Univerzitet u Tuzli

Fakultet elektrotehnike

Automatika i robotika

Projektni zadatak

Distribuirani sitemi automatizacije

Tema:

ESP32 Smart home

Studenti:

Azur Jusić

Rifet Gazdić

Nuredin Jahić

Profesor: dr.sc Naser Prljača,red.prof.

Asistent: Mirza Hodžić

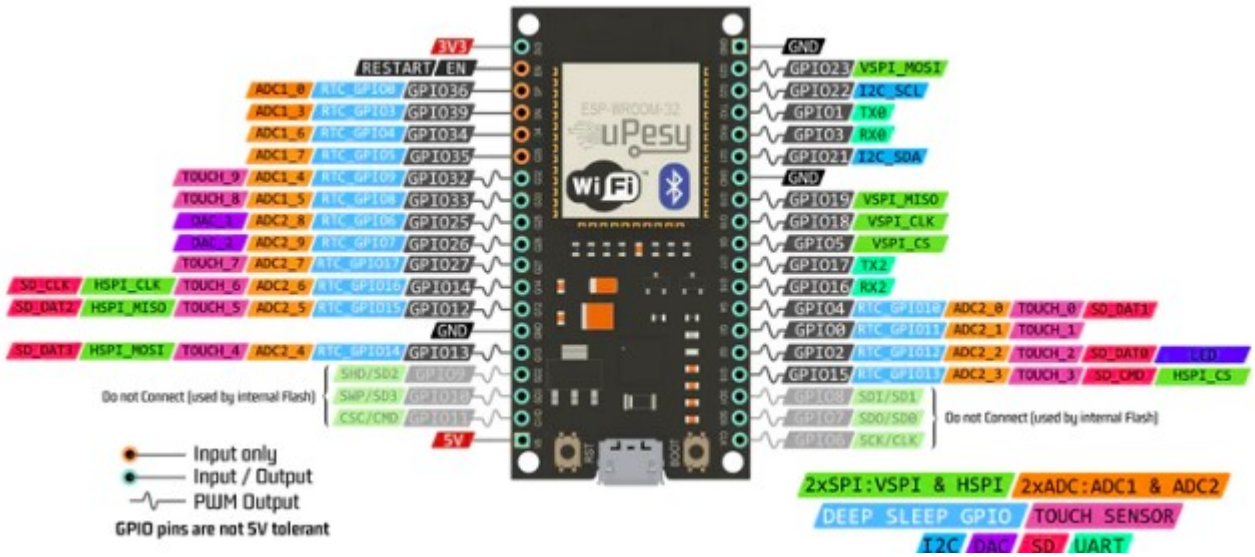
Tuzla,januar 2022

Sadržaj

Popis slika

1.0 Uvod

Ovim projektom pokazat ćemo kako napraviti web server koristeći ESP32 MCU. U našem slucaju korišten je ESP32s (ESP-WROOM-32). Korištene su 3 sijalice koje predstavljaju 3 izlaza koje palimo preko mobitela ili drugog uredjaja pritiskajuci na dugmice (slidere). Projekat može biti idealan primjer pametne kuće gdje umjesto sijalice možemo postaviti releje koji onda mogu da upravljaju paljenjem i gašenjem AC potrosaca u našem domu. U našem primjeru korištena su 3 dugmeta (slidera) koji nam omogućavaju da palimo i gasimo potrosace (u našem slučaju diode). Na svakom pritisku dugmeta pravi se GET request koji onda ide u ESP32 kontroler i tamo se proceusira i na osnovu parametara GET requesta pali se odnosno gasi odgovarajuća dioda. ESP32 je korišten iz razloga sto ima sve komponente potrebne za ovaj projekat. Glavna komponenta zbog koje smo koristili ESP32 umjesto klasičnog arduina je sto ESP32 u sebi ima ugrađen WiFi čip i time nam je olaksao dosta posla. NodeMCU (Node MicroController Unit) je open-source okruženje za razvoj softvera i hardvera izgrađeno oko jeftinog sistema na čipu (SoC) nazvanog ESP32. ESP8266, dizajniran i proizveden od strane Espressif Systems, sadrži ključne elemente računara: CPU, RAM, umrežavanje (WiFi), pa čak i moderan operativni sistem i SDK [1] . To ga čini odličnim izborom za (IoT) projekte svih vrsta. Programira se u mašinskim uputama niskog nivoa koje se mogu tumačiti hardverom čipa [1]. Esp32 će biti programiran u Arduino ide okruženju, te treba voditi brigu da se prije pocetka doda esp32 board u board manageru kao i sve biblioteke koje su korištene.



Slika 1: ESP32s

2.0 Opis problema

2.1 Popis opreme

- ESP32 kao mozak našeg sistema
- **3x** svijetleće diode koje simuliraju signal sa pločice (on/off)
- **3x** otpornika koji nam služe kao current limiting otpornici
- **1x** 0.96“ LCD za prikaz detalja našeg sistema

2.2 Princip rada

ESP32 hosta web server koji prikazuje web stranicu sa tri slajdera. Kada se jedan od slajdera pomakne pravi se HTTP zahtjev prema ESP32 sa novim parametrima.Nakon svakog puta kada pritisnemo slider pravi se novi GET request sa slijedećim parametrima u zavisnosti da li je simulirano paljenje ili gašenje.

/update?output="+element.id+"&state=1

Ili

/update?output="+element.id+"&state=0

- element.id predstavlja broj izlaza koji treba upaliti odnosno ugasiti zavisno od zahtjeva

U zavisnosti od toga koji se slajder pomakne, na pločici će se uključiti jedna od LED dioda, pri čemu postoji mogućnost da se uključe i zajedno.

3.0 Implementacija

Implementiranje koda je vršeno u Arduino IDE okruženju i sastoji se od nekoliko dijelova. Prva stvar koju je potrebno uraditi je da se importuju biblioteke koje služe da bi se napravio WEB server. Te biblioteke su :

- Wifi
- ESPAsyncWebServer
- AsyncTCP

Takođe nam trebaju biblioteke za I2C komunikaciju sa LCD panelom. To su biblioteke:

- SPI
- Wire
- Adafruit_GFX
- Adafruit_SSD1306

3.1 Definicije varijabli

Kako bi koristili LCD potrebno je podesiti odgovarajuće definicije varijabli.

```
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 32 // OLED display height, in pixels
#define OLED_RESET     -1 // Reset pin # (or -1 if sharing Arduino reset pin)
#define SCREEN_ADDRESS 0x3C ///< See datasheet for Address; 0x3D for 128x64, 0x3C for 128x32
```

Slika 2: Definicije varijabli

Pored definicija imamo i konstante koje sluze za kreiranje WEB servera

```
const char* ssid = "Gazda";
const char* password = "rifet123";

const char* PARAM_INPUT_1 = "output";
const char* PARAM_INPUT_2 = "state";
```

Slika 3: Konstante

Pored definicija varijabli imamo i inicijalizaciju Web servera:

```
22
23  // Create AsyncWebServer object on port 80
24  AsyncWebServer server(80);
```

Slika 4: Inicijalizacija web servera

Ovo nam omogucava da kreiramo web server na portu 80.

3.2 WEB stranica

Web stranica za ovaj projekat je prilično jednostavna. Sadrži jedan naslov, jedan pasus i jedan unos raspona tipa. Sav HTML tekst sa uključenim stilovima pohranjen je u varijablu index_html. Sada ćemo proći kroz HTML tekst i vidjeti šta svaki dio radi. Sljedeća <meta> oznaka čini vašu web stranicu responzivnom u bilo kojem pretraživaču.

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

Između oznaka <title> </title> ide naslov našeg web servera.

3.2.1 Stilovi

Između oznaka <style></style> dodajemo CSS kod za stiliziranje web stranice. Pravimo par globalnih stilova za komponente kao i par klasa za slidere.

3.2.2 HTML i Javascript

Unutar oznaka <body></body> dodajemo sadržaj web stranice. Oznake <h2></h2> dodaju naslov web stranici.

```
<h2>Smart Home</h2>
```

Slijedeci HTML kod je definisan kao `%BUTTONPLACEHOLDER%`. Ta vrijednost ce kasnije biti zamijenjena sa 3 dugmica koji se kreiraju na slijedeći način pomoću funkcije processor.

```
// Replaces placeholder with button section in your web page
String processor(const String& var){
  //Serial.println(var);
  if(var == "BUTTONPLACEHOLDER"){
    String buttons = "";
    buttons += "<h4>Dnevni boravak</h4><label class=\"switch\"><input type=\"checkbox\" onchange=\"toggleCheckbox(this)\" id=\"2\" " + outputState(2) + "><span class=\"slider\"></span></label>";
    buttons += "<h4>Kuhinja </h4><label class=\"switch\"><input type=\"checkbox\" onchange=\"toggleCheckbox(this)\" id=\"0\" " + outputState(0) + "><span class=\"slider\"></span></label>";
    buttons += "<h4>Bojler</h4><label class=\"switch\"><input type=\"checkbox\" onchange=\"toggleCheckbox(this)\" id=\"4\" " + outputState(4) + "><span class=\"slider\"></span></label>";
    return buttons;
  }
  return String();
}
```

3.2.3 Kreiranje klizača

Svako dugme ima svoj naslov i odgovarajuće polje kao i label. U nastavku je primjer koda koji kreira jedan od dugmića:

```
<h4>Dnevni boravak</h4><label class=\"switch\"><input type=\"checkbox\" on-
change=\"toggleCheckbox(this)\" id=\"2\" " + outputState(2) + "><span
class=\"slider\"></span></label>
```

Ovaj kod služi kako bi kreirali prvo dugme (sličan kod se koristi i za ostala 2). OnChange funkcija se poziva svaki put kada se klikne na dugme i ta funkcija poziva našu funkciju toggleCheckbox sa odgovarajućim parametrima:

```
<script>function toggleCheckbox(element) {
  var xhr = new XMLHttpRequest();
  if(element.checked){ xhr.open("GET", "/update?output="+element.id+"&state=1", true); }
  else { xhr.open("GET", "/update?output="+element.id+"&state=0", true); }
  xhr.send();
}
```

Element.id = id od dugmeta koje se pritišče koje je ekvivalentno izlazu na ESP32 ploči.

State = Vrijednost koja moze biti 0 ili 1

3. 3 Setup funkcija

Setup funkcija je glavna funkcija u kojoj je definisan način rada naseg kontrolera.Na početku je potrebno podesiti baud rate za serial monitor kao i podesiti iniciajlni text za displej:

```
// Serial port for debugging purposes
Serial.begin(115200);

// initialize OLED display
display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS);
displayText("Welcome - Smart Home");
```

Slika 5: Podešavanja za displej

Nakon toga imamo inicijalizaciju pinova i podesavanje svih pinova na LOW.

```
pinMode(2, OUTPUT);
digitalWrite(2, LOW);
pinMode(0, OUTPUT);
digitalWrite(0, LOW);
pinMode(4 , OUTPUT);
digitalWrite(4, LOW);
```

Slika 6: Inicijalizacija pinova

Slijedeći kod nam omogućava spajanje na wifi mrezu i nakon spajanja na displeju dobijamo ip adresu od spojenog uredjaja.

```
// Connect to Wi-Fi
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(1000);
  Serial.println("Connecting to WiFi..");
  displayText("Connecting to WiFi..");
}

// Print ESP Local IP Address
Serial.println(WiFi.localIP());
displayText("Ip: " + WiFi.localIP().toString());
```

Slika 7: Spajanje na WIFI

Kod za podešavanje index rute:

```
// Route for root / web page
server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
  request->send_P(200, "text/html", index_html, processor);
});
```

Slika 8: Indeks ruta

Nakon sto smo završili sve postavke potrebno je procesuirati GET requeste koje dobijamo preko spojenog uređaja. Koristit ćemo slijedeći kod kako bi to postigli:

```
// Send a GET request to <ESP_IP>/update?output=<inputMessage1>&state=<inputMessage2>
server.on("/update", HTTP_GET, [] (AsyncWebServerRequest *request) {
  String inputMessage1;
  String inputMessage2;
  String Status = "off";
  // GET input1 value on <ESP_IP>/update?output=<inputMessage1>&state=<inputMessage2>
  if (request->hasParam(PARAM_INPUT_1) && request->hasParam(PARAM_INPUT_2)) {
    inputMessage1 = request->getParam(PARAM_INPUT_1)->value();
    inputMessage2 = request->getParam(PARAM_INPUT_2)->value();
    if(inputMessage2.toInt() == 1){
      Status = "on";
    }
    digitalWrite(inputMessage1.toInt(), inputMessage2.toInt());
    displayText("Set output " + inputMessage1 + " to " + Status);

    }
  else {
    inputMessage1 = "No message sent";
    inputMessage2 = "No message sent";
  }
  Serial.print("GPIO: ");
  Serial.print(inputMessage1);
  Serial.print(" - Set to: ");
  Serial.println(inputMessage2);
  request->send(200, "text/plain", "OK");
});
```

Slika 9: Procesuiranje get requesta

Sa svakim novim GET requestom poziva se interna funkcija server.on () koja dobija request i na osnovu requesta vrsi se dalja iteracija.

inputMessage1 = broj izlaza na ploci

inputMessage2 = Status inputa (0|1)

Nakon uspješnog dobijenog GET requesta palimo/gasimo odgovarajuću diodu funkcijom digitalWrite koja kao prvi parametar ima broj izlaza a kao drugi parametar ima vrijednost na koju se izlaz postavlja. Zadnja linija koda iznad vraća request 200 kako bi spojeni uređaj znao da je request bio uspješan. Na kraju naše setup funkcije startujemo server kako bi mogli izvršavati sve gore navedene funkcije. Možete primjetiti da za ispisivanje texta na displeju koristimo funkciju displayText koju smo definisali ovako:

```
void displayText(String text) {
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0,16);
  display.println(text);
  display.display();
}
```

Slika 10: Funkcija za ispis teksta

Funkcija kao parametar prima text i proceusira ga dalje. Funkcija radi na način da se na pocetku obriše sve sa displeja, podesi veličina fonta na 1, boja na WHITE te pozicija kursora na trecem redu (0,16), te nakon toga se poziva funkcija println koja dobija text. Kako bi prikazali taj text potrebno je pozvati funkciju display.display() koja na displej preko I2C komunikacije šalje komandu za ispis texta sa odgovarajucim podešenjima.

3.4 Kod

```
// Import required libraries
#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 32 // OLED display height, in pixels
#define OLED_RESET     -1 // Reset pin # (or -1 if sharing Arduino reset pin)
#define SCREEN_ADDRESS 0x3C ///< See datasheet for Address; 0x3D for 128x64, 0x3C for 128x32
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

// Replace with your network credentials
const char* ssid = "Gazda";
const char* password = "rifet123";

const char* PARAM_INPUT_1 = "output";
const char* PARAM_INPUT_2 = "state";

// Create AsyncWebServer object on port 80
AsyncWebServer server(80);

const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML><html>
<head>
  <title>ESP Web Server</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" href="data:,">
  <style>
    html {font-family: Arial; display: inline-block; text-align: center;}
    h2 {font-size: 3.0rem;}
    p {font-size: 3.0rem;}
    body {max-width: 600px; margin:0px auto; padding-bottom: 25px;}
```

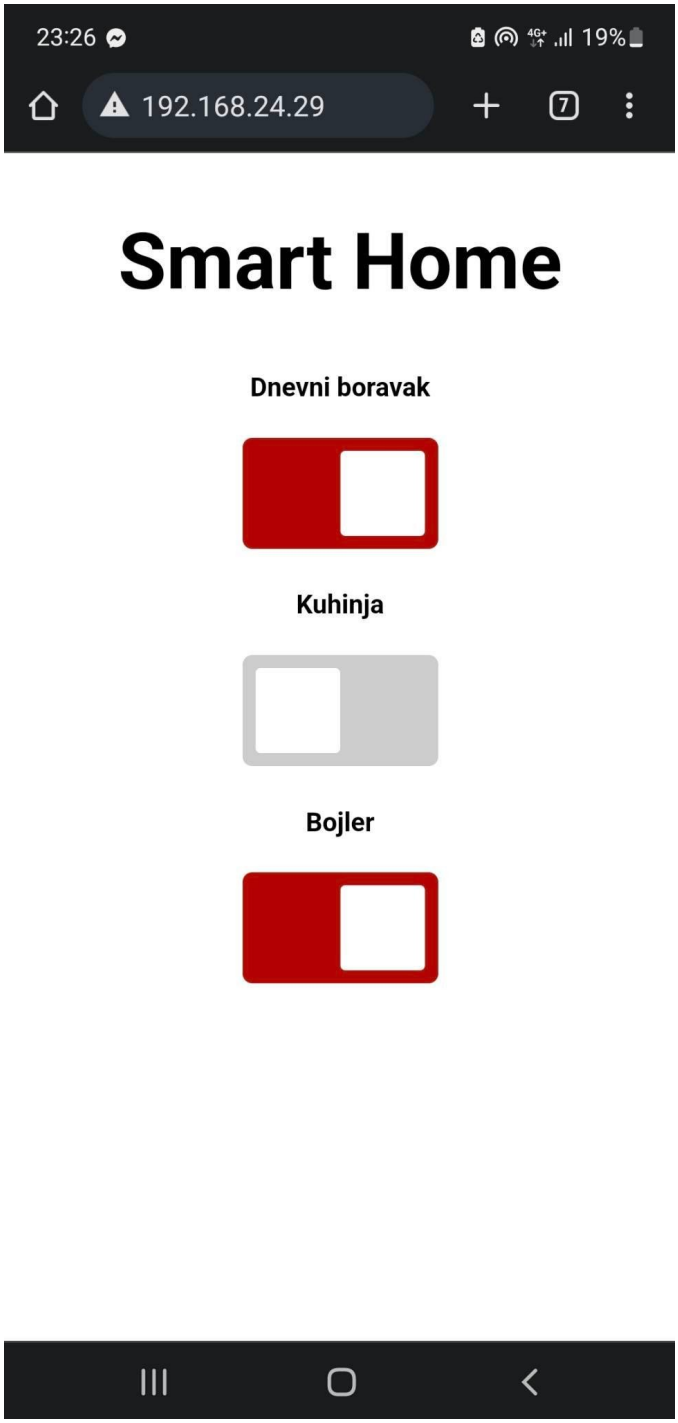
```
    .switch {position: relative; display: inline-block; width: 120px; height: 68px}
    .switch input {display: none}
    .slider {position: absolute; top: 0; left: 0; right: 0; bottom: 0; background-color: #ccc; border-radius: 6px}
    .slider:before {position: absolute; content: ""; height: 52px; width: 52px; left: 8px; bottom: 8px; background-color: #fff; -webkit-transition: .4s; transition: .4s; border-radius: 3px}
    input:checked+.slider {background-color: #b30000}
    input:checked+.slider:before {-webkit-transform: translateX(52px); -ms-transform: translateX(52px); transform: translateX(52px)}
  </style>
</head>
<body>
  <h2>Smart Home</h2>
  %BUTTONPLACEHOLDER%
<script>function toggleCheckbox(element) {
  var xhr = new XMLHttpRequest();
  if(element.checked){ xhr.open("GET", "/update?-output="+element.id+"&state=1", true); }
  else { xhr.open("GET", "/update?output="+element.id+"&state=0", true); }
  xhr.send();
}
</script>
</body>
</html>
)rawliteral";

// Replaces placeholder with button section in your web page
String processor(const String& var){
  //Serial.println(var);
  if(var == "BUTTONPLACEHOLDER"){
    String buttons = "";
    buttons += "<h4>Dnevni boravak</h4><label class=\"switch\"><input type=\"checkbox\" onchange=\"toggleCheckbox(this)\" id=\"2\" " + outputState(2) + "><span class=\"slider\"></span></label>";
    buttons += "<h4>Kuhinja </h4><label class=\"switch\"><input type=\"checkbox\" onchange=\"toggleCheckbox(this)\" id=\"0\" " + outputState(0) + "><span class=\"slider\"></span></label>";
    buttons += "<h4>Bojler</h4><label class=\"switch\"><input type=\"checkbox\" onchange=\"toggleCheckbox(this)\" id=\"4\" " + outputState(4) + "><span class=\"slider\"></span></label>";
    return buttons;
  }
  return String();
}
```

```
String outputState(int output){
  if(digitalRead(output)){
    return "checked";
  }
  else {
    return "";
  }
}

void setup(){
  // Serial port for debugging purposes
  Serial.begin(115200);

  // initialize OLED display
  display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS);
  displayText("Welcome - Smart Home");

  pinMode(2, OUTPUT);
  digitalWrite(2, LOW);
  pinMode(0, OUTPUT);
  digitalWrite(0, LOW);
  pinMode(4 , OUTPUT);
  digitalWrite(4, LOW);

  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi..");
     displayText("Connecting to WiFi..");
  }

  // Print ESP Local IP Address
  Serial.println(WiFi.localIP());
  displayText("Ip: " + WiFi.localIP().toString());

  // Route for root / web page
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/html", index_html, processor);
  });

  // Send a GET request to <ESP_IP>/update?-
output=<inputMessage1>&state=<inputMessage2>
  server.on("/update", HTTP_GET, [] (AsyncWebServerRequest
*request) {
    String inputMessage1;
    String inputMessage2;
    String Status = "off";
```

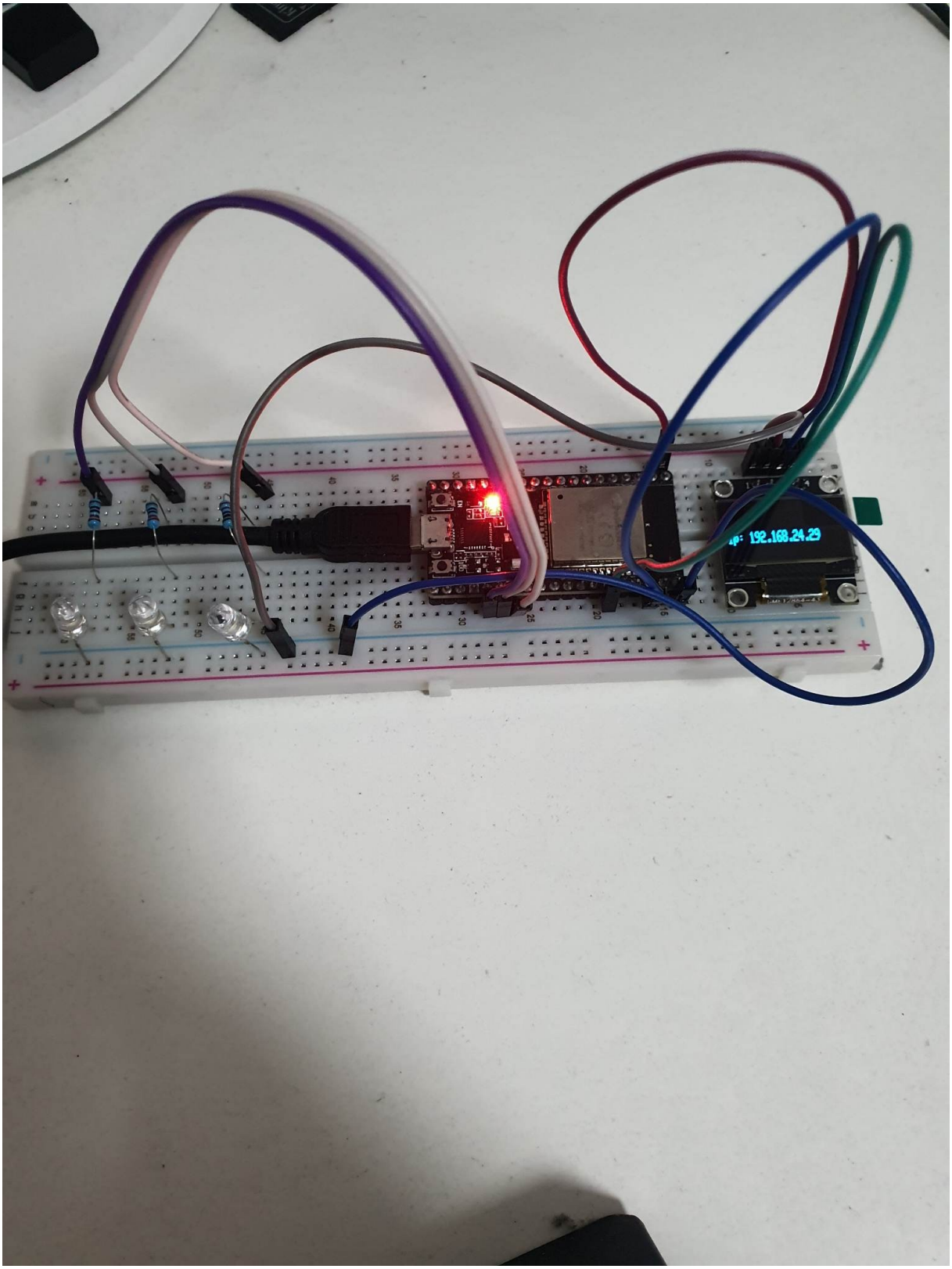
```
118     // GET input1 value on <ESP_IP>/update?-
    output=<inputMessage1>&state=<inputMessage2>
119     if (request->hasParam(PARAM_INPUT_1) && request-
    >hasParam(PARAM_INPUT_2)) {
120       inputMessage1 = request->getParam(PARAM_INPUT_1)->value();
121       inputMessage2 = request->getParam(PARAM_INPUT_2)->value();
122       if(inputMessage2.toInt() == 1){
123         Status = "on";
124       }
125       digitalWrite(inputMessage1.toInt(), inputMessage2.toInt());
126       displayText("Set output " + inputMessage1 + " to " +
    Status);
127
128       }
129     else {
130       inputMessage1 = "No message sent";
131       inputMessage2 = "No message sent";
132     }
133     Serial.print("GPIO: ");
134     Serial.print(inputMessage1);
135     Serial.print(" - Set to: ");
136     Serial.println(inputMessage2);
137     request->send(200, "text/plain", "OK");
138   });
139
140   // Start server
141   server.begin();
142 }
143
144 void displayText(String text) {
145   display.clearDisplay();
146   display.setTextSize(1);
147   display.setTextColor(WHITE);
148   display.setCursor(0,16);
149   display.println(text);
150   display.display();
151 }
152
153 void loop() {
```

Slika 11: Implementirani kod

3.5 Finalni izgled



Slika 12: WEB aplikacija



Slika 13: Izgled projekta

4.0 Literatura

[1] https://www.make-it.ca/nodemcu-details-specifications/

[2] https://randomnerdtutorials.com/esp8266-nodemcu-web-server-slider-pwm/