TEHNIČKA ŠKOLA U ZAGREBU

Zagreb, Palmotićeva 84

ZAVRŠNI RAD

DIGITALNI SAT SA MREŽNOM POVEZANOŠĆU

Mentor:

Siniša Tevelly, dipl.ing.

Učenik:

Niko Pešut, 4e1

Zagreb, lipanj 2023

Digitalni sat sa mrežnom povezanošću

Sadržaj

1. Uvod

Digitalni satovi danas su neizostavni dio našeg svakodnevnog života. Osim što nam pokazuju točno vrijeme, često dolaze s dodatnim funkcijama poput alarmnih funkcija, kalendara i drugih korisnih opcija.

U ovom projektu razvijen je digitalni sat koji koristi hub75 64x32 zaslon i esp32 mikrokontroler te DS3231 RTC sat za prikazivanje vremena i drugih značajki. Jedna od najznačajnijih karakteristika ovog sata je njegova mrežna povezanost, što omogućuje povezivanje s internetom, sinkronizaciju vremena putem mreže te dohvaćanja trenutnih vremenskih uvjeta na specificiranoj lokaciji.

Cilj ovog rada je pokazati da je moguće izgraditi jednostavan, ali funkcionalan digitalni sat koristeći različite tehnologije.

Rad će se u prvom poglavlju baviti s tehnologijama i dijelovima sata, a onda će se fokusirati na njegove funkcionalnosti, te će se na kraju objasniti kôd.

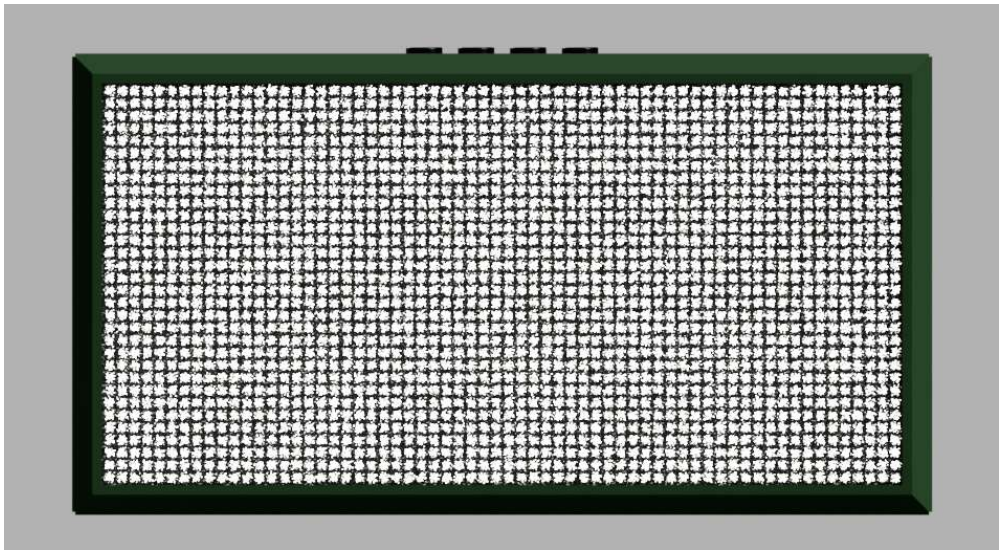
Slika 1: Digitalni sat sa mrežnom povezanošću

2. Tehnologije i dijelovi sata

1.1 CAD dizajn

Sat je sastavljen od 4 stranice kojima je jezgra led matrica (slika 2 i 3). Dizajn je rađen Fusion360 programom.

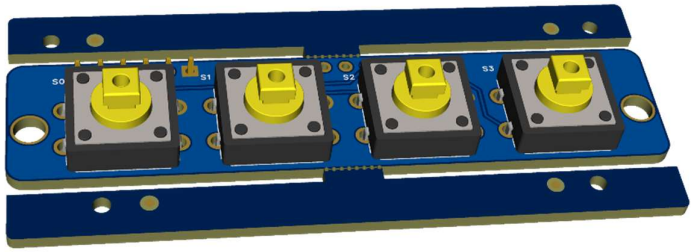
Slika 2: izgled kućišta straga



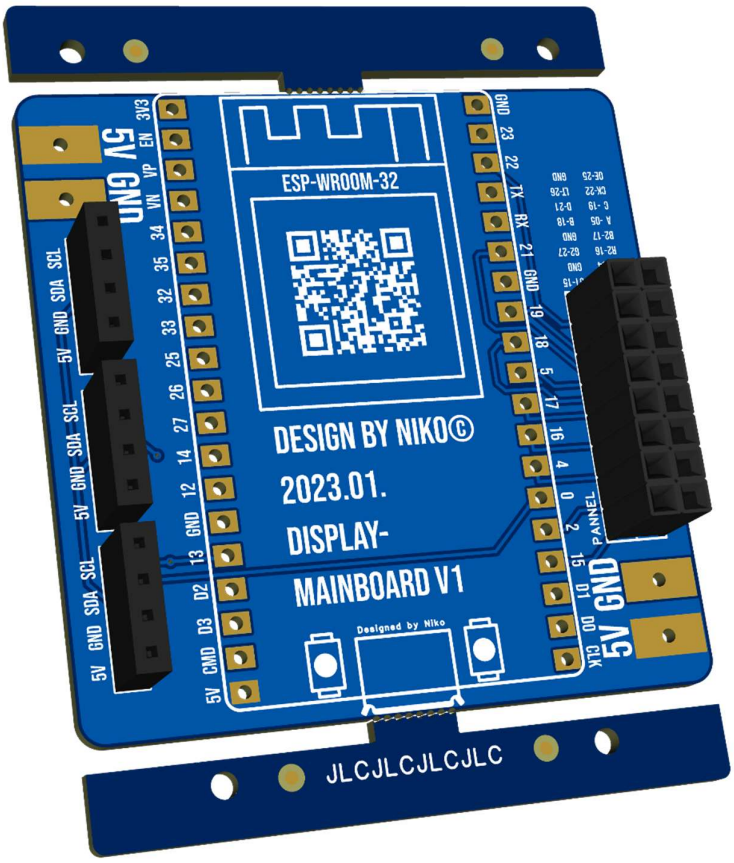
Slika 3: izgled kućišta sprijeda

2.3 Tiskane pločice

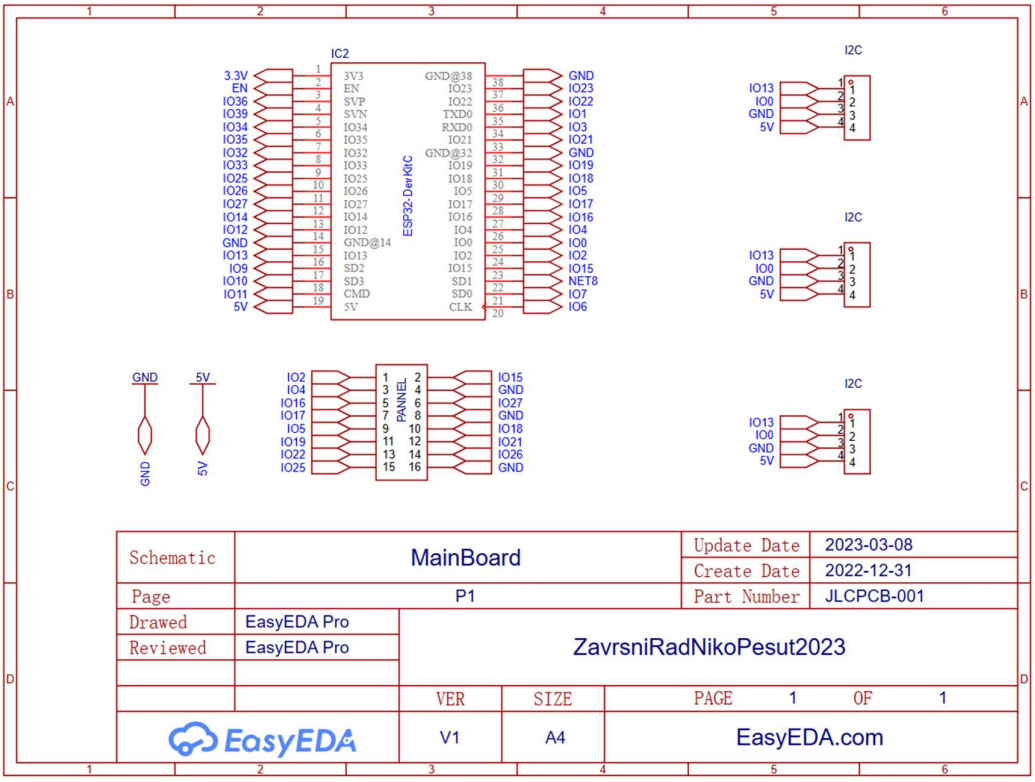
Tiskane pločice, uključujući malu pločicu za gumbe (slika 4 i 7) i veliku pločicu za ESP32 i proširenja (slika 5 i 6), izrađene su putem poznatog kineskog proizvođača JLCPCB-a zbog njihove povoljne izrade. Dizajn pločica napravljen je u programu za projektiranje sklopova EasyEDA Pro.



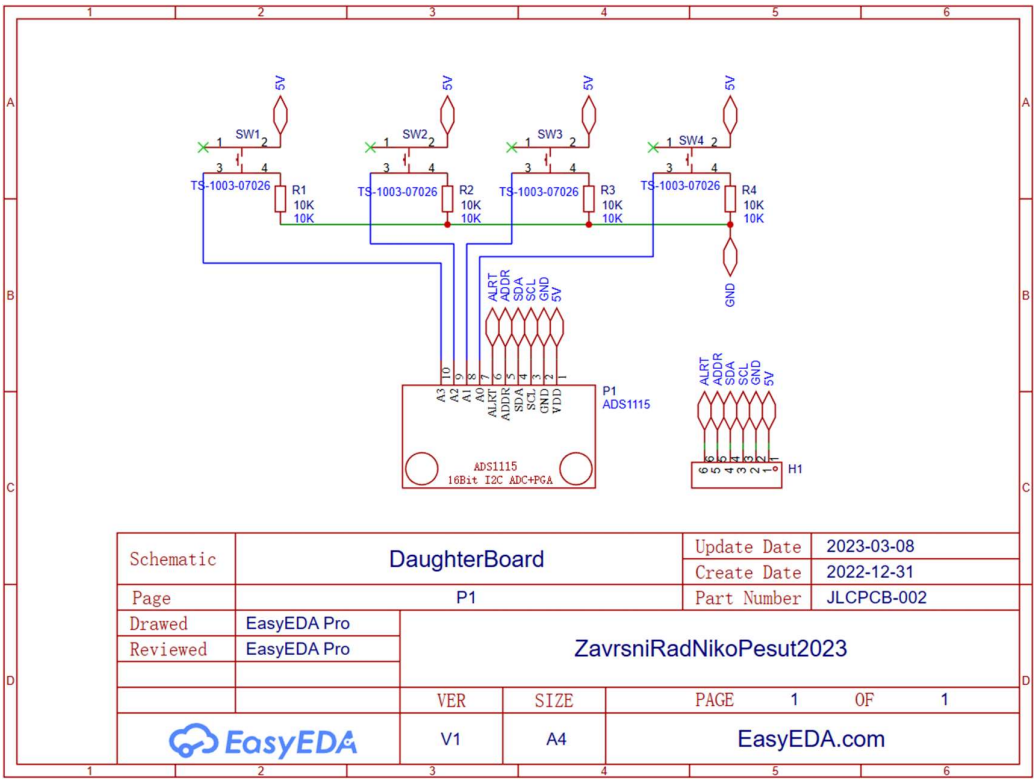
Slika 4: Pločica za gumbe



Slika 5: Pločica za ESP32 i proširenja



Slika 6: Shema pločice za ESP32 i proširenja



Slika 7: Shema pločice za gumbe

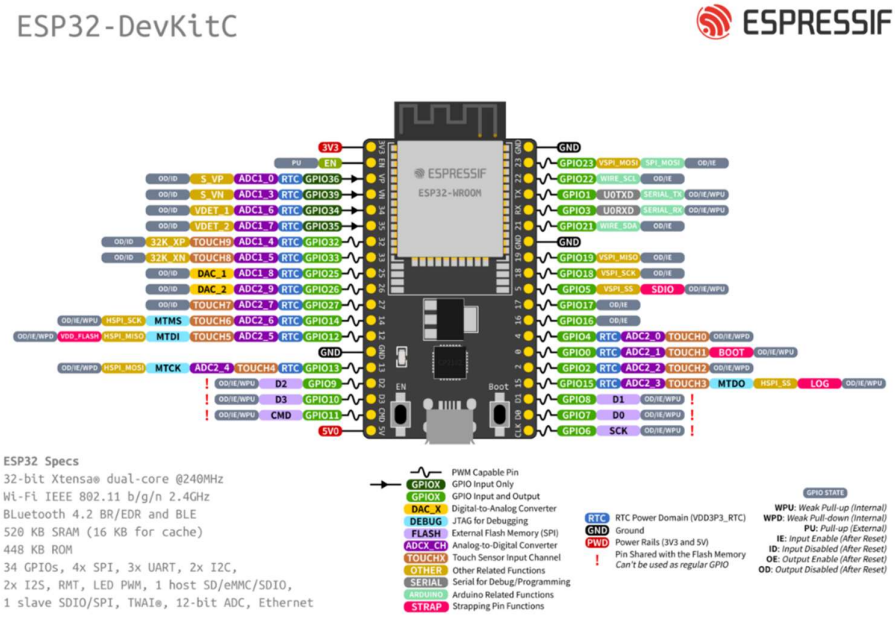
2.4 ESP 32 mikrokontroler

ESP32 mikrokontroler je korišten u ovom projektu za upravljanje digitalnim satom. ESP32 je razvila tvrtka Espressif Systems, kineski proizvođač čipova za bežičnu komunikaciju, koji je poznat po svojim inovativnim rješenjima za IoT uređaje. Pametne kuće, aplikacije, bežični senzori, automobilska industrija su mnogi u nizu koji primjenjuju spomenuti mikroupravljač.

ESP32 devkitc v4 (slika 8) verzija mikrokontrolera korištena u ovom projektu je omogućila višestruke funkcionalnosti digitalnog sata. Jedna od najznačajnijih je njegova sposobnost povezivanja s WiFi mrežom, što je omogućilo povezivanje digitalnog sata s internetom i dohvaćanje podataka s API-ja.

Osim toga, ESP32 je korišten za sinkronizaciju sata putem NTP protokola, što osigurava točno prikazivanje vremena. ESP32 je također omogućio povezivanje s I2C proširenjima, uključujući DS3231 RTC i ADS1115 ADC, što je omogućilo precizno mjerenje vremena i druge podatke koji su prikazani na digitalnom satu.

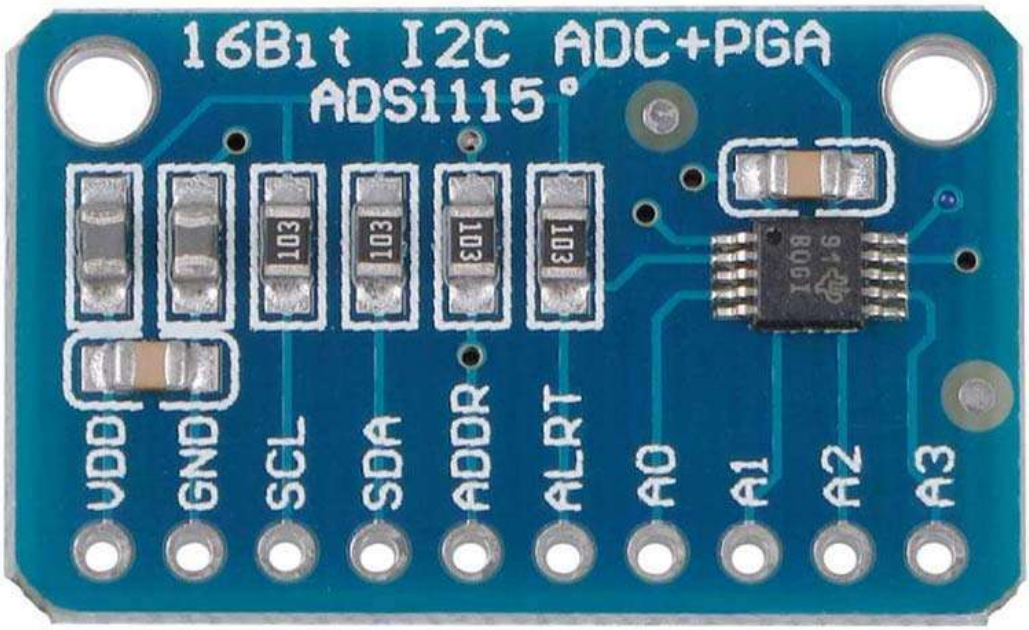
Naposljetku, upravljanje HUB75 64x32 matricom koja je prikazivala vremenske podatke i druge značajke sata također je upotrebljen mikroupravljač.



Slika 8: Prikaz sheme ESP32 devkitc v4 mikrokontrolera

2.5 ADS1115 analogno digitalni pretvornik

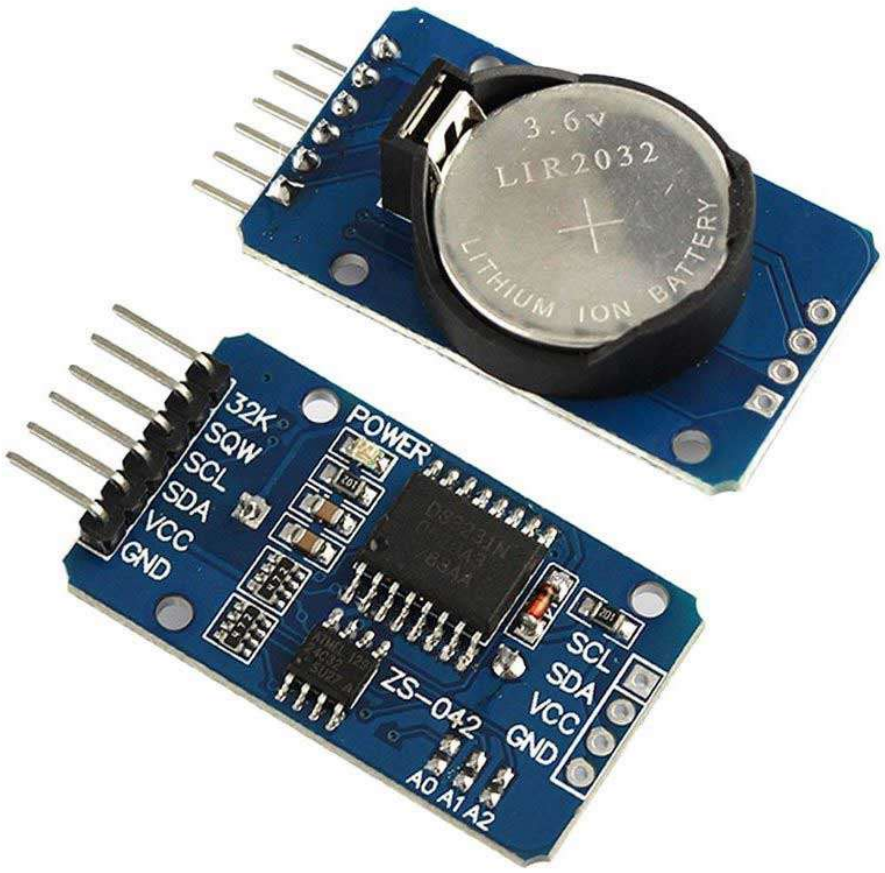
ADS1115 (slika 9) je I2C modul pretvarača analognog signala u digitalni (engl. ADC-Analog-Digital Converter). U ovom projektu, ADS1115 koristi se za detektiranje pritiska gumba koji se nalaze na pločici za upravljanje satom. ADS1115 ima četiri analogna ulaza i može pretvoriti ulazni signal u digitalni oblik na 16-bitnoj rezoluciji, što osigurava precizno mjerenje. Modul ADS1115 je vrlo koristan u situacijama kada je potrebno pretvoriti analogni signal u digitalni, kao što je u ovom slučaju kada se želi očitati pritisak gumba. Također, ADS1115 može biti koristan u drugim projektima kao što su mjerenje temperature i pritiska, detekcija razine tekućine i još mnogo toga. ADS1115 se lako integrira u projekt jer se komunicira putem I2C protokola, što omogućuje jednostavnu povezanost s mikrokontrolerom. Mogao se koristiti i U/I proširivač (engl. I/O expander), ali u trenutku planiranja nije bila dobavljiva verzija s I2C komunikacijom.



Slika 9: Prikaz ADS1115 ADC-a

2.6 DS3231 sat stvarnog vremena

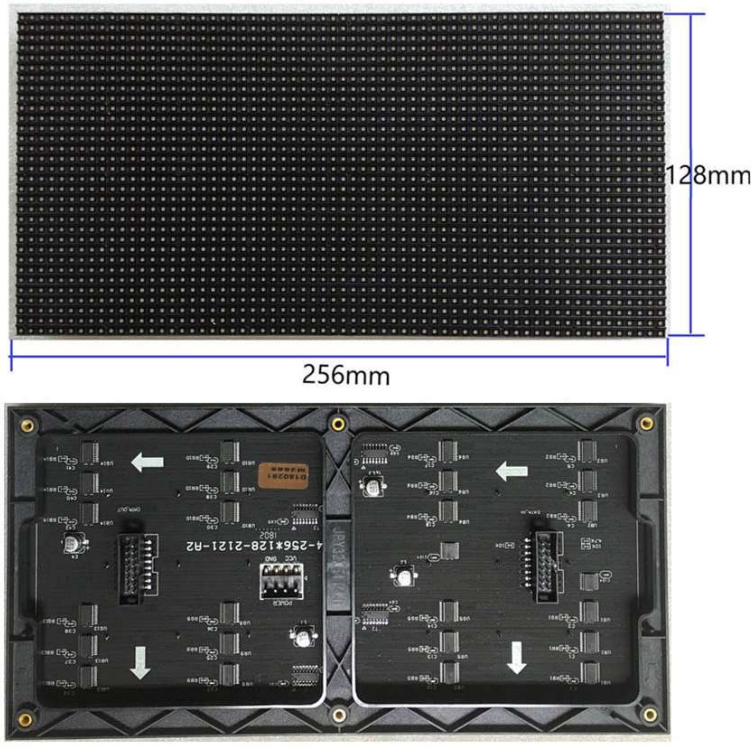
DS3231 RTC (engl. Real-Time Clock) (slika 10) je I2C modul koji se koristi za točno mjerenje vremena i datuma. Ovaj modul ima visoku preciznost i pouzdanost, što ga čini idealnim za korištenje u aplikacijama gdje je potrebna točnost vremena. U ovom projektu, DS3231 RTC modul koristi se za održavanje točnog vremena sata i kalendara. Mikrokontroler (ESP32) periodički čita vrijeme i datum iz DS3231 modula putem I2C komunikacije, te koristi te informacije za točno prikazivanje vremena na HUB75 matrici. DS3231 RTC modul također omogućuje automatsko održavanje ispravnog vremena u slučaju prekida napajanja, što je vrlo korisna funkcija u aplikacijama koje zahtijevaju točno mjerenje vremena.



Slika 10: Prikaz DS3231 RTC-a

2.7 HUB75 64x32 matrica

HUB75 64x32 (slika 11) matrica je LED matrica visoke gustoće koja se sastoji od 2048 pojedinačnih LED dioda, raspoređenih u 64 stupca i 32 reda. Ova matrica može prikazati slike i tekst u visokoj razlučivosti i koristi se u različitim aplikacijama, uključujući LED reklamne ploče, stadionske ekrane i digitalne satove. Matrica se kontrolira pomoću upravljačkog IC-a koji se nalazi na matrici, a podaci se prenose putem paralelnog sučelja koje se zove HUB75. Za prikazivanje teksta ili slike na matrici, potrebno je slati odgovarajuće nizove podataka na HUB75 sučelje, koji će potom biti dekodirani i prikazani na odgovarajućem mjestu na matrici. Za upravljanje matricom u ovom projektu, korišten je ESP32 mikrokontroler, koji se povezuje na matricu putem HUB75 sučelja. Pomoću ESP32 mikrokontrolera, moguće je lako kontrolirati matricu i prikazati različite slike i tekstove na njoj, što čini ovaj projekt idealnim za prikazivanje vremena i drugih korisnih informacija.



Slika 11: Prikaz HUB75 64x32 matrice

3. Funkcije sata

3.1 Postavljanje konfiguracije

Konfiguracija (slika 12) se postavlja tako što se pristupi satu putem web preglednika na adresi 192.168.1.4, ako je upaljena pristupna točka sata, a ako je sat već spojen na mrežu onda konfiguriramo spajanjem na IP adresu koju možemo dobiti pritiskom na prvu tipku na vrhu sata.

Slika 12: Prikaz postavke konfiguracije u web pregledniku

3.2 Dohvaćanje Geo lokacije

Geo lokacija se dohvaća tako što se prilikom konfiguracije pritisne gumb „dobij lokaciju“ i onda se otvara stranica „geoloc.pesut.win“ (slika 13) na kojoj se pritisne gumb i tako se dobije lokacija koja se može prekopirati na konfiguracijsku stranicu.

Slika 13: Prikaz dohvaćanja Geo lokacije u web pregledniku

3.3 Dohvaćanje vremena

Dohvaćanje vremena se odvija tako što se zahtjev s zadanom geografskom širinom, dužinom i API ključem šalje besplatnom API-u (timezonedb.com) s bazom podataka vremena koji nam nazad šalje vrijeme (engl. UNIX timestamp) u obliku XML dokumenta (engl. eXtensible Markup Language) sa specificirane lokacije koje pohranjujemo u DS3231 RTC modul. Dohvaćanje se odvija svakih 5 minuta ako je sat povezan na mrežu.

3.4 Dohvaćanje vremenskih stavki

Dohvaćanje vremena se odvija tako što se zahtjev s zadanom geografskom širinom, dužinom i API ključem šalje API-u (openweathermap.org) koji nam vraća JSON dokument koji sadrži trenutačnu temperaturu, vlagu zraka, vrijeme izlaska i zalaska sunca, trenutačnu brzinu vjetra, tlak zraka, vidljivost te pokrivenost

neba oblacima. Dohvaćanje se odvija svakih 5 minuta ako je sat povezan na mrežu.

3.5 Vremenska crta

Vremenska crta (slika 14) je prikaz izlaza i zalaska sunca, a dobiva se tako što putem podataka dobivenih od API-a preradimo i funkcijom uračunamo trenutačno vrijeme i nacrtamo početak crte kao trenutačno vrijeme. Ona se nalazi u donjem dijelu sata i služi da bismo vidjeli trenutačni položaj sunca na nebu.

Slika 14: Prikaz vremenske crte u 16:30 sati

3.6 Pohrana podataka

Podatci se pohranjuju na internu memoriju ESP32 mikrokontrolera putem SPIFFS biblioteke, a pohranjuju se u obliku JSON (engl. JavaScript Object Notation) (slika 15) datoteke. Podaci koji se pohranjuju su: WiFi ime i šifra, ime i šifra pristupne točke, stanje matrice (prikaz), boje, intenzitet, geografska širina i visina, te API ključevi.

```
{
  "SSID": "WiFi-SSID",
  "PASS": "WiFi-PASS",
  "APssid": "SatZavrsniNiko",
  "APpass": "",
  "stateID": 4,
  "color1Red": 37,
  "color1Green": 167,
  "color1Blue": 105,
  "color2Red": 238,
  "color2Green": 238,
  "color2Blue": 238,
  "color3Red": 244,
  "color3Green": 218,
  "color3Blue": 34,
  "lightIntensity": 200,
  "latitude": 45.88967896,
  "longitude": 15.7041378,
  "weatherApiKey": "weatherApiKljuč",
  "timeApiKey": "timeApiKljuč"
}
```

Slika 15: Prikaz strukture JSON datoteke

3.7 Tipke sata

Na pločici sa gumbima, postoje četiri tipkala.

Pritiskom na prvu tipku prikazuju se detalji o mreži, kao što su trenutna IP adresa i SSID mreže ili pristupne točke ako sat nije spojen na mrežu.

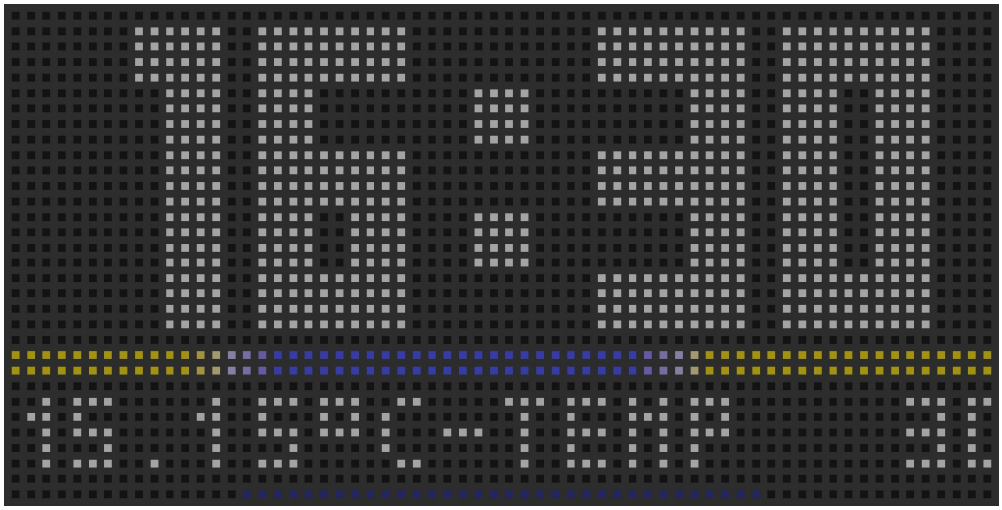
Pritisak na drugu tipku omogućuje prolazak kroz tri zadane boje za matricu. Korisnik tako može prilagoditi izgled sata prema svojim željama i potrebama.

Pritiskom na treću tipku mijenja se intenzitet svjetla, s pet različitih svjetlina koje su dostupne. Ovo je korisna funkcija, posebno u uvjetima smanjene vidljivosti.

Pritisak na četvrtu tipku omogućuje korisniku da promijeni način prikaza na satu.

3.8 Prikaz 1

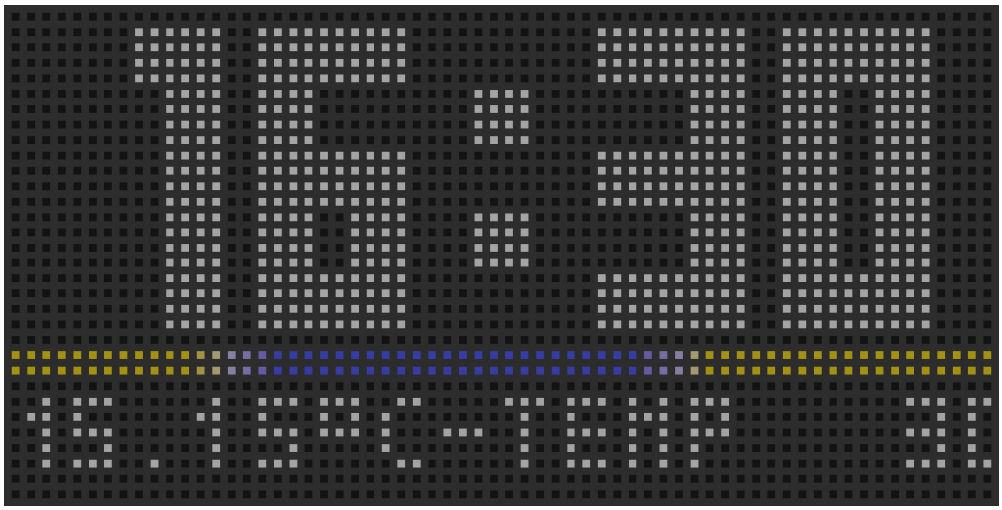
Prikaz 1 (slika 16) sadrži: sat, vremensku crtu, trenutačno vrijeme i WiFi status.



Slika 16: Matrica u 1. prikazu

3.9 Prikaz 2

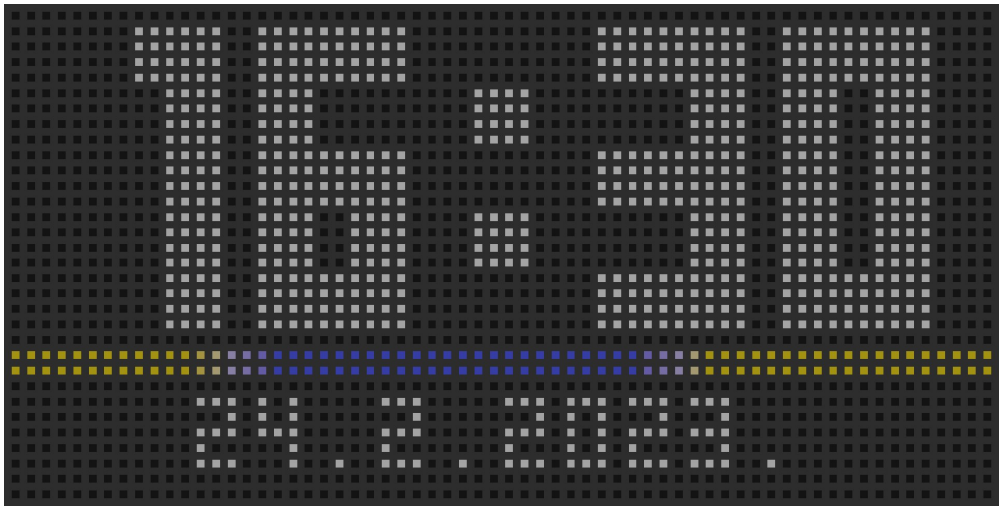
Prikaz 2 (slika 17) sadrži: sat, vremensku crtu i trenutačno vrijeme.



Slika 17: Matrica u 2. prikazu

3.10 Prikaz 3

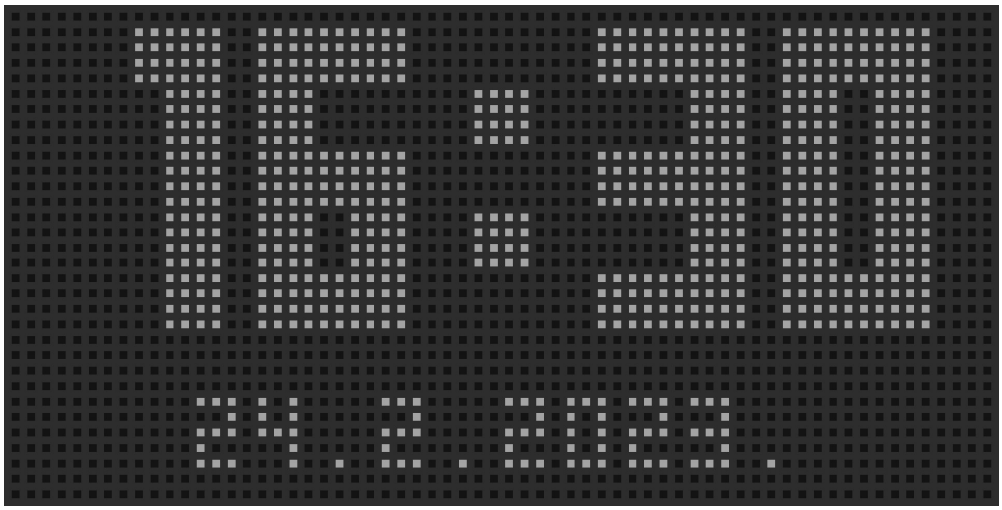
Prikaz 3 (slika 18) sadrži: sat, vremensku crtu i datum.



Slika 18: Matrica u 3. prikazu

3.11 Prikaz 4

Prikaz 4 (slika 19) sadrži: sat i datum.



Slika 19: Matrica u 4. prikazu

4. Kôd

Kôd je pisan u c++ programskom jeziku, te je kompiliran putem PlatformIO (slika 20) proširenja za Visual Studio Code.

Kôd se sastoji od 4 cjeline, a to su: glavni dio programa, konfiguracija i pohrana podataka, HTTP zahtjevi i crtanje po matrici. Još od kôda imamo HTML datoteku za konfiguriranje, te HTML datoteku za dohvaćanje geolokacije koja se nalazi na mojem Raspberry Pi web poslužitelju.

Slika 20: PlatformIO logotip

Slika 21: QR kod na Github repozitorij

Tijekom pisanja rada korišten je Github kao platforma za spremanje rada. Sada je rad javno dostupan putem Github-a (slika 21), a repozitorij sadrži kod, datoteke za izradu tiskanih pločica, datoteke za 3D ispis, program za prikaz slike pomoću brojevnog niza te sam rad.

4.1 Glavni dio programa

Glavni dio programa se nalazi u „main.cpp“ datoteci. Ona sadrži glavno tijelo programa i inicijalizira I2C protokol, matricu, DS3231 RTC modul, ADS1115 ADC modul, WiFi konekciju ili uključuje pristupnu točku ako se ne može spojiti na WiFi i pokreće task manager koji u određenim intervalima poziva funkcije.

4.2 Konfiguracija i pohrana podataka

Konfiguracija i pohrana podataka se odvijaju u „config.h“ datoteci. U njoj se nalaze sve globalne varijable i sve postavke za matricu. Također se u njoj nalaze funkcije za čitanje i pisanje u JSON datoteku koja se nalazi na SPIFFS memoriji ESP32 mikrokontrolera.

4.3 HTTP zahtjevi

HTTP zahtjevi se odvijaju u „HTTP.h“ datoteci. Zahtjevi se obrađuju putem ESPAsyncWebServer biblioteke koja asinkrono sluša zahtjeve od ostalog koda te pri pozivu se aktivira i obrađuje zahtjeve. U ovoj datoteci se nalaze i funkcije za osvježavanje vremena i vremenskih stavki putem API-a, te povezivanje na WiFi mrežu ukoliko se sat odspoji.

4.4 Crtanje po matrici

Crtanje po matrici se odvija u „display.h“ datoteci. Ona sadrži sve funkcije za crtanje po matrici kao što su crtanje sata, broja, slova, teksta, vremenske crte i drugih. Također se može naći funkcija koja osvježava vrijeme sa DS3231 RTC modula i funkcija koja očitava dali je koji od četiri gumba pritisnut.

4.5 Prilog Kôd-a

main.cpp

```
#include <Arduino.h>
#include "config.h"
#include "display.h"
#include "HTTP.h"
TaskHandle_t Task1;


void setup() {
  Wire.begin(I2C_SDA, I2C_SCL);
  adc0.setRate(7);
  if(!SPIFFS.begin(true)) { // Initialize SPIFFS
       Serial.println("An Error has occurred while mounting SPIFFS");
       return;
  }
  Serial.begin(115200);
  Serial.println("JSON LOAD!");
  loadConfig();
  //for(int i=0; i<3; i++) boja1[i]=255;
  //for(int i=0; i<3; i++) boja2[i]=(i!=2?255:0);
  //for(int i=0; i<3; i++) boja3[i]=(i!=1?255:0);
  //saveConfig();
  matrix.addLayer(&backgroundLayer);
  matrix.addLayer(&indexedLayerZ1);
  matrix.addLayer(&indexedLayerZ2);
  matrix.begin();
  matrix.setBrightness(svjetlina);


  indexedLayerZ1.drawString(1,1,1,"Wi-Fi spajanje");
  indexedLayerZ1.drawString(1,26,1,(nName+"  "+nVersion).c_str());
  indexedLayerZ1.swapBuffers();


  rtc.begin();
```

```
 refreshTime();
 myWIFI.begin(SSID.c_str(), PASS.c_str(), APssid.c_str(), APssid.c_str(), APpass.c_str(), "192.168.4.1");
 if (WiFi.status() == WL_CONNECTED)  {
      //mySSDP.begin(SSDP_Name.c_str(), "000000001", nName.c_str(), nVersion.c_str(), "NikoPesut",
"https://kikirikiserver.hopto.org:60001"); Serial.println(F("Start init SSDP"));
   Serial.println(F("Start SSDP"));   //Run SSDP
    //myESPTime.begin(timezone, isDayLightSaving, sNtpServerName, "pool.ntp.org", "time.nist.gov", true,
true);
   clearAllLayers();
   indexedLayerZ1.drawString(0,0,1,"Wi-Fi spojen");
                                          typeText(0,7,(String)("SSID:\n"+(String)myWIFI.getNameSSID()+"\nIP-
WiFi:\n"+myWIFI.getDevStatusIP().substring(9)),&boja2[0]);
   refreshAPIs();
 }
 else
 {
  clearAllLayers();
  indexedLayerZ1.drawString(0,0,1,"Wi-Fi neuspjeh!");
  typeText(0,7,(String)("SSID:\n"+APssid+"\nIP-HOTSPOT:\n192.168.4.1"),&boja2[0]);
 }
 taskManager.scheduleFixedRate(300,refreshAPIs,TIME_SECONDS);//svakih 5 minuta
 taskManager.scheduleFixedRate(15,recconectWifi,TIME_SECONDS);//svakih 15 sekundi
 taskManager.scheduleFixedRate(1,refreshTime,TIME_SECONDS);//svaku 1 sekundi
 taskManager.scheduleFixedRate(100,displayDraw,TIME_MILLIS);//svakih 0.1 sekundi
 taskManager.scheduleFixedRate(99, gumbiSense, TIME_MILLIS);//svakih 0.1 sekundi
 swapAllLayerBuffers();
 httpSetup();
 delay(10000);
 clearAllLayers();
}
void loop(){
 taskManager.runLoop();
}
```

config.h

```
#ifndef GLOBAL_H
#define GLOBAL_H


#include <HTTPClient.h>
#include <MatrixHardware_ESP32_V0.h>
#include <SmartMatrix.h>
#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include "SPIFFS.h"
#include "NetCrtESP.h"
#include <RTClib.h>
#include <Wire.h>
#include <cstring>
#include <ArduinoJson.h>
#include <TaskManagerIO.h>
#include <ADS1115.h>
//#include <TimeLib.h>
//#include <Timezone.h>
//#include "ESPTimeFunc.h"
//#include "time.h"
//#include "ssdpAWS.h"


/**********************JSON_Variables******************/
String filePath = "/myConf.json"; //File for config
String jsonConfig = "{}";
String SSID="", PASS="";
uint8_t stateID=4;
//String mqttPass = "";
//String mqttUsername = "";
//String mqttIP = "";
String APssid = "SatZavrsniNiko";
String APpass = "";
```

```
uint8_t boja1[3]={255,255,255},boja2[3]={128,64,128},boja3[3]={64,64,128}, svjetlina=128;
float lat=0, lng=0;
String timeApiKey="x"; //MCD644HWLRP5
String weatherApiKey="x"; //2f940c078e7a96ecf846d11bba2cbdc7

/**********************Gumbi**************************/
ADS1115 adc0(ADS1115_DEFAULT_ADDRESS);

/***************Serijska Komunikacija******************/
const int I2C_SDA=0,I2C_SCL=13;

/**********************RTC & sat*************************/
RTC_DS3231 rtc;
String daysOfTheWeek[7] = {"NED","PON","UTO","SRI","CET","PET","SUB"};
uint32_t epochNow=0;
bool tocke=true;
//ESPTimeFunc myESPTime(false);
float vrijemeIzlaska=1671261493%86400/60.0/60, vrijemeZalaska=1671301493%86400/60.0/60;
int minuta, sekunda, sat;

/**************************WIFI************************/
NetCrtESP myWIFI;
AsyncWebServer HTTP(80);
//ssdpAWS mySSDP(&HTTP);

String SSDP_Name = "sussyESP";
const String nName = "NikoSat";
const String nVersion = "v0.6.1";



/*********************DISPLAY setup*********************/
#define COLOR_DEPTH 24
const uint16_t kMatrixWidth = 64;
const uint16_t kMatrixHeight = 32;
```

```
const uint8_t kRefreshDepth = 36;
const uint8_t kDmaBufferRows = 4;
const uint8_t kPanelType = SM_PANELTYPE_HUB75_32ROW_MOD16SCAN;
const uint32_t kMatrixOptions = (SM_HUB75_OPTIONS_NONE);
const uint8_t kBackgroundLayerOptions = (SM_BACKGROUND_OPTIONS_NONE);
const uint8_t kScrollingLayerOptions = (SM_SCROLLING_OPTIONS_NONE);
const uint8_t kIndexedLayerOptions = (SM_INDEXED_OPTIONS_NONE);

SMARTMATRIX_ALLOCATE_BUFFERS(matrix, kMatrixWidth, kMatrixHeight, kRefreshDepth, kDmaBufferRows, kPanelType, kMatrixOptions);
SMARTMATRIX_ALLOCATE_BACKGROUND_LAYER(backgroundLayer, kMatrixWidth, kMatrixHeight, COLOR_DEPTH, kBackgroundLayerOptions);
SMARTMATRIX_ALLOCATE_INDEXED_LAYER(indexedLayerZ1, kMatrixWidth, kMatrixHeight, COLOR_DEPTH, kIndexedLayerOptions);
SMARTMATRIX_ALLOCATE_INDEXED_LAYER(indexedLayerZ2, kMatrixWidth, kMatrixHeight, COLOR_DEPTH, kIndexedLayerOptions);

//const int defaultBrightness = (100*255)/100;       // full (100%) brightness
//const int defaultBrightness = (15*255)/100;      // dim: 15% brightness
const int defaultBrightness = (20*255)/100;      // dim: 15% brightness
const int defaultScrollOffset = 0;
const rgb24 defaultBackgroundColor = {0x40, 0, 0};
#define BLACK   0x0000
#define BLUE    0x001F
#define RED     0xF800
#define GREEN   0x07E0
#define CYAN    0x07FF
#define MAGENTA 0xF81F
#define YELLOW  0xFFE0
#define WHITE   0xFFFF

/******************DISPLAY.h varijable*******************/
int dan=0, noc=0;
bool gumbStanje=0, saveStanje=0;
```

```
uint8_t zadnje_stisnuto=0;
String statusTrenutno;
int statusTrenutniPomak=kMatrixWidth;

/********************HTTP.h varijable********************/
float tempCurr=0;
float humCurr=0;
float windCurr=0;
int pressCurr=0;
int visibilityCurr=0;
int skyCurr=0;

/************************ JSON ************************/
bool loadConfig();
bool saveConfig();

bool loadConfig() {
 Serial.println("-----");
 Serial.println("Load!");
 Serial.println("-----");
  if(!SPIFFS.exists(filePath)) {
     Serial.println(F("Failed to open config file"));
     saveConfig();
     return false;
  }
  File configFile = SPIFFS.open(filePath, "r");
  size_t size = configFile.size();
  if (size > 2048) {
     Serial.print(F("Config file size is too large: ")); Serial.println(size);
     configFile.close();
     return false;
  }
  jsonConfig = configFile.readString();
  configFile.close();
```

```
DynamicJsonDocument jsonDoc(5096); //4096
DeserializationError error = deserializeJson(jsonDoc, jsonConfig);
if (error) {
 Serial.print(F("loadConfig() deserializeJson() failed with code "));
 Serial.println(error.c_str());
 return false;
}
JsonObject root = jsonDoc.as<JsonObject>();

//network
SSID = root["SSID"].as<String>();
PASS = root["PASS"].as<String>();

APssid = root["APssid"].as<String>();
APpass = root["APpass"].as<String>();
//mqttPass = root["mqttPass"].as<String>();
//mqttUsername = root["mqttUsername"].as<String>();
//mqttIP = root["mqttIP"].as<String>();

//display
stateID = root["stateID"];
boja1[0] = root["color1Red"] ; boja1[1] = root["color1Green"]; boja1[2] = root["color1Blue"];
boja2[0] = root["color2Red"] ; boja2[1] = root["color2Green"]; boja2[2] = root["color2Blue"];
boja3[0] = root["color3Red"] ; boja3[1] = root["color3Green"]; boja3[2] = root["color3Blue"];
svjetlina = root["lightIntensity"].as<uint8_t>();

//lokacija
lat=root["latitude"].as<float>();
lng=root["longitude"].as<float>();

//api-s
weatherApiKey = root["weatherApiKey"].as<String>();
timeApiKey = root["timeApiKey"].as<String>();
```

```
  return true;
}

bool saveConfig() {
 Serial.println("-----");
 Serial.println("Save!");
 Serial.println("-----");
  DynamicJsonDocument jsonDoc(5096); //4096
  DeserializationError error = deserializeJson(jsonDoc, jsonConfig);
  if (error) {
    Serial.print(F("saveConfig() deserializeJson() failed with code "));
    Serial.println(error.c_str());
    return false;
  }
  JsonObject json = jsonDoc.as<JsonObject>();



  //network mqttUsername mqttIP mqttPass
  json["SSID"] = SSID;
  json["PASS"] = PASS;
  json["APssid"] = APssid;
  json["APpass"] = APpass;
  //json["mqttPass"] = mqttPass;
  //json["mqttUsername"] = mqttUsername;
  //json["mqttIP"] = mqttIP;

  //display
  json["stateID"] = stateID;
  json["color1Red"] = boja1[0]; json["color1Green"] = boja1[1]; json["color1Blue"] = boja1[2];
  json["color2Red"] = boja2[0]; json["color2Green"] = boja2[1]; json["color2Blue"] = boja2[2];
  json["color3Red"] = boja3[0]; json["color3Green"] = boja3[1]; json["color3Blue"] = boja3[2];
  json["lightIntensity"] = svjetlina;

  //lokacija
```

```
  json["latitude"]=lat;
  json["longitude"]=lng;

  //api-s
  json["weatherApiKey"] = weatherApiKey;
  json["timeApiKey"] = timeApiKey;

  serializeJson(json, jsonConfig);
  File configFile = SPIFFS.open(filePath, "w");
  if (!configFile) {
    Serial.println(F("Failed to open config file for writing"));
    configFile.close();
    return false;
  }
  serializeJson(json, configFile);
  serializeJson(json, Serial);
  configFile.close();
  return true;
}


/************************Podatci************************/

bool
brojke[60][13]={{1,1,1,1,1,1,1,1,1,1,1,1,1},{1,1,1,1,1,1,1,1,1,1,1,1,1},{1,1,0,0,0,0,0,0,1,1},{1,1,1,1,1,1,1,1,1,1},{1,1,1,1,1,1
,1,1,1,1},{0,0,0,0,0,0,0,0,0,0},{0,0,0,0,0,0,0,0,0,0},{1,1,0,0,0,0,0,0,0,0},{1,1,1,1,1,1,1,1,1,1},{1,1,1,1,1,1,1,1,1,1
},{1,1,0,0,1,1,1,1,1,1},{1,1,0,0,1,1,1,1,1,1},{1,1,0,0,1,1,0,0,1,1},{1,1,1,1,1,1,0,0,1,1},{1,1,1,1,1,1,0,0,1,1},{1,1,0,
0,1,1,0,0,1,1},{1,1,0,0,1,1,0,0,1,1},{1,1,0,0,1,1,0,0,1,1},{1,1,1,1,1,1,1,1,1,1},{1,1,1,1,1,1,1,1,1,1},{1,1,1,1,1,1,0,
0,0,0},{1,1,1,1,1,1,0,0,0,0},{0,0,0,0,1,1,0,0,0,0},{1,1,1,1,1,1,1,1,1,1},{1,1,1,1,1,1,1,1,1,1},{1,1,1,1,1,1,0,0,1,1},{
1,1,1,1,1,1,0,0,1,1},{1,1,0,0,1,1,0,0,1,1},{1,1,0,0,1,1,1,1,1,1},{1,1,0,0,1,1,1,1,1,1},{1,1,1,1,1,1,1,1,1,1},{1,1,1,1,
1,1,1,1,1,1},{1,1,0,0,1,1,0,0,1,1},{1,1,0,0,1,1,1,1,1,1},{1,1,0,0,1,1,1,1,1,1},{1,1,1,1,0,0,0,0,0,0},{1,1,1,1,0,0,0,0,
0,0},{1,1,0,0,0,0,0,0,0,0},{1,1,1,1,1,1,1,1,1,1},{1,1,1,1,1,1,1,1,1,1},{1,1,1,1,1,1,1,1,1,1},{1,1,1,1,1,1,1,1,1,1},{1,
1,0,0,1,1,0,0,1,1},{1,1,1,1,1,1,1,1,1,1},{1,1,1,1,1,1,1,1,1,1},{1,1,1,1,1,1,0,0,1,1},{1,1,1,1,1,1,0,0,1,1},{1,1,0,0,1,
1,0,0,1,1},{1,1,1,1,1,1,1,1,1,1},{1,1,1,1,1,1,1,1,1,1},{0,0,0,0,0,0,0,0,0,0},{0,0,1,1,0,0,1,1,0,0},{0,0,1,1,0,0,1,1,0,
```

```
0},{0,0,0,0,0,0,0,0,0,0},{0,0,0,0,0,0,0,0,0,0},{0,0,0,0,0,0,0,0,0,0},{0,0,1,1,0,0,1,1,0,0},{0,0,1,1,0,0,1,1,0,0},{0,0,
0,0,0,0,0,0,0,0},{0,0,0,0,0,0,0,0,0,0}};

bool
miniLetters[285][5]={{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0},{1,1,1,0,1},{0,0,0,0,0},{1,1,0,0,0},{0,0,0,0,0},
{1,1,0,0,0},{1,1,1,1,1},{0,1,0,1,0},{1,1,1,1,1},{1,1,1,0,1},{1,1,1,1,1},{1,0,1,1,1},{1,0,0,1,1},{0,0,1,0,0},{1,1,0,0,1},
{0,1,0,1,0},{1,0,1,0,1},{0,1,0,1,1},{0,0,0,0,0},{1,1,0,0,0},{0,0,0,0,0},{0,0,0,0,0},{0,1,1,1,0},{1,0,0,0,1},{1,0,0,0,1},
{0,1,1,1,0},{0,0,0,0,0},{1,1,1,0,0},{1,0,1,0,0},{1,1,1,0,0},{0,0,1,0,0},{0,1,1,1,0},{0,0,1,0,0},{0,0,0,0,0},{0,1,0,0,0},
{1,0,0,0,0},{0,0,1,0,0},{0,0,1,0,0},{0,0,1,0,0},{0,0,0,0,0},{0,0,0,0,1},{0,0,0,0,0},{0,0,0,1,1},{0,1,1,1,0},{1,1,0,0,0},
{1,1,1,1,1},{1,0,0,0,1},{1,1,1,1,1},{0,0,0,0,0},{0,1,0,0,0},{1,1,1,1,1},{1,0,1,1,1},{1,0,1,0,1},{1,1,1,0,1},{1,0,1,0,1},
{1,0,1,0,1},{1,1,1,1,1},{1,1,1,0,0},{0,0,1,0,0},{1,1,1,1,1},{1,1,1,0,1},{1,0,1,0,1},{1,0,1,1,1},{1,1,1,1,1},{1,0,1,0,1},
{1,0,1,1,1},{1,0,0,0,0},{1,0,0,0,0},{1,1,1,1,1},{1,1,1,1,1},{1,0,1,0,1},{1,1,1,1,1},{1,1,1,0,1},{1,0,1,0,1},{1,1,1,1,1},
{0,0,0,0,0},{0,1,0,1,0},{0,0,0,0,0},{0,0,0,0,1},{0,1,0,1,0},{0,0,0,0,0},{0,0,1,0,0},{0,1,0,1,0},{1,0,0,0,1},{0,1,0,1,0},
{0,1,0,1,0},{0,1,0,1,0},{1,0,0,0,1},{0,1,0,1,0},{0,0,1,0,0},{1,0,0,0,0},{1,0,1,0,1},{1,1,1,0,0},{0,1,0,1,1},{1,0,1,0,1},
{1,1,1,0,1},{1,1,1,1,1},{1,0,1,0,0},{1,1,1,1,1},{1,1,1,1,1},{1,0,1,0,1},{0,1,0,1,0},{0,1,1,1,0},{1,0,0,0,1},{1,0,0,0,1},
{1,1,1,1,1},{1,0,0,0,1},{0,1,1,1,0},{1,1,1,1,1},{1,0,1,0,1},{1,0,1,0,1},{1,1,1,1,1},{1,0,1,0,0},{1,0,1,0,0},{0,1,1,1,0},
{1,0,0,0,1},{1,0,1,1,1},{1,1,1,1,1},{0,0,1,0,0},{1,1,1,1,1},{1,0,0,0,1},{1,1,1,1,1},{1,0,0,0,1},{0,0,0,1,1},{0,0,0,0,1},
{1,1,1,1,1},{1,1,1,1,1},{0,0,1,0,0},{1,1,0,1,1},{1,1,1,1,1},{0,0,0,0,1},{0,0,0,0,1},{1,1,1,1,1},{0,1,0,0,0},{1,1,1,1,1},
{1,1,1,1,1},{1,0,0,0,0},{1,1,1,1,1},{1,1,1,1,1},{1,0,0,0,1},{1,1,1,1,1},{1,1,1,1,1},{1,0,1,0,0},{1,1,1,0,0},{1,1,1,1,1},
{1,1,1,1,1},{0,0,0,0,1},{1,1,1,1,1},{1,0,1,1,0},{1,1,1,0,1},{1,1,1,0,1},{1,0,1,0,1},{1,0,1,1,1},{1,0,0,0,0},{1,1,1,1,1},
{1,0,0,0,0},{1,1,1,1,1},{0,0,0,0,1},{1,1,1,1,1},{1,1,1,1,1},{0,0,0,1,1},{1,1,1,1,0},{1,1,1,1,1},{0,0,0,1,0},{1,1,1,1,1},
{1,1,0,1,1},{0,0,1,0,0},{1,1,0,1,1},{1,1,1,0,0},{0,0,1,1,1},{1,1,1,0,0},{1,0,0,1,1},{1,0,1,0,1},{1,1,0,0,1},{0,0,0,0,0},
{1,1,1,1,1},{1,0,0,0,1},{1,1,0,0,0},{0,1,1,1,0},{0,0,0,1,1},{1,0,0,0,1},{1,1,1,1,1},{0,0,0,0,0},{0,1,0,0,0},{1,0,0,0,0},
{0,1,0,0,0},{0,0,0,0,1},{0,0,0,0,1},{0,0,0,0,1},{0,0,0,0,0},{1,0,0,0,0},{0,1,0,0,0},{1,1,1,1,1},{1,0,1,0,0},{1,1,1,1,1},
{1,1,1,1,1},{1,0,1,0,1},{0,1,0,1,0},{0,1,1,1,0},{1,0,0,0,1},{1,0,0,0,1},{1,1,1,1,1},{1,0,0,0,1},{0,1,1,1,0},{1,1,1,1,1},
{1,0,1,0,1},{1,0,1,0,1},{1,1,1,1,1},{1,0,1,0,0},{1,0,1,0,0},{0,1,1,1,0},{1,0,0,0,1},{1,0,1,1,1},{1,1,1,1,1},{0,0,1,0,0},
{1,1,1,1,1},{1,0,0,0,1},{1,1,1,1,1},{1,0,0,0,1},{0,0,0,1,1},{0,0,0,0,1},{1,1,1,1,1},{1,1,1,1,1},{0,0,1,0,0},{1,1,0,1,1},
{1,1,1,1,1},{0,0,0,0,1},{0,0,0,0,1},{1,1,1,1,1},{0,1,0,0,0},{1,1,1,1,1},{1,1,1,1,1},{1,0,0,0,0},{1,1,1,1,1},{1,1,1,1,1},
{1,0,0,0,1},{1,1,1,1,1},{1,1,1,1,1},{1,0,1,0,0},{1,1,1,0,0},{1,1,1,1,1},{1,1,1,1,1},{0,0,0,0,1},{1,1,1,1,1},{1,0,1,1,0},
{1,1,1,0,1},{1,1,1,0,1},{1,0,1,0,1},{1,0,1,1,1},{1,0,0,0,0},{1,1,1,1,1},{1,0,0,0,0},{1,1,1,1,1},{0,0,0,0,1},{1,1,1,1,1},
{1,1,1,1,1},{0,0,0,1,1},{1,1,1,1,0},{1,1,1,1,1},{0,0,0,1,0},{1,1,1,1,1},{1,1,0,1,1},{0,0,1,0,0},{1,1,0,1,1},{1,1,1,0,0},
{0,0,1,1,1},{1,1,1,0,0},{1,0,0,1,1},{1,0,1,0,1},{1,1,0,0,1},{0,0,1,0,0},{1,1,1,1,1},{1,0,0,0,1},{0,0,0,0,0},{1,1,1,1,1},
{0,0,0,0,0},{1,0,0,0,1},{1,1,1,1,1},{0,0,1,0,0},{0,0,1,0,0},{0,1,1,0,0},{0,0,0,0,0}};


#endif
```

HTTP.h

```
void httpSetup();
void updateWeather();
void refreshAPIs();
void updateTime();
void recconectWifi();

void recconectWifi()
{
  if(WiFi.isConnected()==0)
    myWIFI.reconectWIFI();
}

 //timezonedb.com api
void updateTime()
{
  if(WiFi.status()== WL_CONNECTED)
  {
    HTTPClient http;
                                        String                timeApiPath="http://api.timezonedb.com/v2.1/get-time-
zone?key="+timeApiKey+"&format=xml&by=position&lat="+(String)lat+"&lng="+(String)lng;
    http.begin(timeApiPath.c_str());
    int httpResponseCode = http.GET();
    if (httpResponseCode>0) {
      Serial.print("HTTP Response code: ");
      Serial.println(httpResponseCode);
      String payload = http.getString();
      int brojda1=payload.indexOf("<timestamp>")+strlen("<timestamp>");
      int brojda2=payload.indexOf("</timestamp>");
      //Serial.println(payload);
      uint64_t vrijemeSad=strtoul(payload.substring(brojda1,brojda2).c_str(),nullptr,0);
      Serial.println(vrijemeSad);
      rtc.adjust(vrijemeSad);
```

```
   }
   else {
    Serial.print("Error code: ");
    Serial.println(httpResponseCode);
   }
   http.end();
 }
 else {
   Serial.println("WiFi Disconnected");
 }
}

//vrijemeZalaska vrijemeIzlaska
void updateWeather()
{
 if(WiFi.status()== WL_CONNECTED)
 {
   HTTPClient http;
                                                                                                    String
timeApiPath="https://api.openweathermap.org/data/2.5/weather?lat="+(String)lat+"&lon="+(String)lng+"&
appid="+weatherApiKey+"&units=metric";
   http.begin(timeApiPath.c_str());
   int httpResponseCode = http.GET();
   if (httpResponseCode>0) {
    Serial.print("HTTP Response code: ");
    Serial.println(httpResponseCode);
    String payload = http.getString();
    DynamicJsonDocument jsonDoc(5096);
    DeserializationError error = deserializeJson(jsonDoc, payload);
    Serial.println(payload);
    if (error) {
     Serial.print(F("updateWeather() deserializeJson() failed with code "));
     Serial.println(error.c_str());
    }
```

```
    else
    {
     JsonObject root = jsonDoc.as<JsonObject>();
     tempCurr=(float)root["main"]["temp"];
     humCurr=(float)root["main"]["humidity"];
     vrijemeIzlaska=(((int)root["sys"]["sunrise"]+(int)root["timezone"])%86400/60.0/60);
     vrijemeZalaska=(((int)root["sys"]["sunset"]+(int)root["timezone"])%86400/60.0/60);
     windCurr=(float)root["wind"]["speed"];
     pressCurr=(int)root["main"]["pressure"];
     visibilityCurr=(int)root["visibility"];
     skyCurr=(int)root["clouds"]["all"];
    }
   }
   else {
    Serial.print("Error code: ");
    Serial.println(httpResponseCode);
   }
   http.end();
 }
 else {
   Serial.println("WiFi Disconnected");
 }
}
void notFound(AsyncWebServerRequest *request) {
     request->send(404, "text/html", "<!doctype html><html><head><meta http-equiv=\"Content-Type\" content=\"text/html;charset=UTF-8\" /></head> <body>Stranica nije pronađena!<br> <a href=\"/\"> <input id=\"posaljitelj\" type=\"submit\" value=\"Početna stranica!\" style=\"height: 4em; width: 10em;\"></a></body></html>");
}
void httpSetup()
{

 HTTP.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
     request->send(SPIFFS, "/index.html", "text/html");
```

```
  });
  HTTP.on("/reset", HTTP_GET, [](AsyncWebServerRequest *request){
                                        request->send(200,"text/html",          "<!doctype          html>
<html><script>window.location.replace(\"/\");</script></html>");
    delay(1500);
    ESP.restart();
  });

  HTTP.on("/get", HTTP_GET, [] (AsyncWebServerRequest *request) {
    String tmp;
    bool changedAny=0;
    if (request->hasParam("latitude") && request->getParam("latitudechange")->value()=="1") {
      lat=atof((request->getParam("latitude")->value()).c_str());
      changedAny=1;
    }
    if (request->hasParam("longitude") && request->getParam("longitudechange")->value()=="1") {
      lng=atof((request->getParam("longitude")->value()).c_str());
      changedAny=1;
    }
    if (request->hasParam("boja1") && request->getParam("boja1change")->value()=="1") {
      tmp=(request->getParam("boja1")->value());
      tmp = tmp.substring(1, tmp.length() );
      int r,g,b;
      sscanf(tmp.c_str(), "%02x%02x%02x", &r, &g, &b);
      boja1[0]=r;boja1[1]=g;boja1[2]=b;
      changedAny=1;
    }
    if (request->hasParam("boja2") && request->getParam("boja2change")->value()=="1") {
      tmp=request->getParam("boja2")->value();
      tmp = tmp.substring(1, tmp.length() );
      int r,g,b;
      sscanf(tmp.c_str(), "%02x%02x%02x", &r, &g, &b);
      boja2[0]=r;boja2[1]=g;boja2[2]=b;
      changedAny=1;
```

```
  }
  if (request->hasParam("boja3") && request->getParam("boja3change")->value()=="1") {
   tmp=request->getParam("boja3")->value();
   tmp = tmp.substring(1, tmp.length() );
   int r,g,b;
   sscanf(tmp.c_str(), "%02x%02x%02x", &r, &g, &b);
   boja3[0]=r;boja3[1]=g;boja3[2]=b;
   changedAny=1;
  }
  if (request->hasParam("SSID") && request->getParam("SSIDchange")->value()=="1")  {
   SSID=request->getParam("SSID")->value();
   changedAny=1;
  }
  if (request->hasParam("PASSWORD") && request->getParam("PASSWORDchange")->value()=="1") {
   PASS=request->getParam("PASSWORD")->value();
   changedAny=1;
  }
  if (request->hasParam("APSSID") && request->getParam("APSSIDchange")->value()=="1") {
   APssid=request->getParam("APSSID")->value();
   changedAny=1;
  }
  if (request->hasParam("APPASSWORD") && request->getParam("APPASSWORDchange")->value()=="1") {
   APpass=request->getParam("APPASSWORD")->value();
   changedAny=1;

  }
  if (request->hasParam("timezonedb") && request->getParam("timezonedbchange")->value()=="1") {
   timeApiKey=request->getParam("timezonedb")->value();
   changedAny=1;
  }
      if (request->hasParam("openweathermap") && request->getParam("openweathermapchange")-
>value()=="1") {
   weatherApiKey=request->getParam("openweathermap")->value();
   changedAny=1;
```

```
  }
  if (request->hasParam("mod") && request->getParam("modchange")->value()=="1") {
   stateID=(request->getParam("mod")->value()).toInt();
   changedAny=1;
  }
  if(changedAny)
  {
   saveConfig();
   refreshAPIs();
   myWIFI.setConfigWIFI(SSID.c_str(), PASS.c_str(), APssid.c_str(), APssid.c_str(), APpass.c_str());
  }
   tmp="<!doctype html><head><meta http-equiv=\"Content-Type\" content=\"text/html;charset=UTF-8\" /></head>";
  tmp+="<style>input[type=button]{width: 75%; height: 1.6em;}";
  tmp+="html{background-color: #DBDBDB;color: #376DAB;font-family: monospace;font-weight: bolder;}";
  tmp+="input:hover{transition: 0.5s; border: 2px brown solid;}input:focus{background-color: yellow;}";
   tmp+="input{font-size: 2vw;height: 1.5em;padding: auto;margin: 0.1em;color: #376DAB; font-family: monospace;background-color: #EE635240;border: 2px transparent solid;border-radius: 0.2em;font-weight: bolder;position: relative; width: 100%; transition: 0.5s;padding: 0px;height: 1.6em;}</style>";
  tmp+="<body><div style=\"display: flex; flex-direction: column;\"><h1>Poslano!</h1> <br> <a href=\"/\"> <input id=\"posaljitelj\" type=\"submit\" value=\"Vrati se nazad!\" style=\"height: 3em;  width: 99%;\"></a>";
   tmp+="<br><br><br> <form action=\"/reset\"> <a href=\"/\"> <input id=\"posaljitelj\" type=\"submit\" value=\"Restart Sata? (tako se wifi ponovo spaja na novi)\" style=\"height: 3em; width: 99%;\"></a></form><br> </div><body>";

  //debug!
  if(1==0)
  {
  tmp+="<table>";
  tmp+="<tr><th>Varijabla</th><th>Vrijednost</th></tr>";
  tmp+="<tr><td>Geografska Širina</td><td>"+(String)lat+"</td></tr>";
  tmp+="<tr><td>Geografska Dužina</td><td>"+(String)lng+"</td></tr>";
                        tmp+="<tr><td>Boja1</td><td>("+(String)boja1[0]+",          "+(String)boja1[1]+", "+(String)boja1[2]+")</td></tr>";
```

```
                tmp+="<tr><td>Boja2</td><td>("+(String)boja2[0]+",  "+(String)boja2[1]+",
"+(String)boja2[2]+")</td></tr>";
                tmp+="<tr><td>Boja3</td><td>("+(String)boja3[0]+",  "+(String)boja3[1]+",
"+(String)boja3[2]+")</td></tr>";
   tmp+="<tr><td>WiFi SSID</td><td>"+(String)myWIFI.getNameSSID()+"</td></tr>";
   tmp+="<tr><td>WiFi Šifra</td><td>"+(String)myWIFI.getPassSSID()+"</td></tr>";
   tmp+="<tr><td>HotSpot SSID</td><td>"+(String)myWIFI.getNameAPSSID()+"</td></tr>";
   tmp+="<tr><td>HotSpot Šifra</td><td>"+(String)myWIFI.getPassAPSSID()+"</td></tr>";
   tmp+="<tr><td>Timezonedb Api</td><td>"+(String)timeApiKey+"</td></tr>";
   tmp+="<tr><td>openweathermap Api</td><td>"+(String)weatherApiKey+"</td></tr>";
   tmp+="<tr><td>ssdp</td><td>"+(String)myWIFI.getNameSSDP()+"</td></tr>";
   tmp+="</table>";
   }
   request->send(200,"text/html", tmp);

 });
 HTTP.onNotFound(notFound);
 HTTP.begin();

}
void refreshAPIs()
{
   updateTime();
   updateWeather();
}
```

display.h

```
void clearAllLayers();
void drawBroj(int x, int y, int znak, int scaling, uint8_t *boje);
void drawClock(int x, int y, int scaling, uint8_t *boje);
void drawLetter(int x, int y, char znak, uint8_t *boje);
void typeText(int x, int y, String text, uint8_t *boje);
void drawDaytimeLine(int x1, int y1, int x2, int y2, float sat, int povecanje);
void typeTextNoNl(int x, int y, String text, uint8_t *boje);
void drawLetterNoNl(int x, int y, char znak, uint8_t *boje);
```

```
void refreshTime();
void swapAllLayerBuffers();
void clearAllLayerBuffers();
bool gumbPress(uint8_t ulaz);
void gumbiSense();
void displayDraw();

void drawDaytimeLine(int x1, int y1, int x2, int y2, float sat, int povecanje=60)
{
    if (vrijemeZalaska<vrijemeIzlaska)vrijemeZalaska+=24;
    int satx=0;
    for(int i=x1; i<x2; i++)
    {
     satx=i/64.0*24+sat;
     if(satx>=24) satx-=24;
     if(satx>vrijemeIzlaska and satx<vrijemeZalaska)
     {
      if(dan!=255)
      {
       dan+=(dan+povecanje>255?255%povecanje-dan%povecanje:povecanje);
       noc-=(noc-povecanje<0?255%povecanje-dan%povecanje:povecanje);
      }
     }
     else
     {
      if(noc!=255)
      {
       noc+=(noc+povecanje>255?255%povecanje-noc%povecanje:povecanje);
       dan-=(dan-povecanje<0?255%povecanje-dan%povecanje:povecanje);
      }
     }

     backgroundLayer.drawFastVLine(i, y1, y2,{dan,dan,noc});
    }
```

```
}


void drawLetter(int x, int y, char znak, uint8_t *boje)
{
  int trenutnoSlovo=((int)znak)-32;
  for(int j=0 ;j<5; j++)
  {
    int pomBr=0;
     for(int i=trenutnoSlovo*3; i<3+trenutnoSlovo*3; i++)
     {
        int brojkeX=i;
        int ypos=y+j;
        int xpos=x+pomBr++;
                                backgroundLayer.drawPixel(xpos,  ypos,  {*(boje)*miniLetters[brojkeX][j]  ,
*(boje+1)*miniLetters[brojkeX][j], *(boje+2)*miniLetters[brojkeX][j]});
     }
  }
}
void typeText(int x, int y, String text, uint8_t *boje)
{
  int xpoc=x;
  for(int i=0; i<text.length(); i++)
  {
    if((char)text[i]=='\n'){y=y+6; x=xpoc; i+=1; if(i>=text.length()) return;}


    drawLetter(x, y, (char)text[i], boje);
    if(x+6<kMatrixWidth) x+=4; else if(y+11<kMatrixHeight) {y=y+6; x=xpoc;} else return;


  }
}


void drawLetterNoNl(int x, int y, char znak, uint8_t *boje)
{
  int trenutnoSlovo=((int)znak)-32;
```

```
 for(int j=0 ;j<5; j++)
 {
   int pomBr=0;
    for(int i=trenutnoSlovo*3; i<3+trenutnoSlovo*3; i++)
    {
       int brojkeX=i;
       int ypos=y+j;
       int xpos=x+pomBr++;
       if(x>=0 and x<=kMatrixWidth)backgroundLayer.drawPixel(xpos, ypos, {*(boje)*miniLetters[brojkeX][j] ,
*(boje+1)*miniLetters[brojkeX][j], *(boje+2)*miniLetters[brojkeX][j]});
    }
 }
}
void typeTextNoNl(int x, int y, String text, uint8_t *boje)
{
 int xpoc=x;
 for(int i=0; i<text.length(); i++)
 {
   drawLetterNoNl(x, y, (char)text[i], boje);
   x+=4;
 }
}
void drawBroj(int x, int y, int znak, int scaling, uint8_t *boje)
{
   for(int j=0; j<10; j++)
   {
     int pomBr=0;
      for(int i=znak*5; i<5+znak*5; i++)
      {
         int brojkeX=i;
         int ypos=y+j*scaling;
         int xpos=x+pomBr;
         pomBr+=scaling;
         for(int i1=0; i1<scaling; i1++)for(int j1=0; j1<scaling; j1++)
```

```
                                backgroundLayer.drawPixel(xpos+i1,    ypos+j1,    {*(boje)*brojke[brojkeX][j]    ,
*(boje+1)*brojke[brojkeX][j], *(boje+2)*brojke[brojkeX][j]});
      }
   }
}

void drawClock(int x, int y, int scaling, uint8_t *boje)
{

      String   vrijemeIspisa=(String)((sat-sat%10)/10)+   (String)(sat%10)+(tocke   ?   ":":"   ")+(String)((minuta-
minuta%10)/10)+ (String)(minuta%10);



   int tockaMinus=0;
   for(int i=0;i<vrijemeIspisa.length();i++)
    if(vrijemeIspisa[i]==':' and tocke)
      {drawBroj((x+i*6*scaling+tockaMinus), y, 11, scaling, boje); tockaMinus-=scaling;}
    else
    if(vrijemeIspisa[i]!=' ' and vrijemeIspisa[i]!=':')
      drawBroj((x+i*6*scaling+tockaMinus), y, ((String)vrijemeIspisa[i]).toInt(), scaling, boje);
    else
      tockaMinus-=scaling;
//Serial.println("!da!");
//Serial.println(tocke);
}

void clearAllLayers() {
 backgroundLayer.fillScreen((rgb24)BLACK);
 backgroundLayer.swapBuffers();
 indexedLayerZ1.fillScreen(BLACK);
 indexedLayerZ1.swapBuffers();
 indexedLayerZ2.fillScreen(BLACK);
 indexedLayerZ2.swapBuffers();
}
```

```
void clearAllLayerBuffers() {
  backgroundLayer.fillScreen((rgb24)BLACK);
  indexedLayerZ1.fillScreen(BLACK);
  indexedLayerZ2.fillScreen(BLACK);
}
void swapAllLayerBuffers() {
  backgroundLayer.swapBuffers();
  indexedLayerZ1.swapBuffers();
  indexedLayerZ2.swapBuffers();
}

void refreshTime()
{

  DateTime now = rtc.now();
  sat=now.hour();
  minuta=now.minute();
  sekunda=now.second();

  tocke=!tocke;
}

bool gumbPress(uint8_t ulaz)
{
  adc0.setMultiplexer(ulaz);
  if(adc0.getMilliVolts(false)>1000)
    return true;
  else
    return false;
}

void gumbiSense()
{
  if(++zadnje_stisnuto>2) //svakih cca 0.5s
```

```
{
  gumbStanje=0;
}
bool g0=gumbPress(ADS1115_MUX_P0_NG);
bool g1=gumbPress(ADS1115_MUX_P1_NG);
bool g2=gumbPress(ADS1115_MUX_P2_NG);
bool g3=gumbPress(ADS1115_MUX_P3_NG);
bool jelStisnuto=(g0 or g1 or g2 or g3);
if(gumbStanje==0 and jelStisnuto)
{
  gumbStanje=1;
  if(g0)
  {
    stateID+=1;
    if(stateID>4) stateID=1;
    saveStanje=1;
  }
  else if(g1)
  {
    if (svjetlina+50>255) svjetlina=50; else svjetlina+=50;
    matrix.setBrightness(svjetlina);
    saveStanje=1;
  }
  else if(g2)
  {
    uint8_t r= boja1[0],g= boja1[1],b= boja1[2];
    boja1[0]=boja2[0];   boja1[1]=boja2[1];   boja1[2]=boja2[2];
    boja2[0]=boja3[0];   boja2[1]=boja3[1];   boja2[2]=boja3[2];
    boja3[0]=r;          boja3[1]=g;          boja3[2]=b;
    saveStanje=1;
  }
  else if(g3)
  {
    clearAllLayers();
```

```
    indexedLayerZ1.drawString(0,0,1,"Wi-Fi detalji");
    if(WiFi.status() == WL_CONNECTED)
                                        typeText(0,7,(String)("SSID:\n"+(String)myWIFI.getNameSSID()+"\nIP-
WiFi:\n"+myWIFI.getDevStatusIP().substring(9)),&boja2[0]);
    else
      typeText(0,7,(String)("SSID:\n"+APssid+"\nIP-HOTSPOT:\n192.168.4.1"),&boja2[0]);
    swapAllLayerBuffers();
    delay(5000);
   }
   zadnje_stisnuto=0;
 }
 if(saveStanje==1)
   if(zadnje_stisnuto>50 ) //nakon cca 10s
   {
    saveStanje=0;

    saveConfig();
   }
}

void displayDraw()
{
 matrix.setBrightness(svjetlina);
 clearAllLayerBuffers();
 if(stateID==1)
 {
       statusTrenutno= (String)tempCurr + "*C-temp. "+ (String)humCurr+"%RH-vlaga " +
(String)windCurr+"m/s-vjetar "+ (String)pressCurr+"hPa-tlak " +(String) visibilityCurr+"m-vidljivost "+
(String)skyCurr + "%-oblaka ";
   drawClock(4,1,2, &boja1[0]);
   drawDaytimeLine(0,22,64,23,(sat+minuta/60.0),50);
   typeTextNoNl(--statusTrenutniPomak,25,statusTrenutno,&boja2[0]);
   // Serial.println(4*da.length());
   //Serial.println(xda>4*da.length());
```

```
  // Serial.println(xda);
  if(-(int)statusTrenutniPomak>(int)(4*statusTrenutno.length())) {statusTrenutniPomak=kMatrixWidth;}
    // Serial.println(xda);
 }
 else if(stateID==2)
 {
  drawClock(4,1,2, &boja1[0]);
  drawDaytimeLine(0,22,64,23,(sat+minuta/60.0),50);
  String datum=(String)rtc.now().day()+"."+(String)rtc.now().month()+"."+(String)rtc.now().year()+".";
  typeTextNoNl((16-datum.length())/2.0*4,25, datum,&boja2[0]);

 }
 else if(stateID==3)
 {
  drawClock(4,1,2, &boja1[0]);
  String datum=(String)rtc.now().day()+"."+(String)rtc.now().month()+"."+(String)rtc.now().year()+".";
  typeTextNoNl((16-datum.length())/2.0*4,25, datum,&boja2[0]);
 }
 else if(stateID==4)
 {
      statusTrenutno= (String)tempCurr + "*C-temp. "+ (String)humCurr+"%RH-vlaga " +
(String)windCurr+"m/s-vjetar "+ (String)pressCurr+"hPa-tlak " +(String) visibilityCurr+"m-vidljivost "+
(String)skyCurr + "%-oblaka ";
  drawClock(4,1,2, &boja1[0]);
  drawDaytimeLine(0,22,64,23,(sat+minuta/60.0),50);
  typeTextNoNl(--statusTrenutniPomak,25,statusTrenutno,&boja2[0]);
  // Serial.println(4*da.length());
  //Serial.println(xda>4*da.length());
  // Serial.println(xda);
  if(-(int)statusTrenutniPomak>(int)(4*statusTrenutno.length())) {statusTrenutniPomak=kMatrixWidth;}
    // Serial.println(xda);
  if(WiFi.status() == WL_CONNECTED)
    backgroundLayer.drawFastHLine(15,48,31,{0,64,128});
  else
```

```
    backgroundLayer.drawFastHLine(15,48,31,{128,64,0});
  }
  swapAllLayerB uffers();
}
```

HTML geolokacija

```
<!DOCTYPE html>
<html>
<style>
  html {
    background-color: #DBDBDB;
    color: #376DAB;
    font-family: monospace;
    font-weight: bolder;
  }

  table,
  th,
  td {
    border: 1px solid #C14953;
  }

  input,
  select {
    height: 1.5em;
    padding: auto;
    margin: 0.1em;
    color: #376DAB;
    font-family: monospace;
    background-color: lightblue;
    border: 2px transparent solid;
    border-radius: 0.2em;
    font-weight: bolder;
    position: relative;
```

```
  width: 100%;
  transition: 0.5s;
  padding: 0px;
  height: 1.6em;

}

input[type=color],
input[type=button],
select {
  width: 75%;
  height: 1.6em;
}

input:hover {
  transition: 0.5s;
  border: 2px brown solid;
}

input:focus {
  background-color: yellow;
}

label {
  height: 2em;

}

h3 {
  margin-top: 2px;
}

section {
  border-radius: 1em;
```

```
  border: 3px solid #C14953;
  background-color: #30292F;
  padding: 1em;
  margin: 2px;
 }

 a,
 button {
  color: #376DAB;
  border: solid 0.2em gray;
  padding: 0.5em;
  margin: 0.25em;
  border-radius: 0.5em;
  background-color: gainsboro;
  transition: 0.25s;
  -webkit-user-select: none;
  -ms-user-select: none;
  user-select: none;
  font-family: monospace;
  font-weight: bolder;
 }

 a:hover{
  color: #4b91e0;
  background-color: lightgray;
  transition: 0.25s;
 }

 a:active {
  background-color: yellow;
  transition: 0.25s;

 }
</style>
```

```
<head>
 <meta http-equiv="Content-Type" content="text/html;charset=UTF-8" />
</head>

<body>
 <div id="forma"  style="left: 25%; position: absolute; width: 50%;">
  <h1 style="margin-bottom: 0px;">
   Digitalni sat
  </h1>
  <h3 style="margin-top: 0px;">(Završni rad Niko Pešut)</h3>
  <section style="background-color:rgba(128, 128, 128, 0.2); border-color: gray;">
    <h3>Stisni da bi dobio/la lokaciju</h3>
   <a onclick="getLocation()">Stisni me</a>
   <div id="izlaz"></div>
  </section>
  </div>
 </div>

 <script>
  var x = document.getElementById("izlaz");

  function getLocation() {
   if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(showPosition);
   } else {
    x.innerHTML = "Geolokacija nije podržana u ovom pregledniku!";
   }
  }

  function showPosition(position) {
   x.innerHTML = "<h3><br> Geografska Širina: <input type=\"text\" value=\"" + position.coords.latitude +
    "\" id=\"sirina\"><br>Geografska Dužina: <input type=\"text\" value=\"" + position.coords.longitude + "\"
id=\"duzina\"><br><br>Sada te podatke prekopiraj u lokalni form/obrazac.</h3>";
```

```
            x.innerHTML += "Kopiraj klikom:<br><br><a onclick=\"gsir()\">Geografska Širina</a><a
onclick=\"gduz()\">Geografska Dužina</a>";
  }
  function gduz() {
   var copyText = document.getElementById("duzina");
   copyText.select();
   copyText.setSelectionRange(0, 99999);
   navigator.clipboard.writeText(copyText.value);
  }
  function gsir() {
   var copyText = document.getElementById("sirina");
   copyText.select();
   copyText.setSelectionRange(0, 99999);
   navigator.clipboard.writeText(copyText.value);
  }
 </script>

</body>

</html>
```

HTML konfiguracija

```
<!DOCTYPE html>
<html>
<style>
 html {
  background-color: #DBDBDB;
  color: #376DAB;
  font-family: monospace;
  font-weight: bolder;
 }

 table,
 th,
 td {
```

```
  border: 1px solid #C14953;
}

input,
select {
  height: 1.5em;
  padding: auto;
  margin: 0.1em;
  color: #376DAB;
  font-family: monospace;
  background-color: #EE635240;
  border: 2px transparent solid;
  border-radius: 0.2em;
  font-weight: bolder;
  position: relative;
  width: 100%;
  transition: 0.5s;
  padding: 0px;
  height: 1.6em;

}

input[type=color],
input[type=button],
select {
  width: 75%;
  height: 1.6em;
}

input:hover {
  transition: 0.5s;
  border: 2px brown solid;
}
```

```
 input:focus {
  background-color: yellow;
 }

 label {
  height: 2em;

 }

 h3 {
  margin-top: 2px;
 }

 section {
  border-radius: 1em;
  border: 3px solid #C14953;
  background-color: #30292F;
  padding: 1em;
  margin: 2px;
 }

 a {
  color: #376DAB;
 }
</style>

<head>
 <meta http-equiv="Content-Type" content="text/html;charset=UTF-8" />
</head>

<body>

 <form id="forma" action="/get" method="get" style="left: 25%; position: absolute; width: 50%;">
```

```
  <div style="display: flex; flex-direction: column;">


   <h1 style="margin-bottom: 0px;">
    Digitalni sat
   </h1>
   <h3 style="margin-top: 0px;">(Završni rad Niko Pešut)</h3>
   <section style="background-color:rgba(128, 128, 128, 0.2); border-color: gray;">
    <h3 style="margin: 0px;">Stavke koje ne mijenjate se neće postaviti</h3>
   </section>
   <section>
    <h2 style="margin-bottom: 2px; ">Lokacija</h2>
    <h5 style="margin-top: 2px;">(od lokacije se automatski uzima vremenska zona, datum, vrijeme...)</h1>
            <a href="https://geoloc.pesut.win" target="_blank"><input type="button" value="dobij lokaciju"></a><br>
     <label for="latitude">Geografska Širina</label>
     <input type="text" id="latitude" name="latitude"><br>
     <label for="longitude">Geografska Dužina</label>
     <input type="text" id="longitude" name="longitude"><br>


     <input type="hidden" id="longitudechange" name="longitudechange" value="0">
     <input type="hidden" id="latitudechange" name="latitudechange" value="0">
   </section>
   <section>
    <h2>Boje i stil</h2>
    <label for="boja1">1. boja</label><br>
    <input type="color" value="#ffffff" name="boja1" id="boja1"><br>
    <label for="boja2">2. boja</label><br>
    <input type="color" value="#ffffff" name="boja2" id="boja2"><br>
    <label for="boja3">3. boja</label><br>
    <input type="color" value="#ffffff" name="boja3" id="boja3"><br>
    <label for="mod">Modovi</label><br>
    <select name="mod" id="mod">
     <option value="0" disabled selected>-- Odaberi mod --</option>
     <option value="4">Sat, crta dana, trenutačno vrijeme i wifi status</option>
```

```
      <option value="1">Sat, crta dana i trenutačno vrijeme</option>
      <option value="2">Sat, crta dana i datum</option>
      <option value="3">Sat i datum</option>
    </select>

    <input type="hidden" id="boja1change" name="boja1change" value="0">
    <input type="hidden" id="boja2change" name="boja2change" value="0">
    <input type="hidden" id="boja3change" name="boja3change" value="0">
    <input type="hidden" id="modchange" name="modchange" value="0">
  </section>
  <section id="mreza">
    <h2>Mreža</h2>
    <label for="SSID">WiFi SSID</label>
    <input id="SSID" name="SSID" type="text" placeholder="Ime WIFI-a"><br>
    <label for="PASSWORD">WiFi Šifra</label>
    <input id="PASSWORD" name="PASSWORD" type="text" placeholder="Šifra WIFI-a"><br>
    <label for="APSSID">HotSpot SSID</label>
    <input id="APSSID" name="APSSID" type="text" placeholder="Ime accespoint-a"><br>
    <label for="APPASSWORD">HotSpot Šifra</label>
    <input id="APPASSWORD" name="APPASSWORD" type="text" placeholder="Šifra accespoint-a"><br>

    <input type="hidden" id="SSIDchange" name="SSIDchange" value="0">
    <input type="hidden" id="PASSWORDchange" name="PASSWORDchange" value="0">
    <input type="hidden" id="APSSIDchange" name="APSSIDchange" value="0">
    <input type="hidden" id="APPASSWORDchange" name="APPASSWORDchange" value="0">
  </section>
  <section>
    <h2>API-s</h2>
    <label for="timezonedb"><a href="https://www.timezonedb.com">Timezonedb</a> api ključ</label>
    <input id="timezonedb" name="timezonedb" type="text" placeholder="Api Key"><br>
    <label for="openweathermap"><a href="https://www.openweathermap.org">Openweathermap</a> api
ključ</label>
    <input id="openweathermap" name="openweathermap" type="text" placeholder="Api Key"><br>
```

```
      <input type="hidden" id="timezonedbchange" name="timezonedbchange" value="0">
      <input type="hidden" id="openweathermapchange" name="openweathermapchange" value="0">
    </section>
     <input type="submit" id="posaljitelj" type="button" value="Pošalji" style="height: 3em; font-size: 2em;
width: 99%;">
   </div>
 </form>
</body>
<script>
 document.getElementById("boja1change").value = "0";
 document.getElementById("boja2change").value = "0";
 document.getElementById("boja3change").value = "0";
 document.getElementById("latitudechange").value = "0";
 document.getElementById("longitudechange").value = "0";
 document.getElementById("SSIDchange").value = "0";
 document.getElementById("PASSWORDchange").value = "0";
 document.getElementById("APSSIDchange").value = "0";
 document.getElementById("APPASSWORDchange").value = "0";
 document.getElementById("timezonedbchange").value = "0";
 document.getElementById("openweathermapchange").value = "0";
 document.getElementById("modchange").value = "0";
          document.getElementById("boja1").addEventListener("change",          (event)          =>          {
document.getElementById("boja1change").value = "1"; });
          document.getElementById("boja2").addEventListener("change",          (event)          =>          {
document.getElementById("boja2change").value = "1"; });
          document.getElementById("boja3").addEventListener("change",          (event)          =>          {
document.getElementById("boja3change").value = "1"; });
         document.getElementById("latitude").addEventListener("change",          (event)          =>          {
document.getElementById("latitudechange").value = "1"; });
        document.getElementById("longitude").addEventListener("change",          (event)          =>          {
document.getElementById("longitudechange").value = "1"; });
          document.getElementById("SSID").addEventListener("change",          (event)          =>          {
document.getElementById("SSIDchange").value = "1"; });
        document.getElementById("PASSWORD").addEventListener("change",          (event)          =>          {
document.getElementById("PASSWORDchange").value = "1"; });
```

```
        document.getElementById("APSSID").addEventListener("change", (event) => {
document.getElementById("APSSIDchange").value = "1"; });
      document.getElementById("APPASSWORD").addEventListener("change", (event) => {
document.getElementById("APPASSWORDchange").value = "1"; });
       document.getElementById("timezonedb").addEventListener("change", (event) => {
document.getElementById("timezonedbchange").value = "1"; });
     document.getElementById("openweathermap").addEventListener("change", (event) => {
document.getElementById("openweathermapchange").value = "1"; });
        document.getElementById("mod").addEventListener("change", (event) => {
document.getElementById("modchange").value = "1"; });


 let form = document.getElementById("forma");
 let button = document.getElementById("get-location");
 let latText = document.getElementById("latitude");
 let longText = document.getElementById("longitude");
 document.getElementById("get-location").addEventListener("click", (event) => {
  navigator.geolocation.getCurrentPosition((position) => {
   let lat = position.coords.latitude;
   let long = position.coords.longitude;

   latText.value = lat.toFixed(2);
   longText.value = long.toFixed(2);
  }
  );
 }
 );

 document.getElementById("posaljitelj").addEventListener("click", (event) => {
  form.submit();
 }
 );

</script>
</html>
```