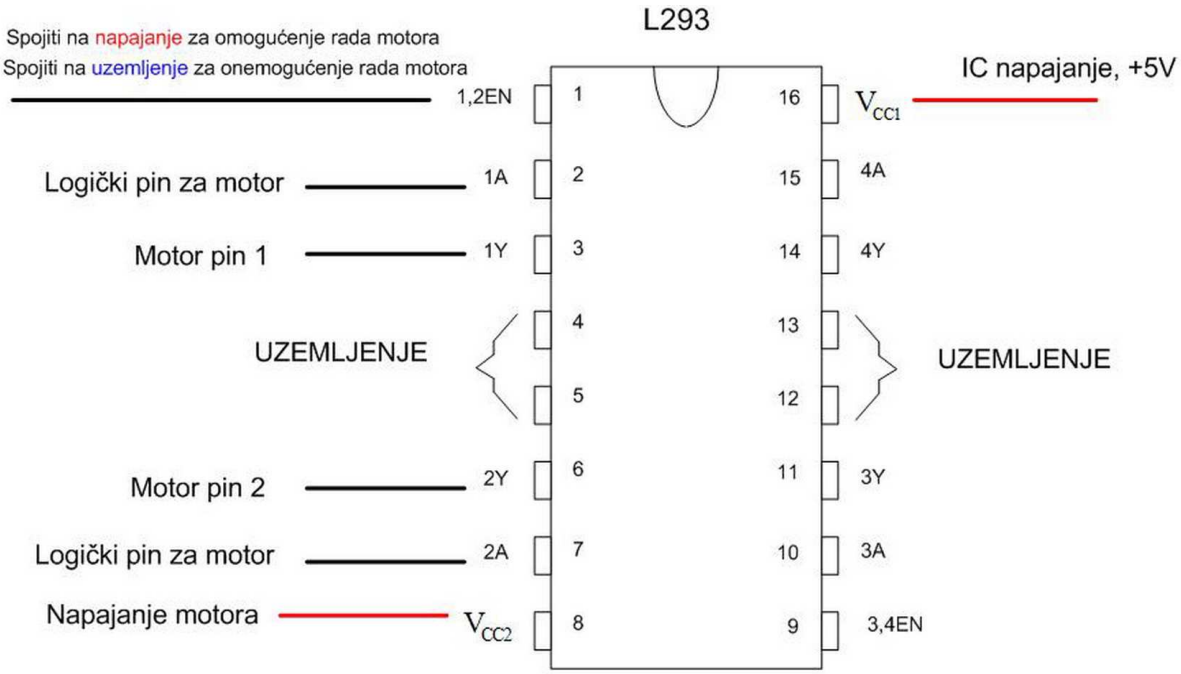
Vrijedi da će promjena temperature promijeniti vrijednost otpornika pa će se na analognom pinu Arduina očitavati različita vrijednost ovisno o padu napona na otporniku R10k.

### 3.3.1. Kalibracija senzora

Kalibracija senzora kod Arduina se radi tako da očitavanja tijekom prvih pet sekundi za vrijeme testnog perioda definiraju minimum i maksimum očekivanih vrijednosti. Postave se inicijalne vrijednosti najvećeg minimuma i čeka se neka vrijednost koja će biti manja od inicijalnog minimum. Tada se ta vrijednost sačuva kao novi minimum. Isto tako se postavi vrijednost najmanjeg maksimuma i čeka se vrijednost koja će ju prekoračiti. Zatim se ta vrijednost postavi kao novi maksimum. Tada se pozove funkcija *map(value, fromLow, fromHigh, toLow, toHigh)* koja skalira vrijednosti jednog opsega u drugi. Vrijednost *fromLow* će se skalirati u vrijednost *toLow*, vrijednost *fromHigh* u vrijednost *toHigh*, a vrijednosti između *fromLow* do *fromHigh* u vrijednosti između *toLow* do *toHigh.*

## 3.4. Spajanje DC motora

Da bi DC motor mijenjao smjer okretanja moramo biti u mogućnosti promijeniti smjer struje u motor. Najlakši način da se to postigne je H-most. U ovom projektu je korišten L293 H-most. S njim je moguće upravljati sa 2 motora. Može raditi sa strujama od 1A i naponima između 4.5 i 36V.




| EN | 1A | 2A | FUNKCIJA |
|---|---|---|---|
| H | L | H | Okretanje u desno |
| H | H | L | Okretanje u lijevo |
| H | L | L | Brzo zaustavljenje motora |
| H | H | H | Brzo zaustavljenje motora |
| L | X | X | Brzo zaustavljenje motora |

L = niska razina,
H = visoka razina,
X = nevažno

**Slika 5: H-most[4]**

Kao što je prikazano u tablici istinitosti na slici 5 rad motor ovisi o stanjima logičkih pinova H-mosta koje postavlja Arduino. Pinovi H-mosta su objašnjeni u nastavku:

Pin 1 (1,2EN) omogućuje i onemogućuje rad motora ovisno o tome je li na njemu niska ili visoka razina. Služi i za dovođenje PWM signala.
Pin 2 (1A) je logički pin za motor (ulaz je ili HIGH ili LOW)
Pin 3 (1Y) je pin na koji se spaja motor
Pin 4-5 – na njih se spaja uzemljenje
Pin 6 (2Y) je pin na koji se spaja motor

Pin 7 (2A) je logički pin za motor (ulaz je ili HIGH ili LOW)
Pin 8 (VCC2) je pin na koji se spaja izvor napona za motor
Pin 9-11 - ne spajaju se jer se koristi samo jedan motor u ovom projektu
Pin 12-13 – na njih se spaja uzemljenje
Pin 14-15 ne spajaju se jer se koristi samo jedan motor u ovom projektu
Pin 16 (VCC1) je spojen na 5V

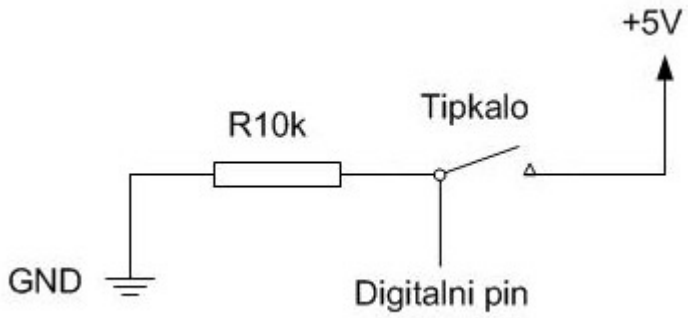
U ovom projektu, pin 1 se spaja na digitalni pin na Arduinu koji može dati PWM signal. Kao što je već navedeno, koristit će se funkcija *analogWrite()* koja će proizvesti PWM signal pomoću kojeg ćemo ubrzavat okretanje motora. Logički pinovi za motor se spajaju na digitalne pinove Arduina koji šalje HIGH i LOW (funkcija *digitalWrite()*) da bi se motor okretao u jednom smjeru ili LOW i HIGH da bi se motor okretao u drugom smjeru. Vanjsko napajanje motora (npr. baterija) se spaja na pin 8 da se motor ne bi napajao iz Arduina.

### 3.4.1. Nailazak motora na prepreku

Jedno od mogućih poboljšanja sustava koji nisu uključeni u ovaj projekt je zaustavljanje motora nailaskom na prepreku. To je vrlo važno zbog sigurnosti u kući, a i zbog mogućih kvarova sustava u slučaju da motor ne stane. Nailaskom motora na prepreku, koja predstavlja prevelik moment tereta, motor će pokušati ući u generatorski način rada (napon na motoru će se početi smanjivati dok na kraju ne okrene polaritet). To se mora spriječiti zaustavljanjem motora. Jedan od mogućih načina je mjerenjem napona na stezaljkama motora ako je poznato uobičajeni napon na motoru. Svako odstupanje od uobičajenih granica kretanja napona će rezultirati zaustavljanjem motora.

## 3.5. Spajanje tipkala i prekidača

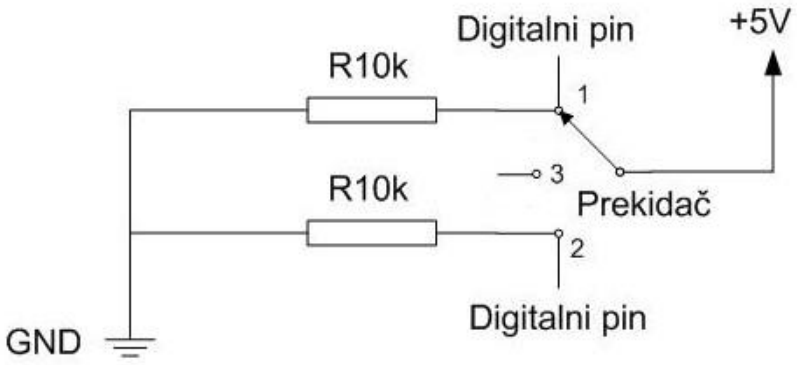
Postoje dva načina za spajanje tipkala i prekidača na Arduino: preko pull-down i pull-up otpornika. O ovom projektu se koristi pull-down metoda. Tipkala se na Arduino spajaju preko otpornika od 10kΩ kao što je prikazano na slici 6.




**Slika 6: Spajanje tipkala na Arduino**

U slučaju da je tipkalo stisnuto zatvorit će se strujni krug preko otpornika na kojem će se dogodit pad napona i na digitalnom pinu će se pomoću funkcije *digitalRead()* očitati visoka razina. U slučaju da tipkalo nije stisnuto krugom ne teče struja, nema pada napona na otporniku i na digitalnom pinu će se očitati niska razina.

Prekidač sa tri stanja se spaja kao što je prikazano na slici 7. U slučaju da je prekidač u položaju 1 na jednom će se digitalnom pinu očitati visoka razina, a na drugom niska. U položaju 2 će biti obratno, dok će se u položaju 3 na oba digitalna pina očitati niska razina.




**Slika 7: Spajanje prekidača na Arduino**

# 4. Programski kod

Programski kod je organiziran tako da je moguće upravljati sa proizvoljnim brojem motora (prozora). Predviđeno je da svaki prozor ima svoja dva senzora koji mjere temperaturu i osvjetljenje u prostoriji u kojoj se prozor nalazi. U programskom kodu svaki prozor se nalazi u polju prozora i napravljen je kao struktura u kojoj imamo razne varijable koje opisuju prozor. Svaki prozor je definiran sa pinovima Arduina na koje se spaja, nekim zastavicama kao što su zastavice smjera, automatske regulacije, daljinskog upravljanja, otvorenosti/zatvorenosti, zastavicom da li je motor omogućen/onemogućen, varijablama u koje se pohranjuju vrijednosti pročitane sa senzora, pinovima na koje se spajaju senzori, te granicama najniže i najviše temperature i osvjetljenja u kojima se treba kretati trenutna temperatura i osvjetljenje.

```
struct structProzor{
  int motor1Pin, motor2Pin, pwmPin, motorSpeed, motorDirection, motorEnable;
  //vrijednosti senzora
  int sensLux, sensTemp, sensOpen, sensClose;
  int switchPinLeft, switchPinRight;
  //zastavice
  int autoRegEnable, autoRegActive, remoteControlActive;
  int minTemp,maxTemp;
  int minLux, maxLux;
  // pinovi na koje su spojeni senzori
  int pin_lux, pin_temp, pin_open, pin_close;
};
```

U slučaju da neke podatke želimo ispisivati na serijski port serijska komunikacija se inicijalizira funkcijom *Serial.begin()*.

## 4.1. Glavni program

Pokretanjem glavnog programa za svaki se prozor učitaju podaci svih senzora koje on posjeduje. Za čitanje osvjetljenja se najprije poziva funkcija *analogRead()* koja prima redni broj analognog pina, na koje je spojen senzor i sa kojega se čita vrijednost koja ovisi o osvjetljenju. Zatim se ta vrijednost predaje funkciji *double toLux(int x)* koja tu vrijednost pretvara u lux-eve i vraća je.

Za računanje temperature se najprije koristi funkcija *analogRead()* koja prima redni broj analognog pina na koji je spojen senzor temperature

i sa kojega se čita vrijednost koja ovisi o temperaturi. Tada se ta vrijednost prosljeđuje funkciji *double toCelsius(int RawADC)* koja vraća vrijednost temperature u stupnjevima celzijusevim.

Za čitanje sa senzora otvorenosti i senzora zatvorenosti koristi se funkcija *digitalRead()*. Ako je roleta spuštena, senzor zatvorenosti će to prepoznati i funkcija će preko digitalnog pina, na koje je senzor spojen, vratiti HIGH. U slučaju da je roleta podignuta vraća se LOW. Obratno vrijedi za senzor otvorenosti. Ako je roleta podignuta, senzor otvorenosti će to prepoznati i funkcija *digitalRead()* će vratiti HIGH. U slučaju da je roleta spuštena senzor će vratiti LOW.

```
for (int i=0; i<brojProzora; i++){
  prozor[i].sensLux = toLux(analogRead(prozor[i].pin_lux));
  prozor[i].sensTemp = toCelsius(analogRead(prozor[i].pin_temp));
  prozor[i].sensOpen = digitalRead(prozor[i].pin_open);
  prozor[i].sensClose = digitalRead(prozor[i].pin_close);
}
```

Zatim se za svaki prozor pokreću funkcije koje upravljaju radom motora:

```
for (int i=0; i<brojProzora; i++){
  CheckSensors(i);
  HandRegulation(i);
  AutomaticRegulation(i);
  }
```

Funkcija *void CheckSensors(int p)* prima redni broj prozora i provjerava podatke senzora otvorenosti/zatvorenosti za svaki prozor. Služi da zaustavi motor, ako je motor došao do kraja zatvaranja/otvaranja. Zaustavljanje motora se obavlja tako da se poziva funkcija *void stopMotor(int p)* koja preko digitalnih pinova Arduina upisuje nisku razinu na digitalne pinove H-mosta na koje je spojen motor. Također se brzina, te zastavice smjera, omogućenja/onemogućenja motora i daljinskog upravljanja postave na 0.

Poziv funkcije *int HandRegulation(int p)* omogućuje upravljanje motorom preko prekidača (ručna regulacija). Funkcija prima redni broj prozora. Predviđeno je da se automatska regulacija isključuje ako je uključena u trenutku početka ručnog upravljanja prozorima (ručna regulacija ima veći prioritet). U navedenoj funkciji se nalaze pozivi funkcija *void setMotorDirection(int p, int d), void enableMotor(p), void runMotor(p)* i *void stopMotor(int p).* Sa digitalnih pinova Arduina, na koje je spojen prekidač, pročitaju se vrijednosti korištenjem funkcije *digitalRead().*Zatim se poziva funkcija *void setMotorDirection(int p, int d)* koja prima redni broj motora i smjer okretanja motora. Ovisno o tome koji je položaj prekidača (položajem je određen smjer kretanja motora) preko digitalnih pinova Arduina postavljaj se pinovi H-mosta na odgovarajuću

razinu i time se omogućuje okretanje motora u raznim smjerovima. Zatim se poziva funkcija *enableMotor(p)* koja omogućuje upravljanje motorom postavljanjem zastavice *motorEnable* u 1. Nakon omogućenja motora poziva se funkcija *void runMotor(p)* koja služi da bi preko PWM digitalnog pin Arduina poslao PWM signal na pin 1 H-mosta i time upravljala brzinom okretanja motora (predviđeno je da se nakon paljenja motora brzina okretanja postepeno povećava dok ne dođe do najviše brzine nakon čega se motor nastavi okretati najvišom brzinom). U toj funkcije moguće je podešavati brzinu okretanje motora mijenjajući radni omjer PWM signala. Radni omjer se mijenja od 0 (motor se ne okreće) pa sve do 255 (najviša brzina motora). U slučaju da je prekidač u položaju kad su na oba digitalna pina logičke 0 poziva se funkcija *void stopMotor(int p).*

Funkcija *void AutomaticRegulation(int p)* prima redni broj prozora. Prvo provjerava da li je automatska regulacija uključena jer je moguće onemogućit automatsku regulaciju na razini inteligentne kuće. U slučaju da je automatska regulacija isključena izlazi se iz funkcije. Zatim se provjerava da je li trenutna temperatura ili osvjetljenje u zadanim granicama. Ako su trenutne vrijednosti više od željenih, poziva se funkcija *void setMotorDirection(int p, int d)* kojoj se prosljeđuje redni broj motora i broj 1 koje predstavlja smjer okretanja motora prema dolje (rolete se spuštaju). Zatim se pozivaju funkcije *void enableMotor(p)* i *void runMotor(p)* koje su ranije opisane. U slučaju da su trenutne vrijednosti manje od željenih također se poziva funkcija *void setMotorDirection(int p, int d)* kojoj se ovaj put predaje redni broj prozora i broj 2, koji predstavlja smjer okretanja motora prema gore (podizanje roleta). Zatim se ponovo pozivaju funkcije *void enableMotor(p)* i *void runMotor(p).* U slučaju da je trenutna temperatura u željenim granicama automatska regulacija nije aktivna.

## 4.2. Funkcije koje se pozivaju od strane koordinatora

Funkciju *void remoteControl(int p, int d)* poziva inteligentna kuća kada se želi upravljati prozorima preko glavnog sustava kuće. Nije predviđeno da čvor koji upravlja prozorima izravno primi podatak od daljinskog upravljača nego da taj podatak zaprimi inteligentna kuća. Tada inteligentna kuća bežično pošalje podatak čvoru koji upravlja prozorima o pokretanju motora. Predaje se redni broj prozora i smjer okretanje (1 ili 2) ovisno o tome želi li se roleta spustiti/podignuti. Radi jednostavnosti se gasi automatsko upravljanje kad se upravlja roletama preko daljinskog upravljača. Zatim se pozivaju već prije opisane funkcije *void setMotorDirection(int p, int d), void enableMotor(p), void runMotor(p).*

Funkciju *void readSensor(int k, byte *HByte, byte *LByte)* također poziva inteligentna kuća kada se želi da se proslijedi vrijednost sa pojedinog senzora. Funkcija prima redni broj senzora i to tako da su za prozor 1 deklarirani senzori 0 i 1 (0 za senzor temperature i 1 za senzor osvjetljenja), za prozor 2 su deklarirani senzori 2 i 3 (2 za senzor temperature i 3 za senzor osvjetljenja) itd. Funkcija rastavlja vrijednost, koju vrati funkcija *double toLux(int x)* ili *double toCelsius(int RawADC)* (ovisno sa kojeg se senzora čita), na dva bajta da bi se ta vrijednost mogla poslati preko bežične mreže. Zato je potrebno predati pokazivače na gornji i donji bajt u koje će se vratit rezultat.

Funkcija *int status(int p)* prima redni broj prozora, a vraća 1 ako je prozor otvoren, 0 ako je zatvoren i -1 ako nije ništa od navedenog, tj. prozor se nalazi na nekoj visini.

## 5. Zaključak

U ovom radu je opisan sustav za bežično, ručno i automatsko upravljanje DC motorom. Prikazano je i kako se spajaju neki od senzora na razvojni sustav Arduino. Ovaj sustav je samo jedan dio realizacije sustava inteligentne kuće koja je u današnje vrijeme sve više aktualna. Postoje još mnogi načini za poboljšanje sustava. Neki od njih su: podatak o trenutnom položaju motora ili funkcija zaustavljanja (i vraćanja, prema potrebi) kod nailaska na prevelik otpor (kao kod podizača stakla na automobilima).

# 6. Literatura

[1] Arduino,
URL: http://www.arduino.cc/
[2] Measure Light Intensity using Light Dependent Resistor (LDR), 2008.
URL: http://www.emant.com/316002.page
[3] Reading a Thermistor, 2008.
http://www.arduino.cc/playground/ComponentLib/Thermistor2
[4] DC motor control using H-bridge, 2009.
URL: http://itp.nyu.edu/physcomp/Labs/DCMotorControl (2009-2-19)
[5] Xavier, Akash. Controlling a DC motor with the Arduino and L293D,
URL: http://akashxav.com/2009/04/18/arduino-l293d-dc-motor/

# 7. Pojmovnik

| Pojam | Kratko objašnjenje | Više informacija potražite na |
|---|---|---|
| ZigBee | ZigBee je bežični komunikacijski protokol namjenjem osobnim mrežama s malom propusnošću i malom potrošnjom energije | http://hr.wikipedia.org/wiki/ZigBee |
| LDR otpornik | Otpornik koji mijenja otpor ovisno o osvjetljenju | http://www.technologystudent.com/elec1/ldr1.htm |
| Termistor | Otpornik kojemu se otpor mijenja ovisno o teperaturi | http://en.wikipedia.org/wiki/Thermistor |
| L293 | Integrirani krug za upravljanje motorima | http://www.st.com/stonline/books/pdf/docs/1328.pdf |
| Arduino | Razvojni sustav | http://www.arduino.cc/ |

# 8. Dodatak

Programski kod:

```
#include <math.h>
//////// KONVENCIJE ///////////
// RIGHT -> DOLE = 1
// LEFT -> GORE = 2
/////// MOTOR LINEARNO UBRZANJE ////////////////////
int topSpeed = 2550; // skalirat ce se za 10
int speedStep = 5;
/////////STANDARDNE POSTAVKE /////////////////////
int DefaultAutoRegEnable = 0;
int DefaultMinTemp  = 20;
int DefaultMaxTemp  = 25;
int DefaultMinLux   = 150;
int DefaultMaxLux   = 350;
////////////////////////////////////////////////
#define brojProzora  1
int pinoviMotora[] = {6,7,9}; //
{motor1Pin_1,motor2Pin_1,pwmPin_1,.....,motor1Pin_n,motor2Pin_n,pwmPin_n}
i tako dalje
int pinoviPrekidaca[] = {4,5}; //
{switchPinLeftn_1,switchPinRight_1,....,switchPinLeftn_n,switchPinRight_n}
i tako dalje
int pinoviSenzora[] = {0,1,2,3}; // brojevi pinova: lux, temp, open, close
-> za jedan prozor


////////////////////////////////////////////////////////////////
//////////      NE MJENJATI NIđŽ´TA NAKON OVE LINIJE     /////////
////////////////////////////////////////////////////////////////

struct structProzor{
  int motor1Pin, motor2Pin, pwmPin, motorSpeed, motorDirection,
motorEnable;
  int switchPinLeft, switchPinRight;
  int sensLux, sensTemp, sensOpen, sensClose;
  ////////////////////
  int autoRegEnable, autoRegActive, remoteControlActive;
  int minTemp,maxTemp;
  int minLux, maxLux;
  // pinovi na koje su spojeni senzori
  int pin_lux, pin_temp, pin_open, pin_close;
};

struct structProzor prozor[brojProzora];

void setup() {
  // set the switch as an input:
  Serial.begin(19200);

  for (int i=0; i<brojProzora; i++){
```

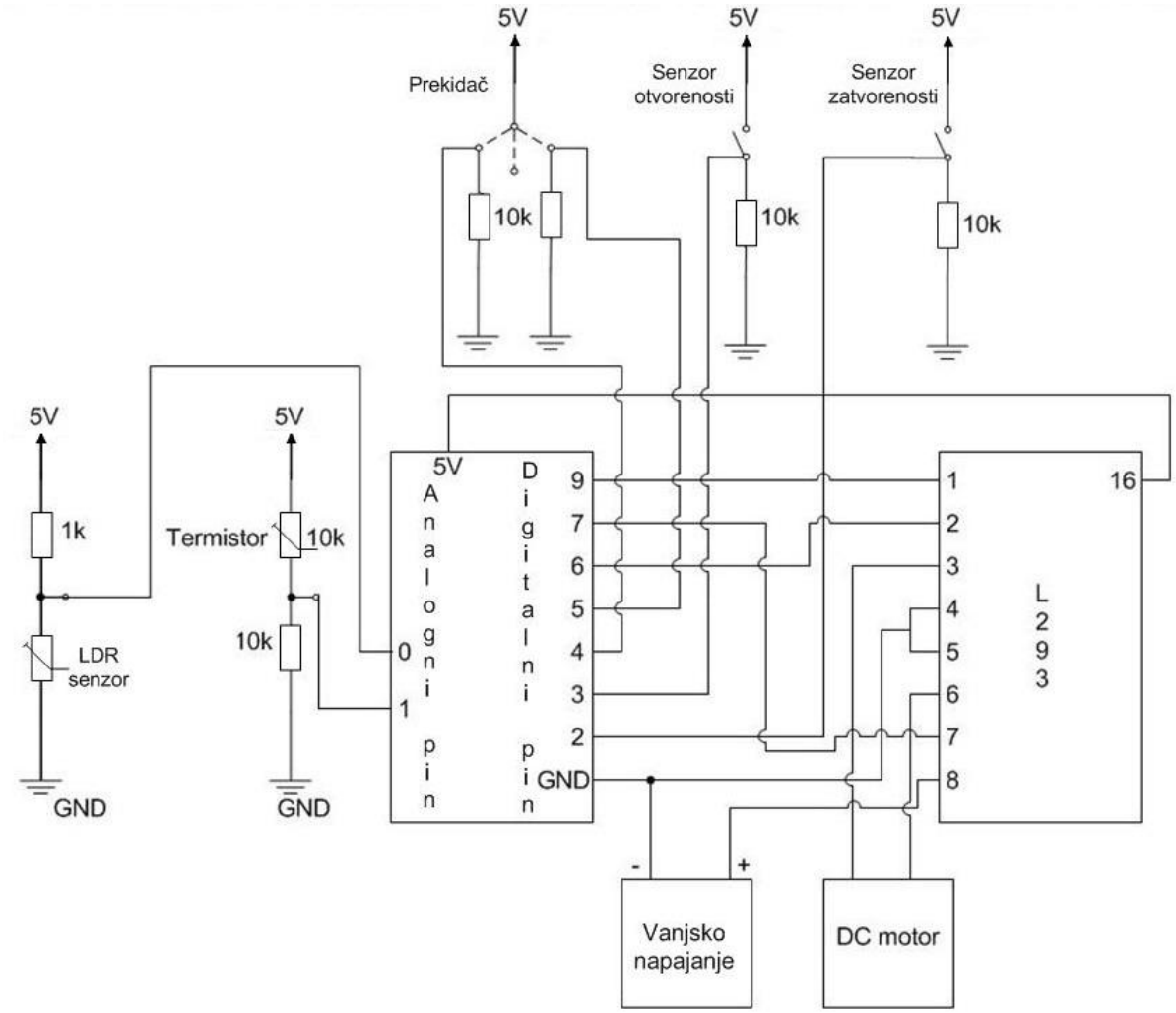
```
    prozor[i].motor1Pin = pinoviMotora[i*3];
    prozor[i].motor2Pin = pinoviMotora[i*3+1];
    prozor[i].motorEnable = 1;
    prozor[i].motorSpeed = 0;
    prozor[i].motorDirection = 0;
    prozor[i].pwmPin = pinoviMotora[i*3+2];
    prozor[i].switchPinLeft = pinoviPrekidaca[i*2];
    prozor[i].switchPinRight = pinoviPrekidaca[i*2+1];
    prozor[i].remoteControlActive = 0;
    pinMode(prozor[i].motor1Pin, OUTPUT);
    pinMode(prozor[i].motor2Pin, OUTPUT);
    pinMode(prozor[i].pwmPin, OUTPUT);
    pinMode(prozor[i].switchPinLeft, INPUT);
    pinMode(prozor[i].switchPinRight, INPUT);
    //digitalWrite(prozor[i].pwmPin, HIGH); //set enablePin high so that
motor can turn on:
    prozor[i].autoRegEnable = DefaultAutoRegEnable;
    prozor[i].autoRegActive = 0;
    prozor[i].minTemp = DefaultMinTemp;
    prozor[i].maxTemp = DefaultMaxTemp;
    prozor[i].minLux = DefaultMinLux;
    prozor[i].maxLux = DefaultMaxLux;
    prozor[i].pin_lux=pinoviSenzora[i*4];
    prozor[i].pin_temp=pinoviSenzora[i*4 + 1];
    prozor[i].pin_open=pinoviSenzora[i*4 + 2];
    prozor[i].pin_close=pinoviSenzora[i*4 + 3];
  }
}

void loop() {
  byte h1;
  byte l1;
  // oďž´itanje senzora svakog prozora
  // svi senzori se citaju u svakom prolazu, potrebno izmjeniti
  for (int i=0; i<brojProzora; i++){
    prozor[i].sensLux = toLux(analogRead(prozor[i].pin_lux));
    prozor[i].sensTemp = toCelsius(analogRead(prozor[i].pin_temp));
    prozor[i].sensOpen = digitalRead(prozor[i].pin_open);
    prozor[i].sensClose = digitalRead(prozor[i].pin_close);

  }


for (int i=0; i<brojProzora; i++){
    CheckSensors(i);
    HandRegulation(i);
    AutomaticRegulation(i);
  }
}

int HandRegulation(int p){
  int left  = digitalRead(prozor[p].switchPinLeft);
  int right = digitalRead(prozor[p].switchPinRight);
  int autoRegActive = prozor[p].autoRegActive;
  int remoteControlActive = prozor[p].remoteControlActive;
  int motor1Pin = prozor[p].motor1Pin;
  int motor2Pin = prozor[p].motor2Pin;
```

```
  int pwmPin = prozor[p].pwmPin;

  if (right == HIGH) {
    if (autoRegActive) prozor[p].autoRegEnable = 0;  //ako se prilikom
pritiska tipke vec desava automatska regulacija, iskljuci je
    setMotorDirection(p,1);
    enableMotor(p);
    runMotor(p);
  }
  else if (left == HIGH){
    if (autoRegActive) prozor[p].autoRegEnable = 0;
    setMotorDirection(p,2);
    enableMotor(p);
    runMotor(p);
  }
  else {
    if (left == LOW and right == LOW){
      if (!autoRegActive and !remoteControlActive){
        stopMotor(p);
      }
    }
  }
  //Serial.println(autoRegActive);
}


void runMotor(int p){
  int pwmPin = prozor[p].pwmPin;
  int motorSpeed = prozor[p].motorSpeed;
  if( ! prozor[p].motorEnable ) return;
  if( motorSpeed < topSpeed )
    prozor[p].motorSpeed += speedStep;
  else
    prozor[p].motorSpeed = topSpeed;
  analogWrite(pwmPin,prozor[p].motorSpeed/10);
}

void stopMotor(int p){
  digitalWrite(prozor[p].motor1Pin, LOW);
  digitalWrite(prozor[p].motor2Pin, LOW);
  prozor[p].motorSpeed = 0;
  prozor[p].motorDirection = 0;
  prozor[p].motorEnable = 0;
  prozor[p].remoteControlActive = 0;

}

void CheckSensors(int p){
  int sensOpen = prozor[p].sensOpen;
  int sensClose = prozor[p].sensClose;
  int motorDirection = prozor[p].motorDirection;

  if( motorDirection == 1 and sensClose == 1 ) stopMotor(p);
  if( motorDirection == 2 and sensOpen == 1 ) stopMotor(p);
}

void setMotorDirection(int p, int d){
```

```
  int motor1Pin = prozor[p].motor1Pin;
  int motor2Pin = prozor[p].motor2Pin;
  if(d==1){
    //right <=> down
    digitalWrite(motor1Pin, LOW);
    digitalWrite(motor2Pin, HIGH);
    prozor[p].motorDirection = 1;
  }
  else if(d==2){
    digitalWrite(motor1Pin, HIGH);
    digitalWrite(motor2Pin, LOW);
    prozor[p].motorDirection = 2;
  }
  else{
    stopMotor(p);
    prozor[p].motorDirection = 0;
  }
}

void enableMotor(int p){
  prozor[p].motorEnable=1;
}

void AutomaticRegulation(int p){
  int sensTemp = prozor[p].sensTemp;
  int sensLux = prozor[p].sensLux;
  int maxTemp = prozor[p].maxTemp;
  int maxLux = prozor[p].maxLux;
  int minTemp = prozor[p].minTemp;
  int minLux = prozor[p].minLux;

  if(!prozor[p].autoRegEnable){
    prozor[p].autoRegActive=0;
    return;
  }

  if(sensTemp < minTemp and sensLux < minLux){
    setMotorDirection(p,1);
    enableMotor(p);
    runMotor(p);
    prozor[p].autoRegActive = 1;
  }
  else if(sensTemp > maxTemp and sensLux > maxLux){
    setMotorDirection(p,2);
    enableMotor(p);
    runMotor(p);
    prozor[p].autoRegActive = 1;
  }
  else{
    prozor[p].autoRegActive = 0;
  }
}
double toLux(int R) {
  double lux;
  float x;
  x=102400/R;
  lux=5*x-500;
```

```
  return (int)lux;
}

double toCelsius(int RawADC) {
  double Temp;
  Temp = log(((10240000/RawADC) - 10000));
  Temp = 1 / (0.001129148 + (0.000234125 * Temp) + (0.0000000876741 * Temp
* Temp * Temp));
  Temp = Temp - 273.15;

  return (int)round(Temp);
}

////////////////////////////////
/// FUNKCIJE ZA KOORDINATORA   ///
////////////////////////////////
// Funkcija prima redni broj prozora i zeljeni smjer
// Za smjer koristiti sljedece:
// RIGHT -> DOLE = 1
// LEFT -> GORE = 2
void remoteControl(int p, int d)
{
  if (prozor[p].autoRegActive) prozor[p].autoRegEnable = 0;
  prozor[p].remoteControlActive = 1;
  setMotorDirection(p,d);
  enableMotor(p);
  runMotor(p);
}

// Funkcija oďż˝itava vrijednosti senzora
// Prima redni broj senzora i pokazivaďż˝e na
// byte varijabe u koje ďż˝e biti spremljene vrijednosti
void readSensor(int k, byte *HByte, byte *LByte)
{
  int data;
  if(!(k%2))
    data = prozor[k/2].sensTemp;
  else
    data = prozor[k/2].sensLux;
  *HByte = (byte)(data >> 8 & 0xff);
  *LByte = (byte)(data & 0xff);
}

int status(int p)
{
 int sensOpen = prozor[p].sensOpen;
 int sensClose = prozor[p].sensClose;
  if  (sensOpen == 1) return 1;
  else if (sensClose == 1) return 0;
  else return -1;
}
```

Shema spajanja senzora i motora na Arduino




**Slika 8: Shema spajanja senzora i motora**