

**Zveza za tehnično kulturo Slovenije**  
**59. srečanje mladih raziskovalcev SLOVENIJE 2025**

**Večnamenska krmilna elektronika za pametno hišo**

Raziskovalno področje ZOTKS: **elektrotehnika, elektronika in  
robotika**

**Avtor: Dejan Dolanc**  
**Mentor: Branko Potisk**  
**Somentor: Mladen Pintar**  
**Srednja šola: Srednja elektro-računalniška šola Maribor**

**Maribor, 2025**

## VSEBINA

Vsebina.....	I
Kazalo tabel:.....	I
Kazalo slik:.....	II
Povzetek:.....	1
Zahvala:.....	2
1 Uvod:.....	3
1.1. Cilj.....	3
1.2. Raziskovalno vprašanje:.....	3
1.3. Hipoteze:.....	3
2 Metodologija dela:.....	4
3 IZBIRA NAČINA KRMILJENJA PAMETNE HIŠE.....	4
4 IZBIRA Wi-fi vmesnika.....	4
5 PREIZKUŠANJE Wi-Fi modula.....	7
6 ZAČETEK IZDELAVE VEČNAMENSKE KRMILE ELEKTRONIKE.....	11
6.1. Načrtovanje in izdelava tiskanine:.....	11
6.2. Načrtovanje in izdelava ohišja:.....	12
6.3. Načrtovanje in izdelava Web-Aplikacije:.....	15
6.4. Pisanje programa za krmiljenje vhodov in izhodov ter povezavo med spletno aplikacijo in krmilno elektroniko:.....	19
7 Princip delovanja:.....	21
8 družbena odgovornost:.....	24
9 zaključek:.....	25
10 viri in literatura:.....	26
10.1. Spletni viri:.....	26
10.2. Knjižni viri:.....	26

## KAZALO TABEL:

Tabela 1:Izbira Wi-Fi vmesnika oziroma Wi-Fi mikrokontrolerja.....	5
Tabela 2:Izbira med ESP32 Wi-Fi mikrokontrolerji.....	5

## KAZALO SLIK:

Slika 1: Primerjava ESP32 mikrokontrolerjev .....	6
Slika 2: Primerjava ESP32 mikrokontrolerjev .....	6
Slika 3: Shematski načrt iz spletne strani tiskanini.....	7
Slika 4: Vezava na preskusni Slika 5: Shematski načrt na shematski sliki .....	8
Slika 6: Ime Wi-Fi povezave      Slika 7: Vpis gesla za Wi-Fi povezavo .....	8
Slika 8: Program na spletni strani .....	9
Slika 9: Delovanje povezave programa s našo preskusno tiskanino .....	10
Slika 10: Sprememba SSID .....	10
Slika 11: Izgled narisan tiskanine      Slika 12: Izgled tiskanine v 3D obliki.....	11
Slika 13: Izgled sestavljene tiskanine.....	12
Slika 14: Spodnji del ohišja narisan v Fusion 360 .....	12
Slika 15: Pokrov ohišja narisan v Fusion 360 .....	13
Slika 16: 3D natisnjen pokrov .....	13
Slika 17: 3D natisnjen pokrov z vidno luknjo za konektorje .....	13
Slika 18: Pri vijáčena krmilna elektronika na dno ohišja.....	14
Slika 19: Pri vijáčena krmilna elektronika na dno ohišja.....	14
Slika 20: Končano ohišje.....	14
Slika 21: Končano ohišje pogled z vrha.....	15
Slika 22: Programska koda za obliko spletne aplikacije .....	16
Slika 23: Programska koda za obliko spletne aplikacije .....	17
Slika 24: Začetna stran naše spletne aplikacije .....	17
Slika 25: Izbira sobe, ki jo želimo krmiliti.....	18
Slika 26: Krmiljenje posameznih naprav v sobi.....	18
Slika 27: Program za delovanje naše krmilne elektronike .....	19
Slika 28: Program za delovanje naše krmilne elektronike .....	20
Slika 29: Primer komunikacije med strežnikom in odjemalcem po standardu HTTP .....	21
Slika 30: Primer standarda Server-Sent Events (SSE) .....	21
Slika 31: Prvi primer komunikacije med odjemalcem in strežnikom .....	22
Slika 32: Prvi test prižiga led luči preko mobilne naprave .....	22
Slika 33: Access point način delovanja.....	23

## **POVZETEK:**

V raziskovalni nalogi bomo spoznali Wi-Fi mikrokontrolerje in krmiljenje preko njih. V raziskovalni nalogi bom raziskal kako uporabiti Wi-Fi mikrokontroler kot vmesnik za krmiljenje pametne hiše. V raziskovalni nalogi je namen izdelati mikrokrmilnik z Wi-Fi vmesnikom, ki ga upravljamo preko mobilnega telefona ali tablice. S pomočjo mobilne naprave bomo tako lahko krmili različne naprave kot so luči, ventilatorji, itd. Raziskali bomo možnost vgradnje takšnega krmilnika, hkrati pa raziskal ali obstaja Wi-Fi vmesnik, ki bi deloval, kot strežnik. Vse skupaj bomo vstavili v lepo plastično ohišje.

**Ključne besede:** modernizacija hiš, krmilna elektronika, spletni strežnik, Wi-Fi komunikacija, mikrokrmilnik.

## **ZAHVALA:**

Iskreno bi se rad zahvalil obema mentorjema za opravljanje mentorskega dela in vse pomoči, ki sta mi ju nudila. Predvsem pri oblikovanju raziskovalne naloge in pri izdelavi izdelka ter pri finančni podpori in da sta mi omogočila delo z vsemi stroji in programske opremo, ki sem jih potreboval za izdelavo te raziskovalne naloge.

# **1 UVOD:**

## **1.1. Cilj**

Cilj raziskovalne naloge je izdelati krmilno elektroniko podprto z Wi-Fi ali BLE za moderniziranje hiš. Krmilna elektronika bo vgrajena v elektro omaro kjer so varovalke. Krmilna elektronika bo imela vgrajen Wi-Fi vmesnik, ki bo omogočal upravljanje naprav preko tablice ali mobilnega telefona. Naš cilj je najti mikrokrmilnik, ki bi omogočal Wi-Fi komunikacijo in bi deloval samostojno kot spletni strežnik, hkrati pa bi nam omogočal krmiliti izhodne in brati vhode. Vhode bi uporabili na primer za vhodne veličine kot so končna stikala, senzor svetlobe ali senzor gibanja itd. Izhode pa bi uporabili za krmiljenje relejev na katere, lahko priključimo npr. luči, žaluzije, ogrevanje, hlajenje itd. Krmilna elektronika bi sprejemala preko Wi-Fi komande z mobilne naprave ter nato krmilila izhode glede na prebrane vhodne signale. Program krmilja pa bi napisali sami. V kolikor bi našel Wi-Fi mikrokontroler, ki omogoča upravljanje tudi spletne strani bi nam to omogočilo nalaganje aplikacije na sam mikrokrmilnik in nebi potrebovali aplikacijskega programa na mobilni napravi. Na mobilni napravi bi potrebovali le brskalnik. Cilj je programiranje Wi-Fi mikrokrmilnika je v jezikih HTML, CSS, JAVA in C ter C++. Te programske jezike moramo dodobra spoznati.

## **1.2. Raziskovalno vprašanje:**

Ali lahko izdelam krmilnik za moderniziranje zgradb, ki omogoča Wi-Fi krmiljenje preko mobilne naprave ter enostavno montažo v elektro omaro ter upravljanje preko brskalnika na mobilni napravi.

## **1.3. Hipoteze:**

Prva hipoteza je, da bomo lahko modernizirali starejše že obstoječe hiše in na enostaven način opremili novogradnje z modernimi funkcijami.

Druga hipoteza je, da bo naš izdelek enostaven za vgradnjo v elektro omaro.

Tretja naša hipoteza je, da bo naš izdelek imel možnost krmiljenja brez aplikacijskega programa na mobilnih napravah, saj bo naš izdelek deloval kot strežnik na katerega se povežemo in preko njega zaženemo spletno aplikacijo za telefon.

Četrta hipoteza je, da bo naš izdelek enostaven za uporabo preko mobilne naprave kot je telefon in bo tako primerna tudi za starejše ljudi, ki si želijo modernizirati svojo hišo in olajšati krmiljenje hišnih naprav kot so električne žaluzije, luči, grelci, klima itd....

## **2 METODOLOGIJA DELA:**

V tem poglavju bi predstavili potek dela raziskovalne naloge. Najprej bomo raziskali na kakšen način bi krmilili naprave v zgradbah. Nato bi raziskali kateri Wi-Fi vmesnik bi uporabili za našo elektroniko. Po določitvi Wi-Fi mikrokontrolerja bi določili osnovne lastnosti krmilja kot so koliko relejskih izhodov in koliko digitalnih vhodov bo imel naš krmilnik. Raziskali bomo načine programiranja in programske jezike za omenjeni Wi-Fi mikrokontroler ter le tega sprogramirali. Na koncu bomo poiskali ohišje za našo elektroniko ali pa ga izdelali s 3D tiskalnikom.

## **3 IZBIRA NAČINA KRMILJENJA PAMETNE HIŠE**

Našo krmilno elektroniko smo nameravali pritrčiti v elektro omarico od koder bi se ta povezala z napravami v hiši. A sem prišel do problema, da imajo hiše le klasično inštalacijo. Na primer naša hiša, ki je novogradnja, hiša od mojega dedija, ki je bila zgrajena pred petdesetimi leti in vse ostale hiše, ki se nahajajo v moji okolici nimajo položenih komunikacijskih kablov, ki so potrebni pri pametni hiši ampak samo navadno inštalacijo. Brez komunikacijskih ali krmilnih kablov naš krmilnik ne more krmiliti naprav iz elektro omarice. Na primer, če želim vključiti luč v sobi in je krmilnik v elektro omarici moram imeti komunikacijski ali krmilni kabel do sobe oziroma do stikala luči, da bi lahko le to vključili.

Zato smo prišli do rešitve, da bi v vsak prostor vgradili po eno krmilno elektroniko, ki bi bila namenjena za krmiljenje naprav v tistem prostoru. Pri tem bi se izognili problemu komunikacijskih kablov, ki jih nimamo v hišah. V določenem prostoru bi dodali samo krmilne kable. Na primer našo napravo vgradimo zraven stikala za luč in namesto stikala sedaj vklaplja luč naš relejski izhod. Rele namesto stikala. S tem bi privarčevali, saj bi porabili manj kablov, če povežemo krmilnik z napravami v istem prostoru, kot, da bi na novo položili komunikacijske kable oz. vsak kabel povezali posebej od naprave v določenem prostoru vse do krmilnika, ki bi se nahajal v elektro omarici.

## **4 IZBIRA WI-FI VMESNIKA**

Pri izbiri Wi-Fi vmesnika sem izbiral med MICROCHIP Wi-Fi vmesniki, STM Wi-Fi vmesniki in ESP32 Wi-Fi vmesniki (Tabela 1). MICROCHIP Wi-Fi vmesniki so se mi zdeli preveč zakomplicirani za uporabo (Tabela 1). STM Wi-Fi vmesniki nudijo veliko opcij, vendar so prezahtevni za moje znanje programiranja (Tabela 1). Zato sem se odločil za ESP32 Wi-Fi vmesnike, ki nudijo brezplačno programsko orodje in zelo dobro tehnično podporo preko internetne strani (Tabela 1).

Iz serije ESP-32 sem se odločal med ESP32-S3 in ESP32-C3 (Tabela 1). Odločil sem se za ESP32-C3 saj je v primerjavi s ESP32-S3 cenovno ugodnejši, pri delovanju porablja manj energije in enostavnejši za uporabo oziroma primernejši za začetnike. Na sliki 1 in sliki 2 pa je prikaz celotne serije ESP32 mikrokontrolerjev in njihovih funkcij.

	LASTNOSTI	PRIMEREN/NEPRIMEREN
<b>MICROCHIP:</b> - PIC32MZ2051W - PIC32MZ1025W - WFI32E01 - WFI32E03	Prezahteven za začetnike! Preveč vhodno izhodnih pinov, preveč dodatnih funkcij, plačljivo programsko orodje.	Neprimeren!
<b>STM:</b> - STM32W	Nudi veliko opcij Prezahteven za začetnike	Neprimeren
<b>ESPRESSIF:</b> - ESP32 serija	Brezplačna programsko orodje Odlična tehnična podpora Enostaven za uporabo	Primeren

Tabela 1: Izbira Wi-Fi vmesnika oziroma Wi-Fi mikrokontrolerja

ESPRESSIF	LASTNOSTI	PRIMEREN/NEPRIMEREN
ESP32-S3	Prezahteven dvojni mikrokontroler Večja poraba energije Več spomina	Neprimeren
ESP32-C3	Primeren za začetnike Dobavljiv Manjša poraba enregije Manj spomina	Primeren

Tabela 2: Izbira med ESP32 Wi-Fi mikrokontrolerji

Comparison of ESP32 Series Microcontrollers

	ESP32	ESP32-S2	ESP32-S3	ESP32-C2
Release Date	2016	2020	2020	2022
Processor	Xtensa dual-core 32-bit LX6	Xtensa single-core 32-bit LX7	Xtensa dual-core 32-bit LX7	32-bit single-core RISC-V
Frequency	160/240 MHz	Up to 240 MHz	Up to 240 MHz	Up to 120 MHz
SRAM	520 KB	320 KB	512 KB	272 KB
ROM	448 KB	128 KB	384 KB	576 KB ★
Flash	Up to 4 MB	Up to 4 MB	Up to 8 MB ★	Up to 4 MB
Wi-Fi	802.11 b/g/n, 2.4 GHz	802.11 b/g/n, 2.4 GHz	802.11 b/g/n, 2.4 GHz	802.11 b/g/n, 2.4 GHz
Bluetooth	v4.2 BR/EDR and BLE	✗	v5.0 BLE	v5.0 BLE
Zigbee	✗	✗	✗	✗
GPIO	34	43	45 ★	14
ADC	Two 12-bit, 18 channels	Two 13-bit, 20 channels ★	Two 12-bit, 20 channels	One 12-bit, 5 channels
DAC	Two 8-bit channels ★	Two 8-bit channels ★	✗	✗
SPI	4	4	4	3
I2C	2	1	2	1
I2S	2 ★	1	2 ★	1
RMT	8 channels ★	4 channels	8 channels ★	✗
Touch Sensor	10	14 ★	14 ★	✗
Hall Sensor	✓ ★	✗	✗	✗
LCD Interface	✓	✓	✓	✗
Camera Interface	✓	✓	✓	✗
Deep Sleep	~10 µA	~20 µA	~10 µA	~5 µA ★
Size	5x5 mm or 6x6 mm	7x7 mm	7x7 mm	4x4 mm ★
Other Features	✗	USB OTG	USB OTG	✗

Slika 1: Primerjava ESP32 mikrokontrolerjev

Comparison of ESP32 Series Microcontrollers

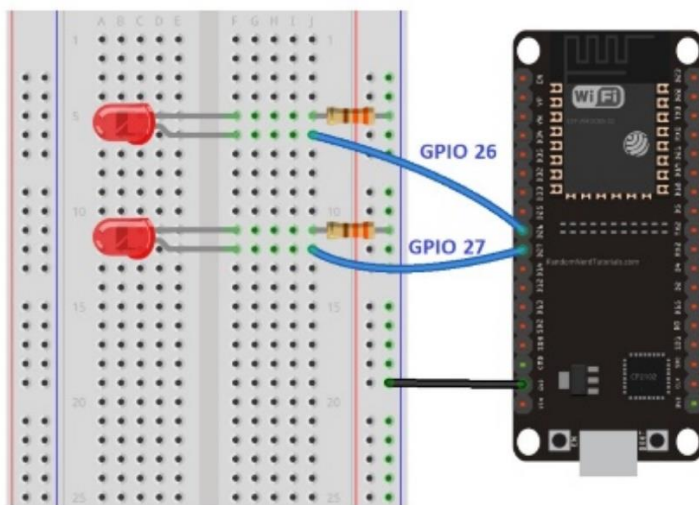
	ESP32-C3	ESP32-C5	ESP32-C6	ESP32-H2	ESP32-P4
Release Date	2020	2022	2021	2021	2023 ★
Processor	32-bit single-core RISC-V	32-bit single-core RISC-V	32-bit single-core RISC-V	32-bit single-core RISC-V	Dual-core 32-bit RISC-V ★
Frequency	Up to 160 MHz	Up to 240 MHz	Up to 160 MHz	Up to 96 MHz	Up to 400 MHz (main), 40 MHz (low-power core) ★
SRAM	400 KB	400 KB	512 KB	256 KB	768 KB ★
ROM	384 KB	384 KB	320 KB	128 KB	8 KB TCM
Flash	Up to 4 MB	Up to 4 MB	Up to 4 MB	External	External
Wi-Fi	802.11 b/g/n, 2.4 GHz	802.11 ax, 2.4/5 GHz ★	802.11 ax, 2.4 GHz	✗	✗
Bluetooth	v5.0 BLE	v5.2 BLE	v5.3 BLE ★	v5.0 BLE	✗
Zigbee	✗	802.15.4 (Thread, Zigbee)	802.15.4 (Thread, Zigbee)	✗	✗
GPIO	22	20	30	19	50+ ★
ADC	Two 12-bit, 6 channels	One 12-bit, x channels	One 12-bit, 7 channels	✗	✗
DAC	✗	✗	✗	✗	✗
SPI	3	2	2	2	✗
I2C	1	2	2	1	✗
I2S	1	1	1	✗	✗
RMT	4 channels	2 channels	2 channels	✗	✗
Touch Sensor	✗	✗	✗	✗	✗
Hall Sensor	✗	✗	✗	✗	✗
LCD Interface	✗	✗	✗	✗	Yes (via MIPI-DSI) ★
Camera Interface	✗	✗	✗	✗	Yes (via MIPI-CSI) ★
Deep Sleep	~5 µA ★	~5 µA ★	~5 µA ★	~5 µA ★	N/A
Size	5x5 mm	5x5 mm	5x5 mm	4x4 mm ★	Custom
Other Features	✗	✗	✗	✗	HMI, USB OTG, Ethernet, AI features ★

Slika 2: Primerjava ESP32 mikrokontrolerjev

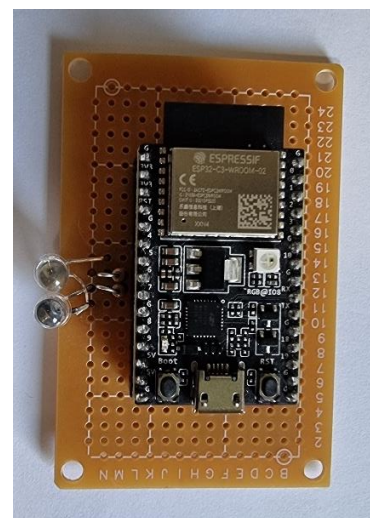
## 5 PREIZKUŠANJE WI-FI MODULA

Ko smo se odločili za Wi-Fi modul ESP32-C3-Devboard-V2, smo le tega naročili. Ko smo ga dobili, smo poiskali spletno stran s primerom uporabe. Na tej spletni strani je lepo opisano, kako naj pričnemo s delom. Najprej je opisano katero programsko orodje, si moramo naložiti za delovanje. Prenesli smo si Arduino IDE, ter knjižnico za ESP32, ki nam omogoči programiranje ESP32 mikrokontrolerjev. Odločili smo se za uporabo ESP32-C3-Devboard-V2 modula, kot Web Server Access Point, tako kot je primer na internet strani. Potem smo šli po naslednjih korakih.

Najprej smo, kot je opisano na interne strani (slika 3) priključili naš ESP32-C3-Devboard-V2 modul, le da smo ga priključili na našo preizkusno tiskanino (slika 4). Napajanja nismo potrebovali, saj modul napajanje dobiva preko USB kablo, ki je povezan na naš osebni računalnik.

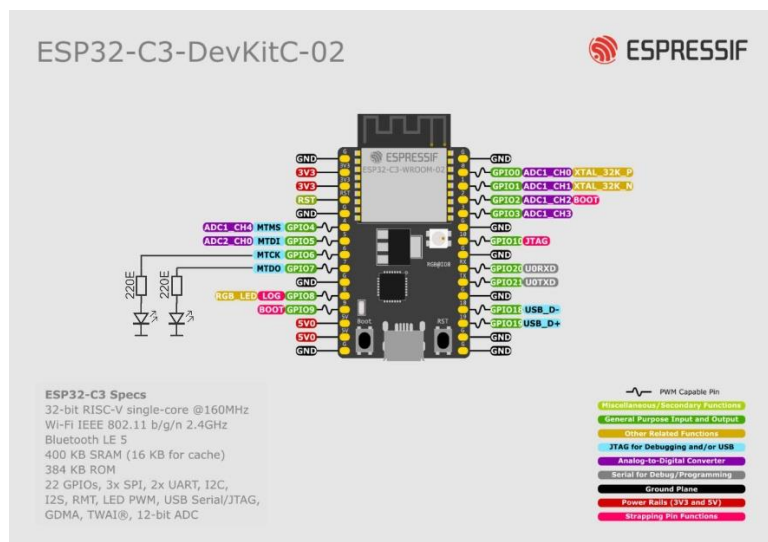


Slika 3: Shematski načrt iz spletne strani



Slika 4: Vezava na preskusni tiskanini

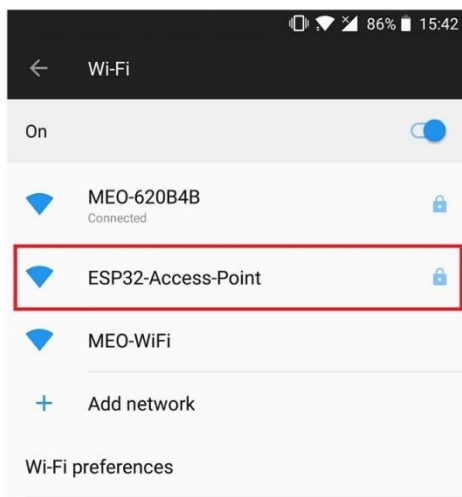
Na shematski sliki kateri so lepo opisane funkcije pinov smo narisali naše komponente (dva upora in dve led diodi) (slika 5).



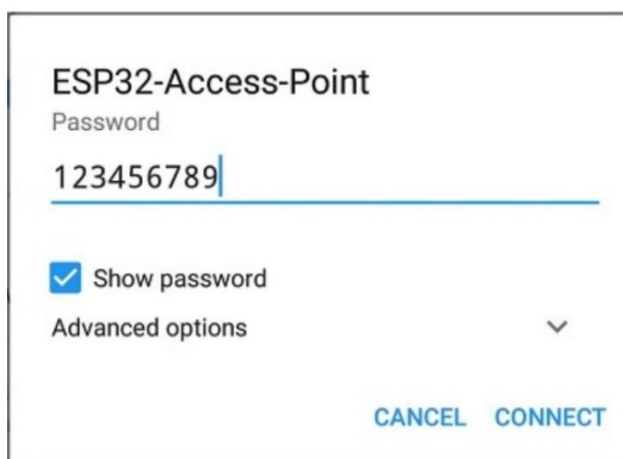
Slika 5: Shematski načrt na shematski sliki

S spletne strani smo prenesli enostavni program »WiFiAccessPoint«, ter ga odprli v Arduino IDE.

Sedaj smo povezali naš modul z osebnim računalnikom preko USB kabla, ter prenesli program s pomočjo Arduino IDE. Po prenosu programa smo uporabili mobilni telefon ter na njem poiskali Wi-Fi ime našega modula, ki se je glasilo (ESP32-Access-Point)(Slika 6), ter vpisali geslo(Slika 7).

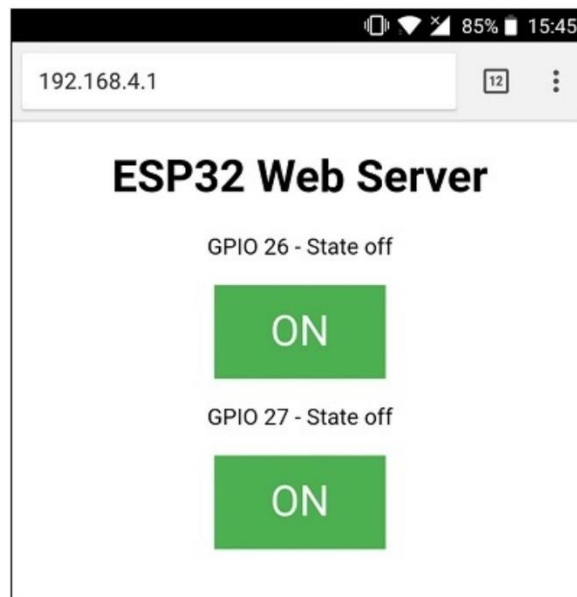


Slika 6: Ime Wi-Fi povezave



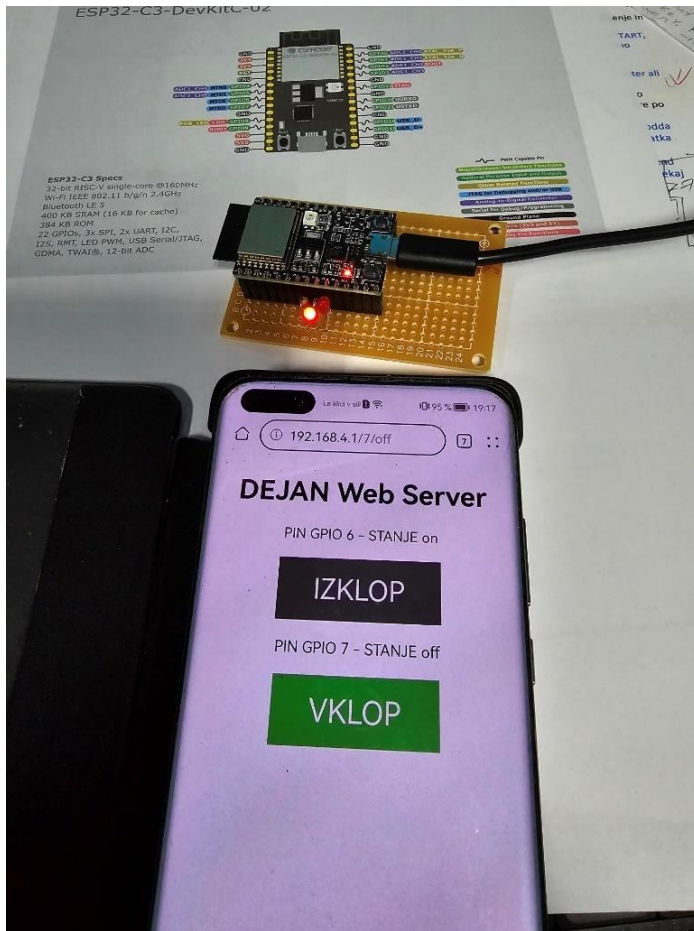
Slika 7: Vpis gesla za Wi-Fi povezavo

Po vzpostavljeni povezavi Wi-Fi smo na mobilni napravi odprli brskalnik, ter vanj vpisali IP naslov (IP Address). Ob vpisu IP naslova se nam je odprla spletna stran s izgledom (Slika 8):



*Slika 8: Program na spletni strani*

Povezava Wi-fi modula po navodilih iz spletne strani je bila uspešna. Na spletni strani je lepo opisano delovanje programa, kjer je razvidno, kako program deluje. Zanimivo nam je bilo, da program pozna programske jezike C, C++, HTML, in JAVA. Po uspešnem prvem preizkusu delovanja, smo želeli spremeniti izgled internetne strani (slika 9). Na omenjeni spletni strani je lepo opisano delovanje programa zato smo lahko z malo truda razumeli delovanje in nastavitve ter lahko spreminjali izgled naše spletne strani (slika 9). Naš modul res deluje kot spletni server. Zaradi pozitivnega presenečenja smo se začeli učiti programske jezike: C, C++, HTML, in JAVA, s pomočjo vodičev na You Tube in spletno stranjo W3SCHOOL. V programu smo spremenili SSID (Slika 10) V samem HTML programu pa obliko gumbov in spremembo besedila S remi spremembami smo še preverili našo razumevanje delovanje programa.



Slika 9: Delovanje povezave programa s našo preskusno tiskanino

## ESP32 Access Point

In this example, we'll modify an [ESP32 Web Server](#) from a previous tutorial to add access point capabilities. What we'll show you here can be used with any ESP32 web server example.

Upload the sketch provided below to set the ESP32 as an access point.

```

/*****
  Rui Santos
  Complete project details at https://randomnerdtutorials.com
  *****/

// Load Wi-Fi library
#include <WiFi.h>

// Replace with your network credentials
const char* ssid = "ESP32-Access-Point";
const char* password = "123456789";

// Set web server port number to 80
WiFiServer server(80);

// Variable to store the HTTP request
String header;

// Auxiliar variables to store the current output state
String output26State = "off";
String output27State = "off";

// Assign output variables to GPIO pins
const int output26 = 26;
const int output27 = 27;

```

Slika 10: Sprememba SSID

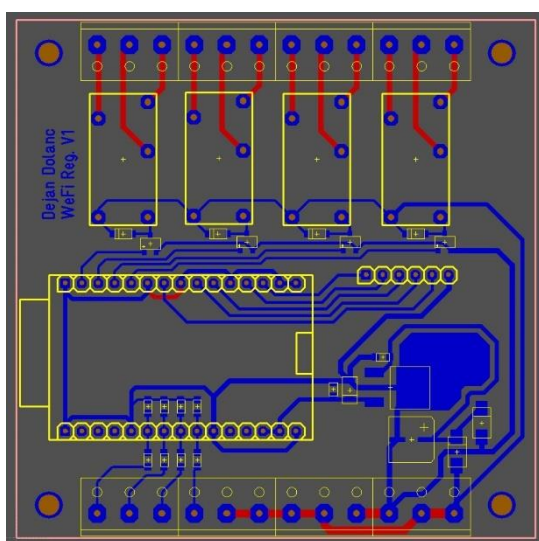
## 6 ZAČETEK IZDELAVE VEČNAMENSKE KRMILE ELEKTRONIKE

Prvotni preizkus našega Wi-Fi modula je uspešno deloval, prav tako smo razumeli njegovo delovanje. Zato smo se lahko lotili načrtovanja krmilne elektronike. Za osnovo smo si določili, da bo krmilna elektronika imela 4 relejske izhode in 4 digitalne vhode. Mislimo, da bi nam to zadostovalo za osnovno krmilno elektroniko. Za izdelavo večnamenske krmilne elektronike se bomo držali naslednjih korakov:

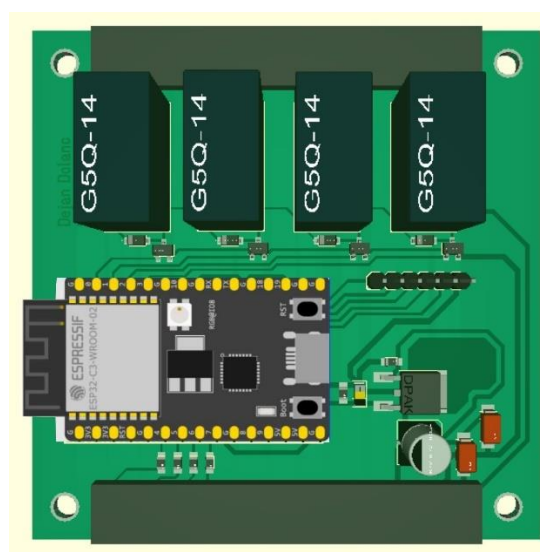
- Izdelava krmilne elektronike z digitalnimi vhodi in relejskimi izhodi.
- Izdelava ohišja v katerega bomo pritrtili našo krmilno elektroniko in katerega bomo lahko nato na enostaven način pritrtili v vsak posamičen prostor.
- Izdelava spletne aplikacije za Wi-Fi krmilno elektroniko oziroma več aplikacij za različne prostore v hiši oziroma različna krmilja.

### 6.1. Načrtovanje in izdelava tiskanine:

Pred začetkom izdelave našega vezja smo si postavili vprašanje: «Koliko digitalnih vhodov in relejskih izhodov bo vsebovalo našo vezje.» Prišli smo do odločitve, da bomo uporabili štiri digitalne vhode in štiri relejske izhode. Mislimo, da bi bilo za začetek to dovolj. Vklapljali in izklapljali bi lahko štiri naprave v tistem prostoru lahko pa bi jih tudi krmilili oz. gledali določene vrednosti kot so temperatura oz. svetloba v nekem prostoru s pomočjo senzorjev, ki jih lahko priključimo na naše vhode. Na primer lahko bi vklapljali štiri luči neodvisno eno od druge. V drugem primeru bi lahko na primer z enim relejskim izhodom odpirali žaluzije z drugim pa zapirali žaluzije, lahko pa bi tudi s pomočjo vhodov gledali temperaturo prostora in z izhodi vklapljali grelce ali pa gledali svetlobo v prostoru in avtomatsko s pomočjo izhodov vklapljali luči ali pa odprli oz. zaprli žaluzije. Sedaj smo pričeli z načrtovanjem tiskanega vezja za našo krmilno elektroniko. Za načrtovanje in risanje tiskanine smo uporabili programsko orodje Target 3001. Narisali smo tiskano vezje (slika 11) in preverili njen izgled v 3D obliki (slika 12) kar nam omogoča Target 3001. Po tem smo dali v izdelavo naša tiskana vezja. Po prejemu tiskanih vezij smo nanje pri spajkali vse potrebne komponente (slika 13).

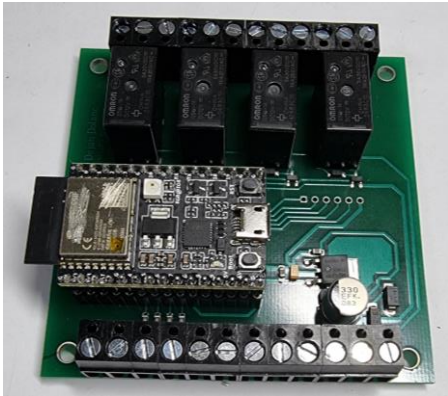


Slika 11: Izgled narisane tiskanine



Slika 12: Izgled tiskanine v 3D obliki

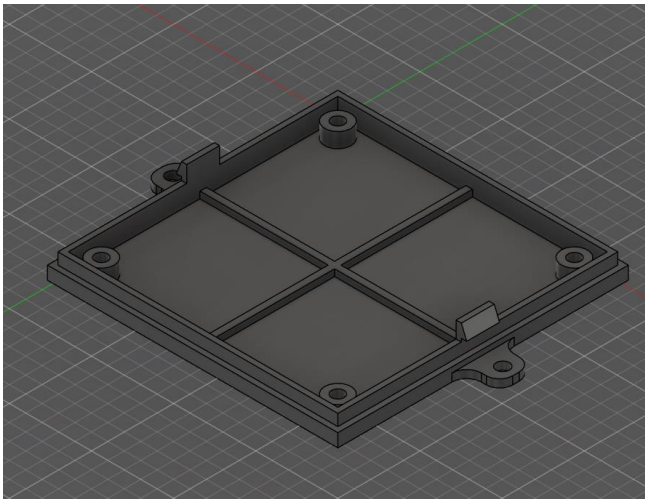
Na tej sliki lahko vidimo končano krmilno elektroniko na kateri so položene vse komponente.



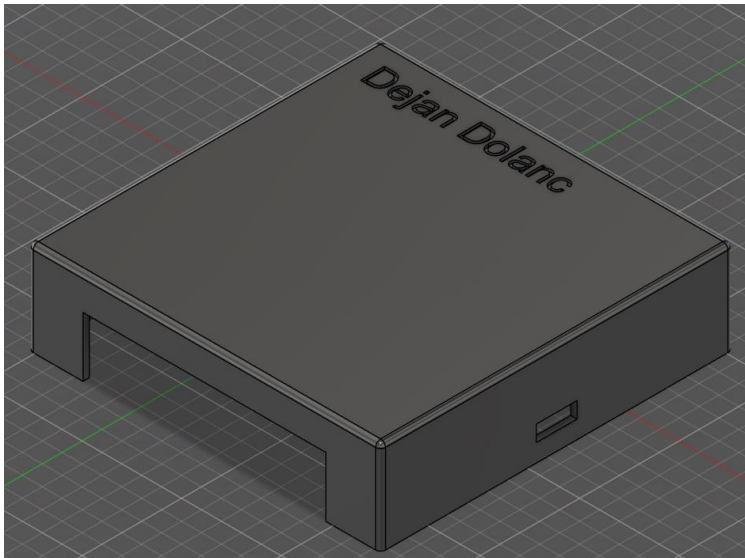
Slika 13: Izgled sestavljene tiskanine

## 6.2. Načrtovanje in izdelava ohišja:

Sedaj ko smo imeli narejena tiskana vezja z vsemi potrebnimi pritrjenimi komponentami smo izmerili potrebno višino, širino in dolžino naše krmilne elektronike, ki jih potrebujemo za izdelavo ohišja. Ohišje smo narisali iz dveh delov in sicer, da je spodnji del enostaven za montažo zaradi česar smo dodali dve ušesi z luknjami za vijake in pokrov, ki je narejen tako, da ga lahko na enostaven način namestimo in odstranimo ko priklopljamo vse potrebne kable saj je narejen tako, da se samo zaskoči na spodnji del, kar lahko vidimo na spodnjih slikah 14 in 15. Vse skupaj smo narisali v programu FUSION 360.



Slika 14: Spodnji del ohišja narisani v Fusion 360



*Slika 15: Pokrov ohišja narisano v Fusion 360*

Nato smo oba dela natisnili s 3D tiskalnikom. Naše ohišje je narejeno iz PLA plastike. Na spodnjih slikah 17 in 18 lahko vidimo naš pokrov natisnjen s 3D tiskalnikom:

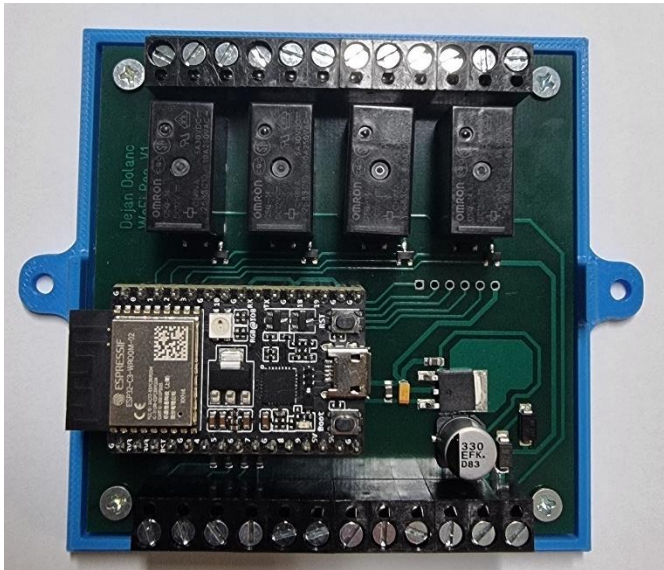


*Slika 16: 3D natisnjen pokrov*

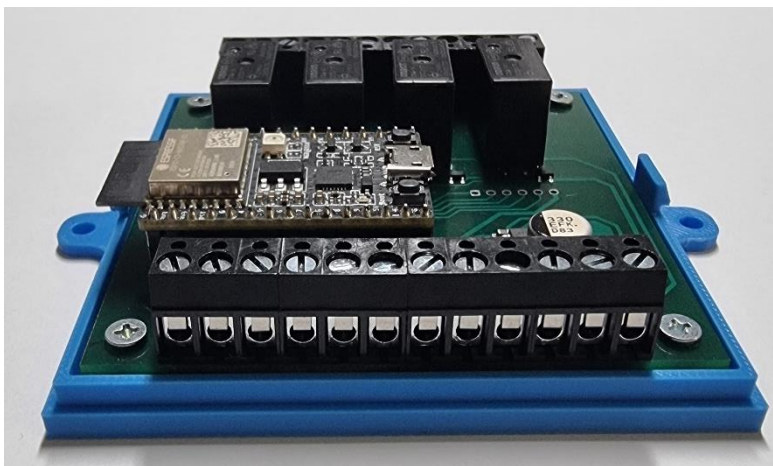


*Slika 17: 3D natisnjen pokrov z vidno luknjo za konektorje*

Na spodnjih slikah 18 in 19 lahko vidimo našo krmilno elektroniko pri vijačeno na naše dno ohišja:



*Slika 18: Pri vijačena krmilna elektronika na dno ohišja*



*Slika 19: Pri vijačena krmilna elektronika na dno ohišja*

Na spodnjih slikah 20 in 21 pa lahko vidimo končni izgled našega ohišja, ki ima v sebi pri vijačeno našo krmilno elektroniko in luknje za konektorje.



*Slika 20: Končano ohišje*



*Slika 21: Končano ohišje pogled z vrha*

### **6.3. Načrtovanje in izdelava Web-Aplikacije:**

Sedaj, ko smo naredili vezje in ohišje smo pričeli s programiranjem naše Web-Aplikacije. Za pisanje programske kode HTML, CSS ter Java smo si izbrali programsko opremo Visual Studio Code, saj je najbolj priljubljena programska oprema za programiranje kot ga potrebujemo mi.

Pričeli smo s pisanjem programa v HTML, CSS ter Java scriptu kjer smo oblikovali našo spletno aplikacijo. Kako izgleda spletna aplikacija v HTML lahko vidimo na slikah 22 in 23.

```

living.html > ...
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <link rel="stylesheet" href="living.css">
7   <title>DEJAN Web Server</title>
8 </head>
9 <body>
10  <a href="menu.html">
11    <button class="b-btn">BACK</button>
12  </a>
13
14
15  <h2 class="luči">LUČI</h2>
16  <h2 class="luč-1-2">LUČ_1 LUČ_2</h2>
17  <button class="1-1" id="toggleButton1-lg"></button>
18  <button class="1-2" id="toggleButton2-lg"></button>
19
20  <h2 class="žaluzije">ŽALUZIJE</h2>
21  <h2 class="okno-1-2">OKNO_1 OKNO_2</h2>
22  <button class="o-1" id="toggleButton3-lg"></button>
23  <button class="o-2" id="toggleButton4-lg"></button>
24
25  <script>
26    window.onload = function() {
27
28      const buttons = [
29        { id: "toggleButton1-lg", key: "buttonState1-lg" },
30        { id: "toggleButton2-lg", key: "buttonState2-lg" },
31        { id: "toggleButton3-lg", key: "buttonState3-lg" },
32        { id: "toggleButton4-lg", key: "buttonState4-lg" }
33      ];
34
35
36      buttons.forEach(button => {
37        const savedState = localStorage.getItem(button.key);
38        if (savedState) {
39          document.getElementById(button.id).textContent = savedState;
40        }
41      });
42
43
44      document.getElementById("toggleButton1-lg").onclick = function() {
45        this.textContent = this.textContent === "ON" ? "OFF" : "ON";
46        localStorage.setItem("buttonState1-lg", this.textContent);
47      };
48
49      document.getElementById("toggleButton2-lg").onclick = function() {
50        this.textContent = this.textContent === "ON" ? "OFF" : "ON";
51        localStorage.setItem("buttonState2-lg", this.textContent);
52      };
53
54      document.getElementById("toggleButton3-lg").onclick = function() {
55        this.textContent = this.textContent === "GOR" ? "DOL" : "GOR";
56        localStorage.setItem("buttonState3-lg", this.textContent);
57      };
58
59      document.getElementById("toggleButton4-lg").onclick = function() {
60        this.textContent = this.textContent === "GOR" ? "DOL" : "GOR";
61        localStorage.setItem("buttonState4-lg", this.textContent);
62      };
63    }
64  </script>
65
66 </body>
67 </html>

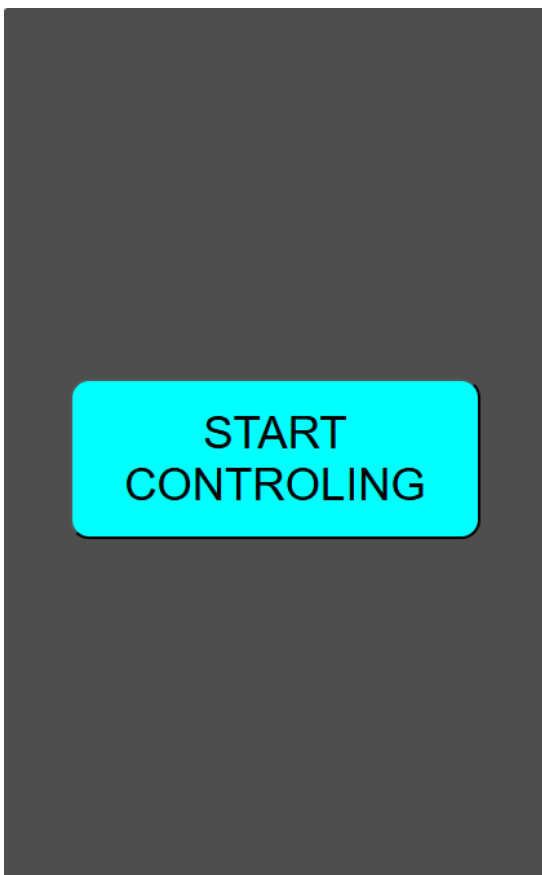
```

Slika 22: Programski kod za oblikovanje spletnih aplikacija

```
menu.html > html > body > a > button.dd-btn
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <link rel="stylesheet" href="menu.style.css">
7   <title>DEJAN Web Serber</title>
8 </head>
9 <body>
10
11   <a href="index.html">
12     <button class="b-btn">BACK</button>
13   </a>
14
15   <h1 class="menu-t">MENU</h1>
16
17   <a href="living.html">
18     <button class="lg-btn">LIVING<br />ROOM</button>
19   </a>
20
21   <a href="bath.html">
22     <button class="dg-btn">BATH<br />ROOM</button>
23   </a>
24
25   <a href="kids.html">
26     <button class="ld-btn">KIDS<br />ROOM</button>
27   </a>
28
29   <a href="bad.html">
30     <button class="dd-btn">BED<br />ROOM</button>
31   </a>
32
33 </body>
34 </html>
```

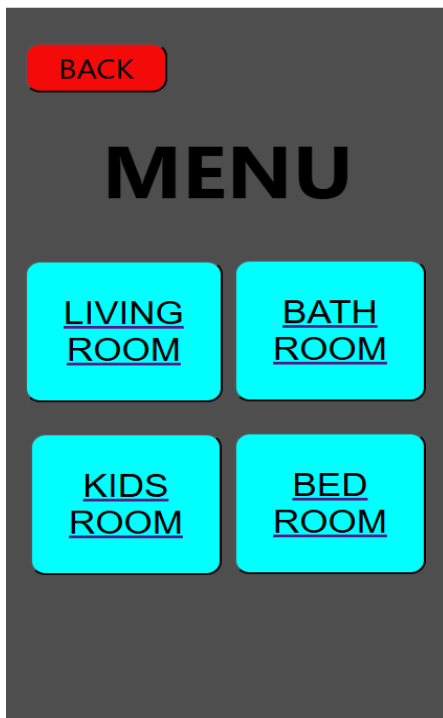
Slika 23: Programska koda za obliko spletne aplikacije

Na sliki 23 lahko vidimo kako izgleda začetna stran naše spletne aplikacije, vidimo lahko, da je oblikovan tako, da se nam pojavi gumb, ki ga pritisnemo ko želimo začeti krmiliti nek prostor.



Slika 24: Začetna stran naše spletne aplikacije

Ko smo pritisnili gumb na začetnem zaslonu se nam odpre naslednje okno kjer izberemo katero sobo želimo krmiliti to lahko vidimo na sliki 25, imamo pa možnost, da pritisnemo gumb za nazaj na začetni zaslon:



Slika 25: Izbira sobe, ki jo želimo krmiliti

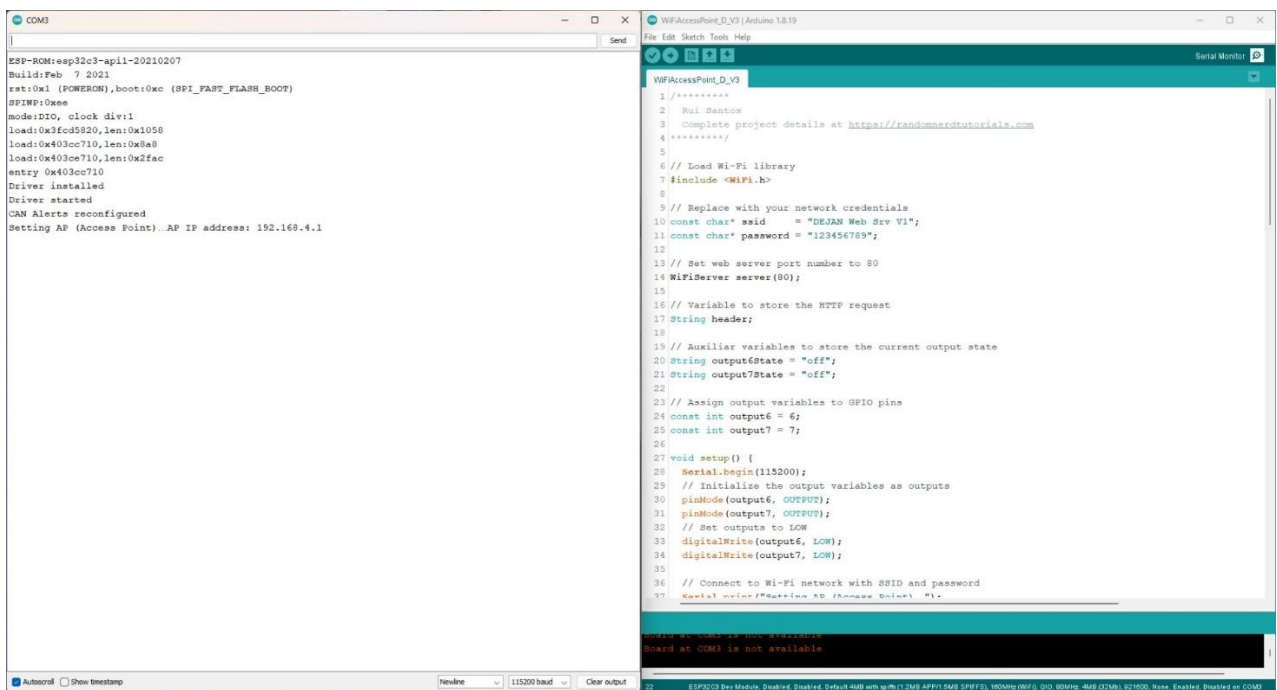
Ko izberemo katero sobo želimo krmiliti se nam odpre naslednje okno, kjer izberemo kaj želimo krmiliti v tisti sobo torej v našem primeru krmiljenje luči in žaluzij to vidimo na sliki 26, imamo pa prav tako gumb za nazaj s katerim pridemo na prejšnji zaslon za izbiro sob:



Slika 26: Krmiljenje posameznih naprav v sobi

## 6.4. Pisanje programa za krmiljenje vhodov in izhodov ter povezavo med spletno aplikacijo in krmilno elektroniko:

Ko smo oblikovali našo spletno aplikacijo smo se lotili pisanja programa za povezavo med našo spletno aplikacijo in vhodi ter izhodi na naši krmilni elektroniki, torej program, ki gleda kaj smo pritisnili na naši spletni aplikaciji na primer prižig luči in nam na osnovi teg preko relejskega izhoda prižge luč ali pa gleda vhode na naši krmilni elektroniki in na osnovi signalov iz senzorjev, ki so povezani na te vhode naredi nekaj na izhodih ali pa nam kaj izpiše oz. naredi na naši spletni aplikaciji. Ta program smo napisali v programski opremi za delo z Arduino saj je ta ESP-32 del Arduino družine. Program lahko vidimo na slikah 27 in 28.



```
ESP-ROM:esp32u3-ap11-20210207
Build:Feb  7 2021
rst:0x1 (POWERON),boot:0xc (SPI_FAST_FLASH_BOOT)
SPIWP:0xee
mode:DIO, clock div:1
load:0x3fcd5820, len:0x1058
load:0x4030ce710, len:0x8e8
load:0x4030ce710, len:0x2fac
entry 0x4030ce710
Driver installed
Driver started
CAN Alerts reconfigured
Setting AP (Access Point)...AP IP address: 192.168.4.1

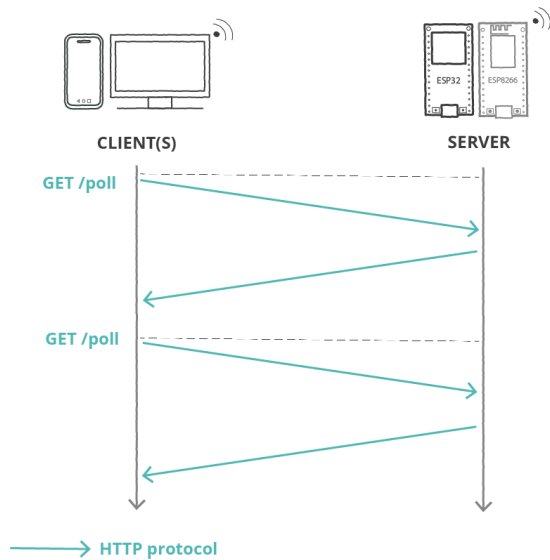
WiFiAccessPoint_D_V3 | Arduino 1.8.19
File Edit Sketch Tools Help
WiFiAccessPoint_D_V3
1 //*****
2 Rui Santos
3 Complete project details at https://randomertutorials.com
4 //*****
5
6 // Load Wi-Fi library
7 #include <WiFi.h>
8
9 // Replace with your network credentials
10 const char* ssid = "DEJAN Web Srv V1";
11 const char* password = "123456789";
12
13 // Set web server port number to 80
14 WiFiServer server(80);
15
16 // Variable to store the HTTP request
17 String header;
18
19 // Auxiliar variables to store the current output state
20 String output6State = "off";
21 String output7State = "off";
22
23 // Assign output variables to GPIO pins
24 const int output6 = 6;
25 const int output7 = 7;
26
27 void setup() {
28   Serial.begin(115200);
29   // Initialize the output variables as outputs
30   pinMode(output6, OUTPUT);
31   pinMode(output7, OUTPUT);
32   // Set outputs to LOW
33   digitalWrite(output6, LOW);
34   digitalWrite(output7, LOW);
35
36   // Connect to Wi-Fi network with SSID and password
37   WiFi.begin(ssid, password);
38
39   while (WiFi.status() != WL_CONNECTED) {
40     delay(5000);
41     Serial.print("Connecting to WiFi network: ");
42     Serial.println(WiFi.status());
43   }
44
45   Serial.println("WiFi connected");
46   Serial.println("IP address: ");
47   Serial.println(WiFi.localIP());
48 }
49
50 void loop() {
51   // Get the HTTP request and parse it:
52   // Here we check if the browser asks for the root URL
53   if (server.hasArg()) {
54     // If there's a "GET /" request, then set the output state
55     String request = server.arg(0);
56     if (request == "/") {
57       // Turn on the LED
58       digitalWrite(output6, HIGH);
59       digitalWrite(output7, HIGH);
60       output6State = "on";
61       output7State = "on";
62     } else if (request == "/off") {
63       // Turn off the LED
64       digitalWrite(output6, LOW);
65       digitalWrite(output7, LOW);
66       output6State = "off";
67       output7State = "off";
68     }
69   }
70   // Send the response text
71   server.send(200, "text/html", "OK");
72 }
73
74 void printOutputState() {
75   Serial.print("Output 6 state: ");
76   Serial.println(output6State);
77   Serial.print("Output 7 state: ");
78   Serial.println(output7State);
79 }
80
81 void printIP() {
82   Serial.println(WiFi.localIP());
83 }
84
85 void printSSID() {
86   Serial.println(ssid);
87 }
88
89 void printPassword() {
90   Serial.println(password);
91 }
92
93 void printServerPort() {
94   Serial.println(server.port());
95 }
96
97 void printServer() {
98   Serial.println(server);
99 }
100
101 void printHeader() {
102   Serial.println(header);
103 }
104
105 void printRequest() {
106   Serial.println(request);
107 }
108
109 void printResponse() {
110   Serial.println(response);
111 }
112
113 void printStatus() {
114   Serial.println(WiFi.status());
115 }
116
117 void printSSID() {
118   Serial.println(ssid);
119 }
120
121 void printPassword() {
122   Serial.println(password);
123 }
124
125 void printServerPort() {
126   Serial.println(server.port());
127 }
128
129 void printServer() {
130   Serial.println(server);
131 }
132
133 void printHeader() {
134   Serial.println(header);
135 }
136
137 void printRequest() {
138   Serial.println(request);
139 }
140
141 void printResponse() {
142   Serial.println(response);
143 }
144
145 void printStatus() {
146   Serial.println(WiFi.status());
147 }
148
149 void printSSID() {
150   Serial.println(ssid);
151 }
152
153 void printPassword() {
154   Serial.println(password);
155 }
156
157 void printServerPort() {
158   Serial.println(server.port());
159 }
160
161 void printServer() {
162   Serial.println(server);
163 }
164
165 void printHeader() {
166   Serial.println(header);
167 }
168
169 void printRequest() {
170   Serial.println(request);
171 }
172
173 void printResponse() {
174   Serial.println(response);
175 }
176
177 void printStatus() {
178   Serial.println(WiFi.status());
179 }
180
181 void printSSID() {
182   Serial.println(ssid);
183 }
184
185 void printPassword() {
186   Serial.println(password);
187 }
188
189 void printServerPort() {
190   Serial.println(server.port());
191 }
192
193 void printServer() {
194   Serial.println(server);
195 }
196
197 void printHeader() {
198   Serial.println(header);
199 }
200
201 void printRequest() {
202   Serial.println(request);
203 }
204
205 void printResponse() {
206   Serial.println(response);
207 }
208
209 void printStatus() {
210   Serial.println(WiFi.status());
211 }
212
213 void printSSID() {
214   Serial.println(ssid);
215 }
216
217 void printPassword() {
218   Serial.println(password);
219 }
220
221 void printServerPort() {
222   Serial.println(server.port());
223 }
224
225 void printServer() {
226   Serial.println(server);
227 }
228
229 void printHeader() {
230   Serial.println(header);
231 }
232
233 void printRequest() {
234   Serial.println(request);
235 }
236
237 void printResponse() {
238   Serial.println(response);
239 }
240
241 void printStatus() {
242   Serial.println(WiFi.status());
243 }
244
245 void printSSID() {
246   Serial.println(ssid);
247 }
248
249 void printPassword() {
250   Serial.println(password);
251 }
252
253 void printServerPort() {
254   Serial.println(server.port());
255 }
256
257 void printServer() {
258   Serial.println(server);
259 }
260
261 void printHeader() {
262   Serial.println(header);
263 }
264
265 void printRequest() {
266   Serial.println(request);
267 }
268
269 void printResponse() {
270   Serial.println(response);
271 }
272
273 void printStatus() {
274   Serial.println(WiFi.status());
275 }
276
277 void printSSID() {
278   Serial.println(ssid);
279 }
280
281 void printPassword() {
282   Serial.println(password);
283 }
284
285 void printServerPort() {
286   Serial.println(server.port());
287 }
288
289 void printServer() {
290   Serial.println(server);
291 }
292
293 void printHeader() {
294   Serial.println(header);
295 }
296
297 void printRequest() {
298   Serial.println(request);
299 }
300
301 void printResponse() {
302   Serial.println(response);
303 }
304
305 void printStatus() {
306   Serial.println(WiFi.status());
307 }
308
309 void printSSID() {
310   Serial.println(ssid);
311 }
312
313 void printPassword() {
314   Serial.println(password);
315 }
316
317 void printServerPort() {
318   Serial.println(server.port());
319 }
320
321 void printServer() {
322   Serial.println(server);
323 }
324
325 void printHeader() {
326   Serial.println(header);
327 }
328
329 void printRequest() {
330   Serial.println(request);
331 }
332
333 void printResponse() {
334   Serial.println(response);
335 }
336
337 void printStatus() {
338   Serial.println(WiFi.status());
339 }
340
341 void printSSID() {
342   Serial.println(ssid);
343 }
344
345 void printPassword() {
346   Serial.println(password);
347 }
348
349 void printServerPort() {
350   Serial.println(server.port());
351 }
352
353 void printServer() {
354   Serial.println(server);
355 }
356
357 void printHeader() {
358   Serial.println(header);
359 }
360
361 void printRequest() {
362   Serial.println(request);
363 }
364
365 void printResponse() {
366   Serial.println(response);
367 }
368
369 void printStatus() {
370   Serial.println(WiFi.status());
371 }
372
373 void printSSID() {
374   Serial.println(ssid);
375 }
376
377 void printPassword() {
378   Serial.println(password);
379 }
380
381 void printServerPort() {
382   Serial.println(server.port());
383 }
384
385 void printServer() {
386   Serial.println(server);
387 }
388
389 void printHeader() {
390   Serial.println(header);
391 }
392
393 void printRequest() {
394   Serial.println(request);
395 }
396
397 void printResponse() {
398   Serial.println(response);
399 }
400
401 void printStatus() {
402   Serial.println(WiFi.status());
403 }
404
405 void printSSID() {
406   Serial.println(ssid);
407 }
408
409 void printPassword() {
410   Serial.println(password);
411 }
412
413 void printServerPort() {
414   Serial.println(server.port());
415 }
416
417 void printServer() {
418   Serial.println(server);
419 }
420
421 void printHeader() {
422   Serial.println(header);
423 }
424
425 void printRequest() {
426   Serial.println(request);
427 }
428
429 void printResponse() {
430   Serial.println(response);
431 }
432
433 void printStatus() {
434   Serial.println(WiFi.status());
435 }
436
437 void printSSID() {
438   Serial.println(ssid);
439 }
440
441 void printPassword() {
442   Serial.println(password);
443 }
444
445 void printServerPort() {
446   Serial.println(server.port());
447 }
448
449 void printServer() {
450   Serial.println(server);
451 }
452
453 void printHeader() {
454   Serial.println(header);
455 }
456
457 void printRequest() {
458   Serial.println(request);
459 }
460
461 void printResponse() {
462   Serial.println(response);
463 }
464
465 void printStatus() {
466   Serial.println(WiFi.status());
467 }
468
469 void printSSID() {
470   Serial.println(ssid);
471 }
472
473 void printPassword() {
474   Serial.println(password);
475 }
476
477 void printServerPort() {
478   Serial.println(server.port());
479 }
480
481 void printServer() {
482   Serial.println(server);
483 }
484
485 void printHeader() {
486   Serial.println(header);
487 }
488
489 void printRequest() {
490   Serial.println(request);
491 }
492
493 void printResponse() {
494   Serial.println(response);
495 }
496
497 void printStatus() {
498   Serial.println(WiFi.status());
499 }
500
501 void printSSID() {
502   Serial.println(ssid);
503 }
504
505 void printPassword() {
506   Serial.println(password);
507 }
508
509 void printServerPort() {
510   Serial.println(server.port());
511 }
512
513 void printServer() {
514   Serial.println(server);
515 }
516
517 void printHeader() {
518   Serial.println(header);
519 }
520
521 void printRequest() {
522   Serial.println(request);
523 }
524
525 void printResponse() {
526   Serial.println(response);
527 }
528
529 void printStatus() {
530   Serial.println(WiFi.status());
531 }
532
533 void printSSID() {
534   Serial.println(ssid);
535 }
536
537 void printPassword() {
538   Serial.println(password);
539 }
540
541 void printServerPort() {
542   Serial.println(server.port());
543 }
544
545 void printServer() {
546   Serial.println(server);
547 }
548
549 void printHeader() {
550   Serial.println(header);
551 }
552
553 void printRequest() {
554   Serial.println(request);
555 }
556
557 void printResponse() {
558   Serial.println(response);
559 }
560
561 void printStatus() {
562   Serial.println(WiFi.status());
563 }
564
565 void printSSID() {
566   Serial.println(ssid);
567 }
568
569 void printPassword() {
570   Serial.println(password);
571 }
572
573 void printServerPort() {
574   Serial.println(server.port());
575 }
576
577 void printServer() {
578   Serial.println(server);
579 }
580
581 void printHeader() {
582   Serial.println(header);
583 }
584
585 void printRequest() {
586   Serial.println(request);
587 }
588
589 void printResponse() {
590   Serial.println(response);
591 }
592
593 void printStatus() {
594   Serial.println(WiFi.status());
595 }
596
597 void printSSID() {
598   Serial.println(ssid);
599 }
600
601 void printPassword() {
602   Serial.println(password);
603 }
604
605 void printServerPort() {
606   Serial.println(server.port());
607 }
608
609 void printServer() {
610   Serial.println(server);
611 }
612
613 void printHeader() {
614   Serial.println(header);
615 }
616
617 void printRequest() {
618   Serial.println(request);
619 }
620
621 void printResponse() {
622   Serial.println(response);
623 }
624
625 void printStatus() {
626   Serial.println(WiFi.status());
627 }
628
629 void printSSID() {
630   Serial.println(ssid);
631 }
632
633 void printPassword() {
634   Serial.println(password);
635 }
636
637 void printServerPort() {
638   Serial.println(server.port());
639 }
640
641 void printServer() {
642   Serial.println(server);
643 }
644
645 void printHeader() {
646   Serial.println(header);
647 }
648
649 void printRequest() {
650   Serial.println(request);
651 }
652
653 void printResponse() {
654   Serial.println(response);
655 }
656
657 void printStatus() {
658   Serial.println(WiFi.status());
659 }
660
661 void printSSID() {
662   Serial.println(ssid);
663 }
664
665 void printPassword() {
666   Serial.println(password);
667 }
668
669 void printServerPort() {
670   Serial.println(server.port());
671 }
672
673 void printServer() {
674   Serial.println(server);
675 }
676
677 void printHeader() {
678   Serial.println(header);
679 }
680
681 void printRequest() {
682   Serial.println(request);
683 }
684
685 void printResponse() {
686   Serial.println(response);
687 }
688
689 void printStatus() {
690   Serial.println(WiFi.status());
691 }
692
693 void printSSID() {
694   Serial.println(ssid);
695 }
696
697 void printPassword() {
698   Serial.println(password);
699 }
700
701 void printServerPort() {
702   Serial.println(server.port());
703 }
704
705 void printServer() {
706   Serial.println(server);
707 }
708
709 void printHeader() {
710   Serial.println(header);
711 }
712
713 void printRequest() {
714   Serial.println(request);
715 }
716
717 void printResponse() {
718   Serial.println(response);
719 }
720
721 void printStatus() {
722   Serial.println(WiFi.status());
723 }
724
725 void printSSID() {
726   Serial.println(ssid);
727 }
728
729 void printPassword() {
730   Serial.println(password);
731 }
732
733 void printServerPort() {
734   Serial.println(server.port());
735 }
736
737 void printServer() {
738   Serial.println(server);
739 }
740
741 void printHeader() {
742   Serial.println(header);
743 }
744
745 void printRequest() {
746   Serial.println(request);
747 }
748
749 void printResponse() {
750   Serial.println(response);
751 }
752
753 void printStatus() {
754   Serial.println(WiFi.status());
755 }
756
757 void printSSID() {
758   Serial.println(ssid);
759 }
760
761 void printPassword() {
762   Serial.println(password);
763 }
764
765 void printServerPort() {
766   Serial.println(server.port());
767 }
768
769 void printServer() {
770   Serial.println(server);
771 }
772
773 void printHeader() {
774   Serial.println(header);
775 }
776
777 void printRequest() {
778   Serial.println(request);
779 }
780
781 void printResponse() {
782   Serial.println(response);
783 }
784
785 void printStatus() {
786   Serial.println(WiFi.status());
787 }
788
789 void printSSID() {
790   Serial.println(ssid);
791 }
792
793 void printPassword() {
794   Serial.println(password);
795 }
796
797 void printServerPort() {
798   Serial.println(server.port());
799 }
800
801 void printServer() {
802   Serial.println(server);
803 }
804
805 void printHeader() {
806   Serial.println(header);
807 }
808
809 void printRequest() {
810   Serial.println(request);
811 }
812
813 void printResponse() {
814   Serial.println(response);
815 }
816
817 void printStatus() {
818   Serial.println(WiFi.status());
819 }
820
821 void printSSID() {
822   Serial.println(ssid);
823 }
824
825 void printPassword() {
826   Serial.println(password);
827 }
828
829 void printServerPort() {
830   Serial.println(server.port());
831 }
832
833 void printServer() {
834   Serial.println(server);
835 }
836
837 void printHeader() {
838   Serial.println(header);
839 }
840
841 void printRequest() {
842   Serial.println(request);
843 }
844
845 void printResponse() {
846   Serial.println(response);
847 }
848
849 void printStatus() {
850   Serial.println(WiFi.status());
851 }
852
853 void printSSID() {
854   Serial.println(ssid);
855 }
856
857 void printPassword() {
858   Serial.println(password);
859 }
860
861 void printServerPort() {
862   Serial.println(server.port());
863 }
864
865 void printServer() {
866   Serial.println(server);
867 }
868
869 void printHeader() {
870   Serial.println(header);
871 }
872
873 void printRequest() {
874   Serial.println(request);
875 }
876
877 void printResponse() {
878   Serial.println(response);
879 }
880
881 void printStatus() {
882   Serial.println(WiFi.status());
883 }
884
885 void printSSID() {
886   Serial.println(ssid);
887 }
888
889 void printPassword() {
890   Serial.println(password);
891 }
892
893 void printServerPort() {
894   Serial.println(server.port());
895 }
896
897 void printServer() {
898   Serial.println(server);
899 }
900
901 void printHeader() {
902   Serial.println(header);
903 }
904
905 void printRequest() {
906   Serial.println(request);
907 }
908
909 void printResponse() {
910   Serial.println(response);
911 }
912
913 void printStatus() {
914   Serial.println(WiFi.status());
915 }
916
917 void printSSID() {
918   Serial.println(ssid);
919 }
920
921 void printPassword() {
922   Serial.println(password);
923 }
924
925 void printServerPort() {
926   Serial.println(server.port());
927 }
928
929 void printServer() {
930   Serial.println(server);
931 }
932
933 void printHeader() {
934   Serial.println(header);
935 }
936
937 void printRequest() {
938   Serial.println(request);
939 }
940
941 void printResponse() {
942   Serial.println(response);
943 }
944
945 void printStatus() {
946   Serial.println(WiFi.status());
947 }
948
949 void printSSID() {
950   Serial.println(ssid);
951 }
952
953 void printPassword() {
954   Serial.println(password);
955 }
956
957 void printServerPort() {
958   Serial.println(server.port());
959 }
960
961 void printServer() {
962   Serial.println(server);
963 }
964
965 void printHeader() {
966   Serial.println(header);
967 }
968
969 void printRequest() {
970   Serial.println(request);
971 }
972
973 void printResponse() {
974   Serial.println(response);
975 }
976
977 void printStatus() {
978   Serial.println(WiFi.status());
979 }
980
981 void printSSID() {
982   Serial.println(ssid);
983 }
984
985 void printPassword() {
986   Serial.println(password);
987 }
988
989 void printServerPort() {
990   Serial.println(server.port());
991 }
992
993 void printServer() {
994   Serial.println(server);
995 }
996
997 void printHeader() {
998   Serial.println(header);
999 }
1000
1001 void printRequest() {
1002   Serial.println(request);
1003 }
1004
1005 void printResponse() {
1006   Serial.println(response);
1007 }
1008
1009 void printStatus() {
1010   Serial.println(WiFi.status());
1011 }
1012
1013 void printSSID() {
1014   Serial.println(ssid);
1015 }
1016
1017 void printPassword() {
1018   Serial.println(password);
1019 }
1020
1021 void printServerPort() {
1022   Serial.println(server.port());
1023 }
1024
1025 void printServer() {
1026   Serial.println(server);
1027 }
1028
1029 void printHeader() {
1030   Serial.println(header);
1031 }
1032
1033 void printRequest() {
1034   Serial.println(request);
1035 }
1036
1037 void printResponse() {
1038   Serial.println(response);
1039 }
1040
1041 void printStatus() {
1042   Serial.println(WiFi.status());
1043 }
1044
1045 void printSSID() {
1046   Serial.println(ssid);
1047 }
1048
1049 void printPassword() {
1050   Serial.println(password);
1051 }
1052
1053 void printServerPort() {
1054   Serial.println(server.port());
1055 }
1056
1057 void printServer() {
1058   Serial.println(server);
1059 }
1060
1061 void printHeader() {
1062   Serial.println(header);
1063 }
1064
1065 void printRequest() {
1066   Serial.println(request);
1067 }
1068
1069 void printResponse() {
1070   Serial.println(response);
1071 }
1072
1073 void printStatus() {
1074   Serial.println(WiFi.status());
1075 }
1076
1077 void printSSID() {
1078   Serial.println(ssid);
1079 }
1080
1081 void printPassword() {
1082   Serial.println(password);
1083 }
1084
1085 void printServerPort() {
1086   Serial.println(server.port());
1087 }
1088
1089 void printServer() {
1090   Serial.println(server);
1091 }
1092
1093 void printHeader() {
1094   Serial.println(header);
1095 }
1096
1097 void printRequest() {
1098   Serial.println(request);
1099 }
1100
1101 void printResponse() {
1102   Serial.println(response);
1103 }
1104
1105 void printStatus() {
1106   Serial.println(WiFi.status());
1107 }
1108
1109 void printSSID() {
1110   Serial.println(ssid);
1111 }
1112
1113 void printPassword() {
1114   Serial.println(password);
1115 }
1116
1117 void printServerPort() {
1118   Serial.println(server.port());
1119 }
1120
1121 void printServer() {
1122   Serial.println(server);
1123 }
1124
1125 void printHeader() {
1126   Serial.println(header);
1127 }
1128
1129 void printRequest() {
1130   Serial.println(request);
1131 }
1132
1133 void printResponse() {
1134   Serial.println(response);
1135 }
1136
1137 void printStatus() {
1138   Serial.println(WiFi.status());
1139 }
1140
1141 void printSSID() {
1142   Serial.println(ssid);
1143 }
1144
1145 void printPassword() {
1146   Serial.println(password);
1147 }
1148
1149 void printServerPort() {
1150   Serial.println(server.port());
1151 }
1152
1153 void printServer() {
1154   Serial.println(server);
1155 }
1156
1157 void printHeader() {
1158   Serial.println(header);
1159 }
1160
1161 void printRequest() {
1162   Serial.println(request);
1163 }
1164
1165 void printResponse() {
1166   Serial.println(response);
1167 }
1168
1169 void printStatus() {
1170   Serial.println(WiFi.status());
1171 }
1172
1173 void printSSID() {
1174   Serial.println(ssid);
1175 }
1176
1177 void printPassword() {
1178   Serial.println(password);
1179 }
1180
1181 void printServerPort() {
1182   Serial.println(server.port());
1183 }
1184
1185 void printServer() {
1186   Serial.println(server);
1187 }
1188
1189 void printHeader() {
1190   Serial.println(header);
1191 }
1192
1193 void printRequest() {
1194   Serial.println(request);
1195 }
1196
1197 void printResponse() {
1198   Serial.println(response);
1199 }
1200
1201 void printStatus() {
1202   Serial.println(WiFi.status());
1203 }
1204
1205 void printSSID() {
1206   Serial.println(ssid);
1207 }
1208
1209 void printPassword() {
1210   Serial.println(password);
1211 }
1212
1213 void printServerPort() {
1214   Serial.println(server.port());
1215 }
1216
1217 void printServer() {
1218   Serial.println(server);
1219 }
1220
1221 void printHeader() {
1222   Serial.println(header);
1223 }
1224
1225 void printRequest() {
1226   Serial.println(request);
1227 }
1228
1229 void printResponse() {
1230   Serial.println(response);
1231 }
1232
1233 void printStatus() {
1234   Serial.println(WiFi.status());
1235 }
1236
1237 void printSSID() {
1238   Serial.println(ssid);
1239 }
1240
1241 void printPassword() {
1242   Serial.println(password);
1243 }
1244
1245 void printServerPort() {
1246   Serial.println(server.port());
1247 }
1248
1249 void printServer() {
1250   Serial.println(server);
1251 }
1252
1253 void printHeader() {
1254   Serial.println(header);
1255 }
1256
1257 void printRequest() {
1258   Serial.println(request);
1259 }
1260
1261 void printResponse() {
1262   Serial.println(response);
1263 }
1264
1265 void printStatus() {
1266   Serial.println(WiFi.status());
1267 }
1268
1269 void printSSID() {
1270   Serial.println(ssid);
1271 }
1272
1273 void printPassword() {
1274   Serial.println(password);
1275 }
1276
1277 void printServerPort() {
1278   Serial.println(server.port());
1279 }
1280
1281 void printServer() {
1282   Serial.println(server);
1283 }
1284
1285 void printHeader() {
1286   Serial.println(header);
1287 }
1288
1289 void printRequest() {
1290   Serial.println(request);
1291 }
1292
1293 void printResponse() {
1294   Serial.println(response);
1295 }
1296
1297 void printStatus() {
1298   Serial.println(WiFi.status());
1299 }
1300
1301 void printSSID() {
1302   Serial.println(ssid);
1303 }
1304
1305 void printPassword() {
1306   Serial.println(password);
1307 }
1308
1309 void printServerPort() {
1310   Serial.println(server.port());
1311 }
1312
1313 void printServer() {
1314   Serial.println(server);
1315 }
1316
1317 void printHeader() {
1318   Serial.println(header);
1319 }
1320
1321 void printRequest() {
1322   Serial.println(request);
1323 }
1324
1325 void printResponse() {
1326   Serial.println(response);
1327 }
1328
1329 void printStatus() {
1330   Serial.println(WiFi.status());
1331 }
1332
1333 void printSSID() {
1334   Serial.println(ssid);
1335 }
1336
1337 void printPassword() {
1338   Serial.println(password);
1339 }
1340
1341 void printServerPort() {
1342   Serial.println(server.port());
1343 }
1344
1345 void printServer() {
1346   Serial.println(server);
1347 }
1348
1349 void printHeader() {
1350   Serial.println(header);
1351 }
1352
1353 void printRequest() {
1354   Serial.println(request);
1355 }
1356
1357 void printResponse() {
1358   Serial.println(response);
1359 }
1360
1361 void printStatus() {
1362   Serial.println(WiFi.status());
1363 }
1364
1365 void printSSID() {
1366   Serial.println(ssid);
1367 }
1368
1369 void printPassword() {
1370   Serial.println(password);
1371 }
1372
1373 void printServerPort() {
1374   Serial.println(server.port());
1375 }
1376
1377 void printServer() {
1378   Serial.println(server);
1379 }
1380
1381 void printHeader() {
1382   Serial.println(header);
1383 }
1384
1385 void printRequest() {
1386   Serial.println(request);
1387 }
1388
1389 void printResponse() {
1390   Serial.println(response);
1391 }
1392
1393 void printStatus() {
1394   Serial.println(WiFi.status());
1395 }
1396
1397 void printSSID() {
1398   Serial.println(ssid);
1399 }
1400
1401 void printPassword() {
1402   Serial.println(password);
1403 }
1404
1405 void printServerPort() {
1406   Serial.println(server.port());
1407 }
1408
1409 void printServer() {
1410   Serial.println(server);
1411 }
1412
1413 void printHeader() {
1414   Serial.println(header);
1415 }
1416
1417 void printRequest() {
1418   Serial.println(request);
1419 }
1420
1421 void printResponse() {
1422   Serial.println(response);
1423 }
1424
1425 void printStatus() {
1426   Serial.println(WiFi.status());
1427 }
1428
1429 void printSSID() {
1430   Serial.println(ssid);
1431 }
1432
1433 void printPassword() {
1434   Serial.println(password);
1435 }
1436
1437 void printServerPort() {
1438   Serial.println(server.port());
1439 }
1440
1441 void printServer() {
1442   Serial.println(server);
1443 }
1444
1445 void printHeader() {
1446   Serial.println(header);
1447 }
1448
1449 void printRequest() {
1450   Serial.println(request);
1451 }
1452
1453 void printResponse() {
1454   Serial.println(response);
1455 }
1456
1457 void printStatus() {
1458   Serial.println(WiFi.status());
1459 }
1460
1461 void printSSID() {
1462   Serial.println(ssid);
1463 }
1464
1465 void printPassword() {
1466   Serial.println(password);
1467 }
1468
1469 void printServerPort() {
1470   Serial.println(server.port());
1471 }
1472
1473 void printServer() {
1474   Serial.println(server);
1475 }
1476
1477 void printHeader() {
1478   Serial.println(header);
1479 }
1480
1481 void printRequest() {
1482   Serial.println(request);
1483 }
1484
1485 void printResponse() {
1486   Serial.println(response);
1487 }
1488
1489 void printStatus() {
1490   Serial.println(WiFi.status());
1491 }
1492
1493 void printSSID() {
1494   Serial.println(ssid);
1495 }
1496
1497 void printPassword() {
1498   Serial.println(password);
1499 }
1500
1501 void printServerPort() {
1502   Serial.println(server.port());
1503 }
1504
1505 void printServer() {
1506   Serial.println(server);
1507 }
1508
1509 void printHeader() {
1510   Serial.println(header);
1511 }
1512
1513 void printRequest() {
1514   Serial.println(request);
1515 }
1516
1517 void printResponse() {
1518   Serial.println(response);
1519 }
1520
1521 void printStatus() {
1522   Serial.println(WiFi.status());
1523 }
1524
1525 void printSSID() {
1526   Serial.println(ssid);
1527 }
1528
1529 void printPassword() {
1530   Serial.println(password);
1531 }
1532
1533 void printServerPort() {
1534   Serial.println(server.port());
1535 }
1536
1537 void printServer() {
1538   Serial.println(server);
1539 }
1540
1541 void printHeader() {
1542   Serial.println(header);
1543 }
1544
1545 void printRequest() {
1546   Serial.println(request);
1547 }
1548
1549 void printResponse() {
1550   Serial.println(response);
1551 }
1552
1553 void printStatus() {
1554   Serial.println(WiFi.status());
1555 }
1556
1557 void printSSID() {
1558   Serial.println(ssid);
1559 }
1560
1561 void printPassword() {
1562   Serial.println(password);
1563 }
1564
1565 void printServerPort() {
1566   Serial.println(server.port());
1567 }
1568
1569 void printServer() {
1570   Serial.println(server);
1571 }
1572
1573 void printHeader() {
1574   Serial.println(header);
1575 }
1576
1577 void printRequest() {
1578   Serial.println(request);
1579 }
1580
1581 void
```

```
WiFiAccessPoint_D_V3 | Arduino 1.8.19
File Edit Sketch Tools Help
WiFiAccessPoint_D_V3
1 /*****
2   Rui Santos
3   Complete project details at https://randomnerdtutorials.com
4 *****/
5
6 // Load Wi-Fi library
7 #include <WiFi.h>
8
9 // Replace with your network credentials
10 const char* ssid    = "DEJAN Web Srv V1";
11 const char* password = "123456789";
12
13 // Set web server port number to 80
14 WiFiServer server(80);
15
16 // Variable to store the HTTP request
17 String header;
18
19 // Auxiliar variables to store the current output state
20 String output6State = "off";
21 String output7State = "off";
22
23 // Assign output variables to GPIO pins
24 const int output6 = 6;
25 const int output7 = 7;
26
27 void setup() {
28   Serial.begin(115200);
29   // Initialize the output variables as outputs
30   pinMode(output6, OUTPUT);
31   pinMode(output7, OUTPUT);
32   // Set outputs to LOW
33   digitalWrite(output6, LOW);
34   digitalWrite(output7, LOW);
```

Slika 28: Program za delovanje naše krmilne elektronike

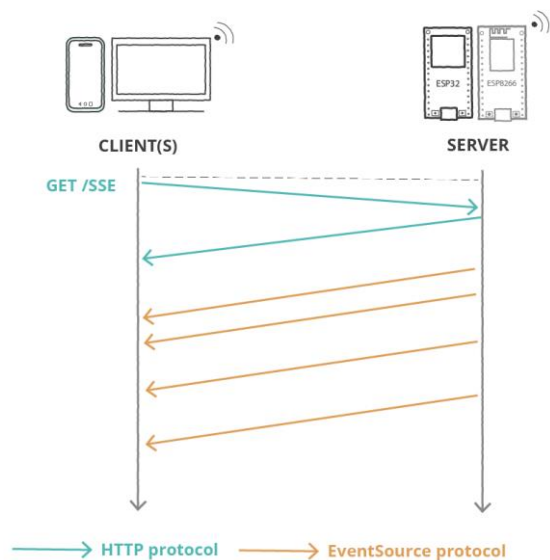
## 7 PRINCIP DELOVANJA:

Naša krmilna elektronika s pomočjo ESP-32 deluje kot strežnik in naša mobilna naprava npr. telefon pa kot odjemalec. To pomeni, da naša mobilna naprava pošlje zahtevo v naš "server" v našem primeru je to ESP-32 in nato mu ta odgovori le v primeru, če dobi zahtevo od naše mobilne naprave drugače ne komunicirata in na osnovi tipke, ki smo jo pritisnili ma mobilni napravi nekaj naredi, to lahko vidimo na sliki 29. To deluje po standardu HTTP.



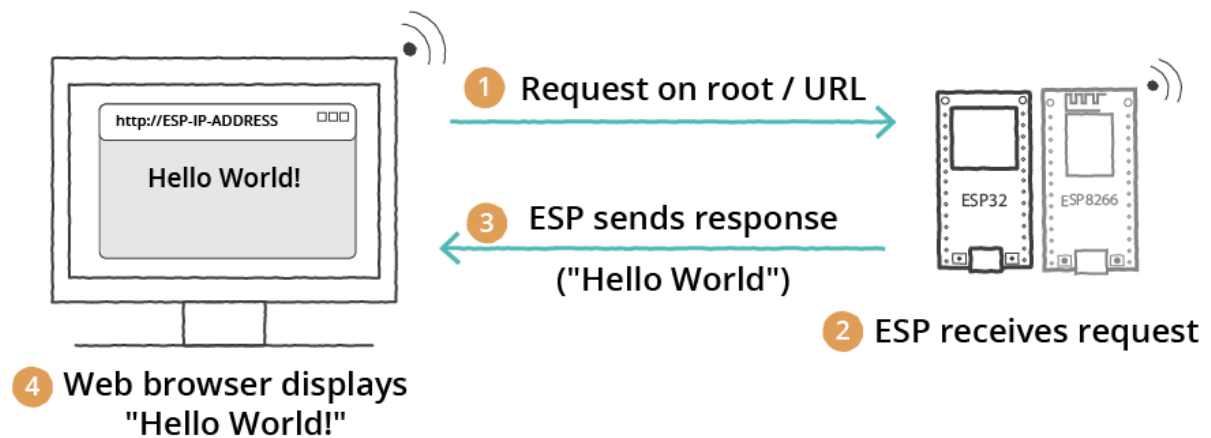
Slika 29: Primer komunikacije med strežnikom in odjemalcem po standardu HTTP

Poznamo pa še metodo Server-Sent Events (SSE), ki pa je zelo koristna, ko želimo le prejemati podatke iz strežnika. Na primer to je zelo uporabno, če želimo v živo spremljati neke podatke kot so recimo temperatura neke sobe, za to le naš odjemalec pošlje zahtevo v strežnik, da ta nazaj pošilja podatke in med tem ne sprejema drugih ukazov torej, na primer, da nam naš "strežnik" v našem primeru ESP-32 le bere senzorske vhode na naši krmilni elektroniki in nam podatke le teh kot je na primer temperatura pošlje v naš odjemalec torej našo mobilno napravo in lahko v živo spremljamo na primer temperaturo nekega prostora.



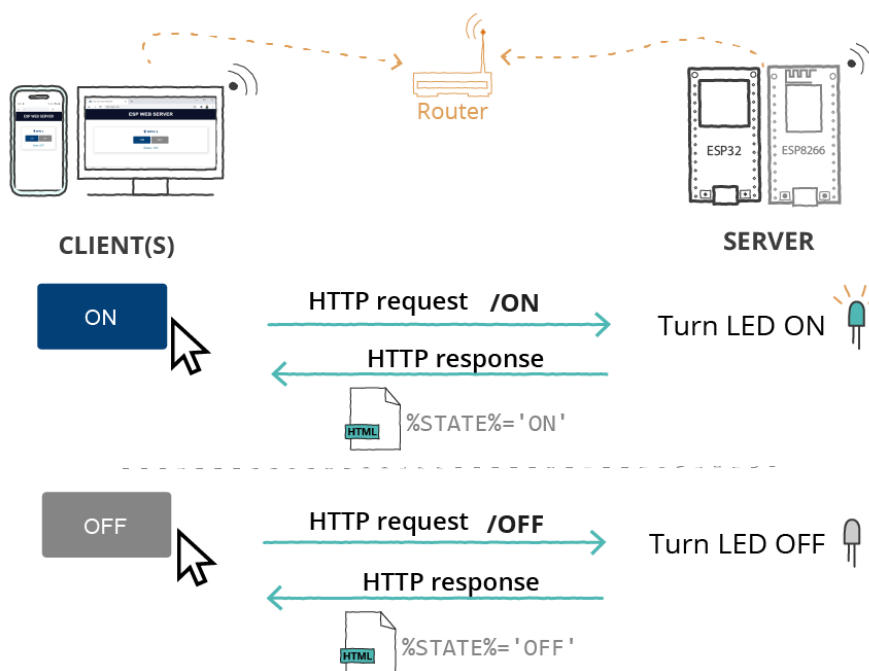
Slika 30: Primer standarda Server-Sent Events (SSE)

Za tem smo naredili prvi testni program, ki smo ga lahko videli prej in deluje tako, da odjemalec torej mobilna naprava pošlje zahtevo v "strežnik" v našem primeru ESP-32 in ta to zahtevo sprejme ter nazaj pošlje odgovor, ki se nam izpiše na mobilni napravi, to je lepo vidno na sliki 32:



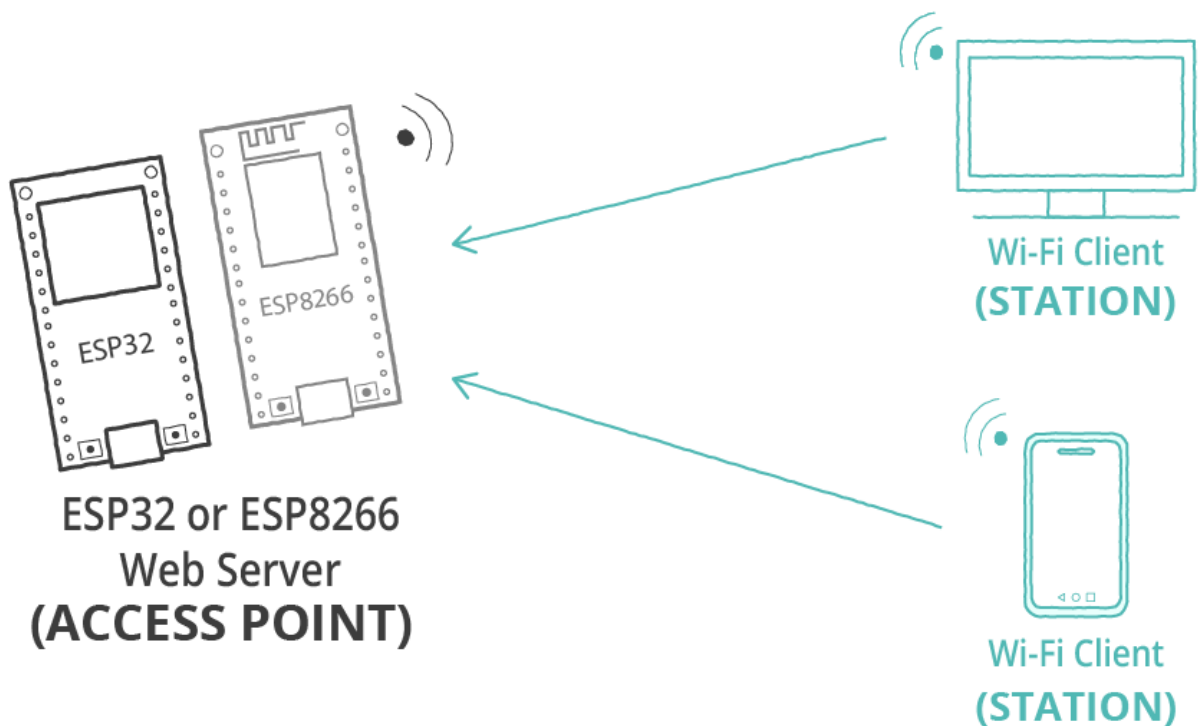
Slika 31: Prvi primer komunikacije med odjemalcem in strežnikom

Nato smo se lotili izdelave programa za testiranje te komunikacije s prižiganjem lučk, kar smo naredili na začetku raziskovalne naloge. Torej naš odjemalec oz. mobilna naprava pošlje podatke in program strežniku kaj naj strežnik naredi ko je pritisnjena tipka. V tem primeru je to prižiganje in ugašanje led luči. Tukaj pa bi še povedal, da je na "strežnik" torej na ESP-32 potrebno naložiti program za spletno aplikacijo in program iz Arduina vsakega posebej saj drugače ne deluje pravilno. Pri programu v Arduino pa je potrebno še program shraniti v pravilno datoteko "data" če jo shranimo drugače potem program prav tako ne bo deloval pravilno.



Slika 32: Prvi test prižiga led luči preko mobilne naprave

Dodal pa bi še, da naša naprava s pomočjo ESP-32, ki služi kot strežnik deluje v načinu access point kar pomeni, da sam ustvari svoje omrežje Wi-Fi z geslom in vsem na katerega se lahko priključijo posamezni odjemalci oz. mobilne naprave, če imajo to geslo. To je lepo vidno na sliki 33.



Slika 33: Access point način delovanja

## **8 DRUŽBENA ODGOVORNOST:**

Naša družbena odgovornost je, da naš krmilnik deluje brez potrebnih dodatnih komunikacijskih kablov, za povezavo s krmilnikom ni potrebna nobena aplikacija ampak samo brskalnik, ki ga ima že vsaka mobilna naprava. S to napravo ohranjamo čisto okolje saj ni potrebe po dodatnih delih kot so instalacija kablov ali vgradnja le teh. Naša naprava odpira možnost proizvodnje in montaže ter s tem odpira nova delovna mesta omogoča pa tudi starejšim in bolnim ljudem, da si olajšajo delo pri upravljanju hiše.

## 9 ZAKLJUČEK:

V zaključku bi radi povedali, da so bile naše raziskave uspešne saj smo potrdili tri od naših štirih hipotez. Potrdili smo prvo hipotezo saj s pomočjo našega izdelka res lahko na enostaven način moderniziramo starejše hiše in novogradnje. Ovrgli smo našo drugo hipotezo, da bi naša krmilna elektronika bila nameščena v elektro omari saj smo ugotovili, da to ni možno saj starejše hiše pa tudi novogradnje nimajo položenih komunikacijskih kablov od vsake naprave do elektro omarice saj je to zelo drago mi pa bi potrebovali te kable za delovanje naše naprave. Potrdili smo našo tretjo hipotezo saj ima naša krmilna elektronika res možnost krmiljenja brez aplikacijskega programa na mobilnih napravah, saj naš izdelek deluje kot strežnik na katerega se povežemo in preko njega zaženemo spletno aplikacijo za telefon. Potrdili pa smo tudi četrto hipotezo saj je naš izdelek res enostaven za uporabo in prilagojen za uporabo za starejše osebe, ki si želijo olajšati delo v svoji hiši ali pa le to modernizirati.

## 10 VIRI IN LITERATURA:

### 10.1. Spletni viri:

- Primeri uporabe za ESP-32 2024.(2.11.2024). Dostopno na: <https://randomnerdtutorials.com/esp32-access-point-ap-web-server/>
- Programska oprema za Arduino 2024.(8.11.2024). Dostopno na: <https://www.arduino.cc/en/software>
- Primeri uporabe za ESP-32 Access poin 2024.(13.11.2024). Dostopno na: <https://randomnerdtutorials.com/esp32-access-point-ap-web-server/>
- Nastavitve za driver-je 2024.(2.11.2024). Dostopno na: <https://www.silabs.com/developer-tools/usb-to-uart-bridge-vcp-drivers>
- Učenje HTML in CSS 2024.(16.11.2024). Dostopno na: <https://www.youtube.com/watch?v=HGTJBPNC-Gw&t=955s>
- Učenje HTML 2024.(17.11.2024). Dostopno na: <https://www.youtube.com/watch?v=kUMe1FH4CHE&t=7618s>
- Učenje HTML kako narediti tipke 2024.(22.11.2024). Dostopno na: <https://www.youtube.com/watch?v=tDqTXipQmBU&t=166s>
- Nastavitve za Arduino za uporabe ESP-32 2024.(13.12.2024). Dostopno na: <https://www.youtube.com/watch?v=wNtGHCrO7E4>
- Nalaganje driver-jev na ESP-32 2024.(15.12.2024). Dostopno na: <https://www.youtube.com/watch?v=JmDxP4O4Trk>

### 10.2. Knjižni viri:

- Build Web App with ESP32 and ESP8266 (Avtorja: Rui Santos & Sara Santos)
- Firebase Web App with ESP32 and ESP8266 (Avtorja: Rui Santos & Sara Santos)
- Learn ESP32 with Arduino IDE 3rd Edition (Avtorja: Rui Santos & Sara Santos)