

UNIVERZITET U TUZLI  
FAKULTET ELEKTROTEHNIKE

---

SMART LIGHTING SYSTEM  
SATELITSKE TELEKOMUNIKACIJE

STUDENT

AMELA KULIĆ

TUZLA, JUN, 2023. GODINE

---

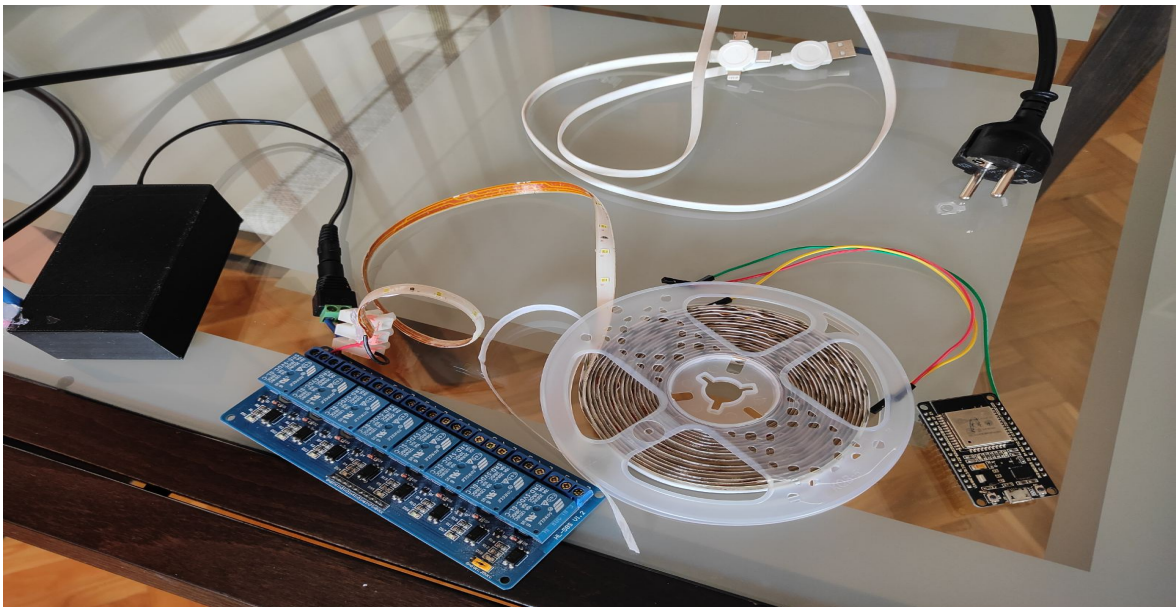
Pametna rasvjeta u kući se odnosi na upotrebu tehnologije i automatizacije kako bi se osvjetljenje u kući kontrolisalo na inteligentan način. Umjesto tradicionalnih prekidača i svjetiljki, koriste se pametne svjetiljke, senzori pokreta, daljinske upravljači ili aplikacije za pametne telefone kako bi se prilagodilo osvjetljenje prema potrebama korisnika. Evo nekoliko prednosti i mogućnosti koje pametna rasvjeta u kući može pružiti:

- Prilagodljivost
- Automatizacija
- Daljinsko upravljanje
- Integracija sa drugim pametnim uređajima

## Daljinsko upravljanje

Kroz ovaj izvještaj upoznat ćete se sa tehnikama izrade pametne rasvjete u smislu uključivanja i isključivanja rasvjete i u slučajevima kada niste prisutni u prostoriji. Za ovaj projekat korištena je sljedeća oprema:

- ESP32 mikrokontroler
- Relay modul
- Jumperi
- LED traka
- USB kablo
- 220 V to 12 V converter



ESP32 je popularni mikrokontroler koji se koristi za razvoj IoT (Internet of Things) projekata. IoT (Internet of Things) projekti su projekti koji uključuju povezivanje fizičkih uređaja s internetom kako bi se omogućila razmjena podataka, upravljanje i nadzor tih uređaja putem mreže. U ovom slučaju pokazat ćemo kako postaviti ESP32 kao pristupnu tačku koristeći Arduino IDE. U nastavku se nalazi objašnjenje koda u Arduino IDE alatu pomoću kojeg se vrši instalacija ESP32 mikrokontrolera kao pristupne tačke.

```
sketch_jun16a | Arduino IDE 2.1.0
File Edit Sketch Tools Help

sketch_jun16a.ino
1 #include <WiFi.h> // Učitavanje WiFi biblioteke
2
3 const char* ssid = "SatKom";
4 const char* password = "123456789";
5
6 // Port 80 je standardni HTTP port koji se često koristi za komunikaciju između web servera i klijenata.
7 WiFiServer server(80);
8
9 // Varijabla za storiranje HTTP zahtjeva
10 String header;
11 String output26State = "off";
12 String output27State = "off";
13
14 // GPIO pinovi koji pokreću blinkanje LED lampi
15 const int output26 = 26;
16 const int output27 = 27;
17
18 void setup() {
19   Serial.begin(115200); //inicijalizacija serijske komunikacije između mikrokontrolera i računara. Podešava se brzina prenosa podataka na 115200 bps.
20   // postavljanje izlaznih pinova
21   pinMode(output26, OUTPUT);
22   pinMode(output27, OUTPUT);
23   // isključivanje povezanih uređaja
24   digitalWrite(output26, LOW);
25   digitalWrite(output27, LOW);
26 }
```

```
sketch_jun16a | Arduino IDE 2.1.0
File Edit Sketch Tools Help

sketch_jun16a.ino
25 digitalWrite(output27, LOW);
26
27 // konfiguracija uređaja kao pristupne tačke za wifi mrežu.
28 Serial.print("Setting AP (Access Point)...");
29
30 WiFi.softAP(ssid, password);
31
32 IPAddress IP = WiFi.softAPIP(); //IP adresu pristupne tačke smješta se u promenljivu IP
33 Serial.print("AP IP address: ");
34 Serial.println(IP);
35
36 server.begin(); // ova linija koda pokreće server za WiFi komunikaciju
37
38
39 void loop(){
40   WiFiClient client = server.available(); //provjerava da li postoji dolazna konekcija klijenta na serveru, ako postoji objekat client nije prazan
41
42   if (client) {
43     Serial.println("New Client."); // ispisuje se poruka na serijskom portu da je klijent povezan sa serverom
44     String currentLine = ""; // String za pohranu vrijednosti korisnika
45     while (client.connected()) { // obrada podataka koji stižu od klijenta i petlja traje sve dok je klijent povezan
46       if (client.available()) { // provjerava da li postoje dostupni podaci za čitanje od klijenta i metoda client.available() vraća broj dos
47         char c = client.read(); // čita se jedan bajt podataka iz klijenta i smješta u varijablu c
48         Serial.write(c); // ispisivanje na serijski port
49         header += c; // dodavanje podataka u string
50         if (c == '\n') { // u slučaju newline karaktera, znači da je završen redak HTTP zahteva koji je poslao klijent
```

```
sketch_jun16a | Arduino IDE 2.1.0
File Edit Sketch Tools Help

sketch_jun16a.ino
49 header += c; // dodavanje podataka u string
50 if (c == '\n') { // u slučaju newline karaktera, znači da je završen redak HTTP zahteva koji je poslao klijent
51
52   if (currentLine.length() == 0) {
53     // Ova linija koda provjerava da li je currentLine string prazan
54     // ako jest to znači da je prethodni redak bio prazan i da je to kraj HTTP zahteva te server šalje odgovor klijentu
55
56     client.println("HTTP/1.1 200 OK"); //HTTP statusni odgovor "200 OK" šalje se klijentu što znači da je zahtjev uspješno primljen i obraden
57     client.println("Content-type:text/html"); // šalje se HTTP zaglavlje koje definiše tip sadržaja kao "text/html" što znači da će se odgovor
58     client.println("Connection: close"); //zatvaranje konekcije nakon slanja odgovora
59     client.println(); //prazan red koji označava kraj HTTP zaglavlja
60
61     // paljenje i gasenje GPIO pinova
62     // provjerava se sadržaj header stringa kako bi se identifikovalo koji zahtjev je poslao klijent
63     if (header.indexOf("GET /26/on") >= 0) {
64       Serial.println("GPIO 26 on");
65       output26State = "on";
66       digitalWrite(output26, HIGH);
67     } else if (header.indexOf("GET /26/off") >= 0) {
68       Serial.println("GPIO 26 off");
69       output26State = "off";
70       digitalWrite(output26, LOW);
71     } else if (header.indexOf("GET /27/on") >= 0) {
72       Serial.println("GPIO 27 on");
73       output27State = "on";
74       digitalWrite(output27, HIGH);
```

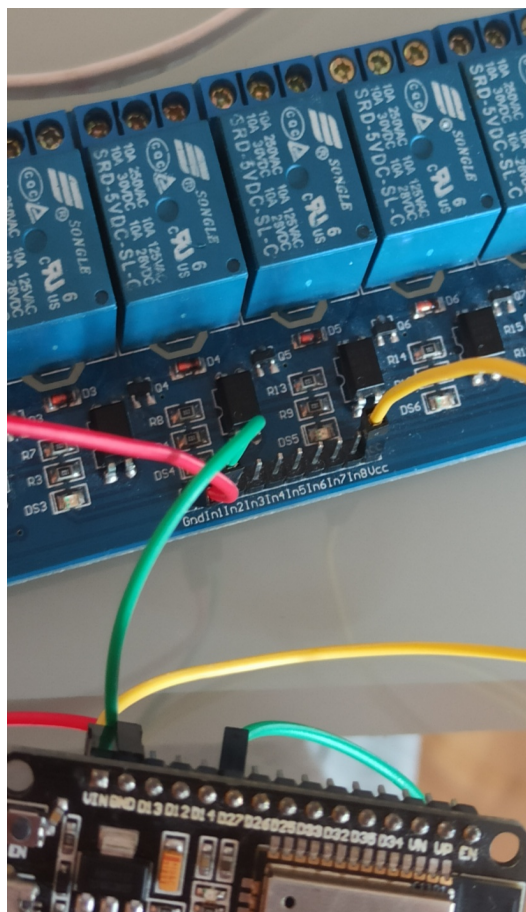
- Znači, pristupna tačka (AP) omogućava ESP32 mikrokontroleru da se ponaša kao samostalna WiFi mreža. To znači da uređaji mogu povezati s ESP32 direktno, bez potrebe za postojećom WiFi mrežom.
- Korištenje ESPAsyncWebServer biblioteke omogućava brzo i jednostavno postavljanje web servera na ESP32 mikrokontroleru. Ova biblioteka podržava asinhroni rad, što omogućava obradu više zahtjeva istovremeno.
- Uz pomoć HTML, CSS i JavaScript koda, kreirali smo vlastitu web stranicu koja se prikazuje kada korisnik poveže s ESP32 pristupnom tačkom.

U zaključku, postavljanje ESP32 mikrokontrolera kao pristupne tačke i web servera pruža nam mogućnost da kreiramo samostalne WiFi mreže i kontroliramo uređaje putem web interfejsa.

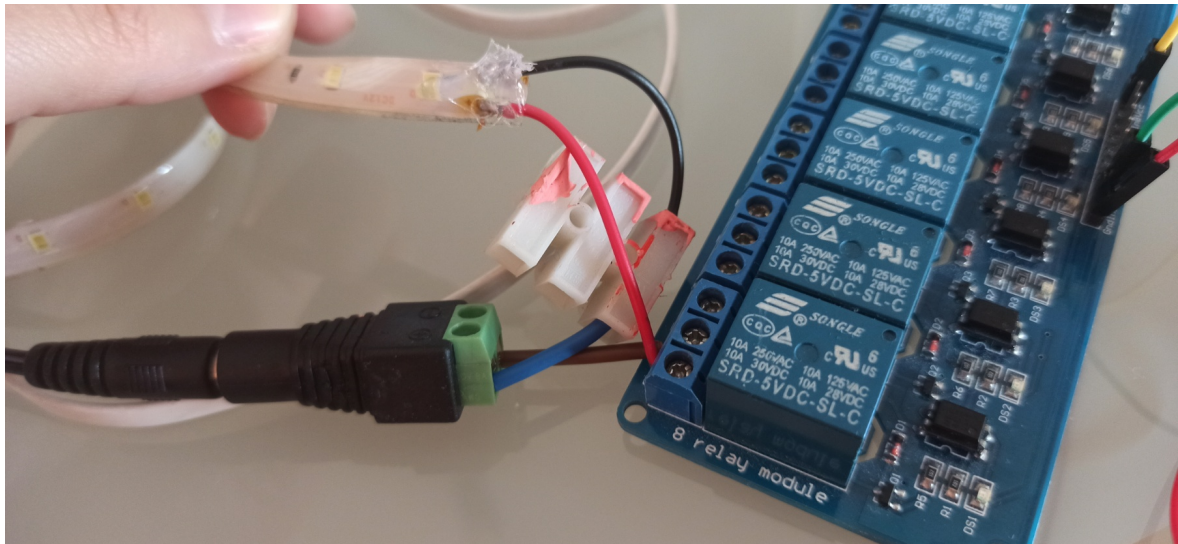
Nakon instaliranja ESP32 ploče u Arduino IDE, izvršeno je spajanje mikrokontrolera sa relay modulom. Korištena su 3 jumpera i spajanje je izvršeno na sljedeći način:

- GND mikrokontrolera spaja se za GND relaya
- Vin pin mikrokontrolera spaja se na Vcc pin relaya
- GPIO26/27 pin mikrokontrolera spaja se na input1 relaya

Prethodno opisani postupak prikazan je na sljedećoj slici:



Nakon toga se mikrokontroler spaja na izvor napajanja preko USB kablova. Zalemili smo vodiče na izlazne pinove LED trake i povezali ih na relay modul. Kada povežemo LED traku sa relay modulom, relay modul će djelovati kao prekidač za napajanje LED trake. Relay modul ima ulazni pin koji je povezan sa mikrokontrolerom ESP32, a izlazni pin koji je povezan sa napajanjem LED trake. Također, LED traka se priključuje na izvor napajanja preko 220V to 12V konvertera.



Kao što je navedeno na slikama koje objašnjavaju kod, postavlja se naziv mreže u varijablu "ssid". SSID (Service Set Identifier) predstavlja ime Wi-Fi mreže koja se koristi za identifikaciju i pristup mreži. U kontekstu priloženog koda, SSID je varijabla koja sadrži naziv pristupne tačke koju ESP32 mikrokontroler stvara.

Zatim moramo dobiti IP adresu pristupne tačke pomoću softAPIP() metoda.

WiFi.softAP(ssid, password); je funkcija u ESP32 Arduino biblioteci koja se koristi za postavljanje ESP32 mikrokontrolera kao pristupne tačke (Access Point - AP) u Wi-Fi mreži. Ova funkcija omogućava ESP32 da kreira vlastitu Wi-Fi mrežu na koju se drugi uređaji mogu povezati. Argumenti ssid i password predstavljaju naziv i lozinku mreže koju će ESP32 kreirati.

```
IPAddress IP = WiFi.softAPIP();  
Serial.print("AP IP address: ");  
Serial.println(IP);
```

Nakon izvršenja ovog dijela koda, na serijskom monitoru će se ispisati IP adresa pristupne tačke.

```
COM4
ets Jun  8 2016 00:22:57

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:812
load:0x40078000,len:0
load:0x40078000,len:11392
entry 0x40078a9c
Setting AP (Access Point)...AP IP address: 192.168.4.1
```

Nakon povezivanja na WiFi mrežu koju smo napravili, u web browseru kucamo IP adresu pristupne tačke koju stvara ESP32 mikrokontroler. U ovom slučaju to je 192.168.4.1. Pri-  
kazuje nam se izgled aplikacije definisane pomoću HTML, CSS i JavaScript-a. Ovisno o  
tome koji pin mikrokontrolera je spojen na relay, vrši se paljenje i gašenje rasvjete. U našem  
slučaju, u kodu smo tu mogućnost definisali za pinove GPIO26 i GPIO27. Izgled aplika-  
cije prikazana je na sljedećoj slici, a sama funkcionalnost će biti predstavljena na odbrani  
završnog projekta.

