Ante Grgat
0036460770

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

Automatizacija zastora

- Rad u sklopu projekta „Pametna kuća"
- Namijenjen svima zainteresiranima
- Osnove programiranja
- Programiranje Arduina i spajanje komponenti

16. lipnja 2016

Sažetak

Automatizirani zastori omogućuju korisnicima stambenog prostora ugodniji boravak i uštedu energije. Razvijeni sustav omogućuje automatsko otvaranje i zatvaranje zastora obzirom na detekciju pokreta, količinu upadnog svjetla i vanjsku temperaturu. Prednosti ovakvog sustava leže u niskoj cijeni izrade, niskoj potrošnji električne energije i predstavlja široko područje za daljnju nadogradnju.

Sadržaj

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

1. Uvod

Zastori igraju značajnu ulogu u komfornosti i privatnosti doma. Automatizirani zastori, uz stvaranje ugodnog ambijenta i osvjetljenja, omogućuju postizanje ugodne temperature unutar stambenog prostora na prirodan način. Naime, povećanjem vanjske temperature, zastori se automatski spuštaju čime se onemogućava povećanje temperature unutarnjeg prostora; dok se smanjenjem temperature zastori ponovno podižu.

Korisniku su omogućena dva načina rada sustava. U prvom se načinu rada detektira količina upadnog svjetla te se na temelju dobivenih rezultata upravlja servo motorom koji potom otvara ili zatvara zastore.

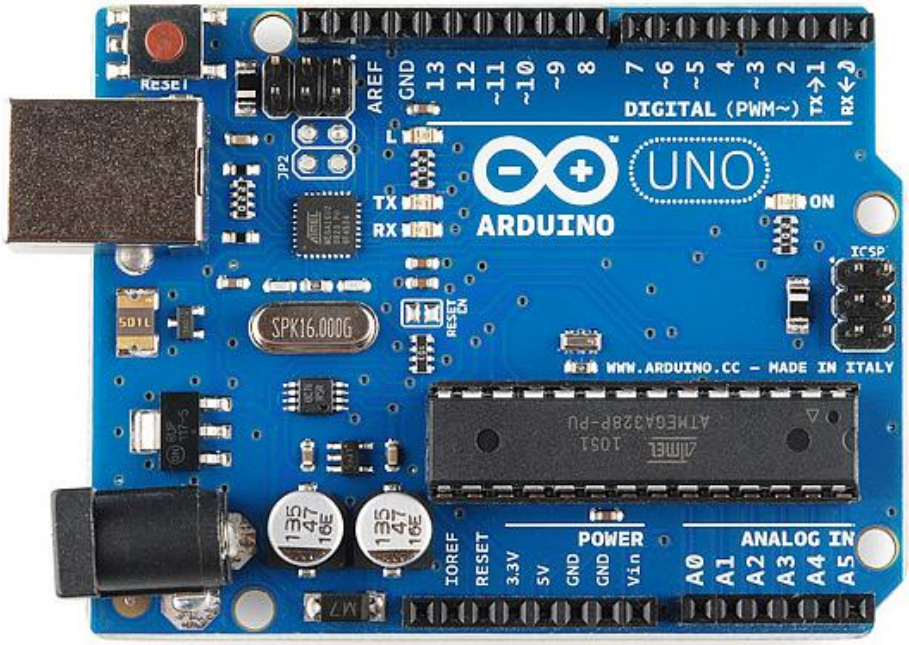
Korištenjem takvog sustava koji detektira količinu upadnog zračenja te na istu pravodobno reagira, smanjuje se utjecaj štetnih UVA sunčevih zraka. Nadalje, za razliku od rashladnih klima uređaja, automatizirani zastori predstavljaju zdravstveno bolje i ekonomski isplativije rješenje.

U drugom načinu rada se pasivnim infracrvenim senzorom detektira prisustvo osobe te se na temelju toga donosi odluka o otvaranju ili zatvaranju zastora. Prednost u korištenju PIR senzora leži u činjenici da se isti može koristi u raznim dijelovima „pametne kuće" – primjerice u alarmnim sustavima.

2. Opis sustava

2.1. Arduino UNO

Arduino je elektronička prototipna platforma namijenjena kreiranju elektroničkih projekata. Sastoji se od hardware dijela koji je zapravo fizički elektronički programabilni strujni krug i software dijela koji se naziva IDE – Integrated Development Environment. Namijenjen je dizajnerima, elektroničarima i svima koji su zainteresirani za stvaranje interaktivnih objekata i okruženja.

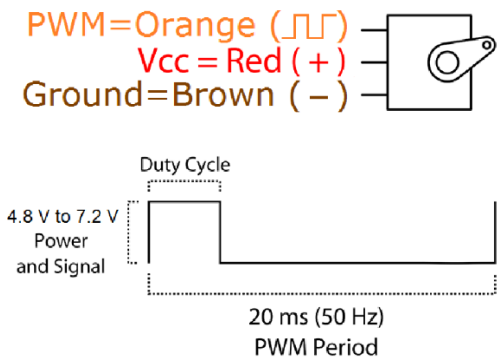


Slika 1. Arduino UNO

2.2. Servo motor

Servo motor je pomoćni motor s električnim, pneumatskim ili hidrauličkim pogonom koji se automatski uključuje u procese upravljanja i reguliranja. U ovom projektu korišten je servo motor oznake Tower Pro MG995. Servo motorom se upravlja PWM (eng. Pulse Width Modulation) signala. Modulacija širine impulsa PWM je postupak kojim se analogni signal pretvara u digitalni. Napon, odnosno struja se dovodi kao niz impulsa, a informacija o amplitudi analognog signala predstavlja se širinom (trajanjem) impulsa PWM signala.

Slika 2. TowerPro MG995



Slika 3. Pulse Width Modulation

2.3. Fotootpornik

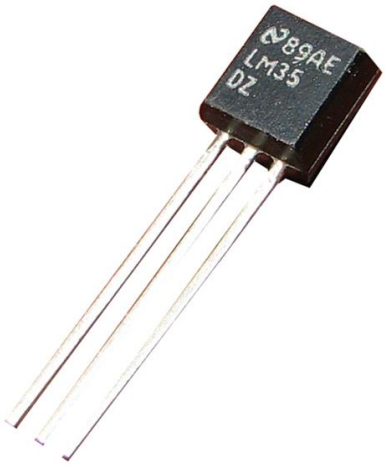
Fotootpornik (eng. Photoresistor ili light dependent resistor – LDR) je otpornik čiji se električni otpor smanjuje s povećanjem inteziteta ulazne svjetlosti.

Fotootpornik se izrađuje od poluvodiča s velikim električnim otporom. Ako svjetlo padne na fotootpornik, sa dovoljno velikom frekvencijom (granična frekvencija), poluvodič će upiti fotone svjetlosti i izbaciti elektrone, koji stvaraju električnu struju u zatvorenom strujnom krugu. Fotootpornici se spajaju u seriju s otpornikom iznosa 10 kΩ.

Slika 4. Fotootpornik

2.4 Temperaturni senzor

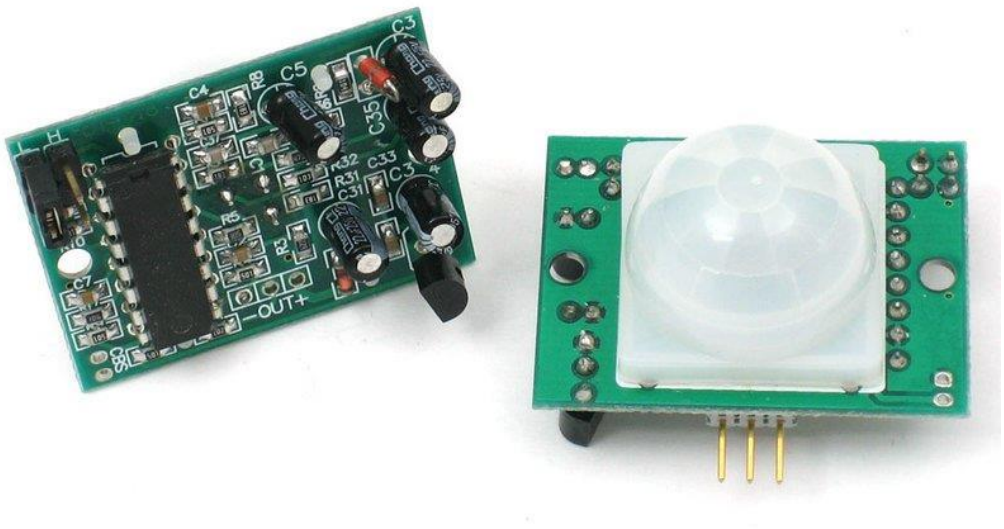
Temperaturni senzori temeljeni na osnovama promjene električnog otpora zovu se otpornički temperaturni senzori (eng. Resistance temperature detectors – RTD). U realizaciji sustava korišten je LM35 temperaturni otpornik čiji je izlazni napon proporcionalan Celzijevoj temperaturi. LM35 temperaturni senzor odlikuju dobra linearna svojstva te mu prednost leži u činjenici da mu je za ispravan rad potrebna mala struja.



Slika 5. LM35 temperaturni senzor

2.4. PIR senzor

PIR (eng. Passive InfraRed) je elektronički senzor koji mjeri infracrveno zračenje sa objekata u njegovom vidnom polju. Osnovni dio PIR senzora je piroelektrični element koji omogućuje detekciju infracrvenog zračenja. Princip rada temelji se na detekciji promjene primljenog infracrvenog zračenja na jednom od dva dijela koja čine piroelektrični element. Ukoliko jedna od njih primi više zračenja od druge, senzor će reagirati na tu promjenu. PIR senzori su uglavnom kompaktni i mogu se uklopiti u gotovo bilo koji elektronski uređaj . Ne trebaju vanjski izvor napajanja i mogu mjeriti vrijednosti temperatura neovisno o udaljenosti tijela kojeg se mjeri.



Slika 6. PIR senzor

3. Programska realizacija sustava

```
# include <Servo.h>

Servo myservo; // create a Servo object

// configuration for Pin

int servoPin = 9; // input pin for Servo
int photocellPin = 0; // input pin for photocell
int buttonPin = 2; // input pin for button (manual control)
int temperaturePin = 5; // input pin for temperature LM35 sensor
int pirPin = 7; // input pin for PIR sensor
int LEDPin = 13; // input pin for LED

// configuration for PIR sensor

int calibrationTime = 10; // time given for sensor calibration
long unsigned int pause = 5000; // amount of milliseconds the
sensor has to be LOW before we assume all motion has stopped
long unsigned int lowIn; // the time when the sensor outputs a low
impulse
boolean lockLow = true;
boolean takeLowTime;

// configuration for LM35 temperature sensor and Photocell sensor

int temperatureReading; // raw analog temperature reading
float temperatureC; // calibrated temperature reading
int photocellReading;
int minLight;
int maxLight;
int adjustedLightLevel; // calibrated photocell light reading

// configuration for Servo motor

int pos = 0; // Servo position
int spd = 100; // Servo speed
int state =0; // Keep track of state
int previous_state = 0; // Prior Servo motor state;
int dest = 0; // Servo destination
```

```
// configuration for button ( PIR sensor Servo motor control )

int buttonPushCounter = 0; // counter for the number of button
presses
int buttonState = 0; // current state of the button
int lastButtonState = 0; // previous state of the button
long time = 0;
long debounce = 200;
int stateLED = 0;

void setup(){

  // attach Servo to pin 9

  myservo.attach(servoPin);

  // begin serial communication - Baud rate = 9600

  Serial.begin(9600);

  // initialize the button pin as a INPUT

  pinMode(pirPin, INPUT);

  // initialize the LED as an OUTPUT

  pinMode(LEDPin, OUTPUT);

  // initialize Photocell as an INPUT

  pinMode(photocellPin, INPUT);

  // initialize Temperature LM35 sensor

  pinMode(temperaturePin, INPUT);

  // Setup the starting light level limits
  photocellReading = analogRead(photocellPin);
  minLight = photocellReading - 20;
  maxLight = photocellReading;
```

```
// give PIR sensor time to calibrate

  Serial.println("Calibrating PIR sensor");
  for(int i = 0;i < calibrationTime; i++){
    Serial.print(".");
    delay(1000);
  }
  Serial.print("\n");
  Serial.println("Done");

  // wait until the PIR`s output is low before ending setup

  while(digitalRead(pirPin)==HIGH){
    delay(500);
  }
  Serial.print("\nSensor Active");
}

void loop(){

  // read the state of the button pin

  buttonState = digitalRead(buttonPin);

  // turn on a LED when the button is pressed and let it on until
the button is pressed again

  if(buttonState == HIGH && lastButtonState == LOW && millis() -
time > debounce){
    if(stateLED == HIGH){
      buttonPushCounter++;
      stateLED = LOW;
      Serial.println("\nButton OFF");
      Serial.print("\n Number of button pushes :");
      Serial.println(buttonPushCounter);
    } else {
      stateLED = 1;
      Serial.println("\nButton ON");
    }

    time = millis();
  }

  digitalWrite(LEDPin, stateLED);
  lastButtonState = buttonState;
```

```
// if LED == HIGH enable Servo motor control via PIR sensor

  if(stateLED==1){

    // if PIR output is HIGH, turn Servo ON

  if(digitalRead(pirPin)==HIGH){

    Serial.println("PIR Sensor Servo Motor Control");

    // read raw data from temperature LM35 sensor

    temperatureReading = analogRead(temperaturePin);

    // calibrate the raw temperature read data

    temperatureC = (5.0 * temperatureReading * 100) / 1024;
    Serial.print("\nCalibrated Temperature reading :");
    Serial.print(temperatureC);
    delay(500);

    // opening and closing blinds depending on the temperature

    if(temperatureC > 30){
      Serial.print("\n High temperature - Closing blinds");
      dest = 180;
      state = 5;
    } else if(temperatureC > 22 && temperatureC < 30){
      Serial.print("\n Medium temperature - Blinds Half Closed");
      dest = 360;
      state = 6;

    } else if(temperatureC < 22){
      Serial.print("\n Low temperature - Open Blinds");
      dest = 0;
      state = 7;
    }
```

```
// make sure we wait for a transition to LOW before further output is
made

    if(lockLow){
      lockLow=false;
      Serial.println("---");
      Serial.println("\nMotion detected at ");
      Serial.print(millis()/1000);
      Serial.println(" sec");
      delay(50);
    }
    takeLowTime = true;
  }

  if(digitalRead(pirPin)==LOW){

    if(takeLowTime){
      lowIn = millis(); // save the time of the transition from HIGH
to LOW
      takeLowTime = false; // make sure this is only done at the
start of a LOW

    }

    // if the sensor is low for more than the given pause, we can
assume the motion has stopped

    if(!lockLow && millis() - lowIn > pause){

      // make sure this block is only executed again after a new
motion sequence has been detected

      lockLow = true;
      Serial.print("\nMotion ended at ");
      Serial.print((millis()-pause)/1000);
      Serial.println(" sec");
      delay(50);
    }
  }

  if(state != previous_state){
    motor_control();
  }

  previous_state = state;
  delay(5000);
```

```
  } else {
    photocellReading = analogRead(photocellPin);

    if(minLight > photocellReading){
      minLight = photocellReading;
    }
    if(maxLight < photocellReading){
      maxLight = photocellReading;
    }

    adjustedLightLevel = map(photocellReading, minLight, maxLight,
0, 100);
    Serial.print("\n Light reading:");
    Serial.println(adjustedLightLevel);
    Serial.print("Position :");
    Serial.println(pos);
    Serial.print("State :");
    Serial.println(state);
    if(adjustedLightLevel > 0 && adjustedLightLevel < 10){
      Serial.println("Night");
      dest = 180;
      state = 1;
    } else if(adjustedLightLevel > 10 && adjustedLightLevel < 30){
      Serial.println("Dusk");
      dest = 135;
      state = 2;
    } else if(adjustedLightLevel > 30 && adjustedLightLevel < 84){
      Serial.println("Day");
      dest = 85;
      state = 3;
    }  else  if(adjustedLightLevel  >  84  &&  adjustedLightLevel  <
100){
      Serial.println("Very bright day");
      dest = 20;
      state = 4;
    } else {
      Serial.println("Readings out of range");
    }
    if( state != previous_state){
      motor_control();
    }
    previous_state = state;
    delay(5000);
  }


}
```

```
void motor_control(void) {
      Serial.println("\nState Change");
    //Connect to servo
    myservo.attach(servoPin);
    // If the current position is great than the destination then
we must subtract
    if (pos > dest){
       // Change current position to desired position, one degree
at a time.
         while (pos > dest)
    {
      // tell servo to go to position in variable 'pos'

      myservo.write(pos);
      // waits desired time for the servo to reach the position

      delay(spd);

      pos--;

    }
    //Detach from Servo

    myservo.detach();
  }
  else {
    // If the current position is not greater than the destination
then we must add

    myservo.attach(9);

    // goes from 180 degrees to 0 degrees

    while (pos < dest)
    {

      myservo.write(pos);
      delay(spd);
      pos++;

    }
  }
  myservo.write(pos);
  delay(spd);



  myservo.detach();
}
```

4. Zaključak

Sustav je realiziran korištenjem senzora čijom logikom upravlja Arduino mikrokontroler. Korisniku su omogućena dva načina rada sustava. Kada sustav radi u prvom načinu rada, mjeri se količina upadnog svjetla te se na temelju dobivenih mjerenja kontrolira rad izvršnog člana (servo motora) . U drugom načinu rada, detekcijom prisutnosti (PIR senzorom) osobe, aktivira se mjerenje temperature na temelju kojeg upravljamo radom servo motora; odnosno podižu se ili spuštaju zastori. Sustav bi se mogao povezati i sa ostalim dijelovima „pametne kuće" gdje bi najveću primjenu nalazio kod alarmnih sustava zbog ugrađenog senzora detekcije pokreta.
Realizirani sustav, pokazao se kao ne samo estetski ugodno, već i ekonomski isplativo te energetski učinkovito rješenje.

5. Literatura

[1] https://www.arduino.cc/

[2] https://learn.adafruit.com/photocells/using-a-photocell

[3] https://www.arduino.cc/en/Reference/Servo

[4] http://www.electronicoscaldas.com/datasheet/MG995_Tower-Pro.pdf

[5] http://www.ti.com/lit/ds/symlink/lm35.pdf

[6] http://www.ladyada.net/media/sensors/PIRSensor-V1.2.pdf

6. Pojmovnik

Pojam	Kratko objašnjenje	Više informacija potražite na
Arduino UNO	elektronička prototipna platforma namijenjena kreiranju elektroničkih projekata	https://www.arduino.cc/en/Main/ArduinoBoardUno
PIR senzor	elektronički senzor koji mjeri infracrveno zračenje sa objekata u njegovom vidnom polju	https://en.wikipedia.org/wiki/Passive_infrared_sensor
Servomotor	pomoćni motor s električnim, pneumatskim ili hidrauličkim pogonom	https://en.wikipedia.org/wiki/Servomotor
Fotootpornik	otpornik čiji se električni otpor smanjuje s povećanjem inteziteta ulazne svjetlosti	https://en.wikipedia.org/wiki/Photoresistor
Otpornički temperaturni senzor	Otpornici čiji se otpor mijenja s temperaturom	http://www.electrical4u.com/resistance-temperature-detector-or-rtd-construction-and-working-principle/