MJERENJE TEMPERATURE I VLAŽNOSTI I NJIHOVO PRIKAZIVANJE NA WEB STRANICI

Kušević, Kristijan

VELEUČILIŠTE U KARLOVCU
STROJARSKI ODJEL
PRIJEDIPLOMSKI STRUČNI STUDIJ MEHATRONIKE

KRISTIJAN KUŠEVIĆ

MJERENJE TEMPERATURE I VLAŽNOSTI I NJIHOVO PRIKAZIVANJE NA WEB STRANICI

ZAVRŠNI RAD

KARLOVAC, 2024

VELEUČILIŠTE U KARLOVCU
STROJARSKI ODJEL
PRIJEDIPLOMSKI STRUČNI STUDIJ MEHATRONIKE

KRISTIJAN KUŠEVIĆ

MJERENJE TEMPERATURE I VLAŽNOSTI I NJIHOVO PRIKAZIVANJE NA WEB STRANICI

ZAVRŠNI RAD

mr. sc. Vedran Vyroubal, v. pred.

KARLOVAC, 2024

IZJAVA

Izjavljujem da sam samostalno radio ovaj završni rad. Ovaj rad sam napravio sa stečenim znanjima tijekom obrazovanja i korištenjem strane i domaće literature.

Ovom prilikom bih se htio zahvaliti svim svojim članovima obitelji, prijateljima i kolegama koji su mi davali podršku kroz cijeli studij.

Također se zahvaljujem mentoru Vedranu Vyroubalu na stečenome znanju i mentorstvu ovoga završnoga rada.

Karlovac, 2024. godina

Kristijan Kušević

_____

SAŽETAK

Glavna tema ovoga završnoga rada je pokazati kako preko senzora za mjerenje temperature i vlažnosti, mikrokontrolera i web servera se može postići mjerenje temperature i vlage i prikazivanje mjerenja preko grafova prikazani na web stranici. Ovaj rad je savršen primjer kako se može nadzirati temperatura i vlaga željenog prostora bilo gdje na svijetu koristeći web stranicu na kojoj će se vrijednosti prikazivati u stvarnome vremenu.

KLJUČNE RIJEČI

Temperatura, vlaga, ESP32, Web-server, DHT11

SUMMARY

The main topic of this undergraduate thesis is to show how temperature and humidity measurement can be achieved through sensors for measuring temperature and humidity, microcontrollers and web servers and displaying the measurements via the graphs displayed on the website. This work is a perfect example of how one can monitor the temperature and humidity of a desired space anywhere in the world using a website where the values will be displayed in real time.

KEYWORDS

Temperature, humidity, ESP32, Web server, DHT11

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

1. UVOD

Temperaturu i vlažnost zraka uočavamo svaki dan. Znatno utječu u oblikovanju klimatskih promjena, ekosustava i ljudskog zdravlja. Ljudska bića su jako osjetljiva na vlažnost zraka i temperaturu i zbog tog razloga je jako bitno nadgledati te dvije fizikalne veličine.

Temperatura predstavlja mjeru kinetičke energije čestica u zraku u određenome prostoru. Promjene temperature utječu na gustoću zraka i formiranje atmosferskoga tlaka. Kroz promjene u temperaturi možemo bolje razumjeti utjecaj na poljoprivredu, ekonomsku aktivnost i energetsku potrošnju. Najviše korištene mjerne jedinice za temperaturu su:

°C(stupanj Celzijev), F(Fahrenheit) i K(Kelvin).

Vlažnost zraka se odnosi na prisutnost vodene pare u atmosferi. Igra ključnu ulogu u formiranju oblaka i padalina. Analiza vlažnosti zraka pomaže u razumijevanju stvaranja oblaka te time i formacija padalina čime se mogu predvidjeti mogućnosti oborina, vlažnosti tla i pratećih ekoloških promjena.

U kontekstu završnog zadatka, glavni cilj je demonstrirati kako sustavno pratiti promjene u stvarnom vremenu, koristeći mikrokontroler i senzor kako bismo dublje razumjeli dinamiku temperature i vlažnosti zraka. Za potrebe ovog završnog rada je odabran mikrokontroler ESP32 zbog mogućnosti povezivanja s internetom preko WI-FI-a na kojem će biti spojen senzor temperature i vlage pod imenom DHT11. Sva mjerenja koja senzor prikuplja će biti spremljena u bazu podataka radi pohrane. Analizirat će se korištenje “endpointa” koji će web stranici prosljeđivati mjerenja iz baze podataka i time ih prikazivati u obliku zasebnih grafova za temperaturu i vlagu.

2. ESP32

ESP32 je niz mikrokontrolera tvrtke Espressif, različitih sposobnosti i performansi. Stekao je popularnost sa svojim specifikacijama kao što su:

- Mala cijena
- Niska snaga
- Wi-Fi mogućnosti
- Bluetooth
- Bogato periferno ulazno/izlazno sučelje
- Kompatibilan s Arduino “programskim jezikom”

ESP32 predstavlja vrlo moćan mikrokontroler koji je postao ključan dio u području “IoT” (Internet of Things). Kombinira visokoperformansno obradu podataka s integriranim Bluetooth i Wi-Fi funkcionalnostima koji inženjerima i programerima pruža široki spektar za razvijanje projekata.

ESP32-ova ključna karakteristika je njegova dvostruka jezgra arhitektura koja je omogućila još bržu i efikasniju obradu zadataka od njegovog prethodnika ESP8266. Pored tog, dolazi s različitim perifernim uređajima kao što su analogni-digitalni pretvornik(ADC), digitalne ulazno izlazne pinove, I2C, SPI i UART komunikacijske mogućnosti što daje korisnicima širok opseg opcija povezivanja različitih senzora, aktuatora i ostalih uređaja. Wi-Fi i Bluetooth komunikacije omogućuju bežično povezivanje što je daje mogućnost daljinskog upravljanja i komunikaciju s drugim uređajima na mreži.

Slika 1. ESP32 Chip

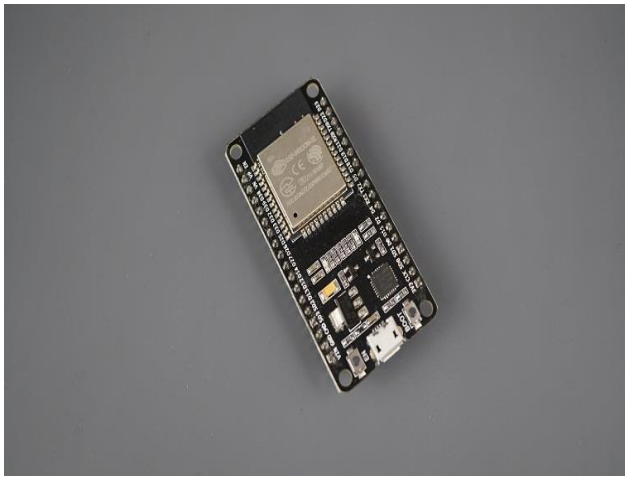
2.1 Razvojne ploče ESP32

Najčešće kada se priča o ESP32, priča se o razvojnoj ploči na koju je ESP32 spojen. Korištenje samog “chipa” nije jednostavno niti praktično, pogotovo pri učenju i testiranju.

Razvojne ploče se sastoje od svih potrebnih dijelova za napajanje i kontroliranje “chipa”.

Najbitnije značajke razvojne ploče su:

- **USB-to-UART sučelje i krug regulatora napona** – Sučelje je bitno jer mu je svrha da spoji ESP32 s računalom kako bi se programski kod mogao prenijeti.
- **BOOT i RESET/EN gumb**: Funkcija im je za resetiranje ploče i stavljanje ploče u “flashing” modu rada u kojemu se kod prenesi na ploču.
- **Broj pinova i konfiguracija pinova** – Svaki pin ima svoju svrhu ovisno kako je postavljen način rada u programskome kodu, stoga treba paziti kako se spaja.
- **Priključak za antenu** – Većina razvojnih ploča dolazi s već ugrađenom Wi-FI antenom. Neke ploče imaju priključak za vanjsku Wi-Fi antenu.
- **Priključak za bateriju** – Neke razvojne ploče imaju priključak za spajanje LiPo baterija, ali većina ploča se napajaju preko USB-B konektora koji se napajaju preko računala ili preko izvora poput “POWERBANK” ili direktno spojen preko adaptera na utičnicu.[1]

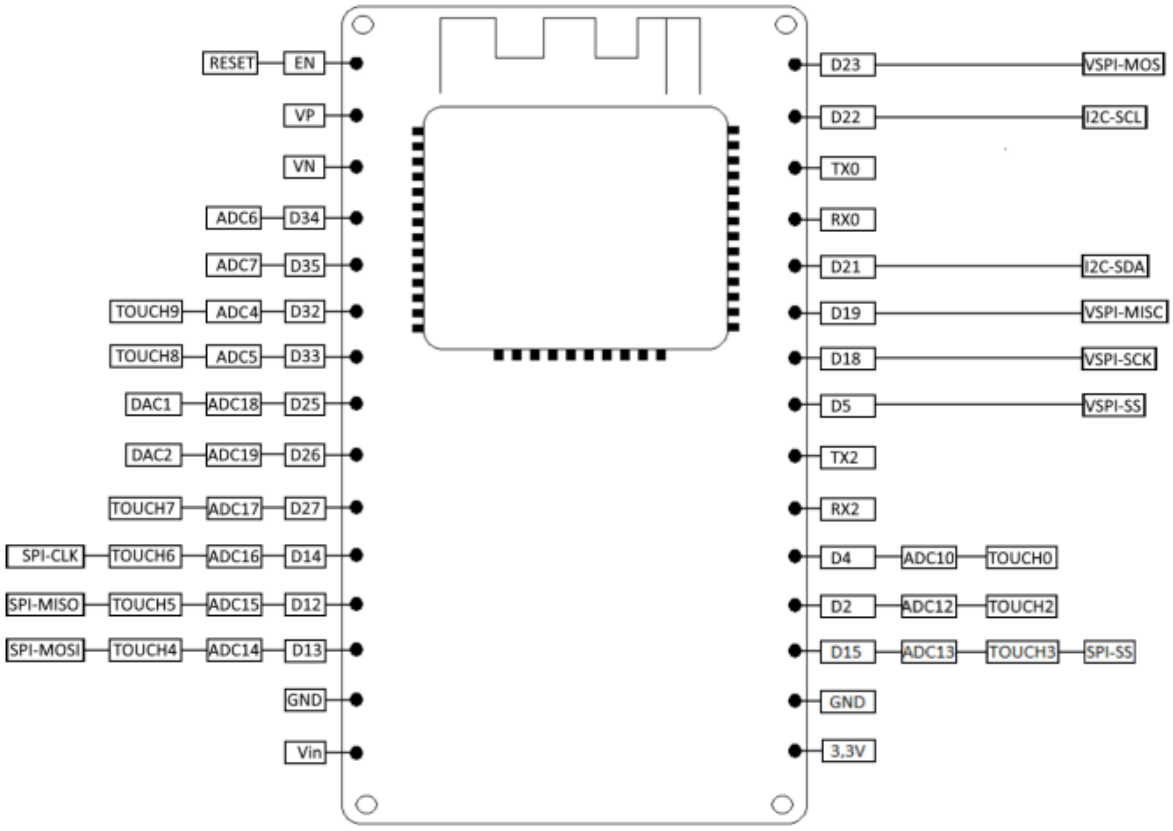


Slika 2. ESP32 razvojna ploča

Za potrebe ovog rada odabrana je NodeMCU ESP32 razvojna ploča od Joy-It kompanije. Ova razvojna ploča je jednostavna za programiranje putem Arduino integriranog razvojnog okruženja. Specifikacije ove razvojne ploče su:

- 2,4 GHz dual mode Wi-Fi i Bluetooth bežična veza.
- 512 kB SRAM i 4 MB memorije
- 2x DAC (Digital to Analog Converter)
- 15x ADC (Analog to Digital Converter)
- 1xSPI (Serial Peripheral Interface)
- 1x I²C (Inter-Integrated Circuit)
- 2x UART (Universal Asynchronous Receiver/Transmitter)

PWM (Pulse Width Modulation) se aktivira na svim digitalnim pinovima. Pregled pinova se može vidjeti na sljedećoj slici:



Slika 3. Položaj pinova na ESP32 razvojnoj ploči

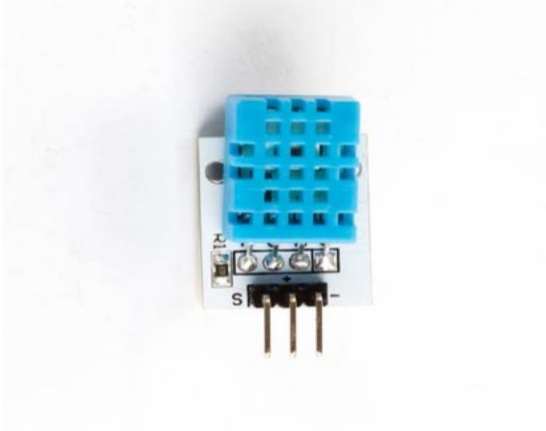
3. DHT11 SENZOR

DHT11 je digitalni senzor za mjerenje temperature i vlage koji se najčešće koristi u svrhe “IoT” (Internet of Things). Senzor omogućuje mjerenje temperature u rasponu od 0 do 50 stupnjeva Celzijusa s točnošću od ±2 stupnja Celzijusa, dok vlažnost mjeri u rasponu od 20 % do 90 % s točnošću od ±5 %. Senzor se sastoji od 3 pina: VCC (napajanje), GND (zemlja) i S pin koji je odgovoran za dvosmjernu komunikaciju između senzora i mikrokontrolera. Komunikacija se odvija u obliku digitalnih impulsa gdje senzor šalje niz bitova koji predstavljaju podatke o temperaturi i vlažnosti.

Tip DHT11 senzora koji je korišten u ovome završnome radu je model VMA311 od proizvođača VELLEMAN. Specifikacije ovoga senzora su prikazani na tablici 1.1, a izgled senzora je prikazan na slici 3.

Napon	5 VDC
Raspon temperature	0-50°C | greška +/- 2°C
Vlažnost	20-90 % | greška +/- 5%
Tip izlaznog signala	Digitalni
Dimenzije	39 x 23 x 10 mm
Maksimalna struja	2.5mA

Tablica 1: Karakteristike DHT11 senzora



Slika 4. Senzor za mjerenje temperature i vlage DHT11

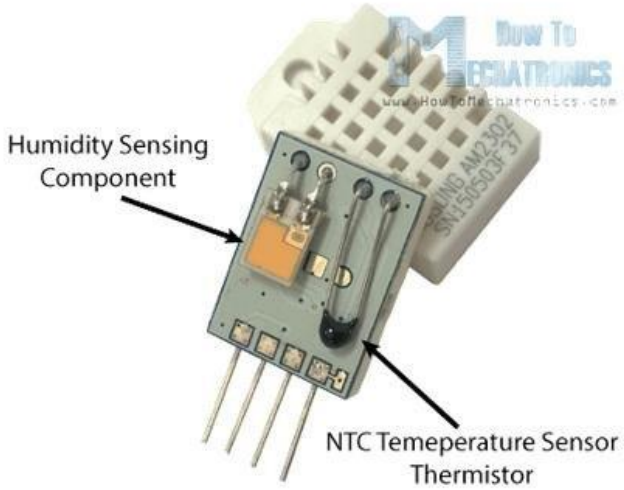
3.1 Način rada DHT11 senzora

Za mjerenje temperature u njemu se nalazi NTC (negativni temperaturni koeficijent) termistor. Termistor je elektronička komponenta čiji otpor ovisi o temperaturi. U ovome slučaju se koristi termistor NTC tipa koji pri porastu temperature njegov otpor pada.

Senzor generira električne impulse koji prolaze kroz termistor, a mjerenje promjene otpornosti omogućuje senzoru da očita temperaturu. To analogno očitanje zatim se pretvara u digitalni format kako bi se lakše prenio mikrokontroleru putem digitalne komunikacije.

Za mjerenje vlažnosti, DHT11 senzor sadrži u sebi kapacitivni senzor vlažnosti. On se sastoji od kondenzatorskog elemenata koji u sebi ima dvije elektrode s dielektričnim elementom koji sadržava vlagu između njih. Kada zrak u okolini postane vlažniji tada taj dielektrični materijal apsorbira vlagu i time mijenja kapacitet kondenzatora. Senzor generira promijenjeni kapacitivni signal koji je proporcionalan vlažnosti zraka.

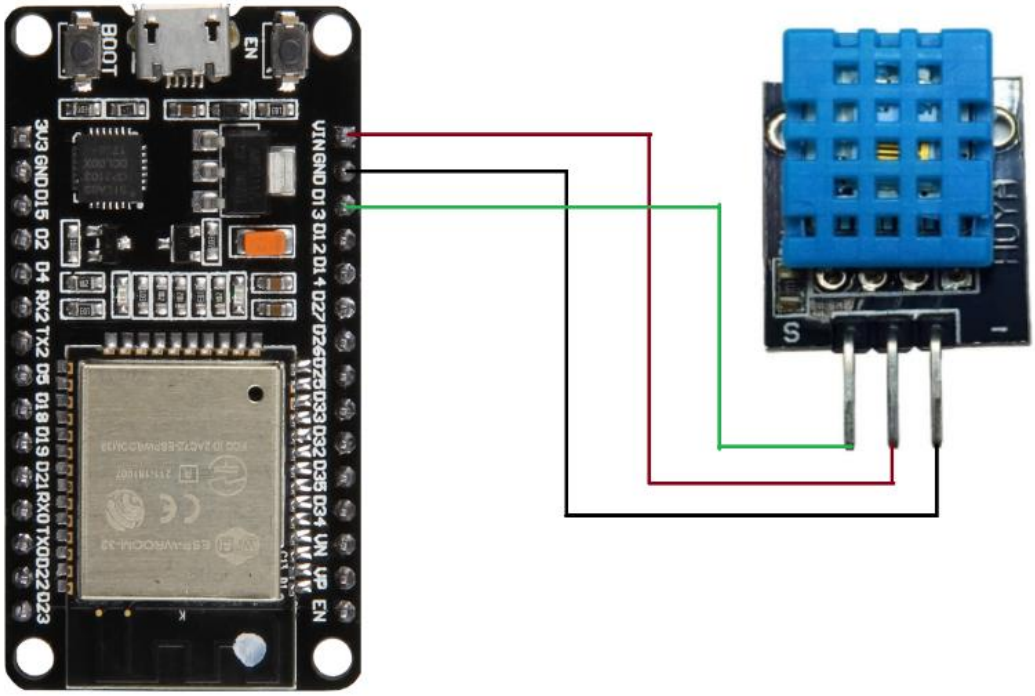
Integrirani krug (IC) u senzoru obrađuje ovaj signal i pretvara ga u digitalni format. Ova digitalna vrijednost vlažnosti zajedno s digitalnim podacima o temperaturi zatim se šalje mikrokontroleru putem digitalne komunikacije. [2]



Slika 5. Konstrukcija DHT11 senzora

4. SPOJ ESP32 RAZVOJNE PLOČE I DHT11 SENZORA

Pravilno povezivanje komponenata ključno je za uspješno funkcioniranje sustava. Spoj je vidljiv na slici 6. i izveden je tako da je + pin DHT11 senzora spojen na napajanje VCC čiji je izlazni napon 5 V na kojem DHT11 zahtijeva za inicijalizaciju. Ako se VCC spoji na 3.3 V senzor neće raditi. – pin senzora je spojen na uzemljenje (GND) ESP32 mikrokontrolera dok je signalni pin (S) spojen na D13 pin ESP32-a.



Slika 6. Shema ESP32 mikrokontrolera i DHT11 senzora

4.1 Arduino Integrirano Razvojno Okruženje (IDE)

Samim spajanjem senzora s razvojnom pločicom neće aktivirati senzor i samim time prikupljati podatke o temperaturi i vlažnosti zraka. Prvo trebamo programirati razvojnu ploču, a to se izvodi preko ARDUINO IDE-a. ESP32 nema svoje razvojno okruženje ali je kompatibilno s ARDUINO IDE-em.

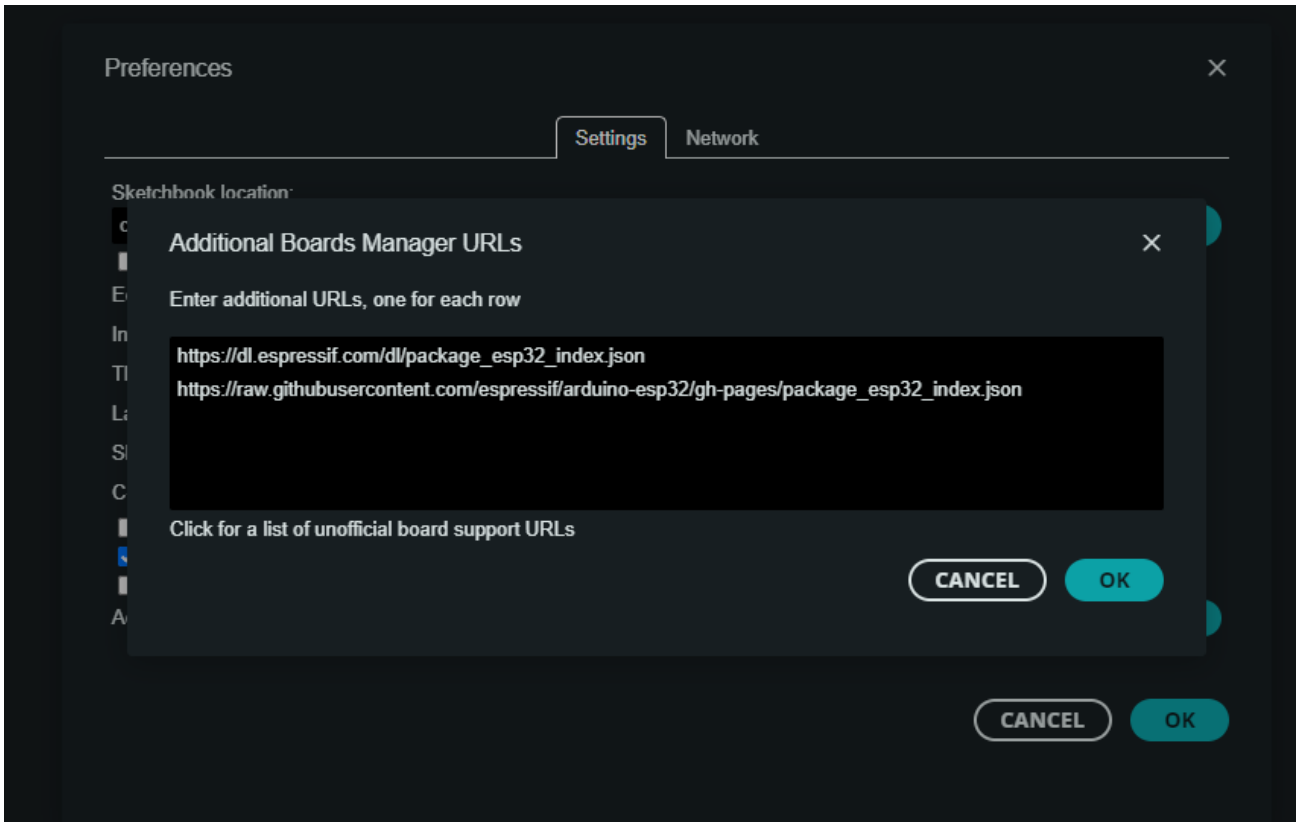
U praksi, velik broj programera koristi integrirano razvojno okruženje (IDE) prilagođeno Arduino platformi, koje je dostupno za različite operacijske sustave. Korisnici mogu pisati kod za Arduino u programskom jeziku C/C++. IDE pruža osnovne značajke poput označavanja sintakse, automatskog dovršavanja i prikaza grešaka radi lakšeg programiranja.

Arduino IDE sadrži mnoge ugrađene primjere i biblioteke koje olakšavaju početnicima u razvoju. Ovo omogućava brzi početak rada s osnovnim projektima. IDE automatski kompilira Arduino kôd u strojni jezik koji je razumljiv mikrokontroleru. Također omogućava prenošenje programa na Arduino ploču putem USB veze. Arduino IDE ima ugrađeni monitor serije koji omogućava praćenje izlaznih podataka iz Arduino mikrokontrolera. To je korisno za dijagnostiku i provjeru ispravnosti rada programa. Na njemu ćemo dobivati podatke od senzora za stanje temperature i vlažnosti zraka.

Arduino IDE je “open-source” softver, što znači da je dostupan za besplatno preuzimanje i prilagodbu prema potrebama korisnika. Velika zajednica korisnika i razvijatelja pridonosi stalnom poboljšavanju i dodavanju novih značajki. [3]

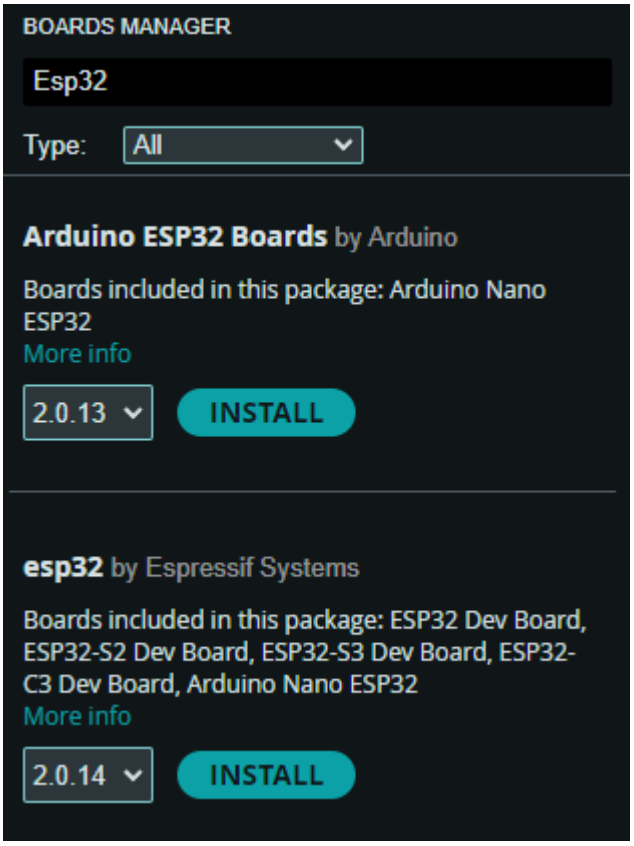
4.2 Inicijalizacija ESP32 s Arduino IDE

Spajanjem ESP32 razvojne ploče s računalom neće odmah omogućiti da se kod napisan u Arduino IDE-u prenese na razvojnu ploču. Kako bi to omogućili treba prvo u postavkama Arduino IDE-a postaviti 2 “linka” u polje za ploče koje nisu Arduino. Postupak je da se klikne na “File” > “Preferences”. Tada u polju "Additional Boards Manager URLs" se zalijepe 2 “linka” kao na slici 7. Ta 2 “linka” predstavljaju adrese mrežnog resursa koji se koristi u Arduino IDE za pristup informacijama o dodatnom paketu potrebnom za podršku ESP32 mikrokontrolera.



Slika 7. Dodavanje podrške za ESP32 razvojne ploče

Idući korak je instalirati biblioteku koja daje podršku za sve vrste ESP32 razvojnih ploča. Postupak je da se klikne pod “Tools” > “Board” > “Board Manager”. U polje za pretraživanje se upiše “esp32” i odabere se paket od “Espressif Systems” i odabere se “install” kao na slici 8.



Slika 8. Odabir paketa za ESP32 razvojnu ploču

Iz padajućeg izbornika odabirom opcije “Tools” > “Board” pojavljuje se “esp32” izbornik u kojemu biramo odgovarajuću razvojnu ploču. U kontekstu ovoga završnoga rada to je “ESP32 Dev Module”. Finalni korak je kada je ESP32 razvojna ploča spojena s računalom da se odabere “PORT” preko kojega računalo komunicira s razvojnom pločom. Sada je razvojna ploča spremna za programiranje. [4]

5. Programiranje ESP32 razvojne ploče

Za ispravan način DHT11 senzora potrebno je instalirati potrebne pakete. Postupak je da iz padajućeg izbornika odaberemo “Tools” > “Manage libraries” i u polje za pretraživanje potraže iduća 2 paketa:

- “DHT sensor library by Adafruit”
- “Adafruit Unified Sensor”

Program za mjerenje temperature i vlažnosti zraka prikazan je kodom:

```
#include <DHT.h>

DHT dht(12, DHT11);

float temp;
float humidity;

void setup() {
  // put your setup code here, to run once:
  dht.begin();
  delay(2000);

  Serial.begin(115200);
  Serial.println();
}

void loop() {
  // put your main code here, to run repeatedly:
  temp = dht.readTemperature();
  humidity = dht.readHumidity();
  Serial.print("Temperatura: ");
  Serial.print(temp);
  Serial.print(" Vlaga: ");
  Serial.print(humidity);
  Serial.println();

  delay(5000);
}
```

Korištenjem ugrađenog “Serial Monitora” u Arduinu, može se vidjeti u stvarnome vremenu što senzor mjeri kao što je prikazano na slici 9.

```
12:22:31.229 -> Temperatura: 23.10 Vlaga: 53.70
12:22:36.267 -> Temperatura: 24.00 Vlaga: 50.30
12:22:41.295 -> Temperatura: 24.00 Vlaga: 50.10
12:22:46.279 -> Temperatura: 24.00 Vlaga: 49.70
12:22:51.346 -> Temperatura: 24.00 Vlaga: 50.00
12:22:56.354 -> Temperatura: 24.10 Vlaga: 50.30
12:23:01.375 -> Temperatura: 24.10 Vlaga: 50.30
12:23:06.416 -> Temperatura: 24.10 Vlaga: 50.60
12:23:11.426 -> Temperatura: 24.10 Vlaga: 50.80
12:23:16.474 -> Temperatura: 24.10 Vlaga: 51.10
12:23:21.488 -> Temperatura: 24.10 Vlaga: 50.50
12:23:26.511 -> Temperatura: 24.20 Vlaga: 50.80
12:23:31.534 -> Temperatura: 24.20 Vlaga: 50.20
12:23:36.567 -> Temperatura: 24.20 Vlaga: 50.00
12:23:41.592 -> Temperatura: 24.20 Vlaga: 50.10
```

Slika 9. Prikazivanje mjerenja temperature i vlažnosti zraka

Vremensko kašnjenje od najmanje 2 sekundi je preporučeno za preciznije mjerenje, ali tu je postavljeno 5 sekundi jer će mjereni podaci kasnije biti poslani prema bazi podataka i radi stabilnosti je uzeto duže vrijeme. Također pri inicijalizaciji senzora je dobro uzeti 2-3 sekunde vremenskog kašnjenja kako bi se mogao izvršiti interni postupak koji uključuje kalibraciju i stabilizaciju senzora. Ako se ne postavi vremensko kašnjenje pri inicijalizaciji dolazi do netočnih rezultata mjerenja ili senzor uopće tada ne mjeri. Svaki model DHT11 ima različite periode inicijalizacije.

6. MYSQL baza podataka

Već smo u stanju napraviti web stranicu, ali ona bi samo pokazivala trenutne vrijednosti temperature i vlage i brisala bi stare vrijednosti jer ih ne bi mogla nigdje spremati. Zbog toga nam je potrebna baza podataka. Za svrhe ovog završnog rada je odabrana MYSQL baza podataka

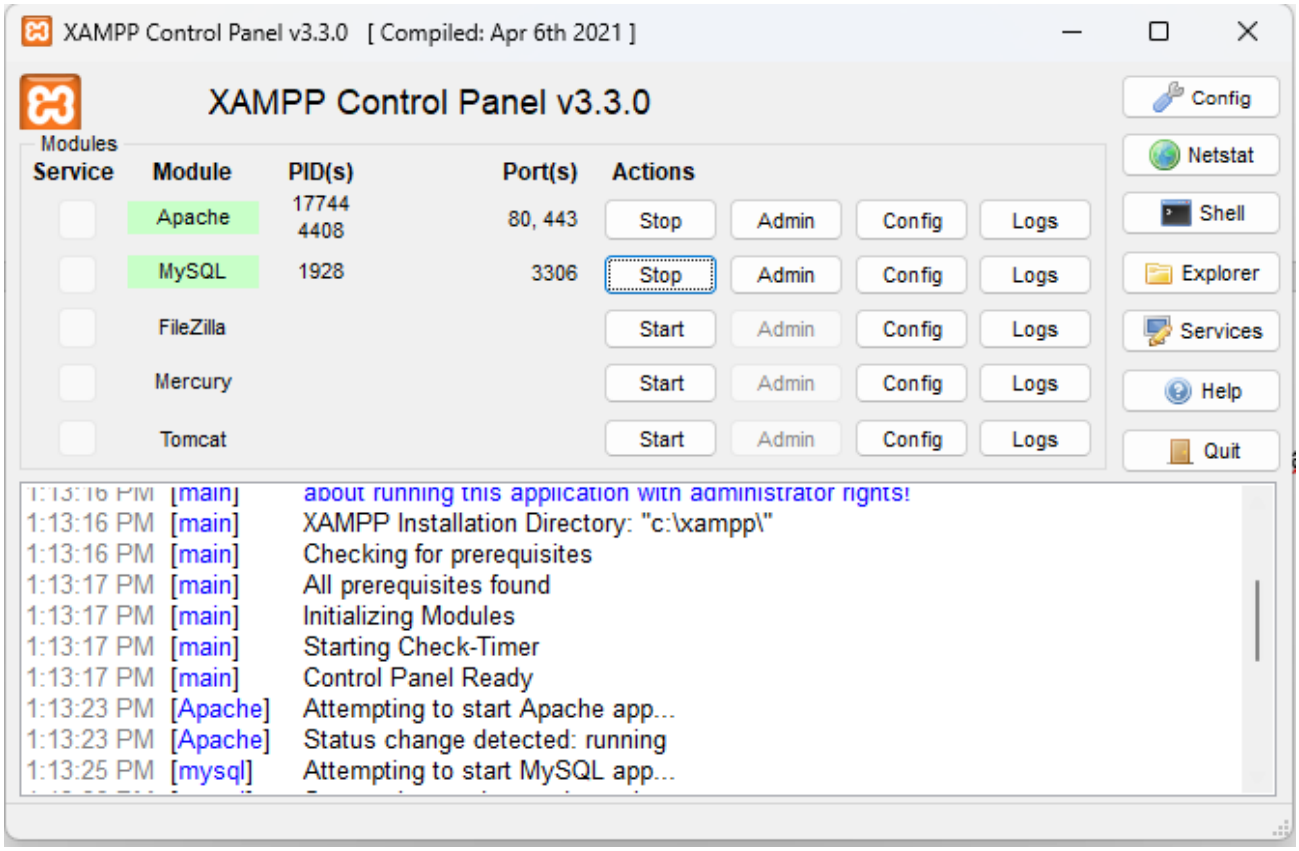
MySQL predstavlja besplatan i otvoren izvor sistema za upravljanje bazama podataka, pružajući učinkovito skladištenje, upravljanje i pristup podacima. Nastao je kao rezultat rada stručnjaka u industriji i stekao je široku popularnost zbog svoje otvorene prirode i snažnih značajki. Centralna karakteristika MySQL-a leži u organizaciji podataka u tablicama te sposobnosti međusobnog povezivanja tih tablica.

MySQL je široko prihvaćen u različitim industrijama, uključujući web razvoj, e-trgovinu i sustave za upravljanje sadržajem. U pogledu sigurnosti, MySQL implementira razne mehanizme kao što su autentikacija, autorizacija i enkripcija kako bi zaštitio podatke. Također, pruža alate i postupke za administraciju baze podataka, olakšavajući upravljanje kompleksnim sustavima.

MYSQL se pokreće preko web servera što znači da se pristupa preko web preglednika. To je najlakša varijanta za koristiti tu bazu, ali se također može pristupiti uz MYSQL Workbench alata ili komandne linije. Za pokretanje web servera se koristi Apache. Njegova glavna funkcija je upravljanje HTTP zahtjevima, obrađivanje dinamičkih sadržaja, i slanje statičkih ili dinamički generiranih stranica korisnicima. [5]

Postoji program koji nam omogućuje paljenje ta 2 servisa vrlo jednostavno. XAMPP je distribucija softvera koja uključuje Apache web server, MySQL bazu podataka, PHP i druge komponente, pružajući potpuno okruženje za razvoj i testiranje web aplikacija na lokalnom računalu. Pri pokretanju XAMPPa i odabirom opcije za pokretanje Apache servera i MySQL baze podataka, oba servisa se pokreću zbog potreba razvojnog okruženja.

XAMPP se može preuzeti s njihove službene stranice i moguće ga je instalirati na Windows/Linux/Mac platformama.[6] Sučelje XAMPPa izgleda kao na slici 10.

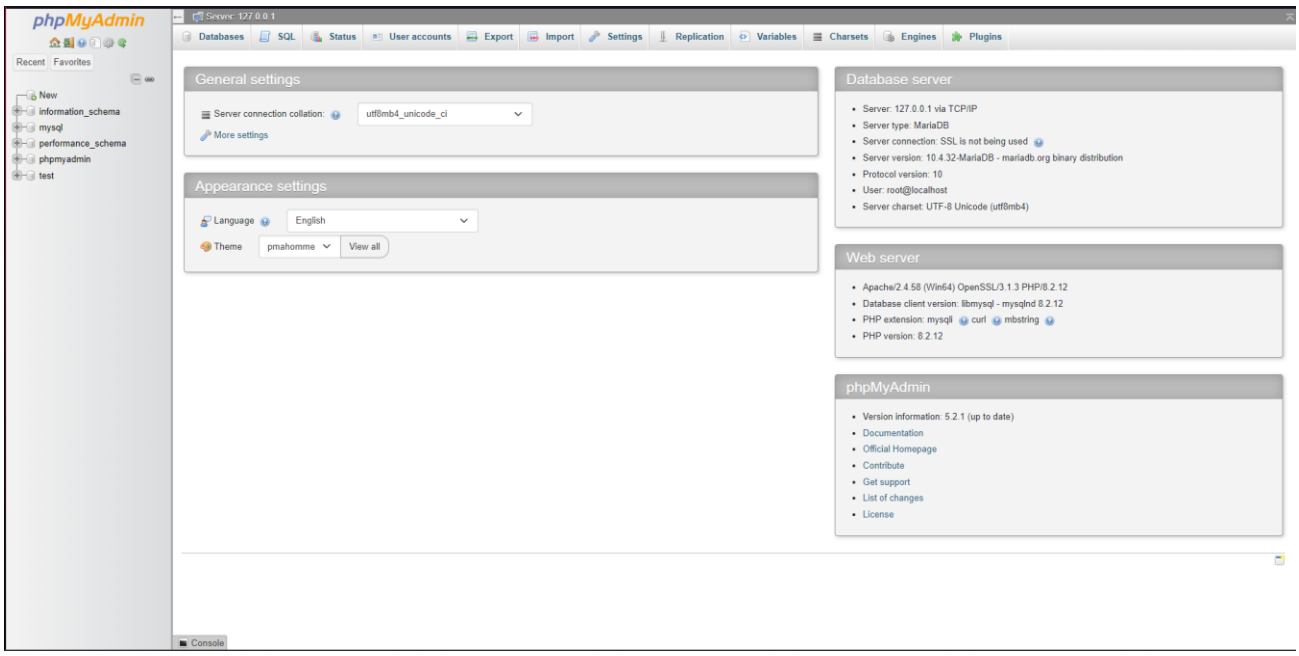


Slika 10. Sučelje XAMPP programa

Kako bi pristupili bazi, u web pregledniku je potrebno upisati idući link:

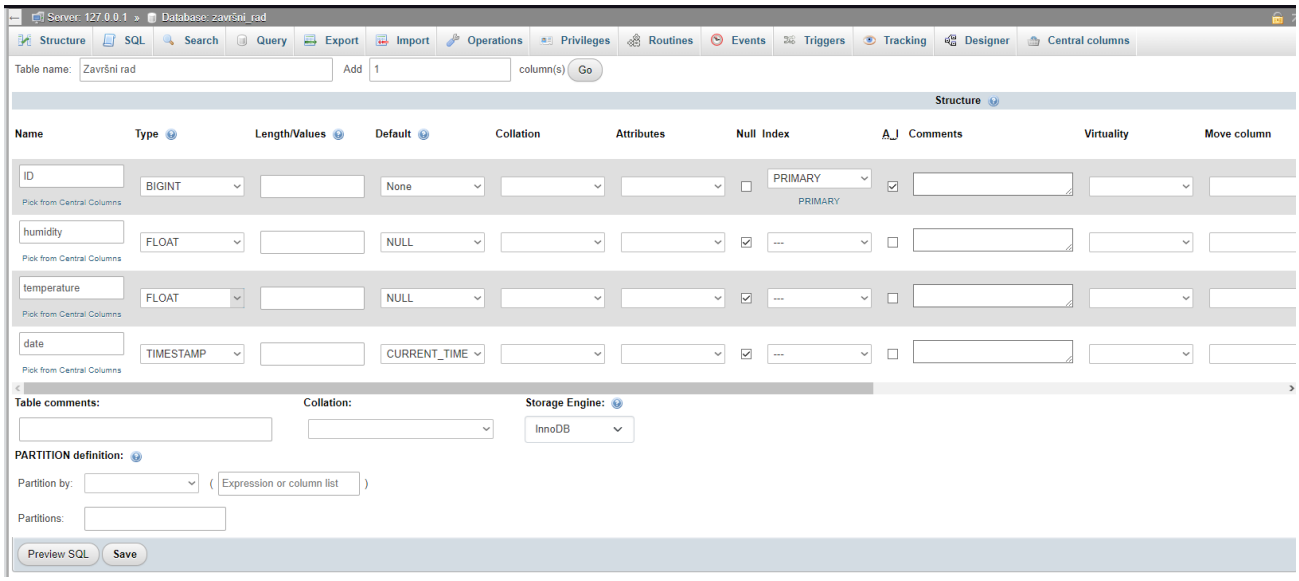
- http://localhost/phpmyadmin/

Izgled stranice s već postojećim različitim bazama podatak izgleda kao na slici 11.



Slika 11. Prikaz web stranice za pristupanje MYSQL bazama podataka.

U lijevom izborniku nalazi se opcija za stvaranje novih baza podataka. Pri stvaranju nove baze potrebno je navesti ime baze, broj redova, nazive redova, te njihov tip. U ovome slučaju ime tablice će biti “Završni_rad”, a broj redova će biti 4. Razlog zašto 4 je taj jer će nam trebati ID reda, zasebni redovi temperature i vlage i red kada je podatak izmjeren. Prikaz toga je prikazan na slici 12.



Slika 12. Prikaz stvaranja tablica u bazi podataka

7. Spremanje mjerenja u bazu podataka

Svako mjerenje koje senzor izmjeri treba odmah spremiti u bazu podataka. Potrebno je prvo locirati instalaciju XAMPP paketa, pronaći mapu pod imenom “htdocs”, te u njoj napraviti novu datoteku s bilo kojim imenom. U njoj će biti spremljena PHP skripta koja se koristi za povezivanje s MYSQL bazom. [7]

```
<?php
class dht11{
 public $link='';
 function __construct($temperature, $humidity){
  $this->connect();
  $this->storeInDB($temperature, $humidity);
 }

 function connect(){
  $this->link = mysqli_connect('localhost','root','') or die('Cannot connect to
the DB');
  mysqli_select_db($this->link,'završni_rad') or die('Cannot select the DB');
 }

 function storeInDB($temperature, $humidity){
  $query = "insert into završni rad set humidity='".$humidity."',
temperature='".$temperature."'";
  $result = mysqli_query($this->link,$query) or die('Errant query:  '.$query);
 }

}
if($_GET['temperature'] != '' and  $_GET['humidity'] != ''){
 $dht11=new dht11($_GET['temperature'],$_GET['humidity']);
}


?>
```

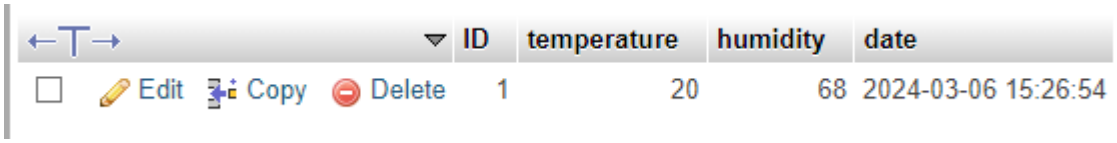
Za svako spremanje mjerenja, skripta mora dobiti ovakav oblik zahtjeva:

- http://localhost/testcode/dht.php?temperature=[mjerenje]&humidity=[mjerenje]

Naprimjer kada ručno u web pregledniku se upiše:

- http://localhost/testcode/dht.php?temperature=20&humidity=68

u bazi podataka će se spremiti da je temperatura bila 20 °C, a vlažnost 68 %



← T →	ID	temperature	humidity	date
☐ Edit Copy Delete	1	20	68	2024-03-06 15:26:54

Slika 13. Ručno spremanje podataka u bazu

Takav princip da se preko HTTP zahtjeva spremaju podaci se mogu ostvariti tako da se u programskome kodu za ESP32 nakon svakoga mjerenja temperature i vlažnosti zraka pošalje HTTP zahtjev i spremi podatke u bazu. Novi kod za ESP32:

```
#include <DHT.h>
#include <WiFiClient.h>
#include <WiFi.h>


DHT dht(12, DHT11);


float temp;
float humidity;

const char* ssid = "Ime_mreže";//
const char* password = "Šifra_mreže";
char server[] = "IP_mreže";


WiFiClient client;

void setup() {
  // put your setup code here, to run once:
  dht.begin();
  delay(2000);

  Serial.begin(115200);

  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
```

```
  }
  Serial.println("");
  Serial.println("WiFi connected");

  Serial.println("Server started");
  Serial.print(WiFi.localIP());
  delay(1000);
  Serial.println("connecting...");
}

void loop() {
  // put your main code here, to run repeatedly:
  temp = dht.readTemperature();
  humidity = dht.readHumidity();
  Sending_To_phpmyadmindatabase();
  delay(5000);
}

void Sending_To_phpmyadmindatabase() {
  if (client.connect(server, 80)) {
    Serial.println("connected");

    // Send HTTP GET request
    client.print("GET /testcode/dht.php?humidity=");
    client.print(humidity);
    client.print("&temperature=");
    client.print(temp);
    client.println(" HTTP/1.1");
    client.println("Host: IP_mreže");
    client.println("Connection: close");
    client.println();

    Serial.print("GET /testcode/dht.php?humidity=");
    Serial.print(humidity);
    Serial.print("&temperature=");
    Serial.println(temp);
  }
  else {
    Serial.println("connection failed");
  }
}
```

Na slici 14 je prikazano kako se šalje HTTP zahtjev, a na slici 15 je prikaz spremanja mjerenja.

```
10:08:57.282 -> GET /testcode/dht.php?humidity=54.50&temperature=23.60
10:09:02.325 -> connected
10:09:02.325 -> GET /testcode/dht.php?humidity=56.90&temperature=23.70
10:09:07.353 -> connected
10:09:07.353 -> GET /testcode/dht.php?humidity=56.80&temperature=23.80
10:09:12.427 -> connected
10:09:12.427 -> GET /testcode/dht.php?humidity=55.90&temperature=23.80
10:09:17.441 -> connected
10:09:17.483 -> GET /testcode/dht.php?humidity=55.10&temperature=23.80
10:09:22.503 -> connected
```

Slika 14. HTTP zahtjevi za spremanje podataka

	ID	temperature	humidity	date
Edit Copy Delete	14	23.7	51.2	2024-03-06 15:49:23
Edit Copy Delete	15	23.7	51.7	2024-03-06 15:49:28
Edit Copy Delete	16	23.7	51.6	2024-03-06 15:49:33
Edit Copy Delete	17	23.7	51.2	2024-03-06 15:49:38
Edit Copy Delete	18	23.7	51.3	2024-03-06 15:49:43
Edit Copy Delete	19	23.7	51.4	2024-03-06 15:49:48
Edit Copy Delete	20	24.4	53.7	2024-03-06 15:49:53
Edit Copy Delete	21	25.3	54.8	2024-03-06 15:49:58
Edit Copy Delete	22	26.1	55.5	2024-03-06 15:50:03
Edit Copy Delete	23	26.9	54.6	2024-03-06 15:50:08
Edit Copy Delete	24	27.8	53.6	2024-03-06 15:50:13
Edit Copy Delete	25	28.3	51.2	2024-03-06 15:50:18
Edit Copy Delete	26	28.1	44.7	2024-03-06 15:50:24
Edit Copy Delete	27	23.6	54.5	2024-03-12 10:08:57
Edit Copy Delete	28	24.1	53.2	2024-03-12 10:11:29
Edit Copy Delete	29	24	53	2024-03-12 10:11:33
Edit Copy Delete	30	24.1	53.2	2024-03-12 10:11:38
Edit Copy Delete	31	24	53	2024-03-12 10:11:43
Edit Copy Delete	32	24	53.2	2024-03-12 10:11:48
Edit Copy Delete	33	24	53	2024-03-12 10:11:53
Edit Copy Delete	34	24.1	53	2024-03-12 10:11:58

Slika 15. Spremljena mjerenja temperature i vlažnosti zraka

8. DJANGO Framework

Za web stranice služit ćemo se Djangom. Django je visokokvalitetni web “framework” napisan u programskom jeziku Python, koji omogućava brzo i efikasno razvijanje sigurnih i skalabilnih web aplikacija. Razvijen je s fokusom na jednostavnost, ponovnu upotrebu koda i ubrzanje procesa razvoja. Ovaj “framework” je otvorenog koda, što znači da je dostupan širokoj zajednici programera i omogućava suradnju na poboljšanjima i ispravkama.[8]

Primarni cilj Djanga je olakšati stvaranje složenih web stranica vođenih bazama podataka. Okvir naglašava mogućnost ponovne upotrebe i "priključivosti" komponenti, manje koda, nisko povezivanje, brzi razvoj i načelo “ne ponavljaj se”. Python se koristi posvuda, čak i za postavke, datoteke i modele podataka.

Glavne osobine Django “frameworka”:

- **ORM (Object-Relational Mapping)**: Django pruža ORM sistem koji omogućava mapiranje objekata u bazi podataka, čime se smanjuje potreba za pisanjem SQL upita. Ovo olakšava rad s bazom podataka i čini aplikaciju neovisnom o određenom sustavu upravljanja bazom podataka.
- **Administracijsko sučelje**: Django dolazi s ugrađenim administracijskim sučeljem koje omogućava jednostavno upravljanje podacima u bazi, bez potrebe za pisanjem posebnog koda. Ovo je korisno za administratore sustava ili razvojne timove koji žele brzo pregledati i urediti podatke.
- **URL rute i pogledi**: Django koristi jasan sustav za definiranje URL ruta i povezivanje s odgovarajućim pogledima (“views”). Ovo omogućava organizaciju ruta u aplikaciji i odvajanje logičkih dijelova koda.
- **Sigurnost**: Django uključuje niz sigurnosnih mehanizama ugrađenih u “framework”, uključujući zaštitu od CSRF (“Cross-site request forgery”) napada, SQL injekcija i drugih sigurnosnih prijetnji.

8.1 Stvaranje virtualnog okruženja

U praksi je najbolje svaki novi Django projekt stvarati u takozvanim virtualnim okruženjima. Virtualno okruženje (eng. virtual environment) je praksa izoliranja Python okoline za svaki projekt, čime se osigurava da projekt ima svoje nezavisne biblioteke i zavisnosti. Biblioteke i zavisnosti će biti samo povezane za taj određeni projekt, neće biti globalno povezane s ostalim projektima na računalu.

Primjena virtualnog okruženja za Django projekte pomaže u održavanju čistoće i organizacije projekta, smanjuje konflikte između različitih projekata te olakšava upravljanje zavisnostima i reprodukciju okoline.

Prvo što je potrebno napraviti je stvoriti mapu na radnoj površini s bilo kojim imenom. U ovome slučaju to će biti “Završni rad”. Nakon toga je potrebno otvoriti naredbeni redak (command prompt – cmd) i u njemu doći do lokacije te mape. Postupak je prikazan na slici 16.

Slika 16. Usmjeravanje u novo kreiranu mapu “Završni rad”.

Nakon pristupanja mapi preko naredbenog retka potrebno je upisati iduću komandu:

- python -m venv venv

Ova komanda stvara virtualno okruženje i u nju će se spremati sve biblioteke koje će se instalirati za ovaj završni rad.

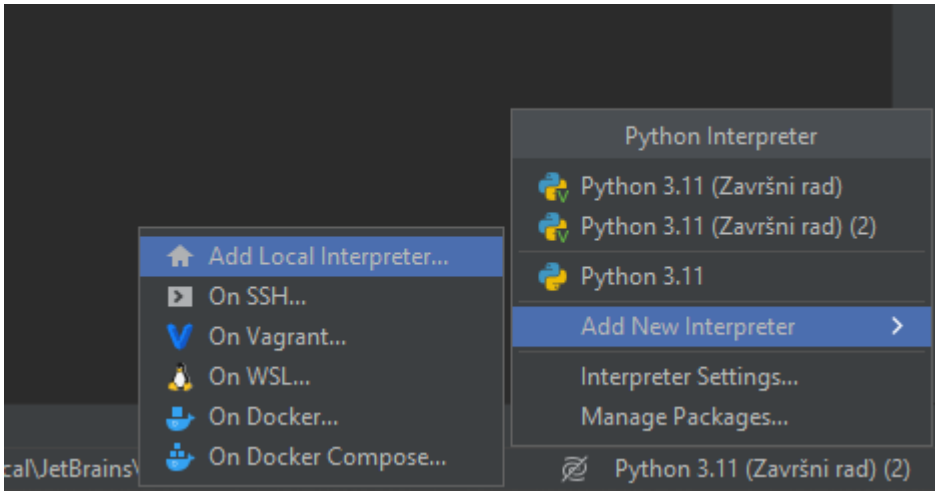
IDE u kojemu je pisan kod za Django projekt je PyCharm, tvrtke JetBrains.

U PyCharm okruženju je potrebno locirati virtualno okruženje kako instalacija potrebnih biblioteka ne bi došla u konflikt s bibliotekama koje su instalirane globalno za cijeli sustav.

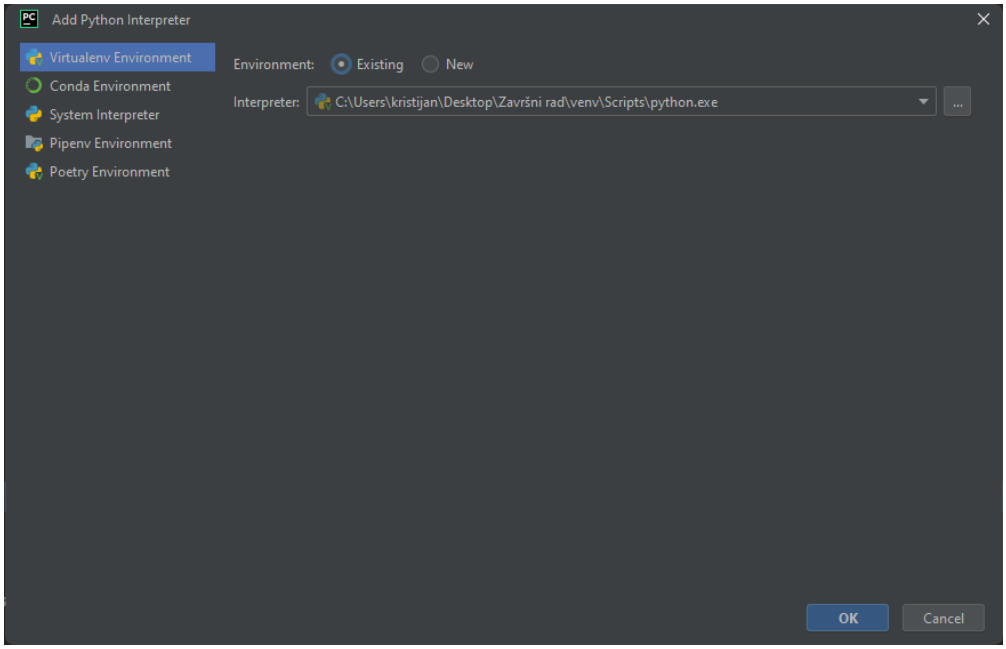
Postupak je sljedeći:

U donjem desnome kutu treba stisnuti na verziju Pythona koju koristimo i dobiti ćemo izbornik. U njemu treba stisnuti “Add new interpreter” i u idućem izborniku odabrati “Add local Interpreter” kao na slici 17.

Potrebno je odabrati opciju "existing" i kao na slici 18. pronađi datoteku “Scripts” u venv datoteci i stisnuti na “python.exe”.



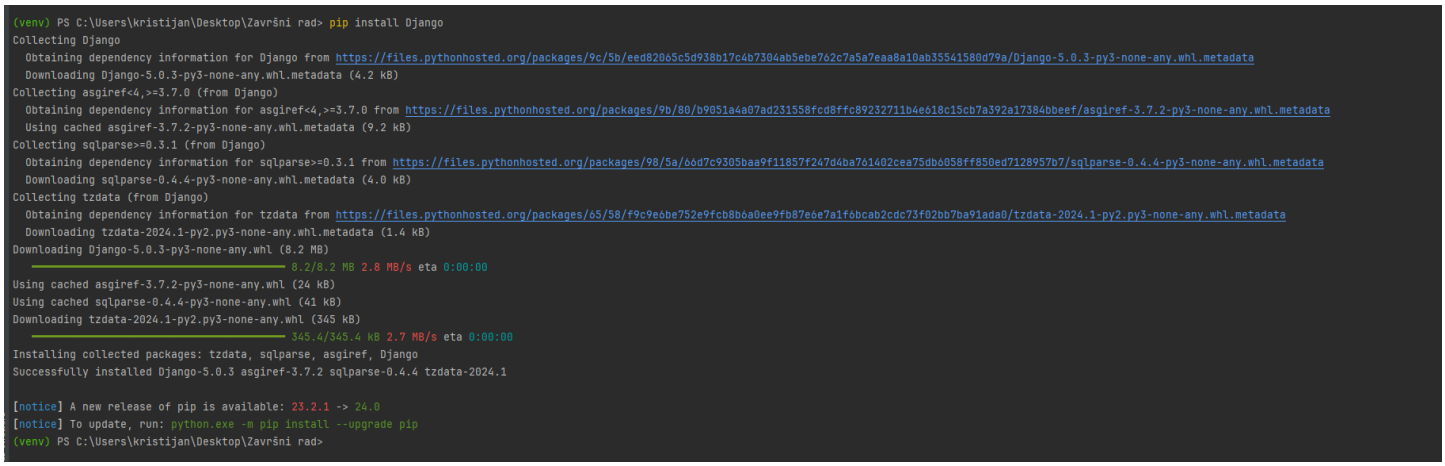
Slika 17. Način dodavanja novog interpretera



Slika 18. Dodavanje virtualnog okruženja u Pycharmu

8.2 Instalacija i postavljanje Django frameworka

Sama instalacija Django “frameworka” je vrlo jednostavna. Svodi se na jednu komandu liniju koju je potrebno upisati u terminal od PyCharm. Postupak je na slici 19.



Slika 19. Instalacija Django frameworka.

Nakon instalacije Django sustava, potrebno je stvoriti Django projekt. Postupak je da se u terminalu upiše komanda:

- django-admin startproject Mjerenje

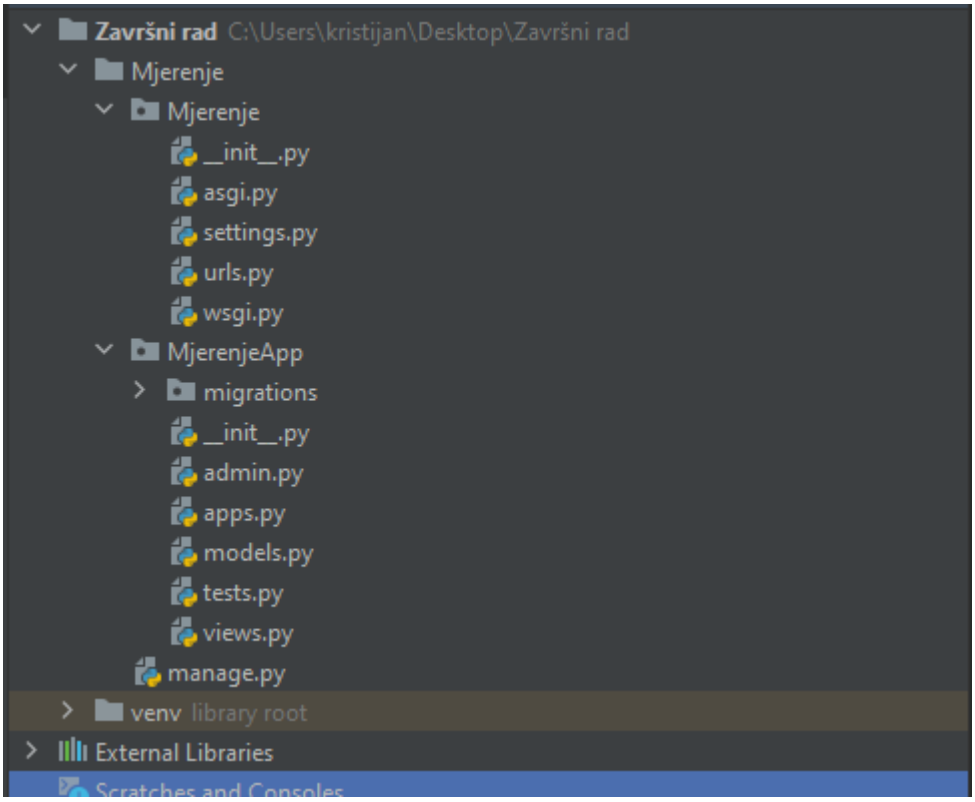
To će stvoriti mapu s potrebnim datotekama za rad Djanga. Zadnji korak je stvoriti takozvanu “aplikaciju” u kojoj će web stranica raditi. Prvo treba u terminalu otvoriti novu mapu koja je bila stvorena pri stvaranju projekta, u ovome slučaju to je mapa “Mjerenje”. Postupak je prikazan s idućom komandom:

- cd Mjerenje

Nakon ulaska u tu mapu, upisuje se iduća komanda:

- django-admin startapp MjerenjeApp

Prikaz mapa sa svim potrebnim datotekama je prikazan na slici 20.



Slika 20. Prikaz mapa Django projekta

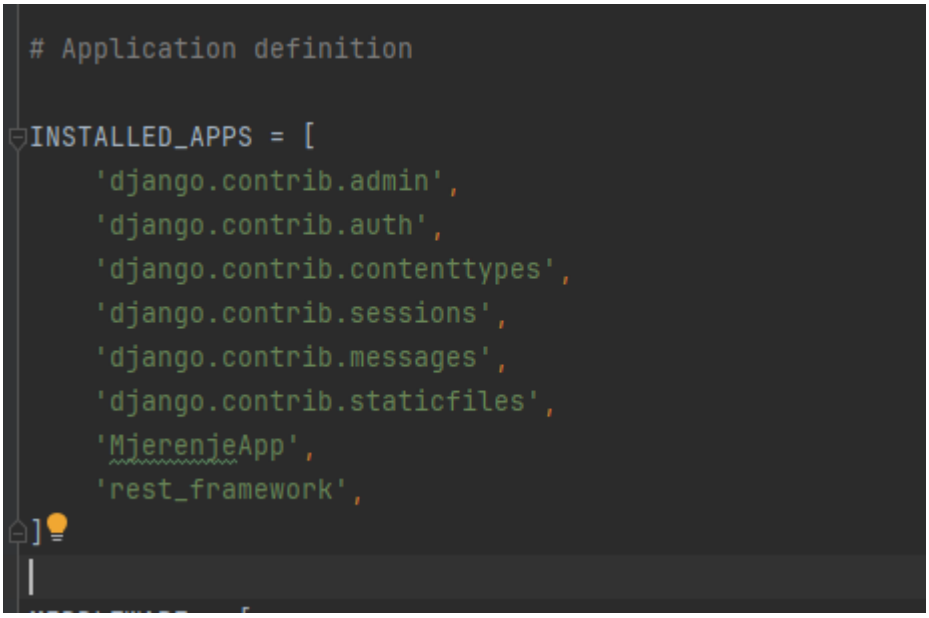
Najveći dio posla će biti u “MjerenjeApp” mapi, dok u “Mjerenje” mapi će se postaviti koju bazu podataka će Django gledati. Za kraj postavljanja projekta je potrebno instalirati par Python paketa:

- djangorestframework==3.14.0
- mysqlclient==2.2.1

Način instalacije je pip install [ime paketa]

Nakon instalacije u mapi “Mjerenje” u settings.py datoteci je potrebno dodati par argumenata.

Prvi argument koji se dodaje je naziv imena aplikacije koja je ranije stvorena. To je “MjerenjeApp” aplikacija. Ona se dodaje kako bi Django inicijalizirao tu aplikaciju, bez toga ona neće raditi. Dodaje se još i ime paketa pod nazivom “rest_framework”. On nam služi kasnije za stvaranje takozvanih “endpointa” koji se kasnije koriste za pozivanje funkcija koje će nam poslati mjerenja iz baze podataka. Način na koji se dodaju argumenti su prikazani na slici 21.



Slika 21. Dodavanje potrebnih argumenata Django projekta

Za kraj je potrebno povezati MYSQL bazu podataka s Djangom. Postupak je prikazan na slici 22.

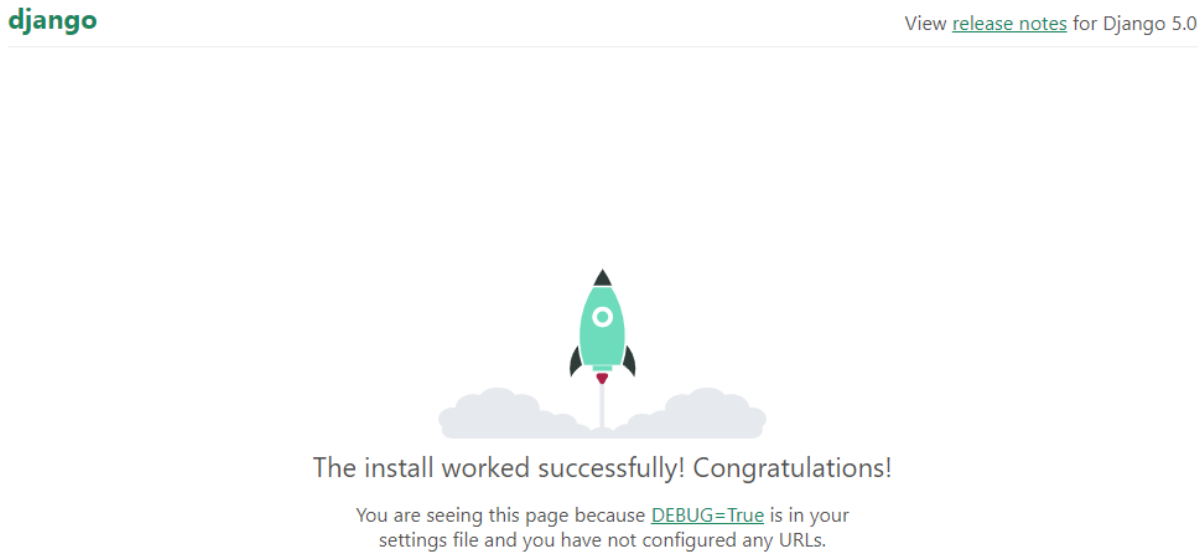
Slika 22. Povezivanje baze podataka s Djangom

Potrebno je paziti na ime baze na koju se veže i da se unesu dobri podaci za autentikaciju.

Kako bi znali da li sve dobro radi u terminalu moramo biti u “Mjerenje” mapi i potrebno je upisati iduću komandu:

- Python manage.py runserver

U terminalu će se pojaviti adresa na koju pristupamo našoj web stranici i ako smo sve dobro napravili na slici 23 je prikaz web stranice na kojoj piše da je instalacija uspješno odrađena.



Slika 23. Prikaz uspješno instalirane web stranice

U terminalu se pojavljuje poruka da imamo neodrađene migracije. To su zapravo baze podataka potrebne Djangu za rad. Kako bi riješili te migracije potrebno je samo upisati iduću komandu:

- python manage.py migrate

9. Endpoint

U Django web “frameworku”, "endpoint" se obično odnosi na određenu URL rutu koja je definirana u Django projektu. “Endpointi” su povezani s funkcijama ili klasama pogleda koje obrađuju zahtjeve poslane na te rute. To znači kada budemo pristupali web stranici u pozadini će se pozvati zahtjevi za dohvaćanje mjerenja temperature i vlažnosti zraka iz baze podataka.

Prvi korak je povezivanje Django modela s bazom podataka. Modeli u Django web “frameworku” su Python klase koje definiraju strukturu podataka web aplikacije. Ovi modeli omogućuju interakciju s bazom podataka na način koji je jednostavan za korištenje i razumijevanje.

Svaki put kada se stvara novi model time se stvara i nova tablica u bazi podataka. U ovome slučaju imamo situaciju gdje već imamo stvorenu tablicu s podacima o mjerenjima temperature i vlažnosti zraka. Način na koji će Django povezati tablicu s modelom se vrši idućim programskim kodom:

```
from django.db import models

class Dht11(models.Model):
    id = models.BigAutoField(db_column='ID', primary_key=True)
    humidity = models.FloatField(blank=True, null=True)
    temperature = models.FloatField(blank=True, null=True)
    date = models.DateTimeField()

    class Meta:
        managed = False
        db_table = 'dht'
```

U modelu potrebno je definirati argumente na isti način kao i u bazi podataka. Također tip argumenta mora biti isti. U klasi “Meta” db_table argument spaja model s tablicom pod imenom “dht”. Nakon svakog stvaranja novoga modela potrebno je izvršiti migracije.

Postupak je takav da se u terminalu upišu iduće dvije komande:

- python manage.py makemigrations
- python manage.py migrate

Prije nego što se počnu raditi funkcije za slanje mjerenja web stranici potrebno je te podatke pretvoriti u odgovarajući oblik. Za to nam služe “serializeri”.

“Serializeri” u Django web “frameworku” služe za pretvaranje kompleksnih tipova podataka (poput Django modela) u formate koji su pogodni za prijenos preko mreže, kao što su JSON, XML, ili drugi. Također, “serializeri” omogućuju obrnuti proces, tj. pretvaranje ovih formata nazad u Python objekte.

Glavna svrha “serializera” je olakšati komunikaciju između web aplikacije i klijentskih aplikacija (npr. JavaScript aplikacija u “Frontendu” ili mobilna aplikacija) putem API-ja (Application Programming Interface).

Serijalizacija je vrlo jednostavna, potrebno je napraviti novu mapu u “MjerenjaApp” i nazvati je “serializers” i u njoj napraviti novu Python datoteku pod bilo kojim imenom. U ovome slučaju smo je nazvali readings.py. Serijalizacija je prikazana idućim programskim kodom:

```
from rest_framework import serializers
from ..models import Dht11

class ReadingsSerializer(serializers.ModelSerializer):

    class Meta:
        model = Dht11
        fields = '__all__'
```

Također u “serializers” mapi je potrebno napraviti __init__.py Python datoteku u kojoj moramo inicijalizirati sve python datoteke koje se nalaze u toj mapi. Ako se to ne napravi Python neće moći prepoznati inicijalizaciju tih datoteka u ostalim Python datotekama. Način na koji se inicijalizira readings.py datoteka je:

- `from .readings import *`

Sada smo spremni za stvaranje funkcija za slanje podataka prema web stranici. Te funkcije stvaramo u views.py datoteci.

Stvorit ćemo jednu funkciju i jednu klasu. U funkciji će biti programski kod za generiranje web stranice u kojoj će biti definirani koji HTML predložak će se stvarati. U klasi ćemo napraviti HTTP GET funkciju u kojoj ćemo prikupljati podatke o temperaturi, vlažnosti zraka, datumu i vremenu mjerenja i ID mjerenja.

```
from django.shortcuts import render
from .models import Dht11
from .serializers.readings import ReadingsSerializer
from rest_framework.response import Response
from rest_framework import generics, status
from django.http import HttpResponse
# Create your views here.


def Index(request):
    return render(request, 'index.html')


class Readings(generics.GenericAPIView):
    """
        Dohvati sva očitanja iz baze podataka.
    """
    queryset = Dht11.objects.all()
    serializer_class = ReadingsSerializer

    def get(self,request):
        readings = Dht11.objects.all()
        reading_data = []
        for reading in readings:
            data = {
                'id': reading.id,
                'temperatura': reading.temperature,
                'Vlažnost': reading.humidity,
                'Timestamp': reading.date
            }
            reading_data.append(data)

        return Response(reading_data,status=status.HTTP_200_OK)
```

Klasa “Readings” predstavlja “endpoint” koji će kasnije biti iskorišten za stvaranje grafova temperature i vlažnosti zraka.

Način na koji će se pozivati je preko URL-a. U “MjerenjeApp” mapi je potrebno napraviti novu Python datoteku pod imenom urls.py. Ova datoteka obično sadrži rute koje su

specifične za funkcionalnosti unutar te aplikacije. Na primjer, imamo aplikaciju za korisnike, urls.py u toj aplikaciji bi sadržavao rute za prijavu, registraciju, postavke profila itd.

Način na koji se definiraju rute su prikazane idućim programskim kodom:

```
from django.urls import path
from .views import Readings, Index

urlpatterns = [
    path('readings/',Readings.as_view()),
]
```

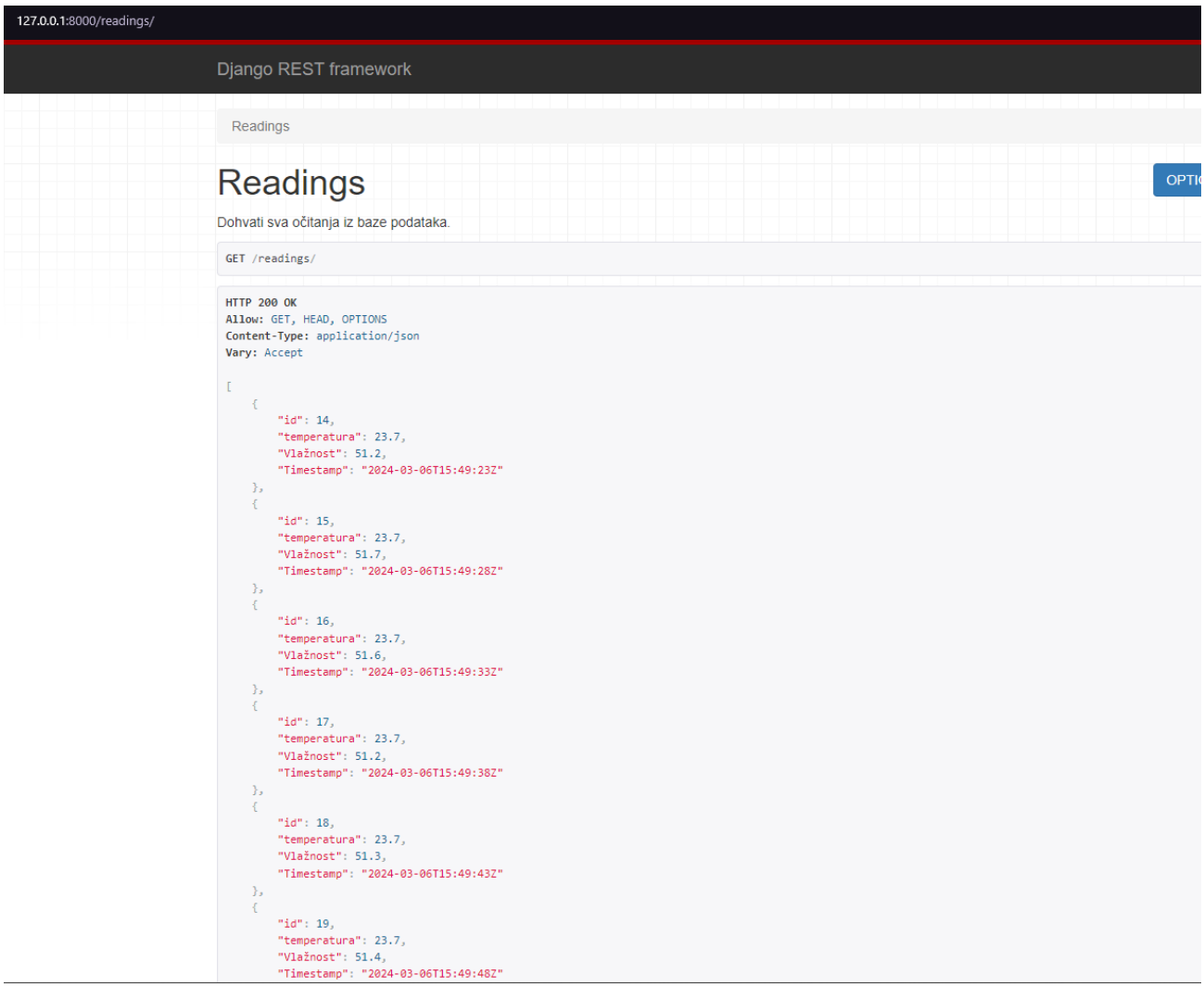
Pokretanjem web servera ova ruta neće raditi jer nismo inicijalizirali urls.py iz "MjerenjeApp" direktorija u projektnom "Mjerenje" direktoriju gdje se isto nalazi pod istim imenom urls.py datoteka. Način inicijalizacije je prikazan idućim programskim kodom:

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('',include('MjerenjeApp.urls')),
]
```

Pristupanju /readings ruti se vrši tako da se u web pregledniku u polju pretraživanja za "linkove" upiše:

- http://127.0.0.1:8000/readings/

Nakon pristupanja kao na slici 24 dobivamo web stranicu u kojoj će biti prikazan JSON format podataka mjerenja koji će kasnije biti poslani web stranici na kojoj ćemo prikazivati mjerenja preko grafova.



Slika 24. Prikaz JSON formata podataka mjerenja

Web stranice se sastoje od 2 strane. “Backend” i “Frontend” strane. Ovo što smo dosada napravili je dio “Backend”. “Backend” web stranice u Djangou predstavljaju serversku stranu aplikacije. To je dio aplikacije koji se nalazi na serveru i koji obrađuje zahtjeve korisnika, komunicira s bazom podataka te šalje odgovore natrag korisnicima. U “Backendu” se stvaraju “endpointi” koji kasnije šalju “Frontend” strani podatke koje trebamo.

10.Sučelje web stranice

Sučelje web stranice je ono što korisnik vidi kada pristupi web stranici. Razvoj tih sučelja se naziva “Frontend” web stranice.. To je sve ono što korisnik vidi i s čime može komunicirati na stranici, uključujući tekst, slike, forme za unos podataka, tipku za navigaciju i druge elemente.

Najjednostavnije web stranice se sastoje od 3 glavna elementa “Frontenda”:

- **HTML(Hypertext Markup Language)** - Osnovni jezik za strukturiranje sadržaja web stranica. HTML se koristi za definiranje strukture i organizacije elemenata na stranici, kao što su naslovi, paragrafi, slike, tabele i forme.
- **CSS(Cascading Style Sheets)** - CSS se koristi za stiliziranje HTML elemenata, određujući izgled i dizajn web stranice. To uključuje postavljanje boja, fontova, veličina i rasporeda elemenata, kao i animacije i prilagodbe za različite uređaje
- **JavaScript** - JavaScript je skriptni jezik koji se koristi za dodavanje interaktivnosti i dinamičkog ponašanja na web stranice. S pomoću JavaScript-a možete dodati funkcionalnosti poput validacije forme, animacija, interakcije s korisnikom i dinamičkog ažuriranja sadržaja bez potrebe za osvježavanjem stranice.

Za profesionalnije web stranice se koriste “frameworki” poput Reacta, Angulara, jQuery...

Za potrebe ovoga završnoga rada su dovoljni HTML,CSS i Javascript.

Prvo što je potrebno napraviti je novu mapu u “MjerenjeApp” i dati joj ime “templates”. U toj mapi će biti stvorena HTML datoteka u kojoj će biti HTML sučelje web stranice.

Naziv HTML datoteke mora biti index.html zbog toga što smo u views.py Python datoteci definirali koji HTML predložak će se prikazati pri ulasku u web stranicu.

Program za prikaz grafova temperature i vlažnosti zraka je prikazan idućim kodom:

```
<!DOCTYPE html>
<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">
    <title>Temperature and Humidity Reading</title>
    <!-- Include Chart.js library -->
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
</head>
<body>
    <h1>Temperature and Humidity Readings</h1>

    <!-- Container for the temperature line chart -->
    <canvas id="temperatureChart" height="50"></canvas>
    <!-- Container for the humidity line chart -->
    <canvas id="humidityChart" height="50"></canvas>

    <script>
        // Inicijaliziranje praznih skupova podataka za temperaturu i vlažnost
        const temperatureData = [];
        const humidityData = [];
        const timestampData = [];

        const temperatureCtx =
document.getElementById('temperatureChart').getContext('2d');
        const temperatureChart = new Chart(temperatureCtx, {
            type: 'line',
            data: {
                labels: [],
                datasets: [
                    {
                        label: 'Temperature',
                        borderColor: 'rgb(255, 99, 132)',
                        data: temperatureData,
                    },
                ],
            },
            options: {
                scales: {
                    x: {
                        type: 'linear',
                        position: 'bottom',
                    },
                    y: {
                        beginAtZero: true,
                    },
```

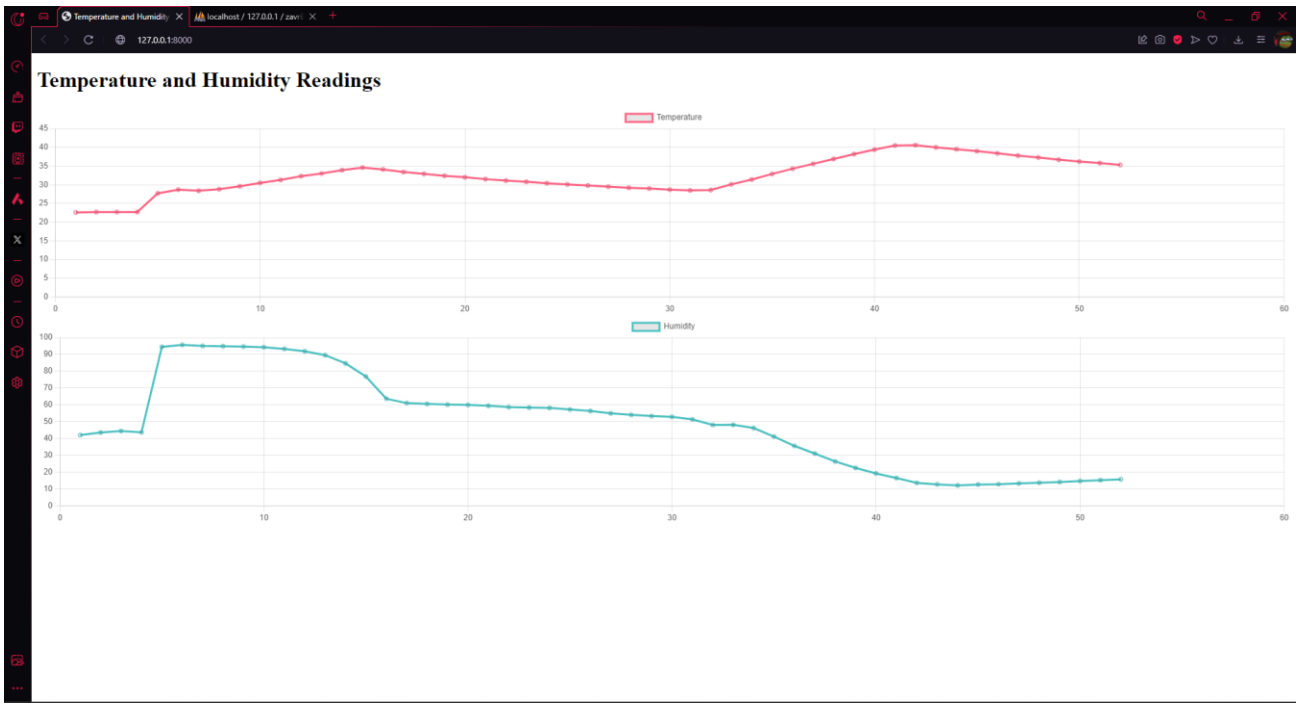
```
                },
                plugins: {
                    tooltip: {
                        callbacks: {
                            label: function (context) {
                                const labelIndex = context.dataIndex;
                                const timestamp = timestampData[labelIndex];
                                const temperature = temperatureData[labelIndex];
                                return `Temperature:
${temperature}°C\nTimestamp: ${timestamp}`;
                            },
                        },
                    },
                },
            },
        });

        const humidityCtx =
document.getElementById('humidityChart').getContext('2d');
        const humidityChart = new Chart(humidityCtx, {
            type: 'line',
            data: {
                labels: [],
                datasets: [
                    {
                        label: 'Humidity',
                        borderColor: 'rgb(75, 192, 192)',
                        data: humidityData,
                    },
                ],
            },
            options: {
                scales: {
                    x: {
                        type: 'linear',
                        position: 'bottom',
                    },
                    y: {
                        beginAtZero: true,
                    },
                },
                plugins: {
                    tooltip: {
                        callbacks: {
                            label: function (context) {
                                const labelIndex = context.dataIndex;
                                const timestamp = timestampData[labelIndex];
```

```
                                const humidity = humidityData[labelIndex];
                                return `Humidity: ${humidity}%\nTimestamp:
${timestamp}`;
                            },
                        },
                    },
                },
            },
        });
    // Poziv API-a iz backenda za dobitak mjerenja iz baze podataka
        function fetchReadings() {
            fetch('http://127.0.0.1:8000/readings/')
                .then(response => response.json())
                .then(data => {
                    updateReadings(data);
                })
                .catch(error => console.error('Error fetching data:', error));
        }

        function updateReadings(data) {
            const timestamps = [];
            temperatureData.length = 0;
            humidityData.length = 0;
            timestampData.length = 0;

            data.forEach(reading => {
                timestamps.push(reading.id);
                temperatureData.push(reading.temperatura);
                humidityData.push(reading.Vlažnost);
                timestampData.push(reading.Timestamp);
            });

            temperatureChart.data.labels = timestamps;
            humidityChart.data.labels = timestamps;

            temperatureChart.update(); // Ažurirajte grafikon temperature novim
podacima
            humidityChart.update(); // Ažurirajte grafikon vlažnosti novim
podacima
        }
        setInterval(fetchReadings, 5000);

        // Početno dohvaćanje
        fetchReadings();
    </script>
</body>
</html>
```

Zadnji korak u “Backendu” koji se mora postaviti je rutu u urls.py Python datoteci u “MjerenjeApp” mapi. To se mora napraviti tako da kada korisnik pristupi linku web stranice da se odmah sučelje generira. To se postiže idućim programskim kodom:

```
from django.urls import path
from .views import Readings, Index

urlpatterns = [
    path('',Index),
    path('readings/',Readings.as_view()),
]
```

Postavljajući senzor blizu radijatora i ostalim dijelovima kućanstva, nakon određenog vremena ovako izgleda web stranica s grafovima temperature i vlažnosti zraka kao na slici 25.



Slika 25. Prikaz mjerenja temperature i vlažnosti zraka preko grafova

Web stranica se sama osvježava svakih 5 sekundi kako bi prikupila najnovije podatke iz baze podataka i prikazivala ih u obliku grafova.

11.Zaključak

Zaključak ovog završnog rada predstavlja integraciju ESP32 mikrokontrolera i DHT11 senzora za precizno mjerenje temperature i vlažnosti zraka. Kroz proces implementacije, razvijen je sustav koji omogućuje kontinuirano praćenje i prikazivanje ovih mjerenja putem web sučelja. Korištenje ESP32 omogućuje pouzdano povezivanje s mrežom, dok je DHT11 senzor pokazao zadovoljavajuću točnost u mjerenju temperature i vlažnosti.

Kroz provedene testove, potvrđeno je da sustav pravilno prikazuje stvarne vrijednosti temperature i vlažnosti zraka. Prikazivanje ovih podataka putem grafova na web stranici omogućuje korisnicima intuitivno praćenje trendova i analizu promjena u okolini. Osim toga, implementacija web sučelja omogućuje udaljeni pristup i praćenje podataka putem interneta, što poboljšava praktičnost i korisnost sustava.

Ovaj rad predstavlja korak naprijed u korištenju “IoT” tehnologija za praćenje okoline. Budući razvoj ovog sustava mogao bi uključivati dodatne senzore ili proširene funkcionalnosti, kao što su upozorenja na ekstremne uvjete ili integracija s drugim “IoT” uređajima. Sveukupno, kombinacija ESP32 mikrokontrolera, DHT11 senzora i web sučelja predstavlja moćan alat za praćenje i analizu temperature i vlažnosti zraka u različitim okruženjima.

Literatura

[1] https://randomnerdtutorials.com/getting-started-with-esp32/

Datum pristupa: 14.3.2024

[2] https://www.elprocus.com/a-brief-on-dht11-sensor/

Datum pristupa: 14.3.2024

[3] https://en.wikipedia.org/wiki/Arduino

Datum pristupa: 16.3.2024

[4] https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/

Datum pristupa: 16.3.2024

[5] https://en.wikipedia.org/wiki/MySQL/

Datum pristupa: 18.3.2024

[6] https://en.wikipedia.org/wiki/XAMPP

Datum pristupa: 18.3.2024

[7] https://forum.arduino.cc/t/sending-data-from-arduino-to-mysql/684331

Datum pristupa: 21.3.2024

[8] https://en.wikipedia.org/wiki/Django_(web_framework)

Datum pristupa: 23.3.2024

Prilozi

1. CD-R