Strukovna škola Đurđevac

WEATHER STATION

UČENICI:

Noah Fel & Antonio Kolar

MENTORI:

Željko Brček,

Ivanka Brček

Đurđevac, siječanj 2023.

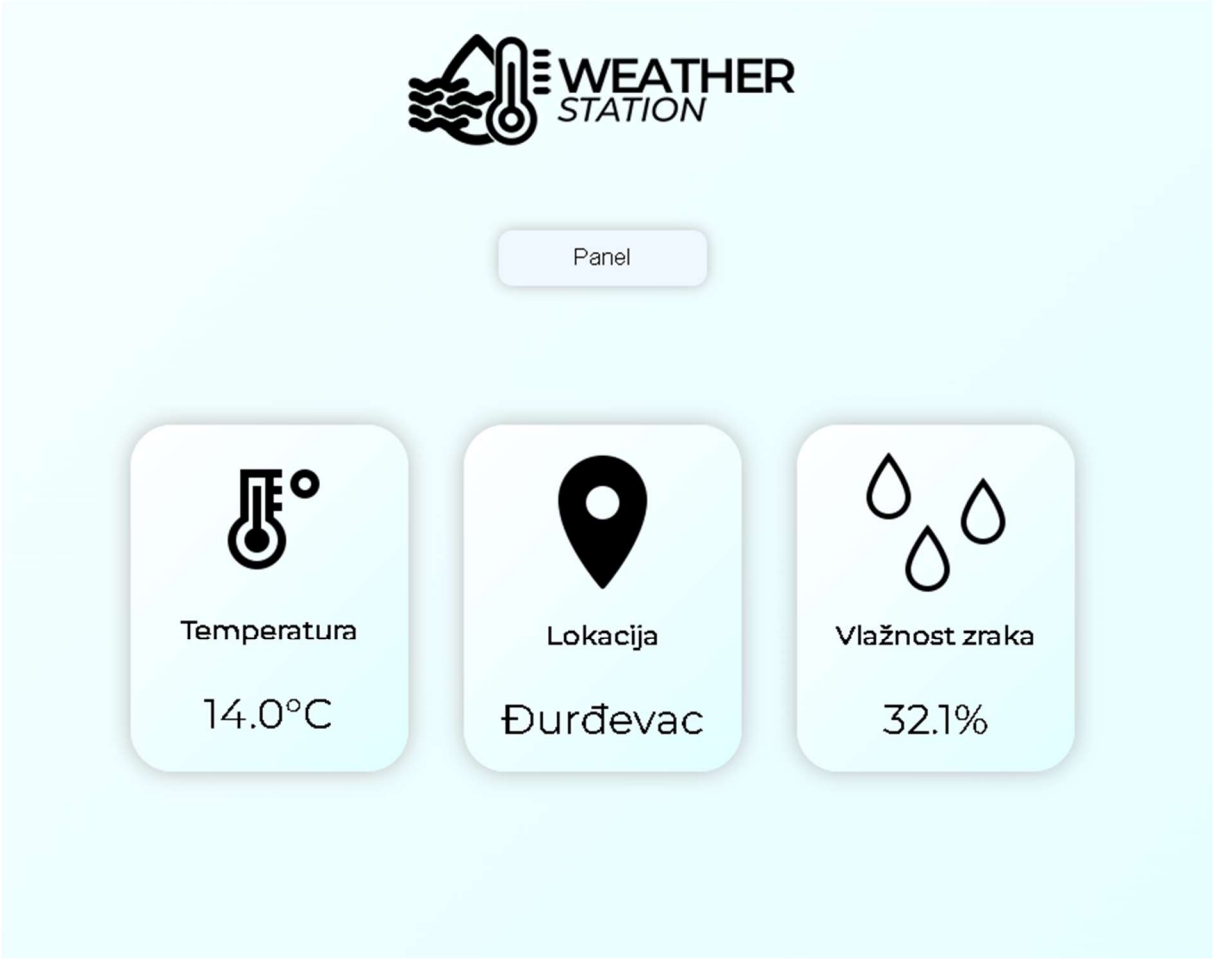
Sadržaj:

1. Uvod

Weather Station namijenjen je za uvid temperature i vlažnosti zraka u zatvorenom prostoru.

Podaci se ažuriraju svaku minutu i prikazuju se na web stranici.

Web stranici se pristupa preko domene, može joj se pristupiti preko bilo kojeg uređaja, jer je stranica prilagođena svim korisnicima.



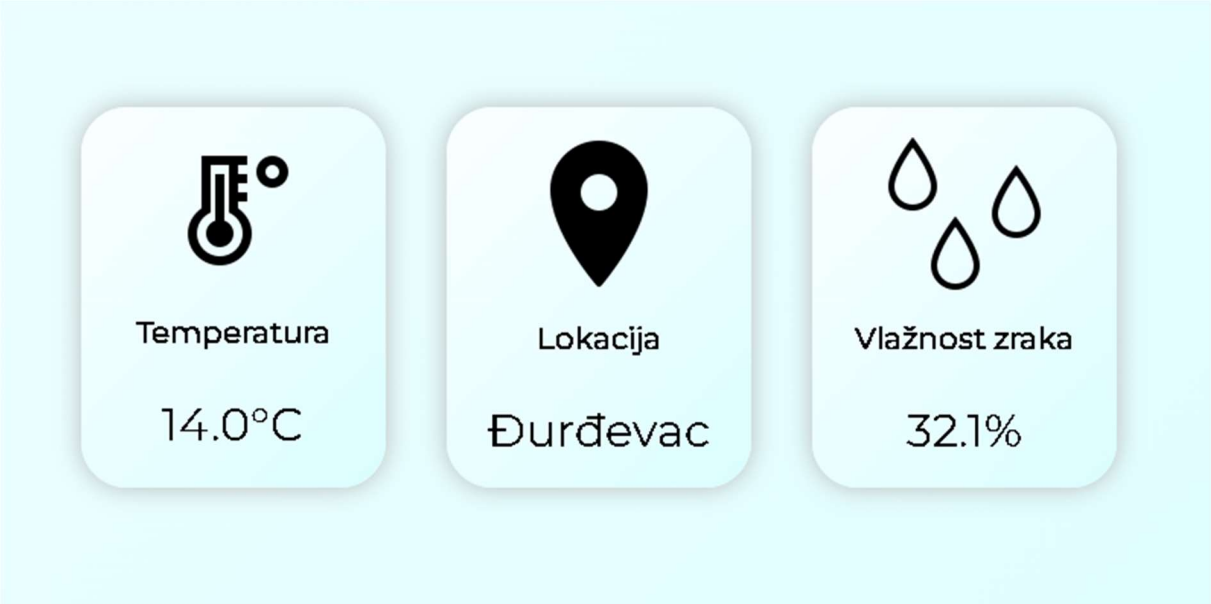
Weather Station - Početna stranica na web-u (**index.html**)

2. Teoretski opis rada

Web stranica prikazuje temperaturu i vlažnost zraka iz baze podatak

Praćenje vremenskih promjena i pregled statistike može se pratiti sa web stranice koja je dostupna preko lokalne IP adrese.

Espressif ESP-32 mikrokontroler šalje podatke sa svojih senzora na HiveMQ MQTT cluster. Sa druge strane pokreće se Django server koji upravlja samom web stranicom, čita podatke koje MQTT cluster šalje, pohranjuje podatke u bazu podataka i te iste podatke prikazuje na web stranici.



Weather Station - Početna stranica na web-u (**index.html**)

2.1. Espressif ESP32 WROWER-E

ESP32 je popularan mikrokontroler otvorenog koda koji je razvio kineski proizvođač čipova Espressif Systems. Evo nekih osnovnih informacija o ESP32:

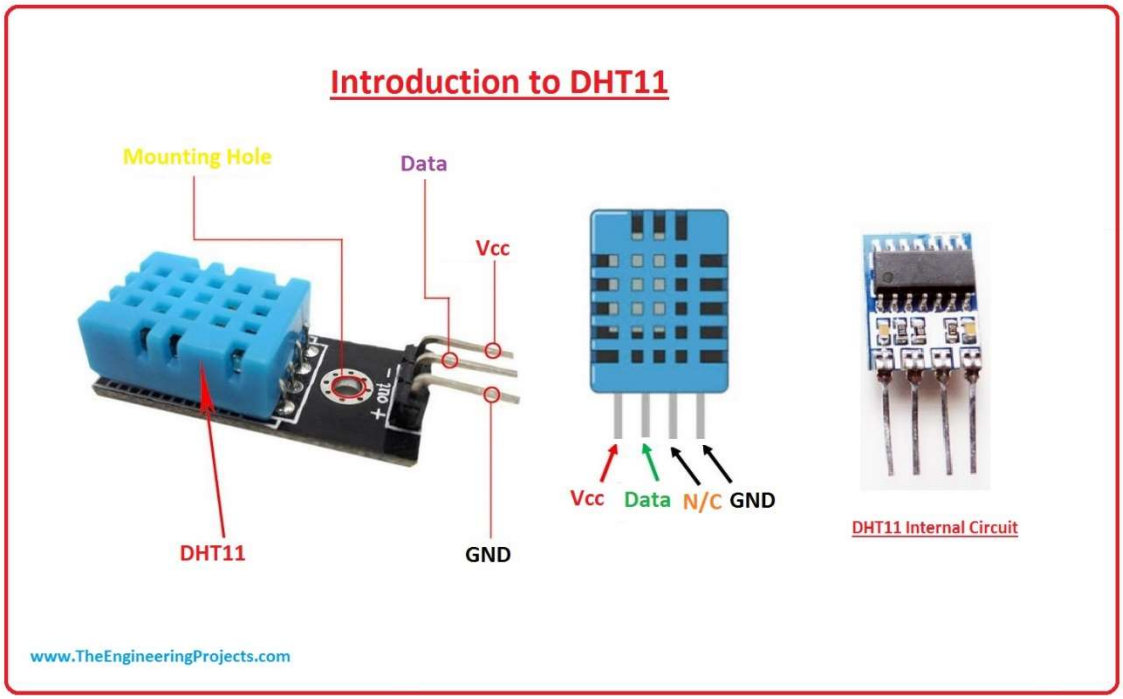
- Arhitektura: ESP32 koristi Xtensa® LX6 32-bitnu dual-core procesorsku jedinicu. Svaki jezgro radi na frekvenciji do 240 MHz.
- Bežične mogućnosti: ESP32 ima ugrađene mogućnosti bežične komunikacije. Podržava Wi-Fi 802.11 b/g/n standard i Bluetooth v4.2 BR/EDR i BLE (Bluetooth Low Energy) protokole
- Radna memorija (RAM): ESP32 dolazi s različitim konfiguracijama RAM-a, uključujući 520 KB SRAM-a i 8 KB RTC RAM-a.
- Ugrađena memorija (Flash): ESP32 može imati različite veličine ugrađene fleš memorije, u rasponu od 4 MB do 16 MB.
- Periferni ulazi/izlazi: ESP32 ima bogat skup perifernih ulaza/izlaza, uključujući digitalne ulaze/izlaze, ADC (analogno-digitalni konvertor), DAC (digitalno-analogni konvertor), PWM (pulsno-širinska modulacija) izlaze, I2C, SPI, UART i druge komunikacijske sučelja.
- Napajanje: ESP32 radi na naponu od 3.3V i podržava različite načine napajanja, uključujući baterijsko napajanje.
- Programiranje: ESP32 se može programirati korištenjem Arduino IDE, MicroPython, Lua ili drugih jezičnih okvira.

Espressif ESP32 WROWER-Ei
(https://www.espressif.com/sites/default/files/documentation/esp32-wrover-e_esp32-wrover-ie_datasheet_en.pdf)

2.2. DHT11 Senzor

DHT11 je precizan senzor temperature (-40°C do 80°C) i vlažanosti zraka. Odstupanje senzora je ±0.5°C i ±1%.

Sa ovim senzorom smo prikupljali podatke o temperaturi i vlažnosti u zatvorenom prostoru.



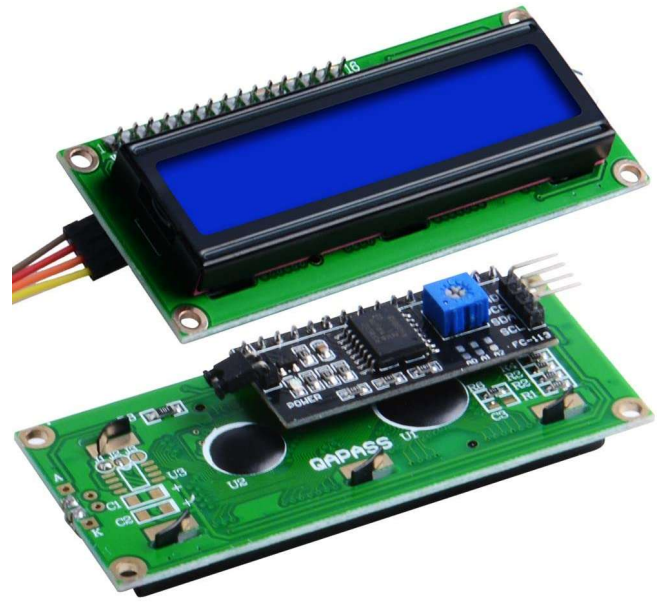
DHT11 Senzor

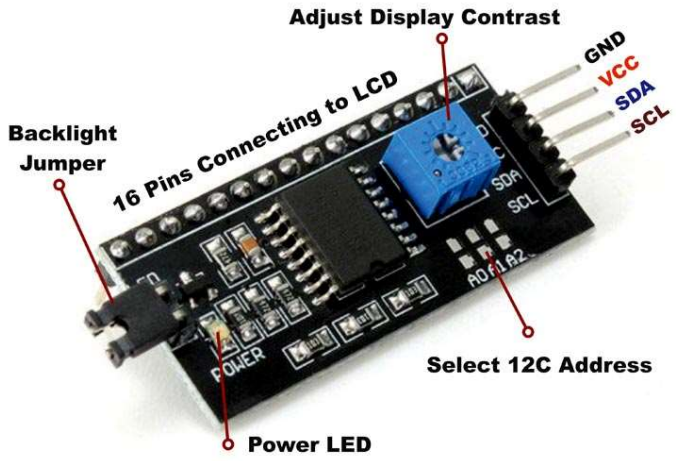
2.3. LCD ekran

Značajke:

- 16bit ekran
- Hitachi HD44780 driver
- 8 data pinova
- R/W pin
- Enable pin

Biblioteka LiquidCrystal nam omogućuje korištenje LCD ekrana u našu svrhu. Pomoću te komponente, dobiti ćemo prikaz trenutne temperature i vlažnosti zraka prikazan na 16 bitnom ekranu.



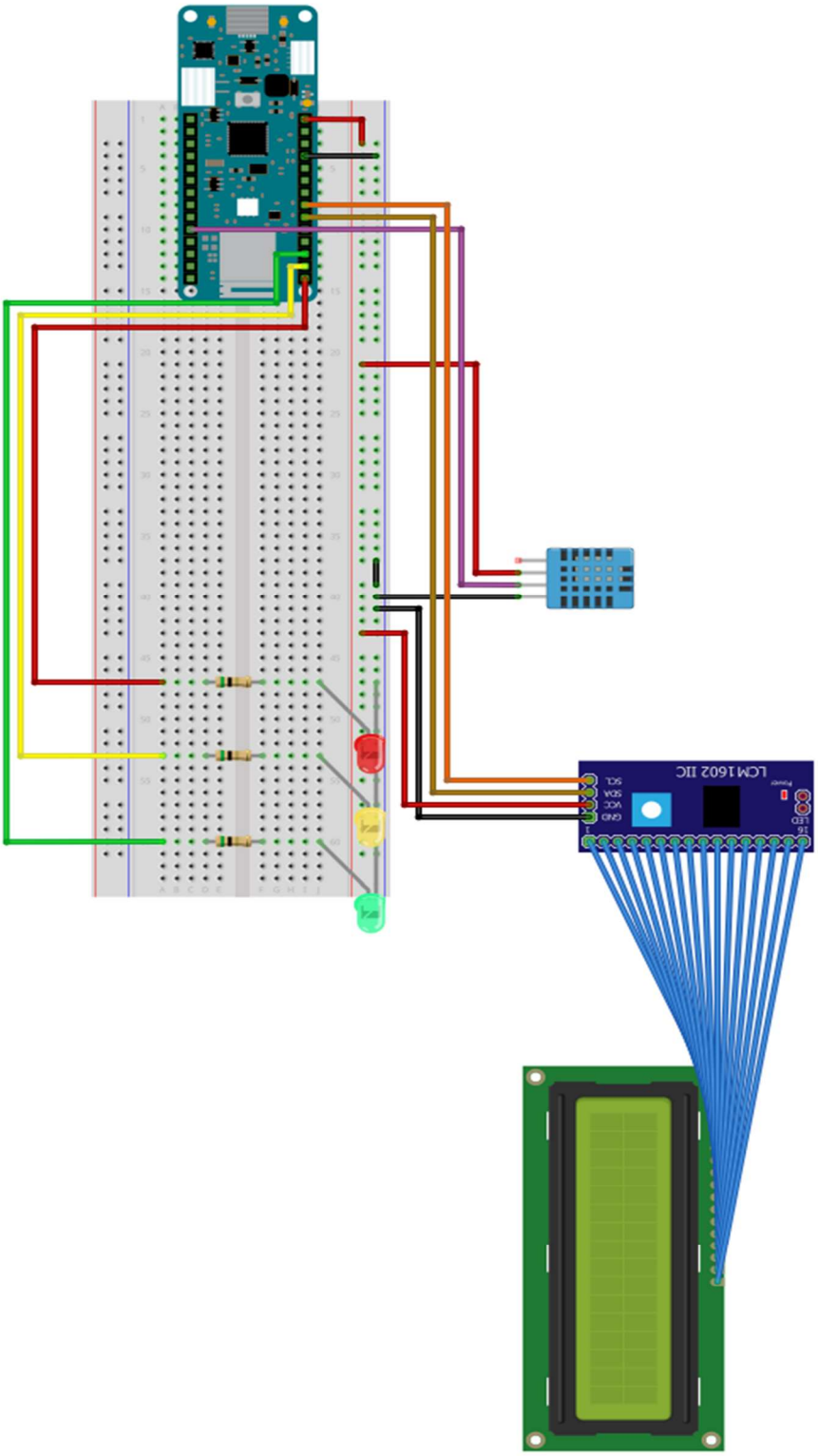


3. Izvedba rada

Potrebni materijali:

- Espressif ESP-32 WROWER-E
- DHT22 senzor
- Žice
- Baterija 12V
- Internet mreža
- LED Dioda (x2)
- LCD ekran

3.1. Shema



3.2. Arduino kod

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <WiFiClientSecure.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>

#include <LiquidCrystal_I2C.h>

#define DHTPIN 23
#define DHTTYPE DHT11

DHT_Unified dht(DHTPIN, DHTTYPE);
LiquidCrystal_I2C lcd(0x27, 16, 2);

//---- WiFi settings
const char *ssid = "PODRUM";
const char *password = "25052005";
//---- MQTT Broker settings
const char *mqtt_server = "39d4d5ac4ee740b1a9deab7d5e3aeea1.s2.eu.hivemq.cloud";
const int mqtt_port = 8883;
const char *mqtt_username = "Rentex";
const char *mqtt_password = "malikostanj";

WiFiClientSecure espClient;
PubSubClient client(espClient);
unsigned long lastMsg = 0;

#define MSG_BUFFER_SIZE (50)
char msg[MSG_BUFFER_SIZE];

// cluster topici
const char *sensor1_topic = "encyclopedia/temperature";
const char *sensor2_topic = "encyclopedia/vlaznost";

static const char *root_ca PROGMEM = R"EOF(
-----BEGIN CERTIFICATE-----
MIIFazCCA1OgAwIBAgIRAIIQz7DSQONZRGPgu2OCiwAwDQYJKoZIhvcNAQELBQAw
```

```
-----BEGIN CERTIFICATE-----
MIIFazCCA1OgAwIBAgIRAIIQz7DSQONZRGPgu2OCiwAwDQYJKoZIhvcNAQELBQAw
TzELMAkGA1UEBhMCVVMxKTAnBgNVBAoTIEludGVybmV0IFNlY3VyaXR5IFJlc2Vh
cmNoIEdyb3VwMRUwEwYDVQQDEwxJU1JHIFJvb3QgWDEwHhcNMTUwNjA0MTEwNDM4
WhcNMzUwNjA0MTEwNDM4WjBPMQswCQYDVQQGEwJVUzEpMCcGA1UEChMgSW50ZXJu
ZXQgU2VjdXJpdHkgUmVzZWFyY2ggR3JvdXAxFTATBgNVBAMTDElTUkcgUm9vdCBY
MTCCAiIwDQYJKoZIhvcNAQEBBQADggIPADCCAgoCggIBAK3oJHP0FDfzm54rVygc
h77ct984kIxuPOZXoHj3dcKi/vVqbvYATyjb3miGbESTtrFj/RQSa78f0uoxmyF+
0TM8ukj13Xnfs7j/EvEhmkvBioZxaUpmZmyPfjxwv60pIgbz5MDmgK7iS4+3mX6U
A5/TR5d8mUgjU+g4rk8Kb4Mu0UlXjIB0ttov0DiNewNwIRt18jA8+o+u3dpjq+sW
T8KOEUt+zwvo/7V3LvSye0rgTBIlDHCNAymg4VMk7BPZ7hm/ELNKjD+Jo2FR3qyH
B5T0Y3HsLuJvW5iB4YlcNHlsdu87kGJ55tukmi8mxdAQ4Q7e2RCOFvu396j3x+UC
B5iPNgiV5+I3lg02dZ77DnKxHZu8A/lJBdiB3QW0KtZB6awBdpUKD9jf1b0SHzUv
KBds0pjBqAlkd25HN7rOrFleaJ1/ctaJxQZBKT5ZPt0m9STJEadao0xAH0ahmbWn
OlFuhjuefXKnEgV4We0+UXgVCwOPjdAvBbI+e0ocS3MFEvzG6uBQE3xDk3SzynTn
jh8BCNAw1FtxNrQHusEwMFxIt4I7mKZ9YIqioymCzLq9gwQbooMDQaHWBfEbwrbw
qHyGO0aoSCqI3Haadr8faqU9GY/rOPNk3sgrDQoo//fb4hVC1CLQJ13hef4Y53CI
rU7m2Ys6xt0nUW7/vGT1M0NPAgMBAAGjQjBAMA4GA1UdDwEB/wQEAwIBBjAPBgNV
HRMBAf8EBTADAQH/MB0GA1UdDgQWBBR5tFnme7bl5AFzgAiIyBpY9umbbjANBgkq
hkiG9w0BAQsFAAOCAgEAVR9YqbyyqFDQDLHYGmkgJykIrGF1XIpu+ILlaS/V9lZL
ubhzEFnTIZd+50xx+7LSYK05qAvqFyFWhfFQDlnrzuBZ6brJFe+GnY+EgPbk6ZGQ
3BebYhtF8GaV0nxvwuo77x/Py9auJ/GpsMiu/X1+mvoiBOv/2X/qkSsisRcOj/KK
NFtY2PwByVS5uCbMiogziUwthDyC3+6WVwW6LLv3xLfHTjuCvjHIInNzktHCgKQ5
ORAzI4JMPJ+GslWYHb4phowim57iaztXOoJwTdwJx4nLCgdNbOhdjsnvzqvHu7Ur
TkXWStAmzOVyyghqpZXjFaH3pO3JLF+l+/+sKAIuvtd7u+Nxe5AW0wdeRlN8NwdC
jNPElpzVmbUq4JUagEiuTDkHzsxHpFKVK7q4+63SM1N95R1NbdWhscdCb+ZAJzVc
oyi3B43njTOQ5yOf+1CceWxG1bQVs5ZufpsMljq4Ui0/1lvh+wjChP4kqKOJ2qxq
4RgqsahDYVvTH9w7jXbyLeiNdd8XM2w9U/t7y0Ff/9yi0GE44Za4rF2LN9d11TPA
mRGunUHBcnWEvgJBQl9nJEiU0Zsnvgc/ubhPgXRR4Xq37Z0j4r7g1SgEEzwxA57d
emyPxgcYxn/eR44/KJ4EBs+lVDR3veyJm+kXQ99b21/+jh5Xos1AnX5iItreGCc=
-----END CERTIFICATE-----
)EOF";

uint32_t delayMS;

void setup()
{


  Serial.begin(9600);
  Serial.print("\nConnecting to ");
  Serial.println(ssid);

  // Pokretanje LCD ekrana
  lcd.init();
  lcd.backlight();


  dht.begin();
  sensor_t sensor;
  dht.temperature().getSensor(&sensor);
  dht.humidity().getSensor(&sensor);

  delayMS = sensor.min_delay / 1000;

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
```

```
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  randomSeed(micros());
  Serial.println("\nPovezan na mrežu\nIP adresa: ");
  Serial.println(WiFi.localIP());

  while (!Serial)
    delay(1);

  espClient.setCACert(root_ca);
  client.setServer(mqtt_server, mqtt_port);
  client.setCallback(callback);


}

void loop(){

  // Dohvaćanje vrijednosti temperature zraka
  sensors_event_t event;
  dht.temperature().getEvent(&event);
  if (isnan(event.temperature)) {
    Serial.println(F("Error reading temperature!"));
  }
  else {
    Serial.print(F("Temperature: "));
    Serial.print(event.temperature);
    Serial.println(F("°C"));


    publishMessage(sensor1_topic, String(event.temperature), true);
  }
  // Dohvaćanje vrijednosti vlažnosti zraka
  dht.humidity().getEvent(&event);
  if (isnan(event.relative_humidity)) {
    Serial.println(F("Error reading humidity!"));
  }
  else {
    Serial.print(F("Humidity: "));
    Serial.print(event.relative_humidity);
    Serial.println(F("%"));
```

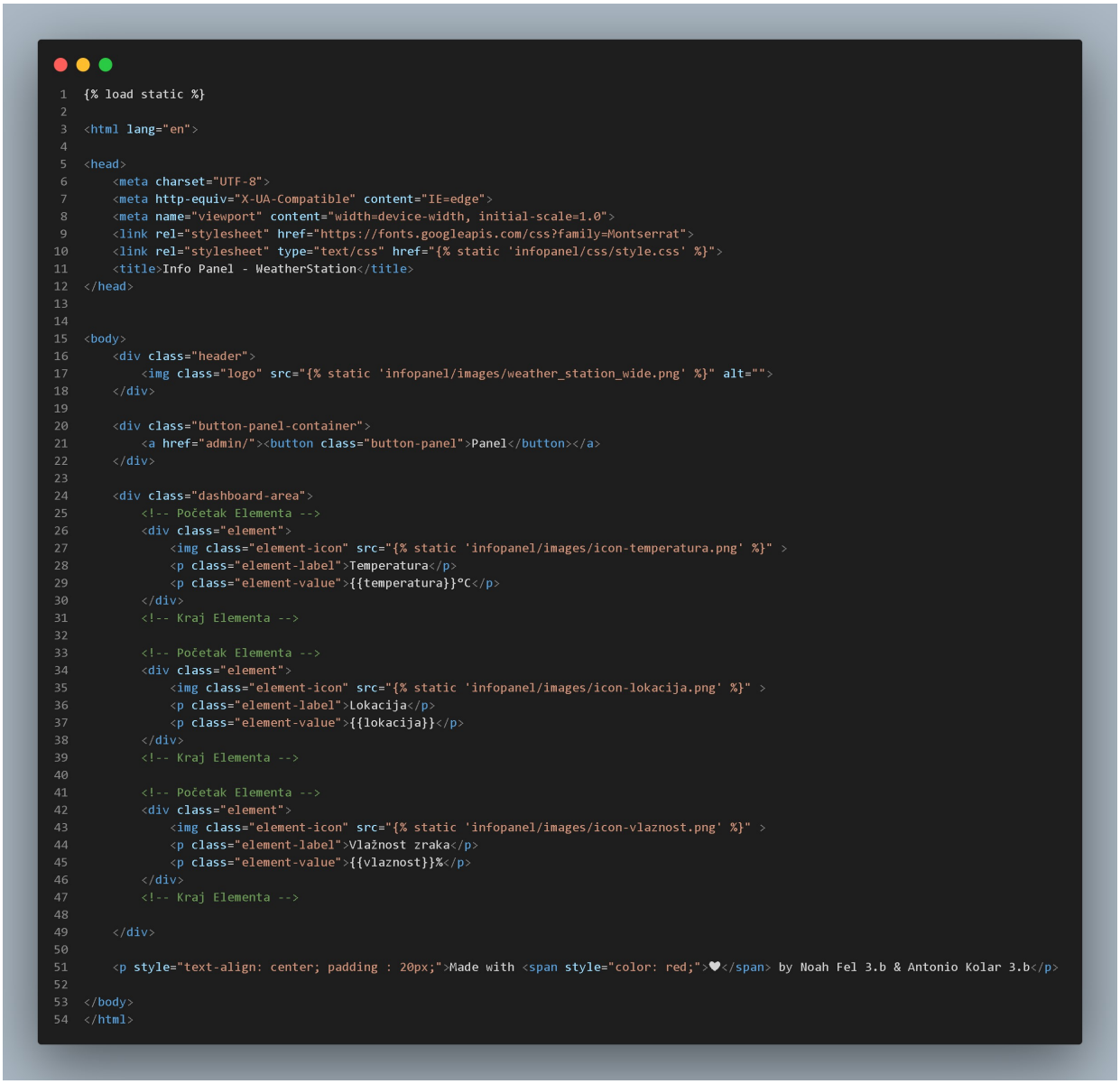
```
137       Serial.print(event.relative_humidity);
138       Serial.println(F("%"));
139       publishMessage(sensor2_topic, String(event.relative_humidity), true);
140     }
141
142
143
144     if (!client.connected())
145       reconnect();
146     client.loop();
147
148     LcdRefresh();
149
150
151   }
152
153   // Funkcija za refresh ekrana, svakih 2 sekunde
154   void LcdRefresh(){
155     delay(2000);
156     sensors_event_t event;
157     dht.temperature().getEvent(&event);
158     lcd.clear();
159     lcd.println("Temp: ");
160     lcd.print(event.temperature);
161     lcd.print(char(223));
162     lcd.print("C");
163
164     lcd.setCursor(0, 1);
165     lcd.println("Vlaga: ");
166     lcd.print(event.relative_humidity);
167     lcd.print("%");
168
169
170   }
171
172   //=========================================================================F
173
174   void reconnect()
175   {
176     while (!client.connected())
177     {
178       Serial.print("Attempting MQTT connection…");
179       String clientId = "ESP32";
180       clientId += String(random(0xffff), HEX);
181       if (client.connect(clientId.c_str(), mqtt_username, mqtt_password))
182       {
183         Serial.println("connected");
```

```
174 void reconnect()
175 {
176   while (!client.connected())
177   {
178     Serial.print("Attempting MQTT connection…");
179     String clientId = "ESP32";
180     clientId += String(random(0xffff), HEX);
181     if (client.connect(clientId.c_str(), mqtt_username, mqtt_password))
182     {
183       Serial.println("connected");
184
185
186     }
187     else
188     {
189       Serial.print("failed, rc=");
190       Serial.print(client.state());
191       Serial.println(" try again in 5 seconds");
192       delay(5000);
193     }
194   }
195 }
196
197
198 void callback(char *topic, byte *payload, unsigned int length)
199 {
200   String incommingMessage = "";
201   for (int i = 0; i < length; i++)
202     incommingMessage += (char)payload[i];
203   Serial.println("Message arrived [" + String(topic) + "]" + incommingMessage);
204 }
205
206 void publishMessage(const char *topic, String payload, boolean retained)
207 {
208   if (client.publish(topic, payload.c_str(), true))
209     Serial.println("Message publised [" + String(topic) + "]: " + payload);
210 }
```

3.3. Python kod (views.py)

```python
from django.shortcuts import render
from .models import Stats
from . import mqtt
from django.db.models.signals import post_save
from django.dispatch import receiver


def display_info(request):

    temperatura = Stats.objects.get(naziv='temperatura').vrijednost
    vlaznost = Stats.objects.get(naziv='vlaznost').vrijednost
    lokacija = Stats.objects.get(naziv="geolokacija").naziv

    print(f"TEMPERATURA: {temperatura}")
    print(f"VLAŽNOST ZRAKA: {vlaznost}")

    context = {'temperatura': temperatura, 'vlaznost': vlaznost, 'lokacija': lokacija}


    return render(request, "infopanel/index.html", context)

@receiver(post_save, sender=Stats)
def refresh_data(sender, instance, **kwargs):
    vlaznost = Stats.objects.get(naziv='vlaznost').vrijednost
    temperatura = Stats.objects.get(naziv='temperatura').vrijednost
    mqtt.client.publish("encyclopedia/temperature", payload=float(temperatura), qos=2)
    mqtt.client.publish("encyclopedia/vlaznost", payload=float(vlaznost), qos=2)


```

3.4. Django HTML kod (templates/index.html)



```
{% load static %}

<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Montserrat">
    <link rel="stylesheet" type="text/css" href="{% static 'infopanel/css/style.css' %}">
    <title>Info Panel - WeatherStation</title>
</head>


<body>
    <div class="header">
        <img class="logo" src="{% static 'infopanel/images/weather_station_wide.png' %}" alt="">
    </div>

    <div class="button-panel-container">
        <a href="admin/"><button class="button-panel">Panel</button></a>
    </div>

    <div class="dashboard-area">
        <!-- Početak Elementa -->
        <div class="element">
            <img class="element-icon" src="{% static 'infopanel/images/icon-temperatura.png' %}" >
            <p class="element-label">Temperatura</p>
            <p class="element-value">{{temperatura}}°C</p>
        </div>
        <!-- Kraj Elementa -->

        <!-- Početak Elementa -->
        <div class="element">
            <img class="element-icon" src="{% static 'infopanel/images/icon-lokacija.png' %}" >
            <p class="element-label">Lokacija</p>
            <p class="element-value">{{lokacija}}</p>
        </div>
        <!-- Kraj Elementa -->

        <!-- Početak Elementa -->
        <div class="element">
            <img class="element-icon" src="{% static 'infopanel/images/icon-vlaznost.png' %}" >
            <p class="element-label">Vlažnost zraka</p>
            <p class="element-value">{{vlaznost}}%</p>
        </div>
        <!-- Kraj Elementa -->

    </div>

    <p style="text-align: center; padding : 20px;">Made with <span style="color: red;">♥</span> by Noah Fel 3.b & Antonio Kolar 3.b</p>

</body>
</html>
```

3.5. Baza podataka (models.py)

```python
from django.db import models

class Stats(models.Model):
    naziv = models.CharField(max_length=16)
    vrijednost = models.FloatField()

    def __str__(self):
        return self.naziv
```

3.6. Python Paho MQTT

```python
import paho.mqtt.client as paho
from paho import mqtt


def on_connect(client, userdata, flags, rc, properties=None):
    print("CONNACK received with code %s." % rc)

def on_publish(client, userdata, mid, properties=None):
    print("mid: " + str(mid))

def on_subscribe(client, userdata, mid, granted_qos, properties=None):
    print("Subscribed: " + str(mid) + " " + str(granted_qos))

def on_message(client, userdata, msg):

    query = msg.topic, str(msg.payload)
    print(query)
    global temperatura
    global vlaznost
    if query[0] == "encyclopedia/vlaznost":
        vlaznost = str(msg.payload).split("'")[1]
        print("Vlaznost zraka je", vlaznost)
    else:
        temperatura = str(msg.payload).split("'")[1]
        print("Temperatura zraka je", temperatura)



client = paho.Client(client_id="", userdata=None, protocol=paho.MQTTv5)
client.on_connect = on_connect

client.tls_set(tls_version=mqtt.client.ssl.PROTOCOL_TLS)
client.username_pw_set("Rentex", "malikostanj")
client.connect("39d4d5ac4ee740b1a9deab7d5e3aeea1.s2.eu.hivemq.cloud", 8883)

client.on_subscribe = on_subscribe
client.on_message = on_message
client.on_publish = on_publish

# subscribe to all topics of encyclopedia by using the wildcard "#"
client.subscribe("encyclopedia/temperature", qos=2)
client.subscribe("encyclopedia/vlaznost", qos=2)
```

4.0. Literatura

- Django framework: https://www.djangoproject.com/
- Paho MQTT: https://pypi.org/project/paho-mqtt/
- HiveMQ cloud: https://www.hivemq.com/
- Espressif ESP32: https://www.espressif.com/sites/default/files/documentation/esp32-wrover-e_esp32-wrover-ie_datasheet_en.pdf
- DHT11: https://learn.adafruit.com/dht/overview
- LCD I2C: https://lastminuteengineers.com/i2c-lcd-arduino-tutorial/