izv. prof. dr. sc. Alen Jakupović, prof. struč. stud.

# INTERNET STVARI

## - PRIRUČNIK -

Europska unija
Zajedno do fondova EU
EUROPSKI STRUKTURNI I INVESTICIJSKI FONDOVI
Strukovna škola Đurđevac

izv. prof. dr. sc. Alen Jakupović, prof. struč. stud.

# INTERNET STVARI

## - PRIRUČNIK -

RIJEKA, 2023.

izv. prof. dr. sc. Alen Jakupović, prof. struč. stud.

**INTERNET STVARI**

*Nakladnik*
Prospekt d.o.o., Dražice 138, Rijeka

*Za nakladnika*
Leonardo Matković

*Recenzenti*
izv. prof. dr. sc. Sanja Čandrlić
Emil Prpić, dipl. ing. el.

*Lektor*
Una Matić Vukelić, prof.

*Slog i prijelom*
Autor

*Dizajn ovitka*
Autor

*Mjesto izdanja*
Rijeka

*Godina izdanja*
2023.

*Elektroničko izdanje (URL)*
https://lms.cekom-kckzz.hr/e-izdanja/

Publikacija je izdana u okviru projekta „Razvoj kompetencija kroz učenje temeljeno na radu“ financiranog od strane Europske unije iz Europskog socijalnog fonda.



ISBN 978-953-50797-0-5

# SADRŽAJ

# 1. ODABRANA POGLAVLJA IZ PROGRAMSKOG JEZIKA C

U ovoj nastavnoj cjelini obrađuju se sljedeći koncepti koji se odnose na programiranje u programskom jeziku C: varijable, standardni ulaz/izlaz, upravljanje tijekom programa, funkcije, polja podataka i pokazivači. Koncepti su obrađeni preko niza primjera koji svi završavaju konkretnim radnim zadacima. Na kraju se nalaze dodatna pitanja i zadaci te popis literature i drugih izvora.
Cilj je ove nastavne jedinice osposobiti čitatelja za programiranje jednostavnijih programa koje će izvoditi mikrokontroleri.

## 1.1. Uspostava razvojnog okruženja za razvoj softvera

Cilj je uspostave razvojnog okruženja na razvojno računalo instalirati i konfigurirati cjelokupan softver koji je nužan za razvoj softvera. O tome koje značajke ima softver koji se razvija te kako se on razvija ovisi i što se sve treba instalirati i konfigurirati na razvojnom računalu (npr. softver razvija dislocirani tim, softver treba komunicirati s bazom podataka, softver je mobilna aplikacija itd.). Jedan od naših ciljeva jest razviti softversku komponentu sustava IoT. U tu svrhu treba nam softver u kojem ćemo pisati programski kôd (engl. IDE – *integrated development environment*). Budući da se u softveru za pisanje programskog koda mogu koristiti različiti programski jezici, bit će potrebno instalirati i dodatke koji će nam omogućiti programiranje u programskom jeziku C. I na kraju, budući da ćemo programirati sustav IoT koji se temelji na razvojnoj pločici ESP32-DevKitC-VIE, trebat će nam i posebna vrsta razvojnog alata koja se naziva razvojni okvir (engl. *framework*), a koji nam olakšava programiranje na način da koristimo već gotove programe koje uključujemo u svoj programski kôd koristeći aplikacijsko programsko sučelje (engl. API – *application programming interface*).

**Nakon usvajanja ove nastavne teme čitatelj će moći:**

1. Dati primjer programske potpore za razvoj vlastitih rješenja zasnovanih na internetu stvari ovisno o određenoj namjeni
2. Instalirati, konfigurirati odabrane programske potpore za razvoj vlastitih rješenja zasnovanih na internetu stvari
3. Proširiti osnovnu funkcionalnost programske potpore pomoću javno dostupnih biblioteka
4. Objasniti koje su prednosti i mane pojedinih dostupnih sklopovskih potpora ovisno o njihovoj namjeni
5. Programirati mikroupravljač koristeći odabranu programsku potporu

### 1.1.1. Opis radnog zadatka

Na razvojno računalo potrebno je instalirati i konfigurirati cjelokupan softver potreban za razvoj softverskog dijela sustava IoT. Prije instalacije potrebno je provjeriti zadovoljava li razvojno računalo odgovarajuće minimalne zahtjeve propisane softverom koji se instalira. Potrebno je instalirati:

1. softver za pisanje programskog koda Visual Studio Code
2. proširenja za Visual Studio Code koji će omogućiti programiranje u programskom jeziku C
3. razvojni okvir ESP-IDF unutar softvera Visual Studio Code koji će nam omogućiti programiranje sustava IoT koji se temelji temeljenog na razvojnoj pločici ESP32-DevKitC-VIE.

### 1.1.2. Osnovni koncepti

Često se za softver koji se instalira na računalo mogu pronaći podaci o minimalnim zahtjevima koje računalo treba zadovoljiti kako bi na njemu softver bio u normalnoj funkciji (engl. *system requirements*). Ti se zahtjevi odnose na hardversku i softversku komponentu računala kao što su: veličina potrebnog prostora za instalaciju softvera, veličina potrebne RAM memorije, brzina procesora, vrsta operacijskog sustava i sl. Budući da ćemo instalirati softver za pisanje programskog koda Visual Studio Code, bit će potrebno provjeriti koji su njegovi hardverski i softverski zahtjevi te ih usporediti s odgovarajućim hardverskim i softverskim karakteristikama razvojnog računala.

Za pisanje programskog koda potrebno je koristiti neki od softvera za pisanje teksta. Tu treba razlikovati ovu vrstu softvera od softvera za obradu teksta koji omogućuju formatiranje teksta (npr. podcrtavanje teksta, upotrebu različitih fontova i veličina slova i sl.). Stoga, softveri za obradu teksta (npr. Microsoft Word) nisu softveri u kojima se može pisati programski kod. Jedino su softveri za pisanje teksta (npr. Notepad, Notepad++) softveri u kojima je moguće pisanje programskog koda.

No postoji i posebna vrsta softvera čija je primarna namjena pisanje programskog koda, a ta se vrsta softvera zove integralno razvojno okruženje (engl. IDE – *integrated development environment*). Integralno razvojno okruženje, osim pisanja programskog koda, ima i niz dodatnih mogućnosti kojima se olakšava rad programera (npr. automatsko formatiranje programskog koda, označavanje mjesta sintaksne pogreške, primjena alata za pronalaženje pogrešaka itd.). Također, integralno razvojno okruženje omogućuje dodavanje proširenja kojima se podržava rad programera (npr. dodavanje proširenja za neki programski jezik, dodavanje kompajlera, dodavanje razvojnog okvira, povezivanje s nekim vanjskim uslugama, npr. Git i sl.). Neki od najčešće korištenih besplatnih integralnih razvojnih okvira jesu Visual Studio Code, NetBeans i Eclipse.

Budući da ćemo programirati u programskom jeziku C koristeći integralno razvojno okruženje Visual Studio Code, potrebno je u njega instalirati i dva proširenja. Jedno je proširenje koje nam omogućuje podršku za pisanje programskog koda u programskom jeziku C (npr. formatiranje programskog koda, automatsko završavanje linija programskog koda, upotrebu alata za otklanjanje programskih pogrešaka i sl.). Drugo proširenje čini kompajler programskog jezika C. Naime, da bi računalo moglo izvesti programski kôd napisan u programskom jeziku C (on se još naziva i izvorni programski kod), taj programski kôd treba pretvoriti u poseban oblik koji se zove izvršni programski kod. Tu pretvorbu izvornoga programskog koda provodi posebna vrsta softvera koja se naziva kompajler. Važno je naglasiti da programeri rade isključivo s izvornim programskim kodom, a ne s izvršnim.

Softver koji ćemo razvijati upravljat će hardverskim dijelovima sustava IoT (npr. senzorom za temperaturu i vlagu), ali i pohranom podataka u lokalnu memoriju te slanjem podataka na udaljenu lokaciju (npr. spremanje podataka na vanjsku memoriju, spremanje podataka na udaljenu bazu podataka i sl.). Kako bi se olakšao razvoj takvog softvera (ali i softvera nekih drugih vrsta, npr. *web*, mobilnih i sl.), programerima je dostupan dodatan razvojni alat koji se zove razvojni okvir (engl. *framework*) u sklopu kojega već postoje gotovi programi koji se koriste unutar softvera koji se razvija. Ti programi već rješavaju određene probleme koje bi inače programeri trebali rješavati (npr. kako očitati temperaturu senzora, spremiti podatke na lokalnu memoriju ili poslati podatke na udaljenu bazu podataka). Programe koji su dio nekog razvojnog okvira programeri dodaju u svoj programski kôd tako da koriste aplikacijsko programsko sučelje tog programa, tj. API. Budući da ćemo za razvoj sustava IoT koristiti razvojnu pločicu ESP32-DevKitC-VIE, bit će potrebno instalirati odgovarajući razvojni okvir koji se zove ESP-IDF (engl. *Espressif IoT Development Framework*).

### 1.1.3. Rješenje radnog zadatka

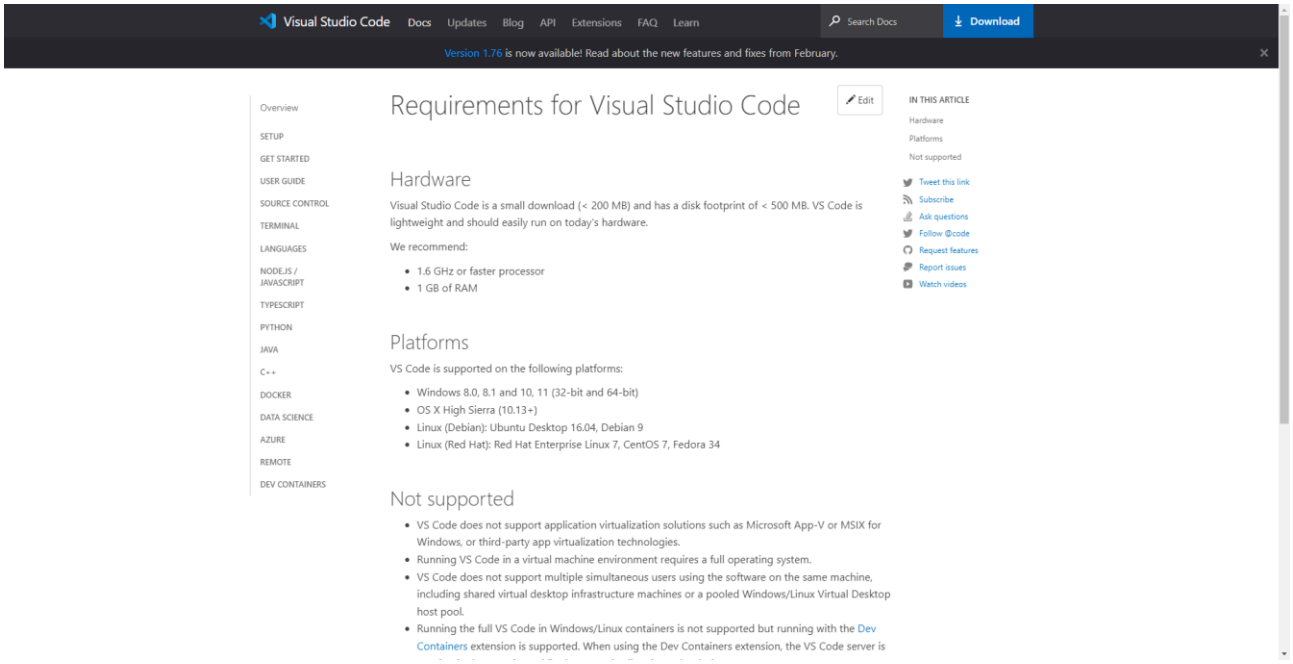
Uspostavu razvojnog okruženja za razvoj softvera čine četiri glavna dijela:

1. provjera zadovoljava li razvojno računalo hardverske i softverske zahtjeve koje postavlja integralno razvojno okruženje Visual Studio Code

2. preuzimanje i instalacija integralnog razvojnog okruženja Visual Studio Code
3. preuzimanje i postavljanje proširenja za Visual Studio Code radi pisanja programskog koda u programskom jeziku C
4. preuzimanje i postavljanje razvojnog okruženja ESP-IDF u Visual Studio Code radi pisanja programskog koda za razvojnu pločicu ESP32-DevKitC-VIE.

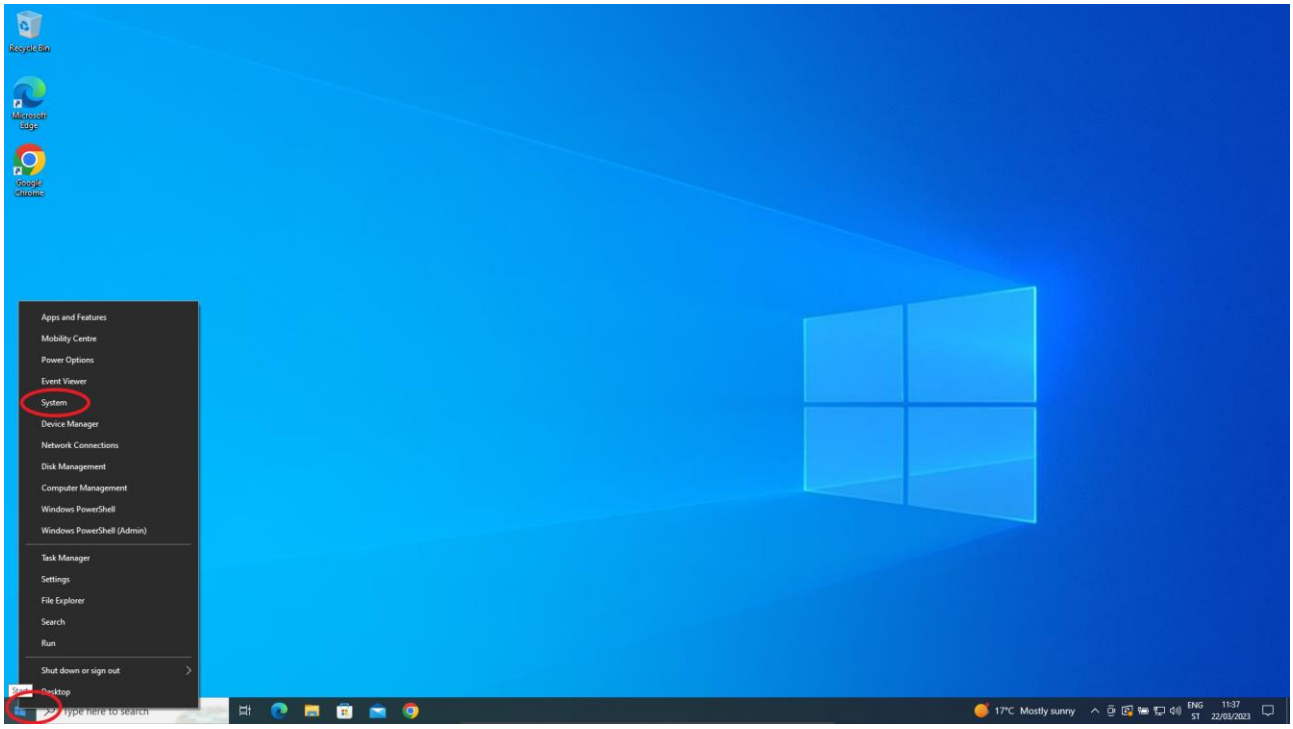
#### 1.1.3.1. Provjera hardverskih i softverskih zahtjeva

Na poveznici https://code.visualstudio.com/docs/supporting/requirements mogu se pronaći hardverski i softverski zahtjevi za instalaciju integralnog razvojnog okruženja Visual Studio Code.



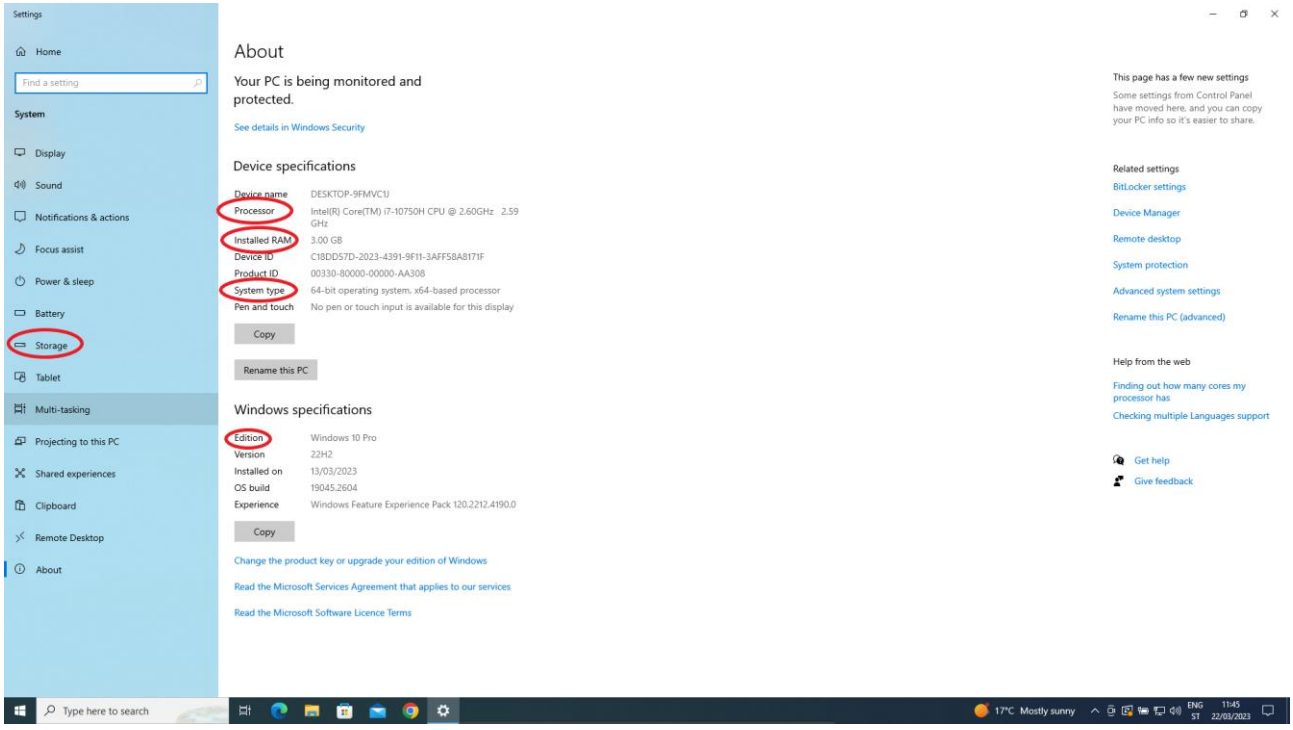

*Slika 1.1.1 Hardverski i softverski zahtjevi za instalaciju softvera Visual Studio Code*

Vidimo da je potrebno 500 MB prostora na disku radi instalacije, da je preporučena brzina procesora 1,6 GHz ili veća te da je potrebno barem 1 GB RAM-a. Također su prikazani i operacijski sustavi za koje postoji mogućnost instalacije. Za nas je važno da postoji podrška za operacijski sustav Windows 10 (32-bitni ili 64-bitni).
Softverskim i hardverskim značajkama razvojnog računala može se pristupiti na više načina. Jedan od njih jest da se s desnom tipkom miša klikne na izbornik Start čime se otvara prozor koji prikazuje niz izbornika. S popisa izbornika zatim treba kliknuti na izbornik System.



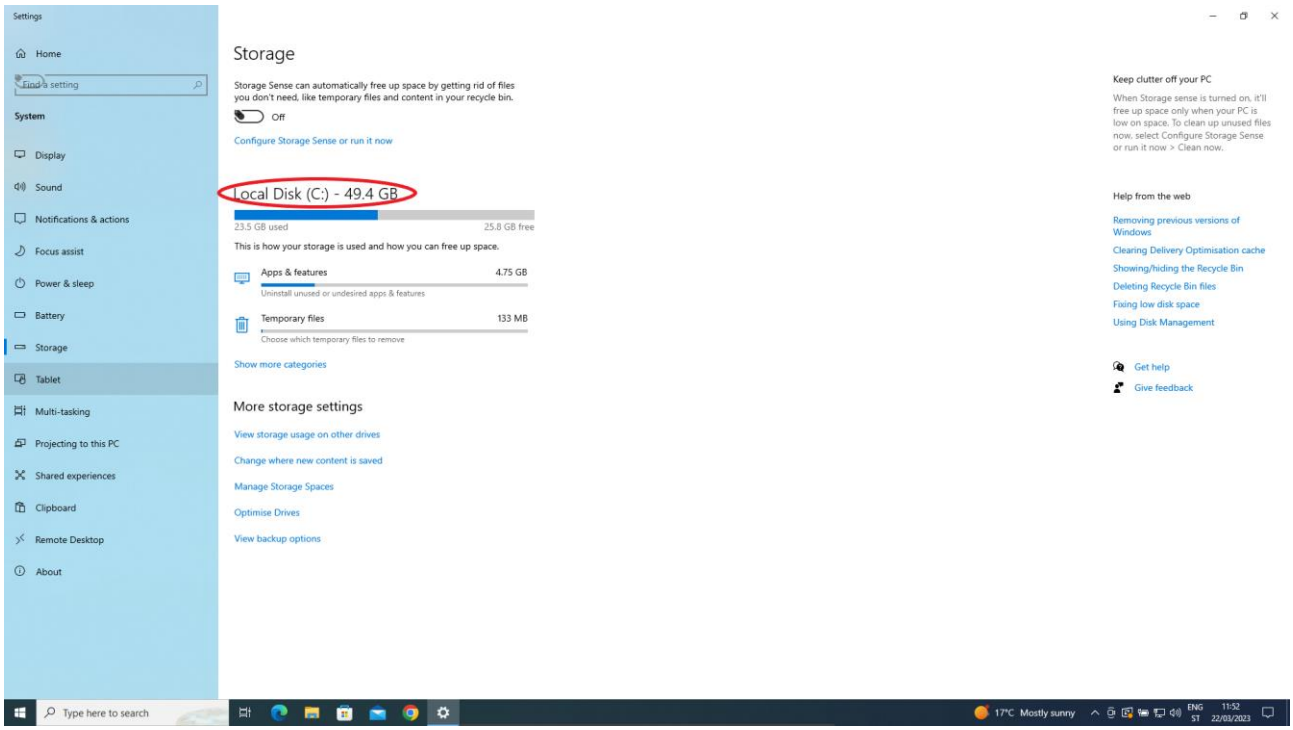

*Slika 1.1.2 Dolazak do izbornika System*

Klikom na izbornik System otvara se prozor s prikazom niza softverskih i hardverskih značajki računala. Može se vidjeti da razvojno računalo ima brzinu procesora od 2,6 GHz, da je veličina RAM-a 3 GB te da je operacijski sustav Windows 10 64-bitni. Da bi se provjerila veličina diska, potrebno je kliknuti na izbornik Storage.




*Slika 1.1.3 Prikaz brzine procesora, veličine RAM memorije, vrste operacijskog sustava i dolaska do informacija o veličini diska*

Klikom na izbornik Storage otvara se prozor u kojem se mogu provjeriti veličina diska i dostupan slobodan prostor. Vidljivo je da je ukupna veličina diska 49,4 GB i da je dostupno 25,8 GB slobodnog prostora.




*Slika 1.1.4 Prikaz veličine diska i dostupan slobodan prostor*

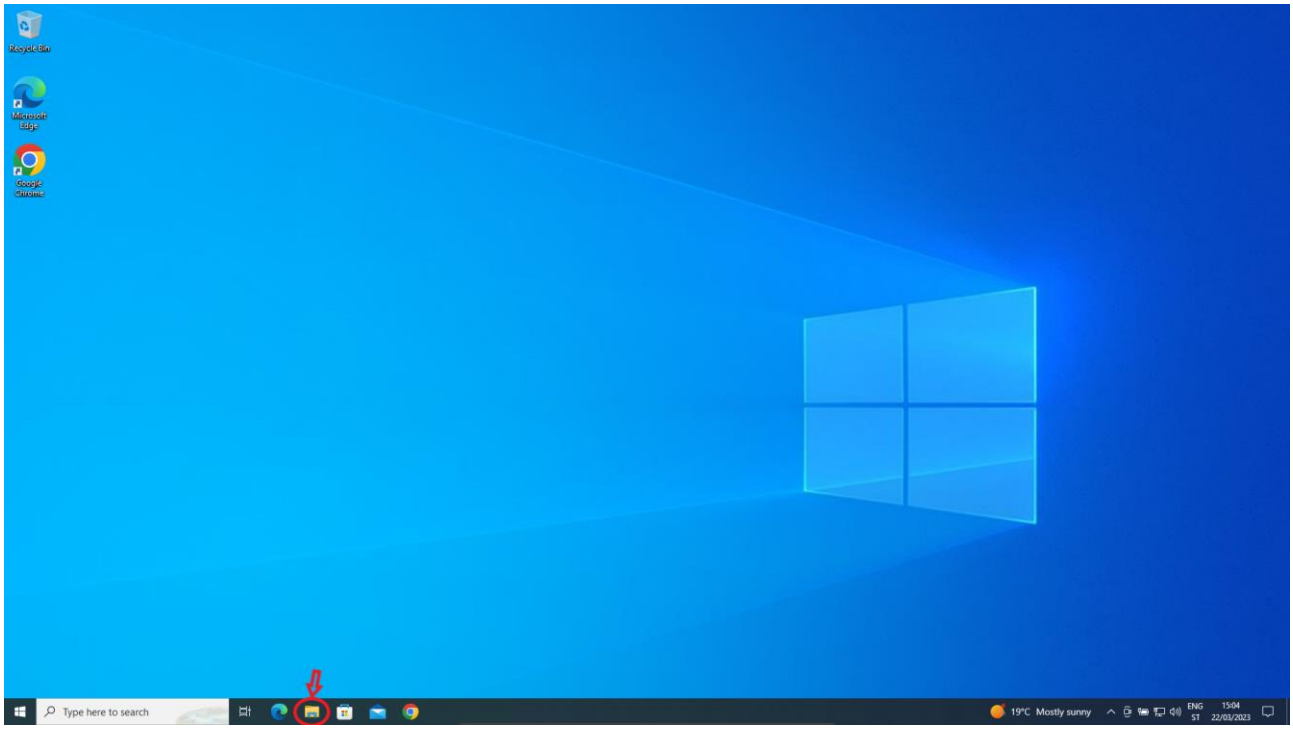
Iz svega navedenoga može se zaključiti da razvojno računalo zadovoljava minimalne zahtjeve koje postavlja Visual Studio Code te se time on može instalirati na računalo.

### 1.1.3.2. Preuzimanje i instalacija softvera Visual Studio Code

Visual Studio Code besplatno je integralno razvojno okruženje koje se može preuzeti preko poveznice https://code.visualstudio.com/Download. Izabrat ćemo verziju x64 jer nam računalo posjeduje procesor koji se temelji na x64 i operacijski je sustav 64-bitni.

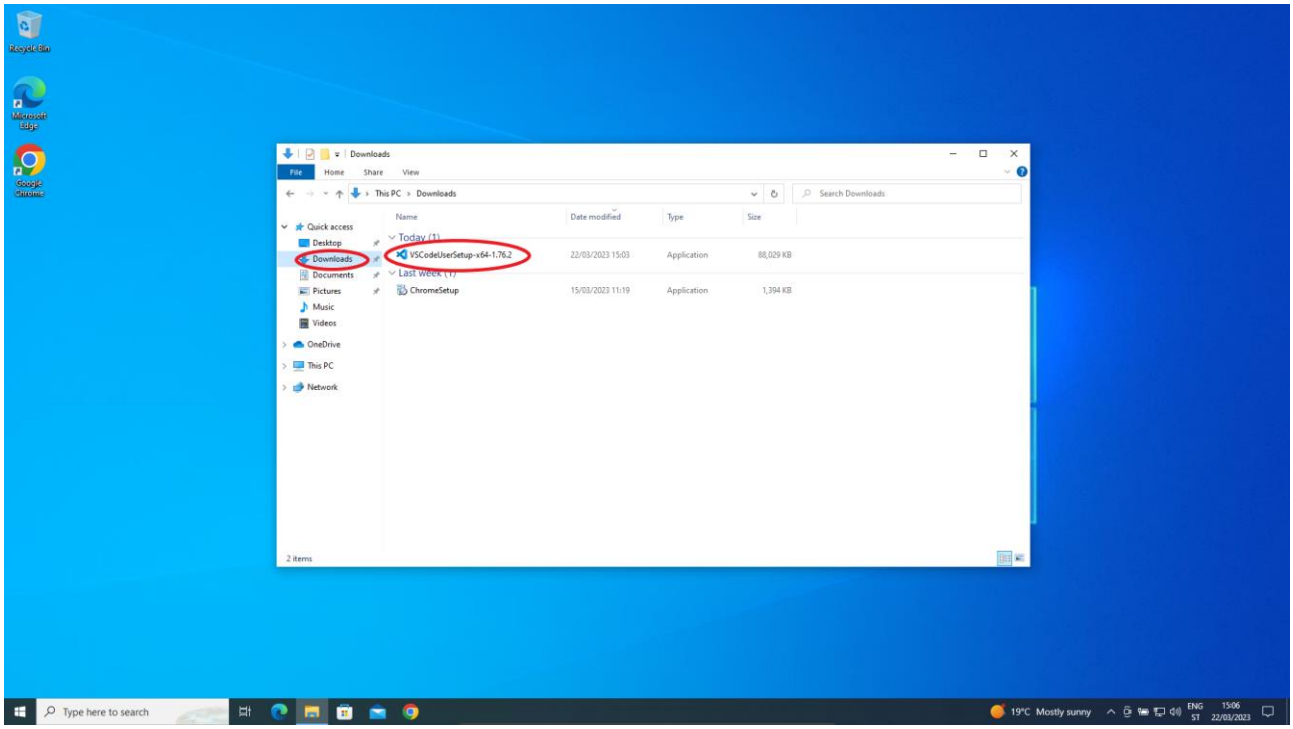

*Slika 1.1.5 Preuzimanje instalacije Visual Studio Code-a*

Nakon završetka preuzimanja instalacijska datoteka nalazi se u mapi Downloads do koje se dolazi preko preglednika datoteka (engl. *File Explorer*).



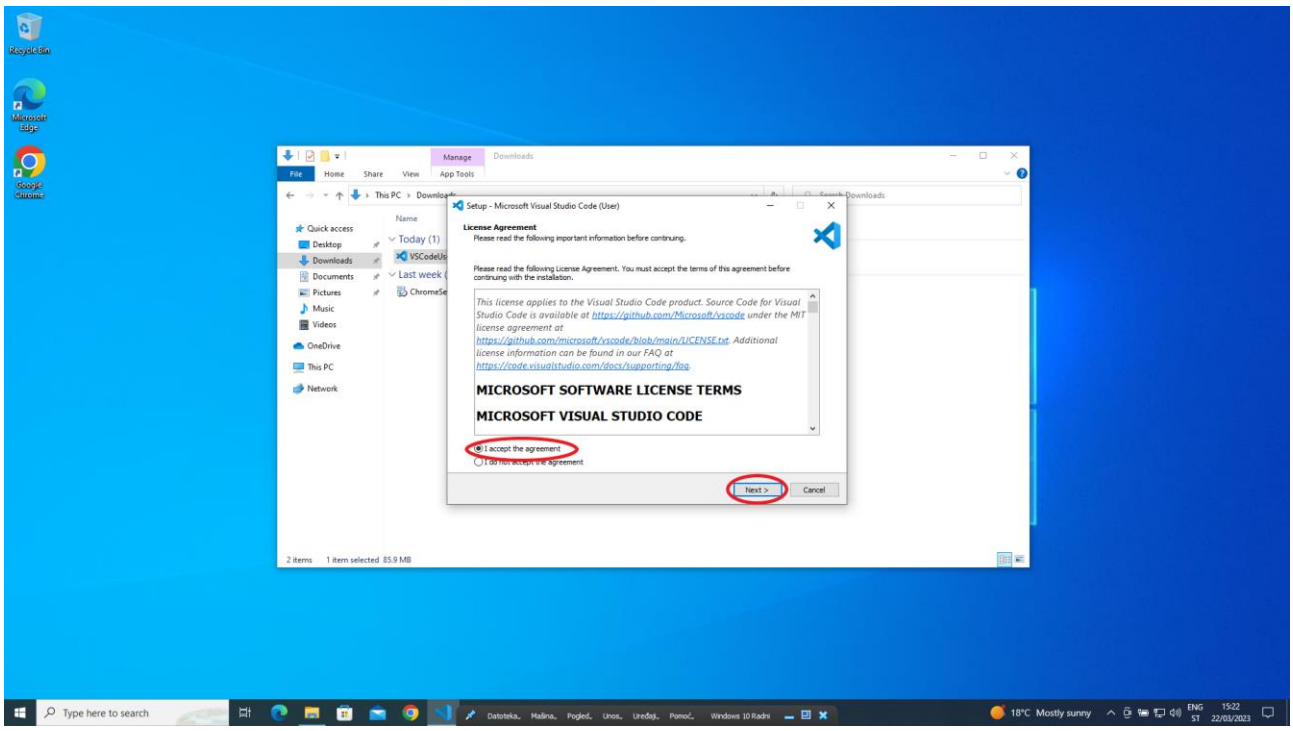

*Slika 1.1.6 Otvaranje preglednika datoteka*

Nakon klika na ikonu za preglednik datoteka otvara se prozor unutar kojega je potrebno odabrati mapu Downloads. Nakon njezina odabira prikazat će se sadržaj mape unutar kojega bi se trebala nalaziti i datoteka naziva VSCodeUserSetup-x64-X.Y.Z gdje X, Y i Z predstavljaju verziju programa.



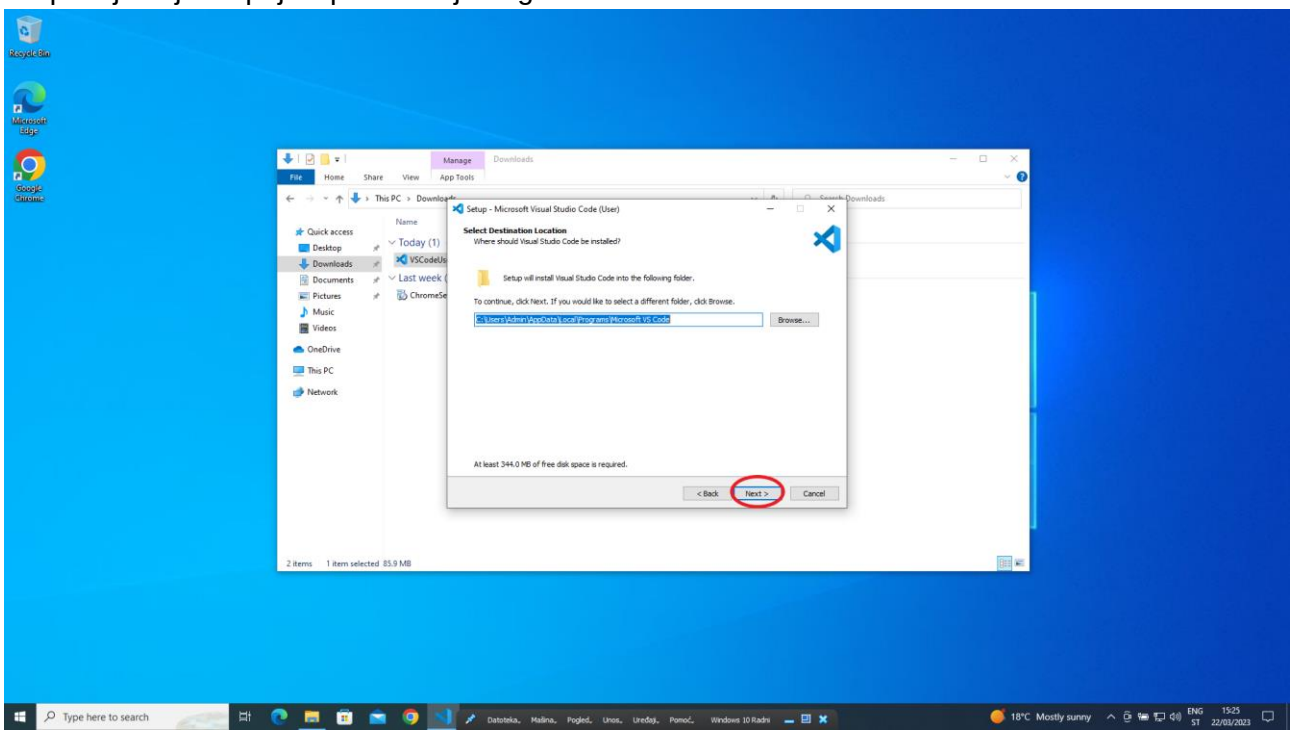

*Slika 1.1.7 Pogled na mapu Downloads*

Kako bi se instalirao softver Visual Studio Code, potrebno je pokrenuti preuzetu datoteku tako da se na nju napravi dvostruki klik lijevom tipkom miša. Pojavljuje se instalacijski čarobnjak (engl. *setup wizard*). Na prvoj stranici trebamo prihvatiti uvjete i kliknuti na Next.



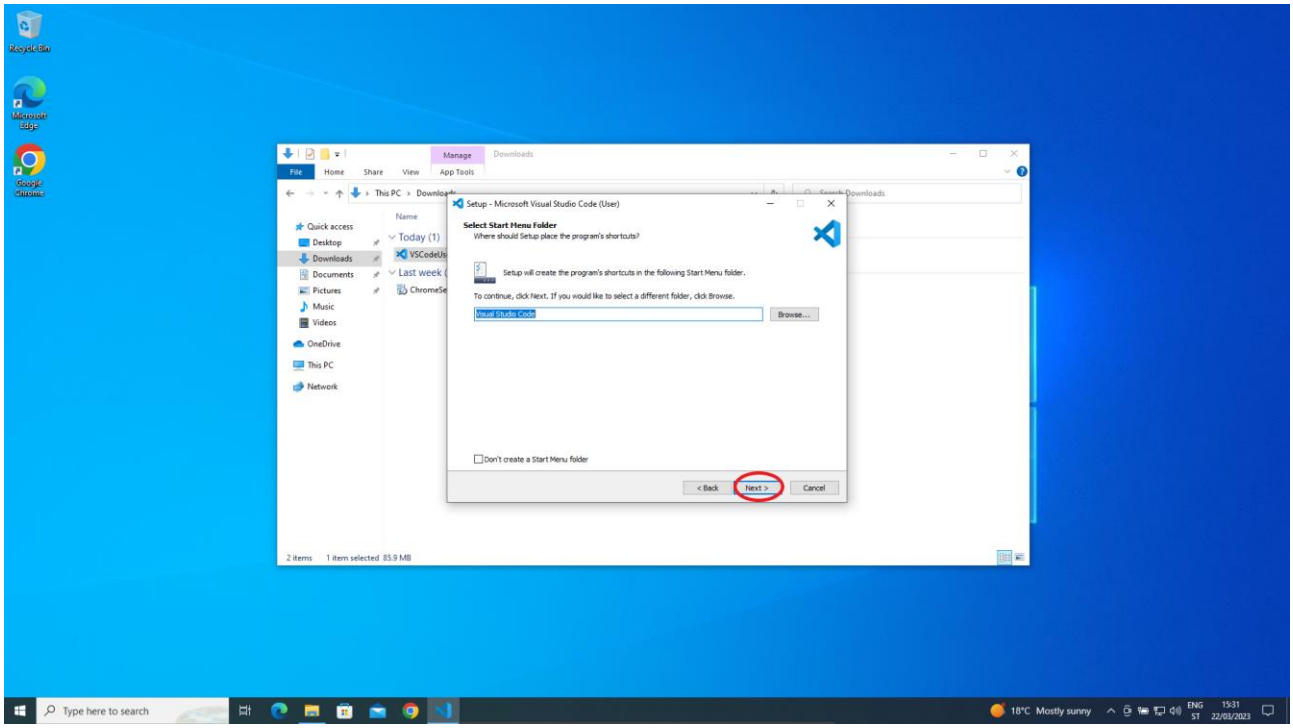

*Slika 1.1.8 Prihvaćanje uvjeta za instalaciju Visual Studio Code-a*

Druga stranica instalacijskog čarobnjaka prikazuje mapu (putanju) u koju će se obaviti instalacija programa. Ostavit ćemo mapu koja je već navedena i kliknuti na Next.
Na pitanje koje se pojavi potrebno je odgovoriti s Yes.



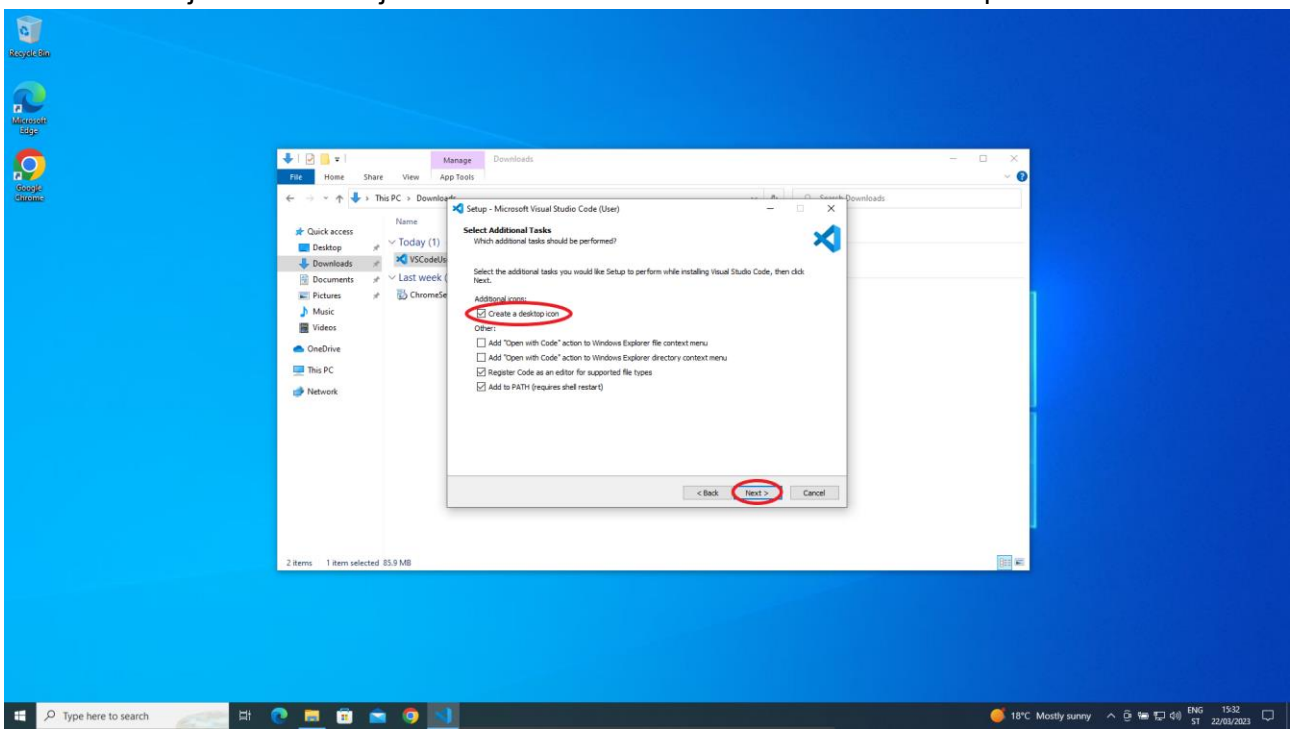

*Slika 1.1.9 Izbor mape u kojoj će se instalirati Visual Studio Code*

Treća stranica instalacijskog čarobnjaka prikazuje mapu u kojoj će se unutar izbornika Start stvoriti prečac do programa. Ostavit ćemo mapu koja je već navedena i kliknuti na Next.



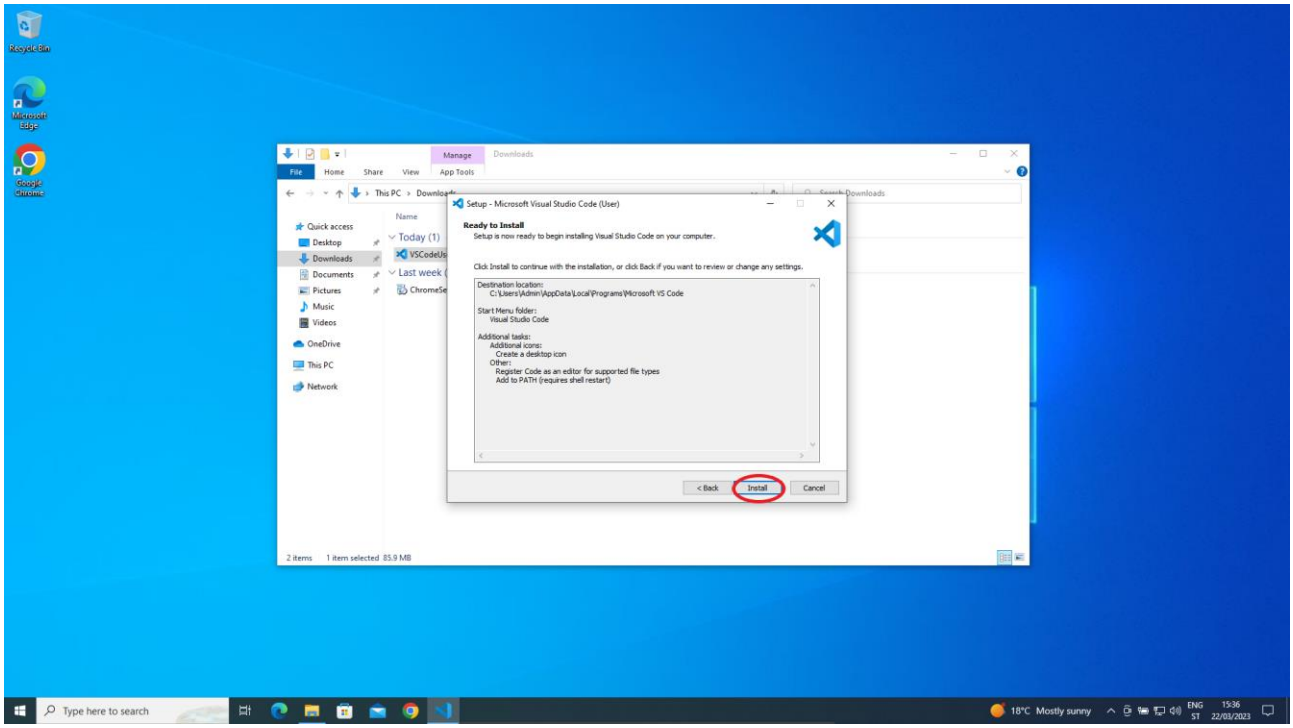

*Slika 1.1.10 Izbor mape za stvaranje prečaca do programa*

Četvrta stranica instalacijskog čarobnjaka omogućuje definiranje još nekih dodatnih aktivnosti koje će se obaviti tijekom instalacije. Tu ćemo odabrati aktivnost Create a desktop icon i kliknuti na Next.



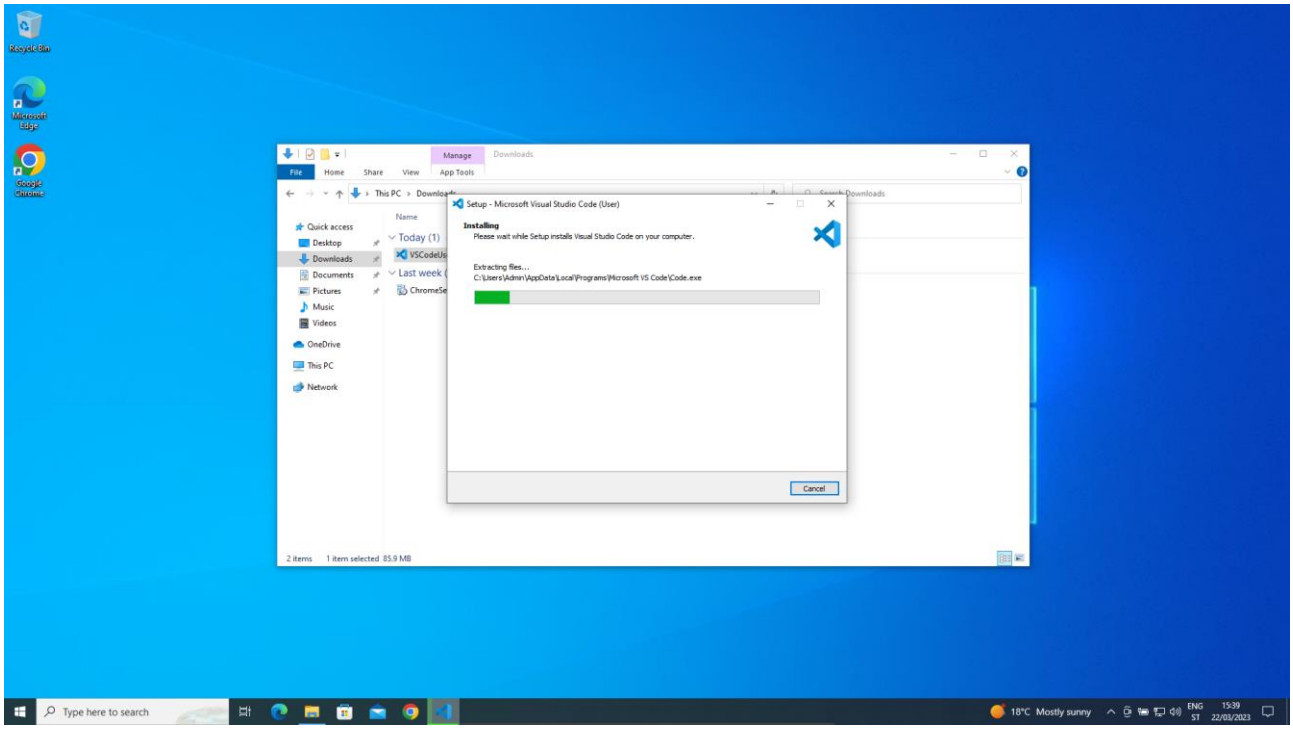

*Slika 1.1.11 Izbor dodatnih aktivnosti u instalacijskom procesu*

Zadnja stranica instalacijskog čarobnjaka prikazuje sažetak postavki instalacijskog procesa. Ako smo njime zadovoljni, možemo kliknuti na Install. Inače se možemo vratiti na prethodne stranice klikom na Back.



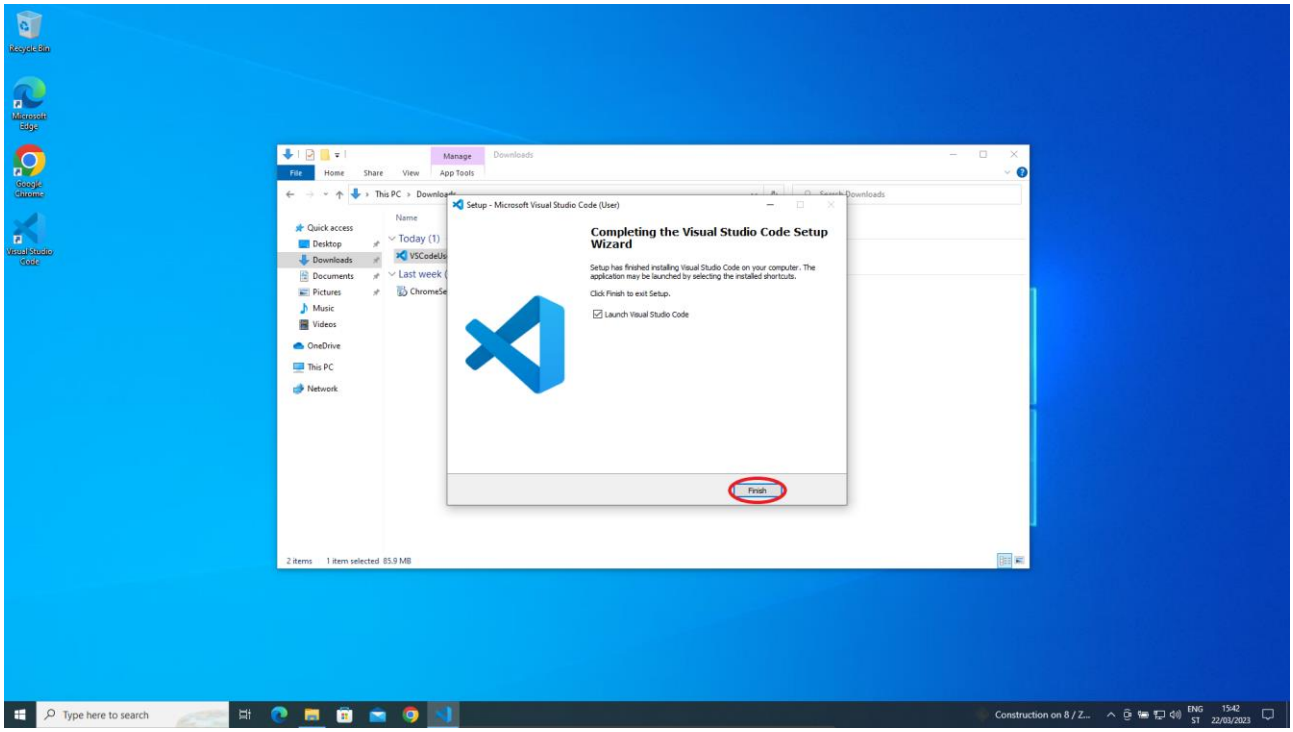

*Slika 1.1.12 Pokretanje instalacijskog procesa*

Nakon pokretanja instalacijskog procesa prikazuje se prozor s progresom instalacije.



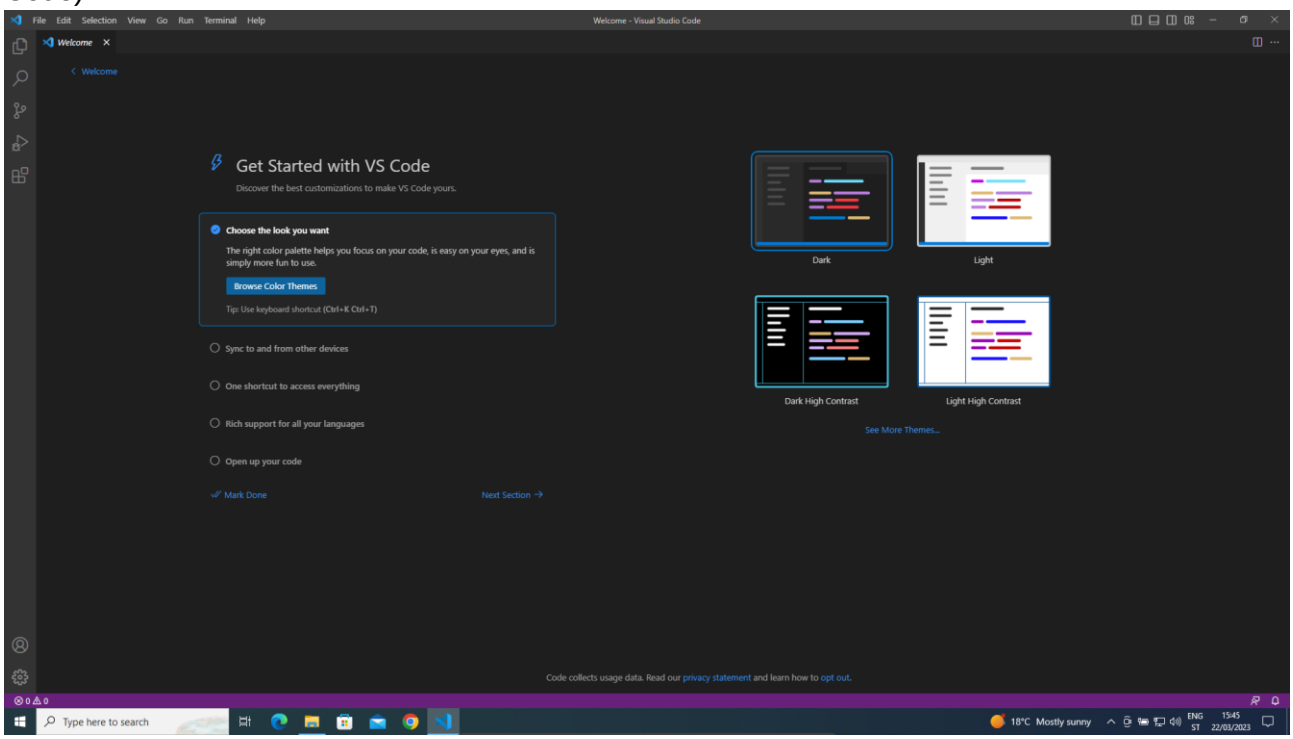

*Slika 1.1.13 Napredak instalacijskog procesa*

Završetkom instalacijskog procesa pojavljuje se poruka o uspješno završenoj instalaciji.




*Slika 1.1.14 Poruka o uspješno završenom procesu instalacije*

Klikom na Finish pokreće se Visual Studio Code (provjerite je li uključena opcija Launch Visual Studio Code).



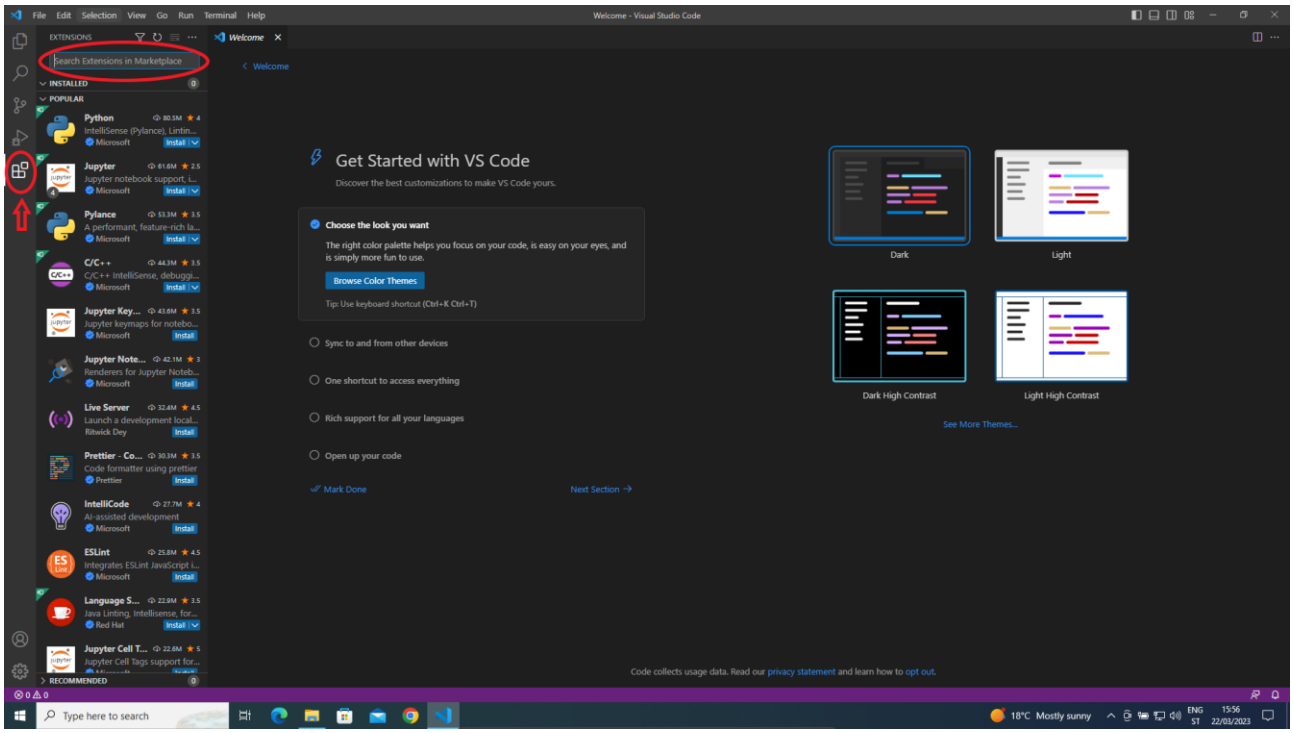

*Slika 1.1.15 Prvo pokretanje softvera Visual Studio Code*

Ovime je postupak preuzimanja i instalacije softvera Visual Studio Code završena.

### 1.1.3.3. Preuzimanje i postavljanje proširenja za programski jezik C

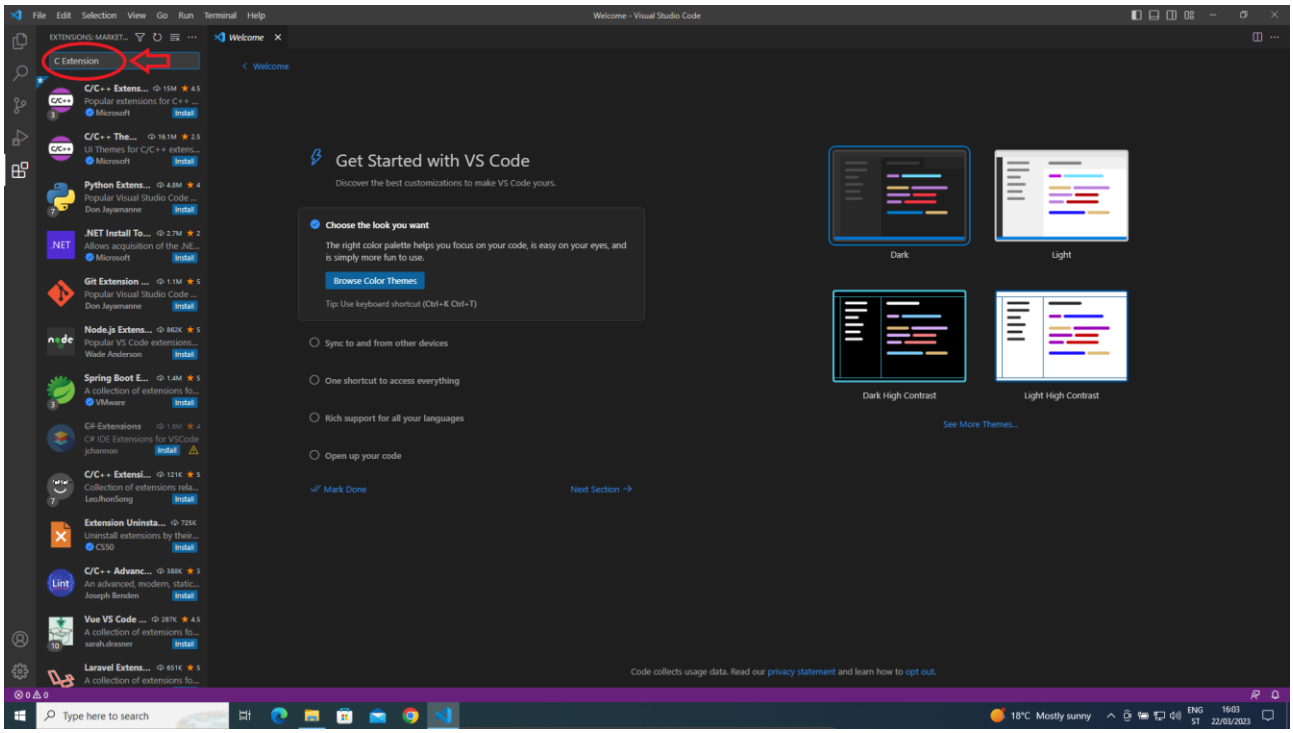
Budući da se u softveru Visual Studio Code može pisati programski kôd u različitim programskim jezicima, mi ga trebamo pripremiti za programski jezik C. To znači da trebamo preuzeti i instalirati dva proširenja: jedno koje nam olakšava pisanje programskog koda u programskom jeziku C i drugo

koji nam omogućuje pokretanje napisanoga programskog koda u programskom jeziku C. Do proširenja za Visual Studio Code dolazi se preko izbornika Extensions. Za pretraživanje proširenja koristi se polje Search Extensions in Marketplace.



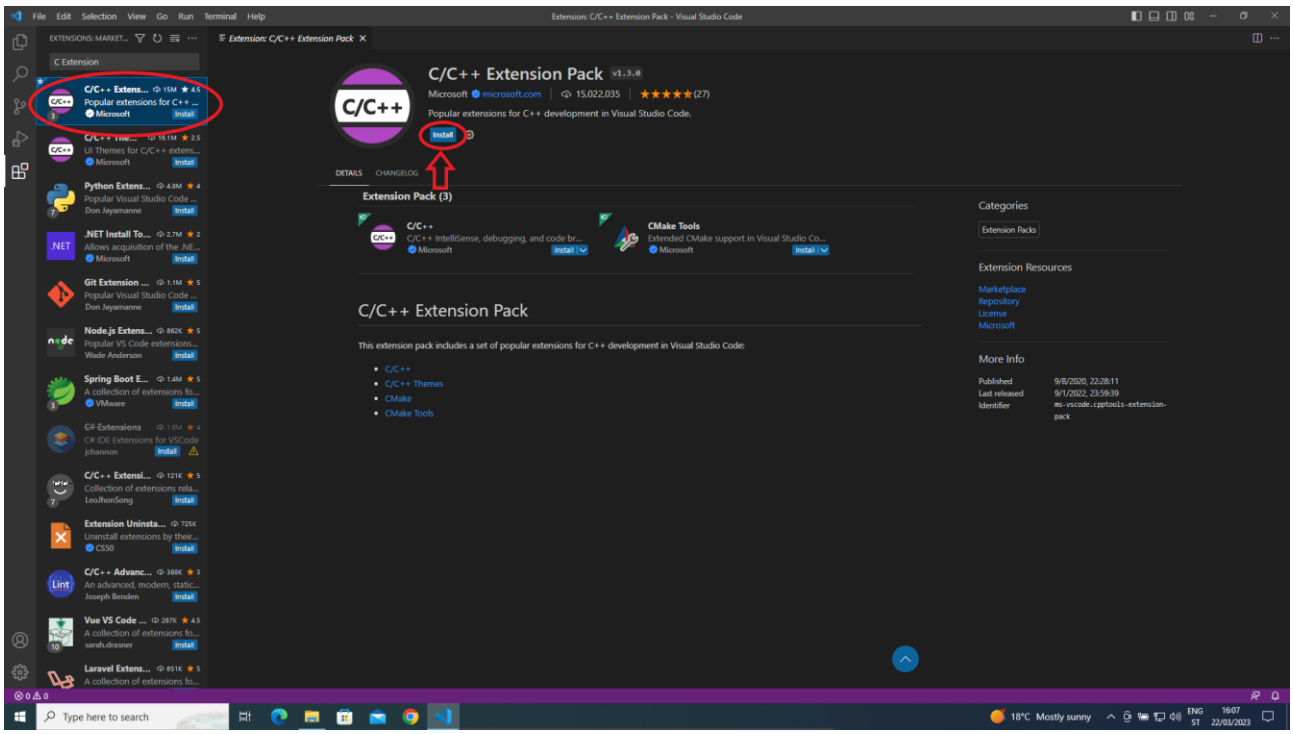

*Slika 1.1.16 Pokretanje prozora s proširenjima*

Prvo ćemo potražiti proširenje za pisanje programskog koda u programskom jeziku C upisom C Extension u polje za pretraživanje. Kao rezultat dobit ćemo proširenja koja se odnose na programski jezik C.



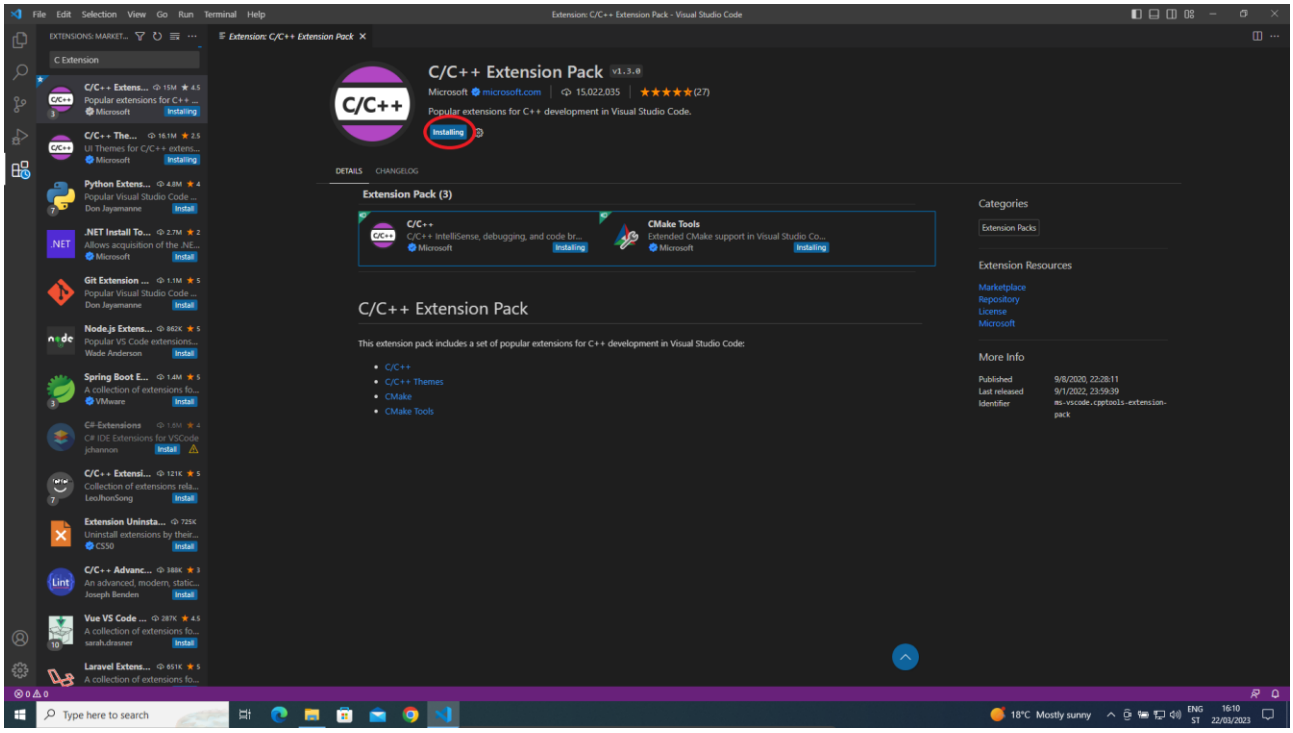

*Slika 1.1.17 Pretraživanje proširenja za programski jezik C*

Iz popisa proširenja koja se odnose na programski jezik C treba kliknuti na proširenje C/C++ Extension i nakon toga na Install.



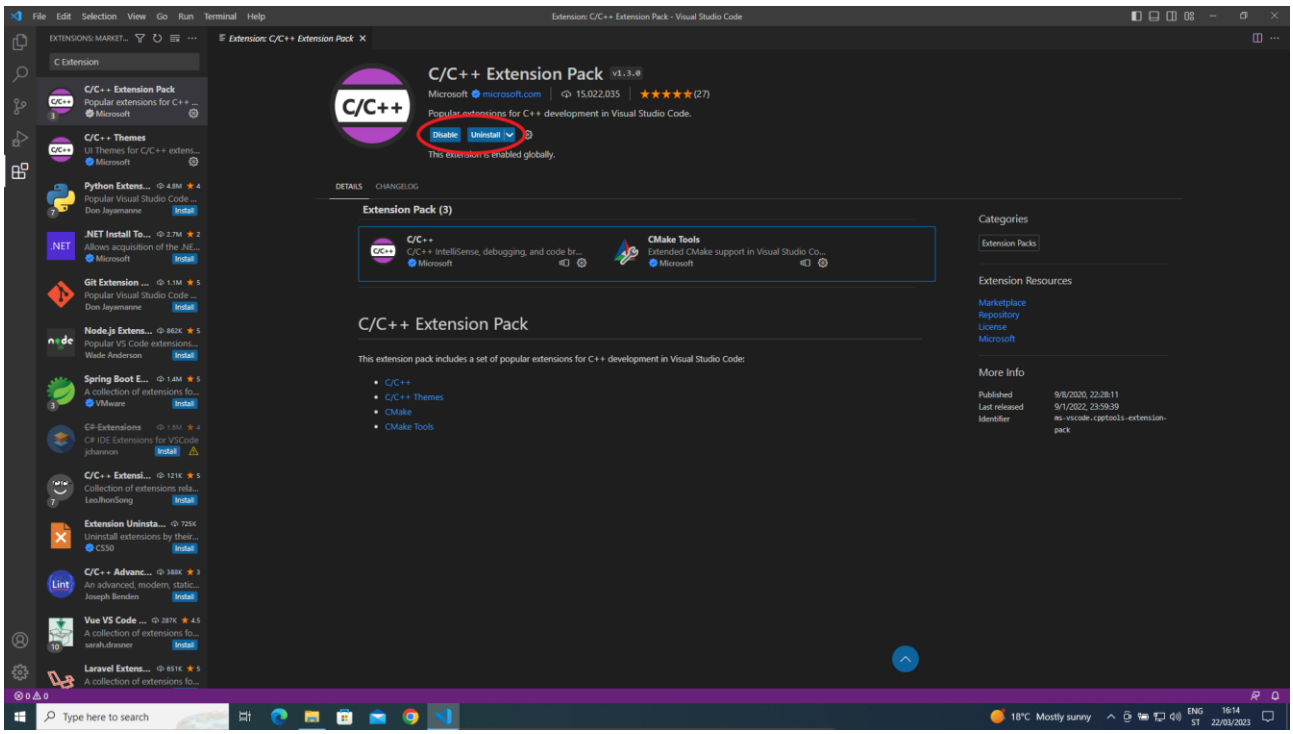

*Slika 1.1.18 Instalacija proširenja za programski jezik C*

Pokretanjem instalacije proširenja pokreće se proces kako je to prikazano na slici.



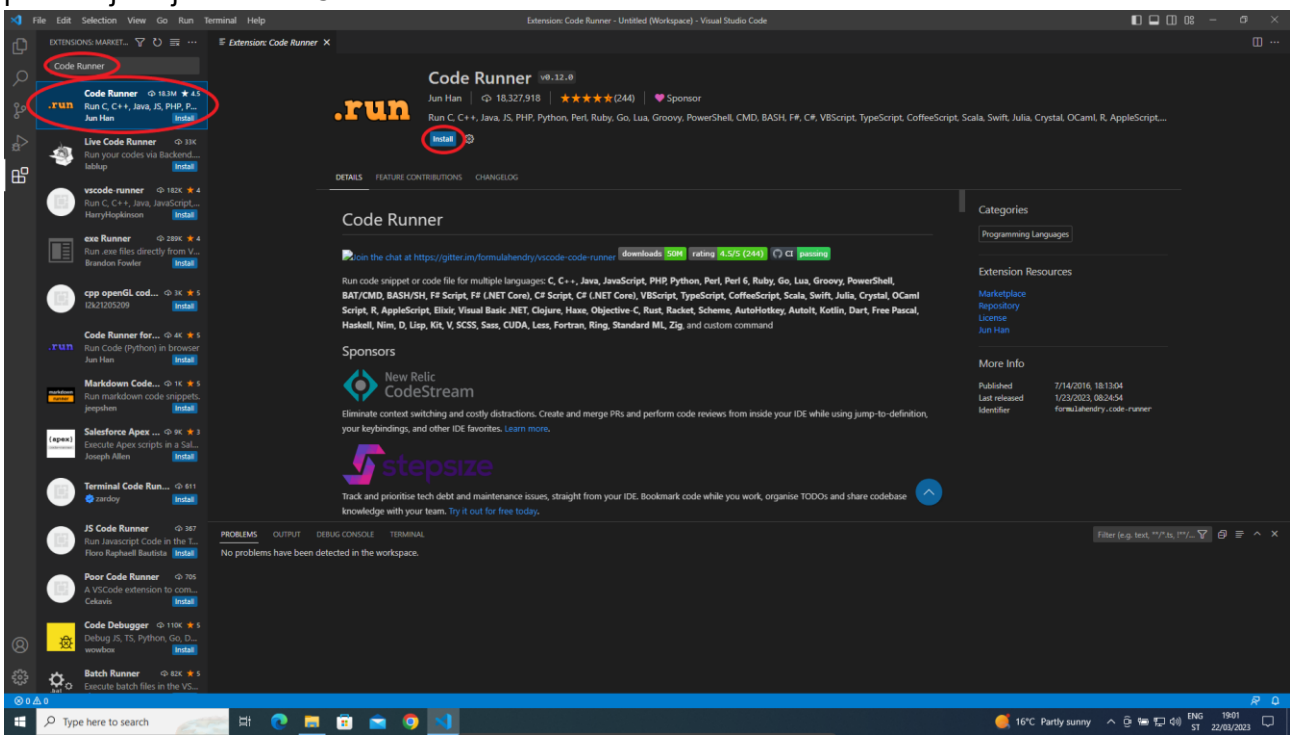

*Slika 1.1.19 Pokrenuti proces instalacije proširenja za programski jezik C*

Završetkom instalacije proširenja pojavljuju se dva nova gumba kojima se mijenja status proširenja u Onemogućeno ili se pokreće deinstalacija proširenja.



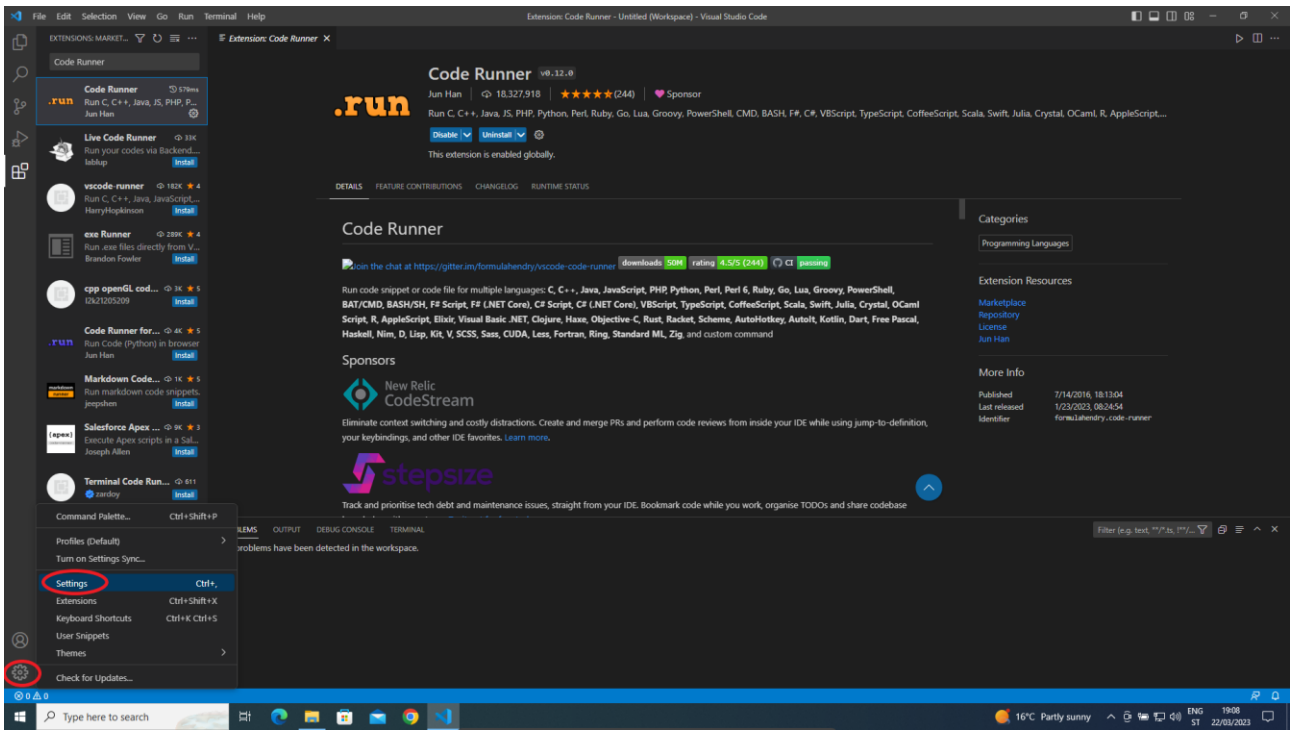

*Slika 1.1.20 Završetak procesa instalacije proširenja za programski jezik C*

Za pokretanje programskog koda pisanog u programskom jeziku C trebat će instalirati još jedno proširenje koje se zove Code Runner.



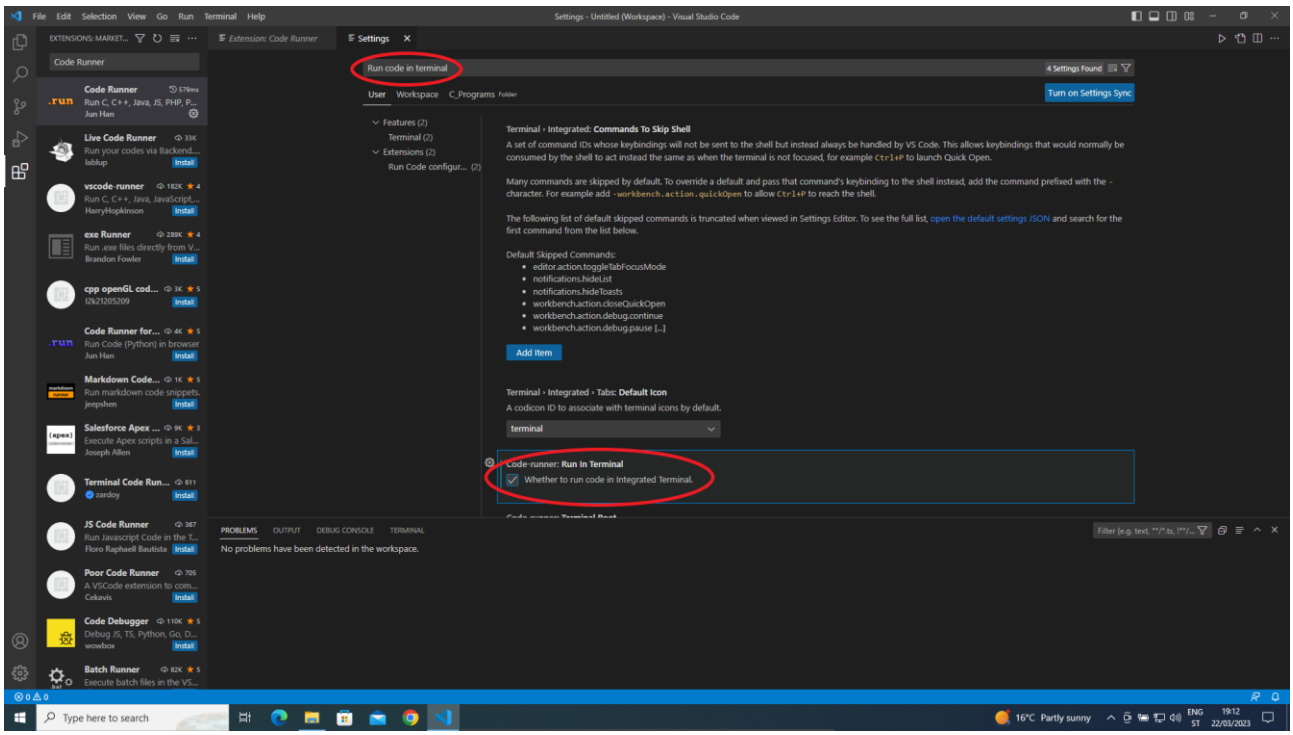

*Slika 1.1.21 Instalacija proširenja Code Runner*

Završetkom instalacije proširenja potrebno je u softveru Visual Studio Code navesti da će se programski kôd pokretati u terminalu računala.



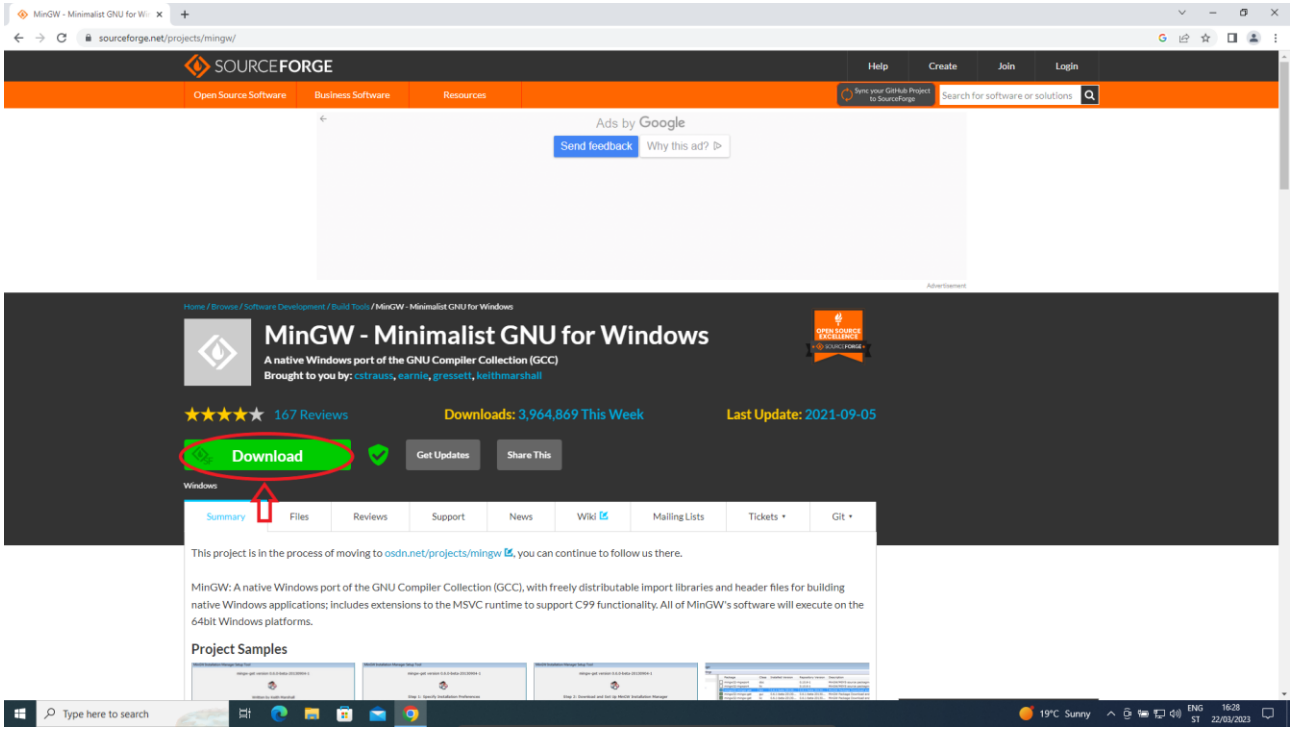

*Slika 1.1.22 Pokretanje postavljanja postavki u Visual Studio Code*

Nakon otvaranja prozora s postavkama potrebno je pronaći postavku tako da se u pretraživač upiše Run code in terminal. Označimo Code-runner: Run In Terminal.



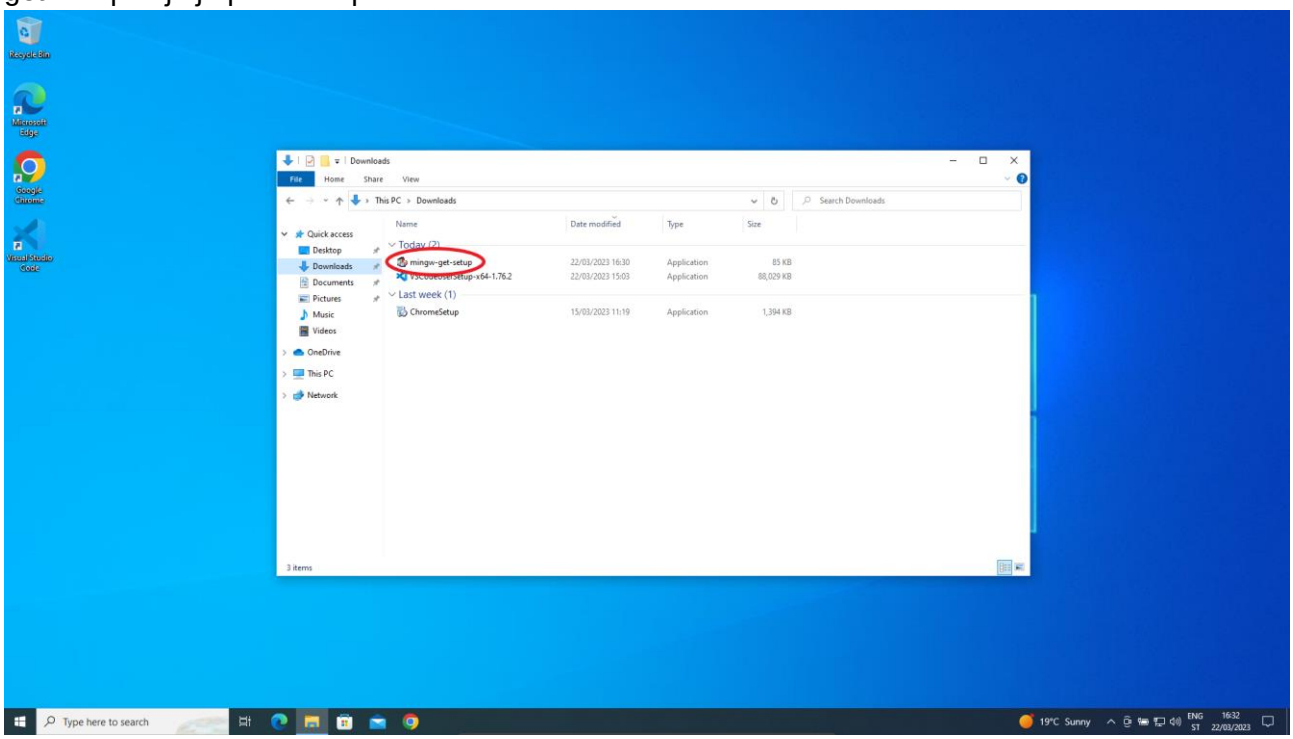

*Slika 1.1.23 Definiranje postavke za pokretanje programskog koda u terminalu*

Nakon instalacije proširenja za pisanje programskog koda u programskom jeziku C i definiranja dodatni postavki potrebno je preuzeti i instalirati kompajler koji će prevesti izvorni programski kôd u izvršni i tako omogućiti da ga računalo izvede. Preuzimanje kompajlera radimo preko poveznice https://sourceforge.net/projects/mingw/ na kojoj kliknemo na Download.



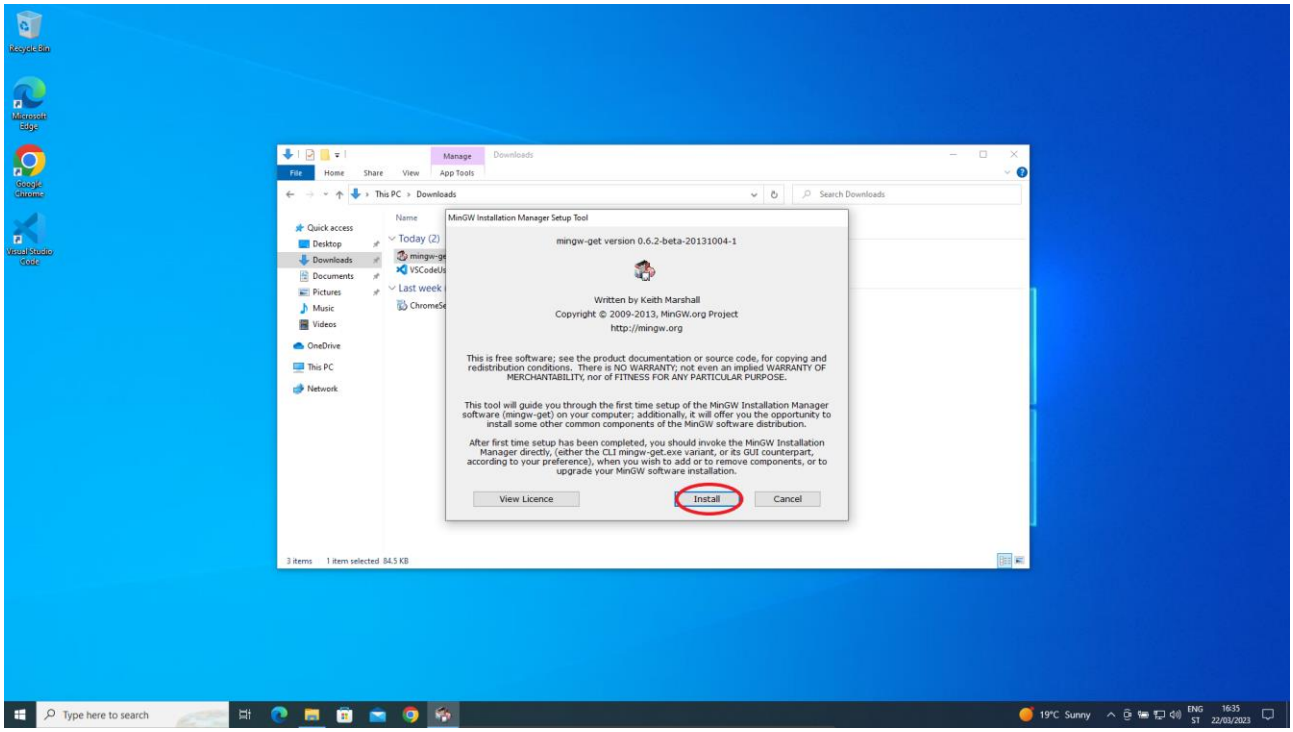

*Slika 1.1.24 Preuzimanje kompajlera za programski jezik C*

Nakon završetka preuzimanja u mapi Downloads nalazi se preuzeta instalacijska datoteka mingw-get-setup koju je potrebno pokrenuti dvostrukim klikom miša.



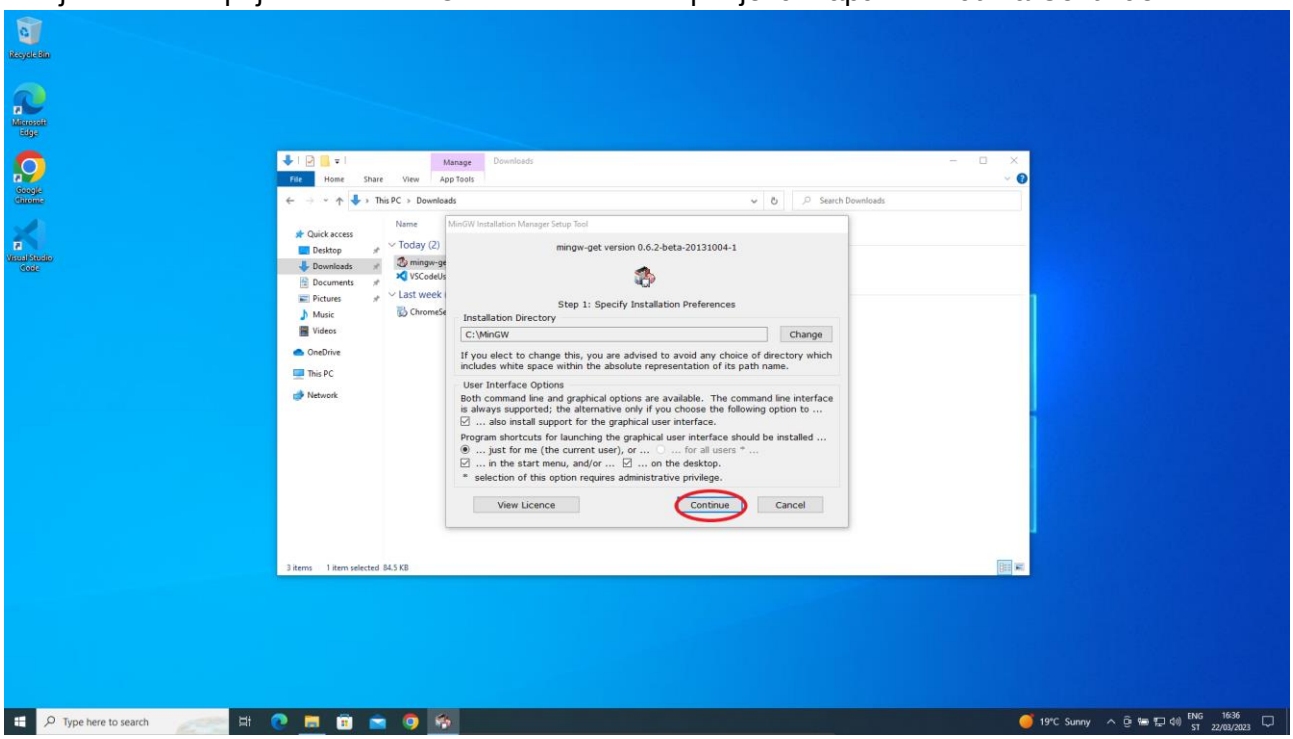

*Slika 1.1.25 Preuzeta instalacijska datoteka kompajlera za programski jezik C*

Nakon pokretanja instalacijske datoteke pojavljuje se prozor s osnovnim informacijama o kompajleru. Instalacija se pokreće klikom na Install.



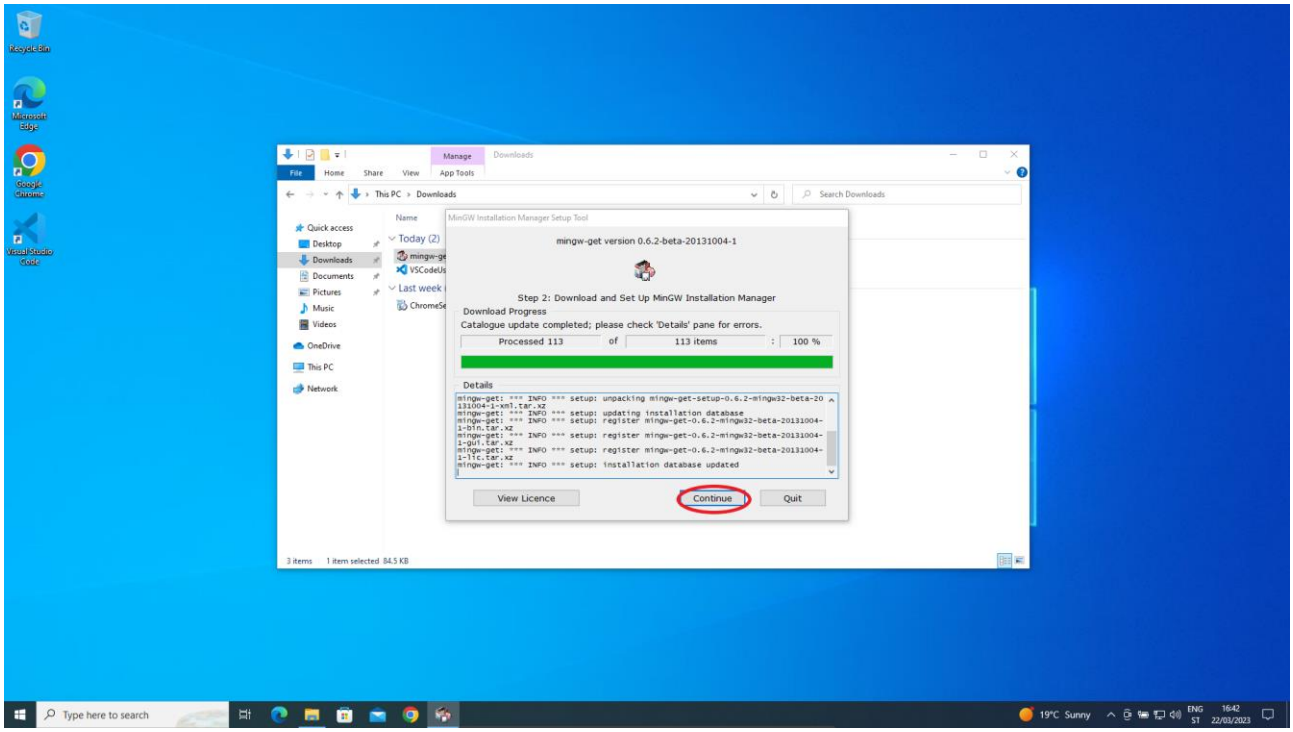

*Slika 1.1.26 Pokrenuta instalacijska datoteka kompajlera za programski jezik C*

Klikom na Install pojavljuje se prvi korak instalacijskog postupka u kojemu je moguće odabrati mapu u koju će se kompajler instalirati. Ostavit ćemo već ispunjenu mapu i kliknuti na Continue.



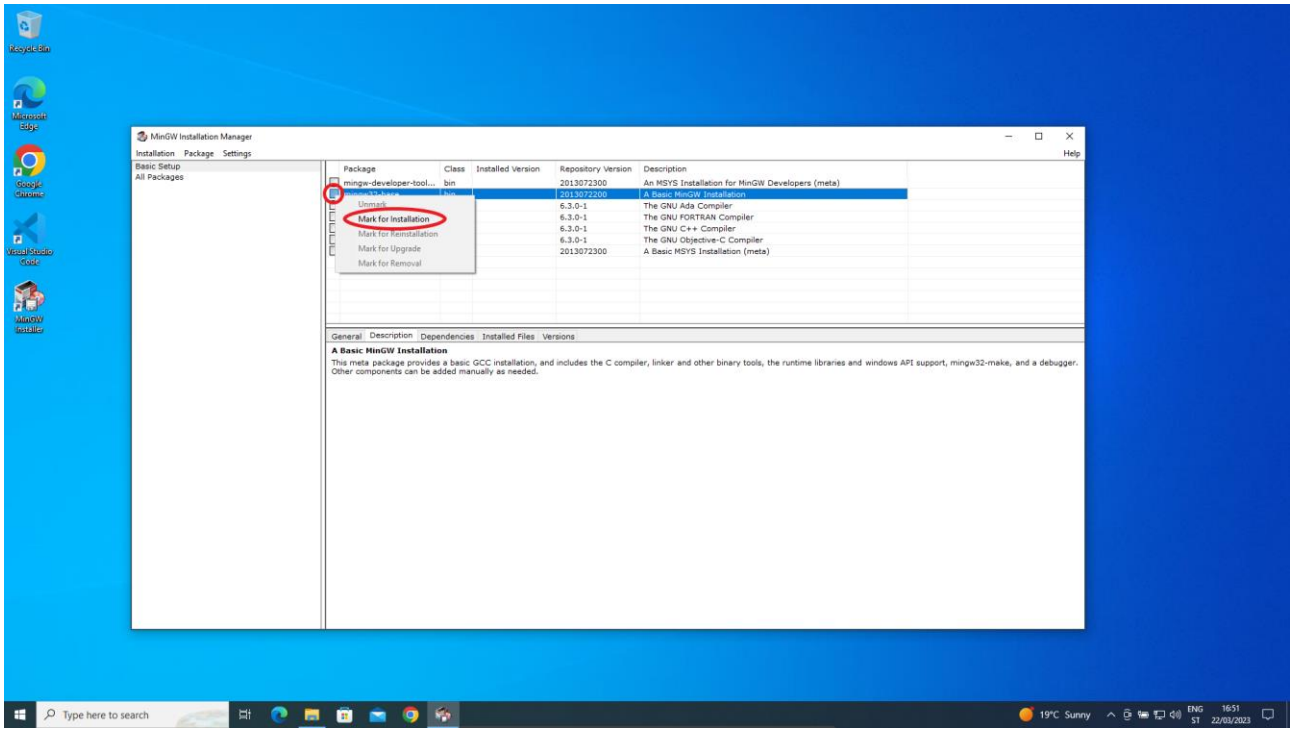

*Slika 1.1.27 Izbor mape za instalaciju kompajlera*

U drugom koraku instalacijskog postupka prikazuje se napredak instalacije. Nakon završetka potrebno je kliknuti na Continue.



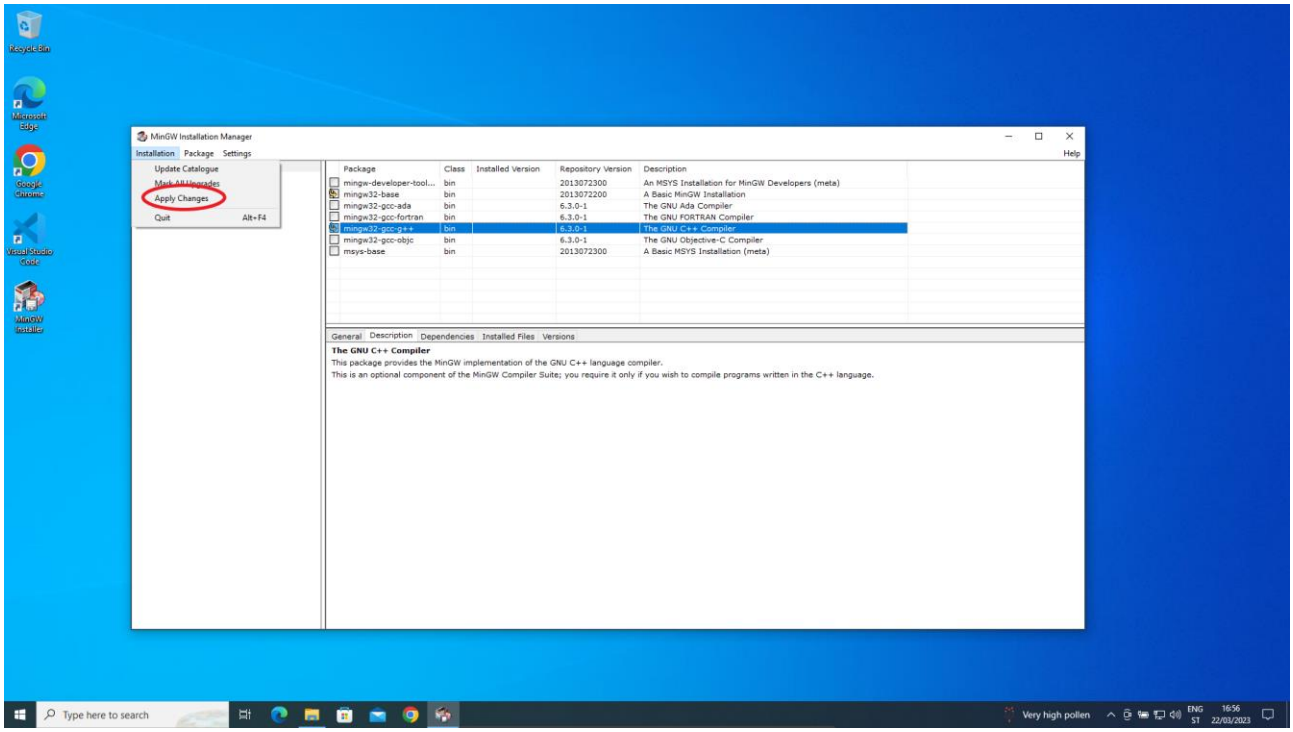

*Slika 1.1.28 Progres instalacije kompajlera*

Nakon završetka instalacije prikazuje se prozor MinGW Installation Manager na kojem je potrebno izabrati dva paketa (**mingw32-base** i **mingw32-gcc-g++**) kako bi se omogućilo kompajliranje i pokretanje programa pisanog u programskom jeziku C unutar softvera Visual Studio Code. Odabir paketa sradi se jedan po jedan i na način da se mišem klikne na područje potvrdnog okvira (engl. *checkbox*). Tada se otvara izbornik na kojem je potrebno odabrati Mark for Installatio.



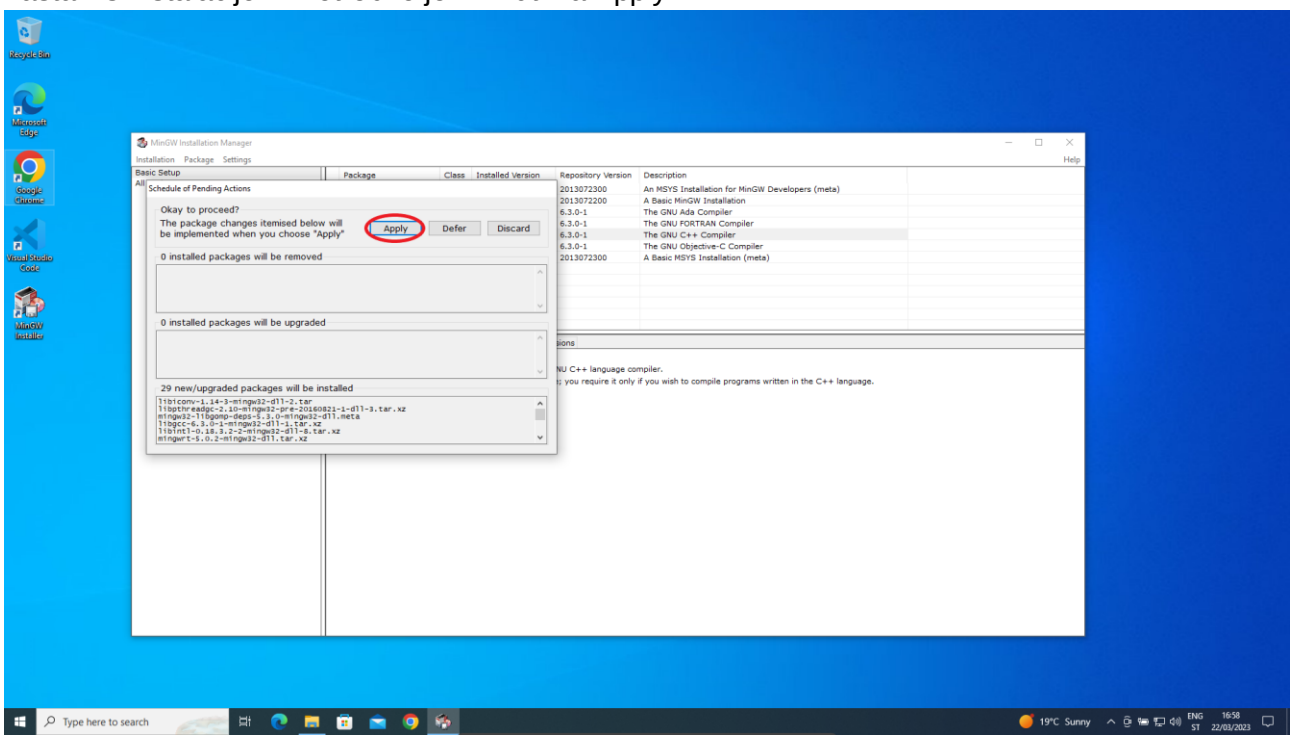

*Slika 1.1.29 Izbor paketa potrebnih za pokretanje i kompajliranje programa pisanih u programskom jeziku C*

Nakon odabira potrebnih paketa oni su posebno označeni. Da bi se ti paketi instalirali, potrebno je odabrati izbornik Apply Change (unutar izbornika Installation).



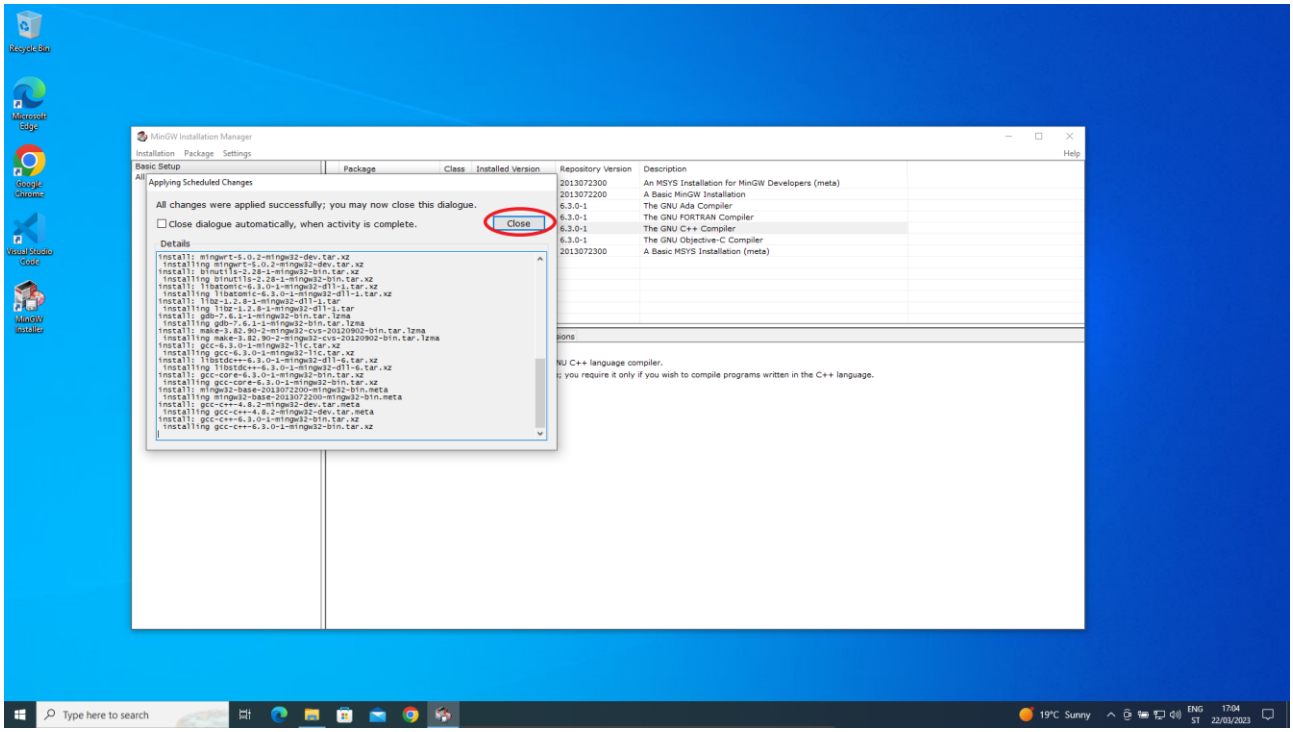

*Slika 1.1.30 Pokretanje instalacije izabranih paketa kompajlera*

Prije samog procesa instalacije pojavljuje se još jedan prozor koji pokazuje koji će efekt biti ako se nastavi s instalacijom. Potrebno je kliknuti na Apply.



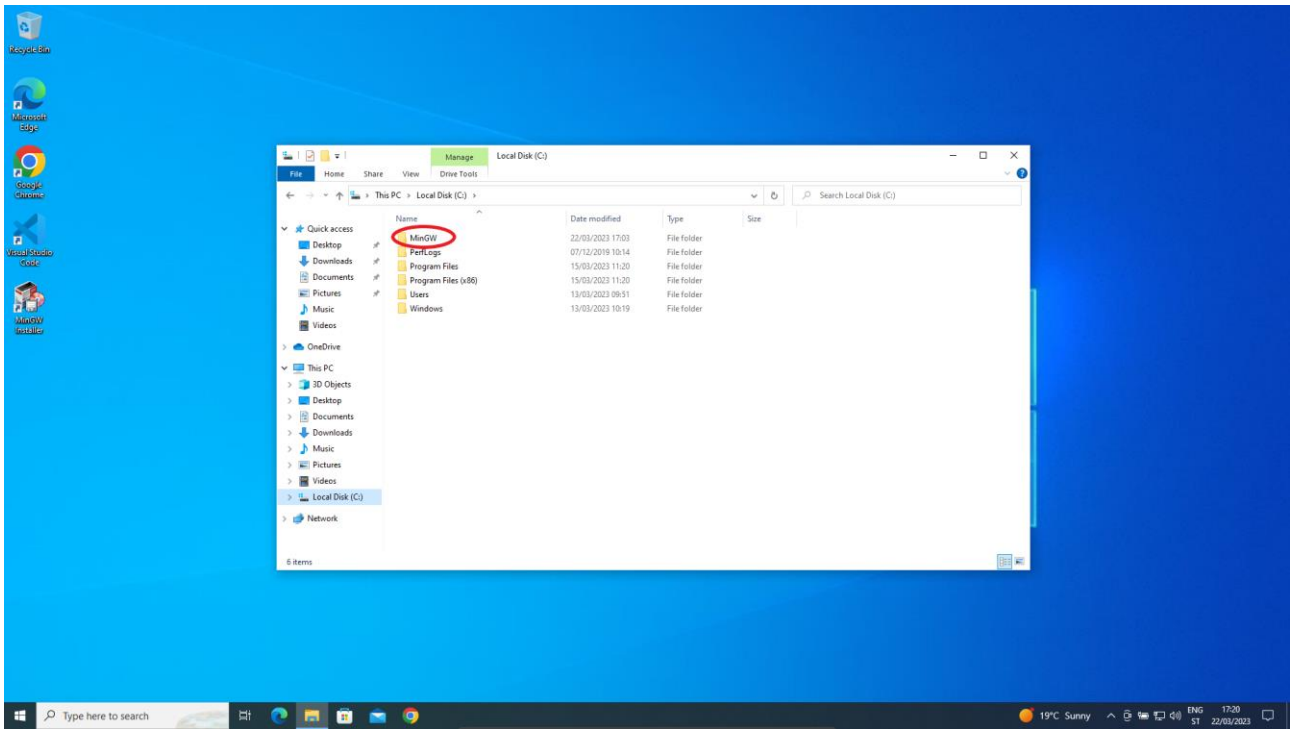

*Slika 1.1.31 Prikaz efekta prije instalacije*

Nakon pokretanja procesa instalacije preuzimaju se potrebne datoteke koje se instaliraju. Po završetku pojavljuje se prozor na kojem je izvještaj što je sve napravljeno. Potrebno je kliknuti Close.



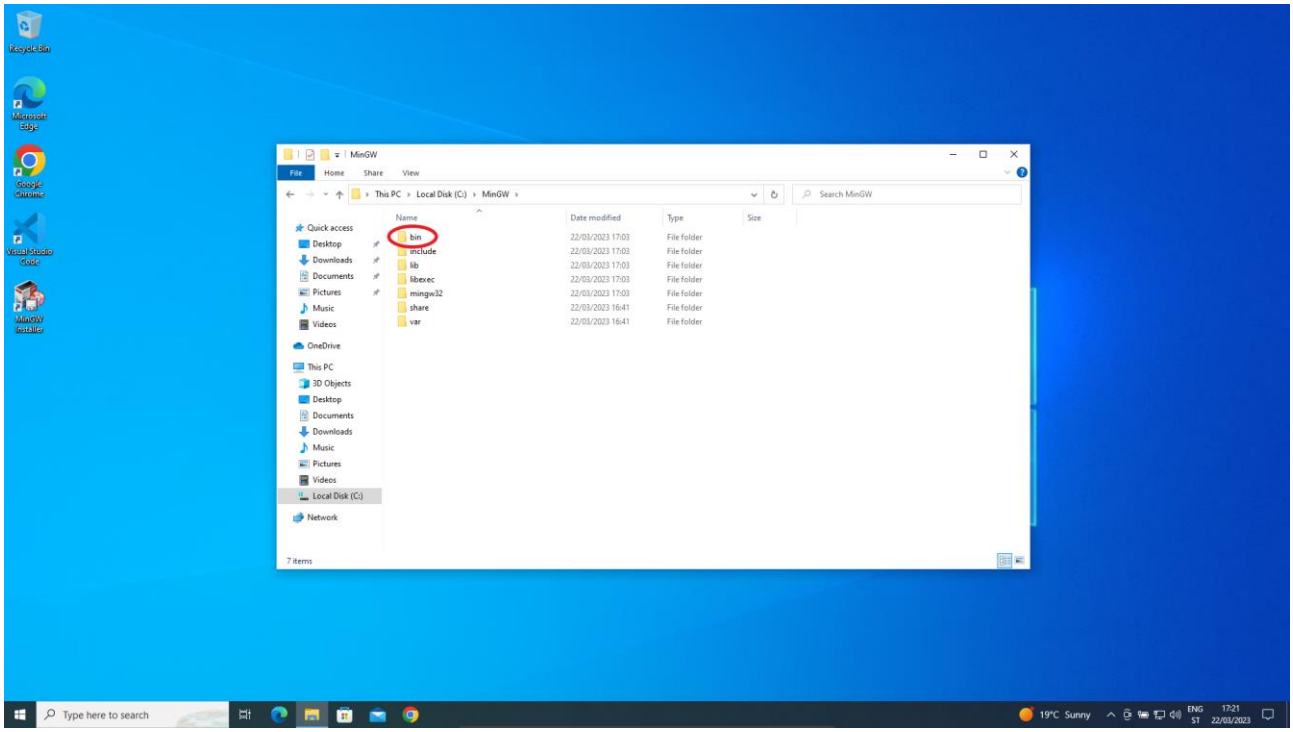

*Slika 1.1.32 Prikaz efekta nakon instalacije*

Kako u softveru Visual Studio Code ne bismo stalno morali pisati putanju do datoteka kompajlera, u varijablu okruženja (engl. *environment variables*) dodat ćemo u Path odgovarajuću putanju do datoteka kompajlera. Da bismo došli do putanje koju trebamo spremiti, potrebno je preko preglednika datoteka otvoriti mapu u koju je instaliran kompajler.



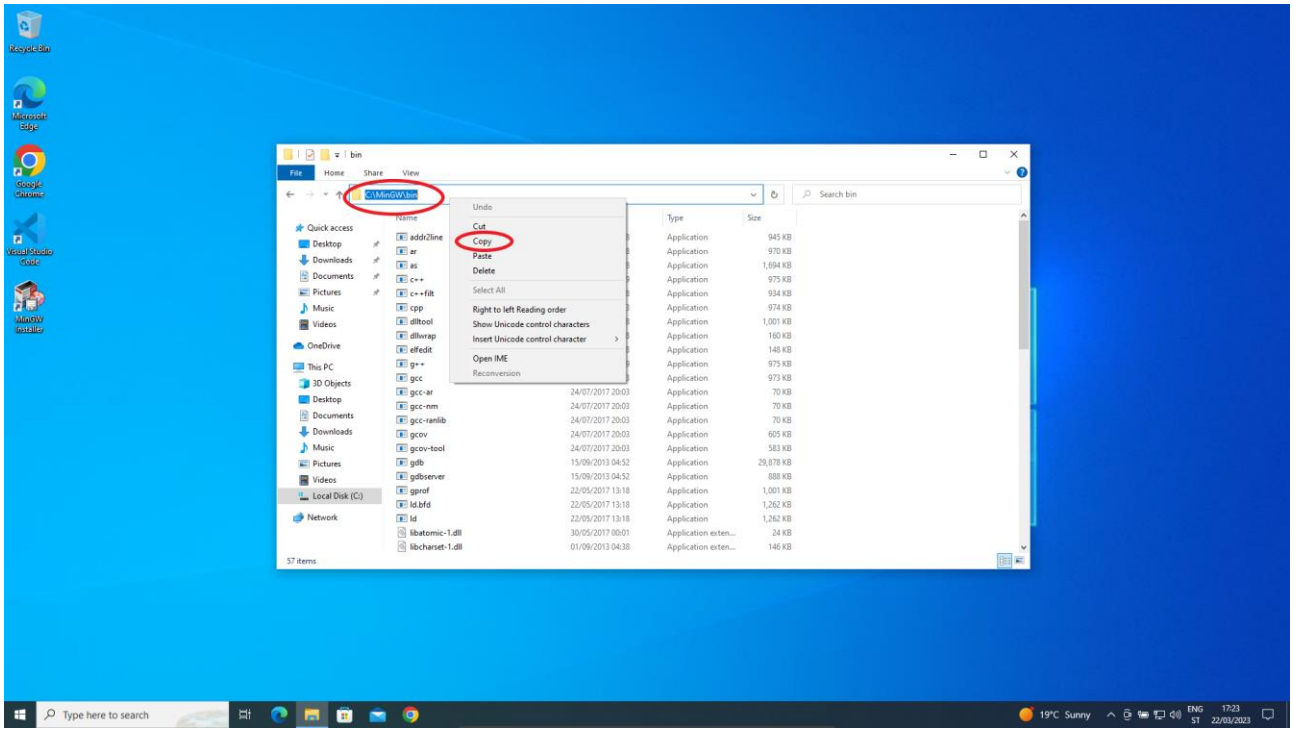

*Slika 1.1.33 Prikaz mape u kojoj je instaliran kompajler*

Unutar te mape treba pronaći podmapu bin.



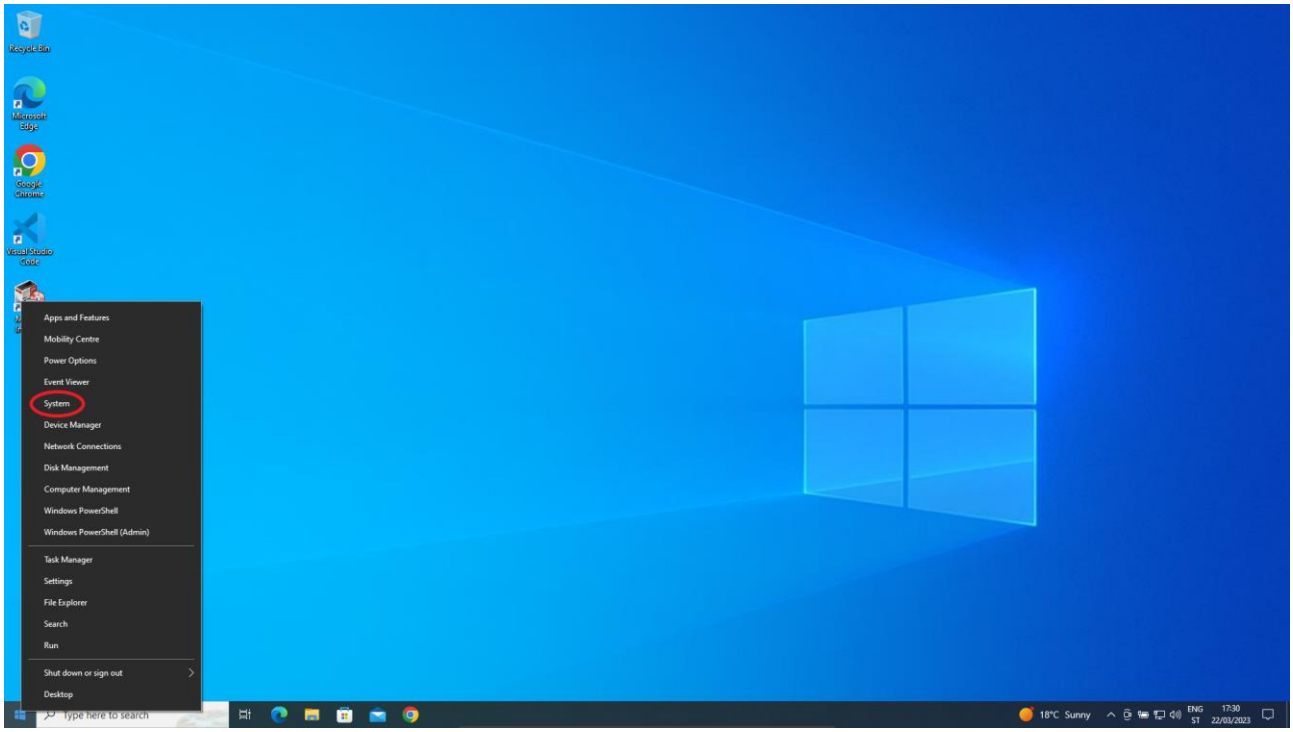

*Slika 1.1.34 Prikaz podmape bin unutar mape u kojoj je instaliran kompajler*

Potrebno je otvoriti mapu bin i kopirati putanju desnim klikom na polje koje prikazuje putanju.



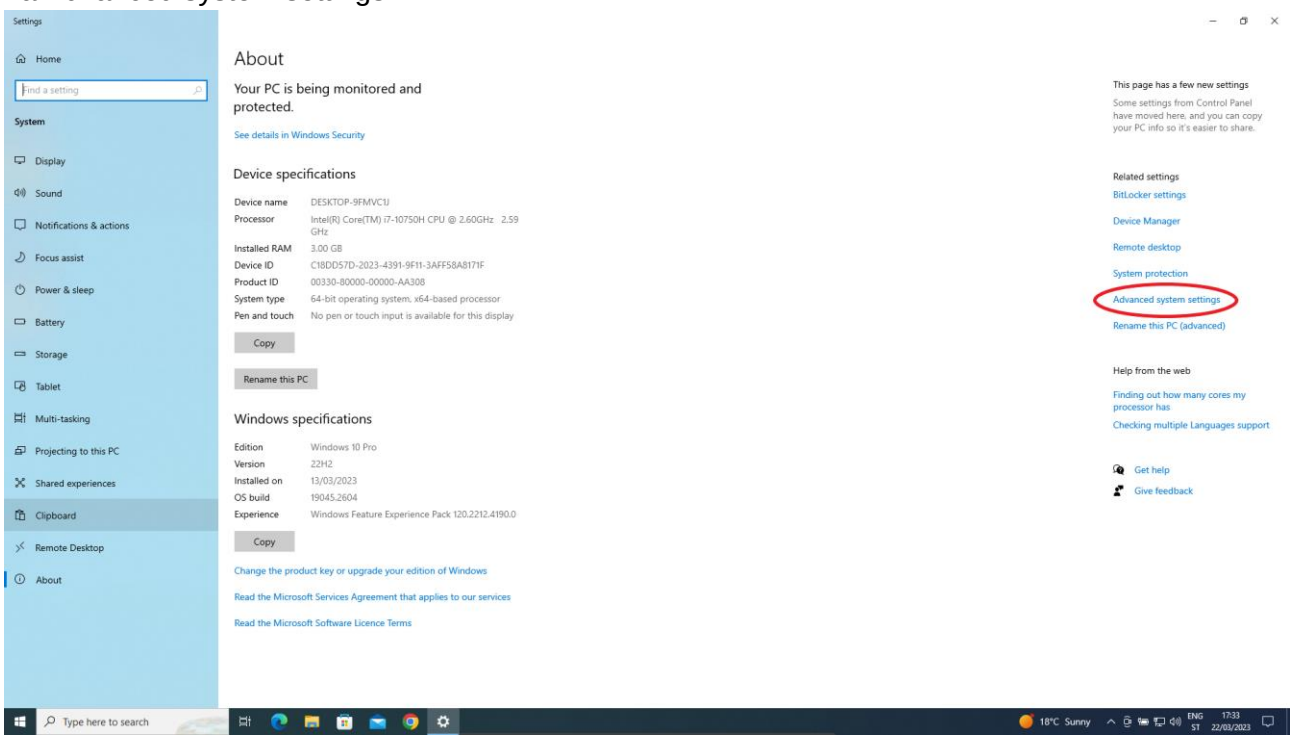

*Slika 1.1.35 Kopiranje putanje do podmape bin unutar mape u kojoj je instaliran kompajler*

Sada je tu kopiranu putanju potrebno postaviti u varijablu okruženja Path. Da bi se to napravilo, potrebno je otvoriti prozor za izmjenu postavki operacijskog sustava do kojega se dolazi desnim klikom miša na izbornik Start i zatim na izbornik System.



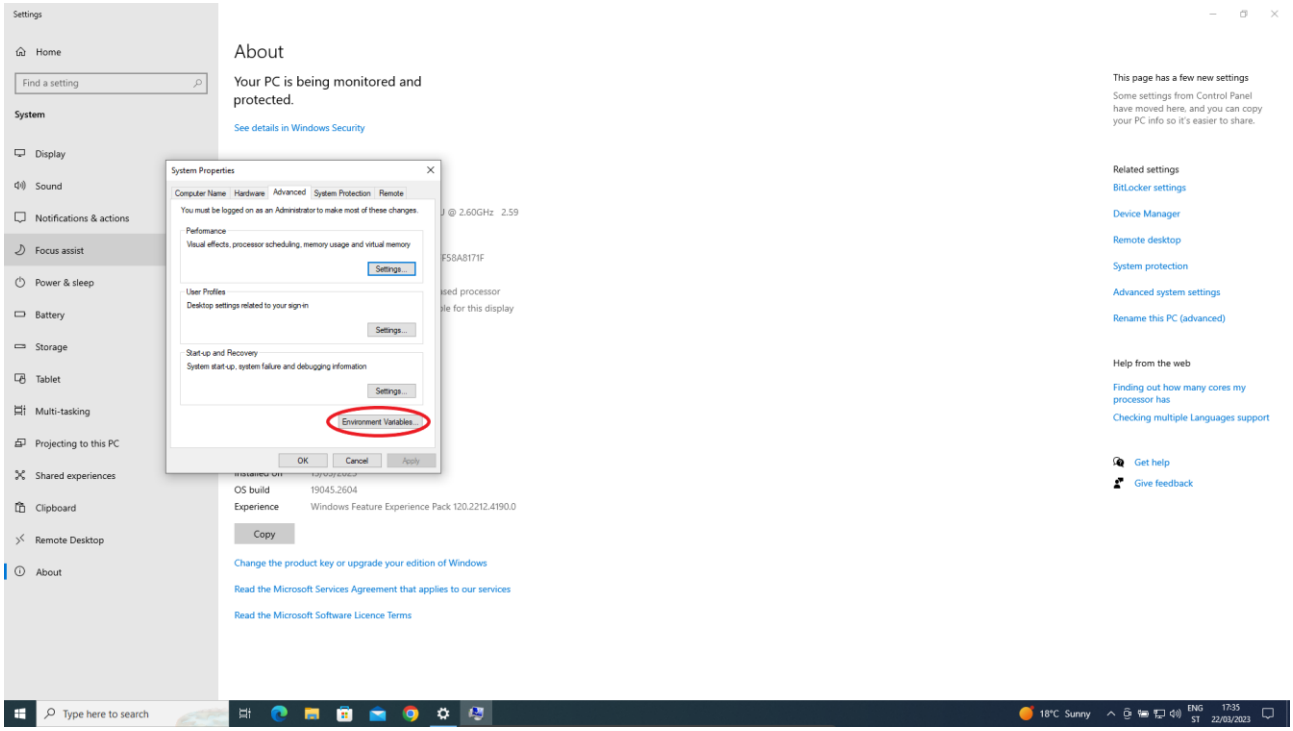

*Slika 1.1.36 Dolazak do prozora za izmjenu postavki operacijskog sustava*

Otvara se prozor s osnovnim značajkama računala i operacijskog sustava. Tu je potrebno kliknuti na Advanced system settings.



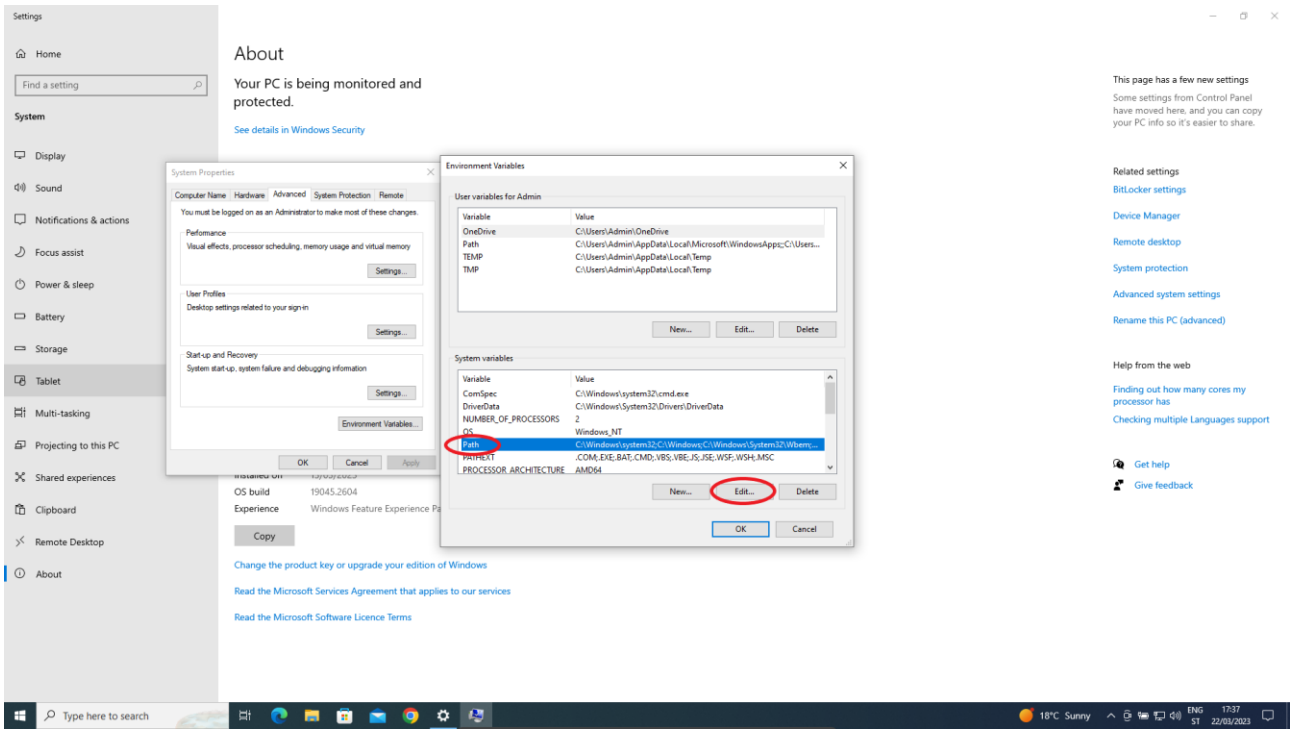

*Slika 1.1.37 Prozor s osnovnim značajkama računala i operacijskog sustava*

Klikom na Advanced system setting otvara se prozor na kojem trebamo kliknuti na Environment Variables….



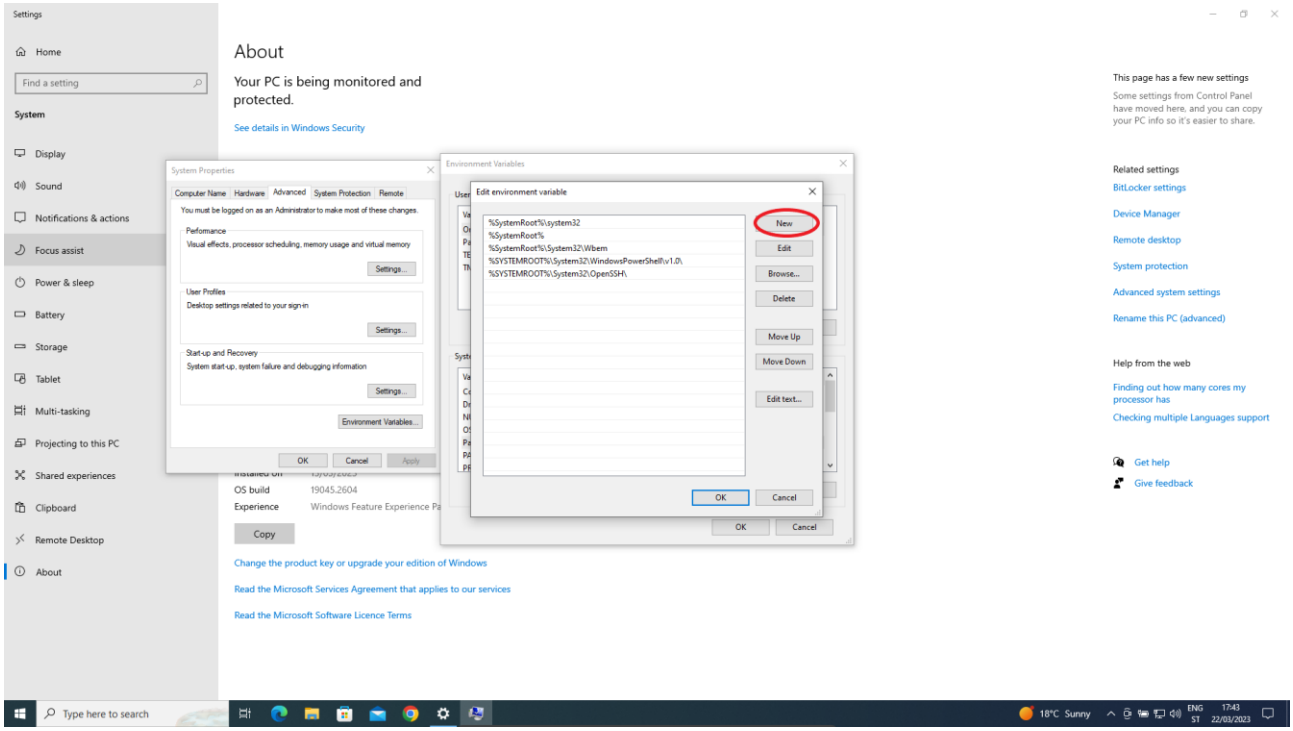

*Slika 1.1.38 Prozor za izmjenu svojstava sustava*

Nakon klika na Environment Variables… otvara se prozor s varijablama okruženja. Nas zanima varijabla Path koja se nalazi među sistemskim varijablama (engl. *system variables*). Nju izaberemo i kliknemo na Edit….



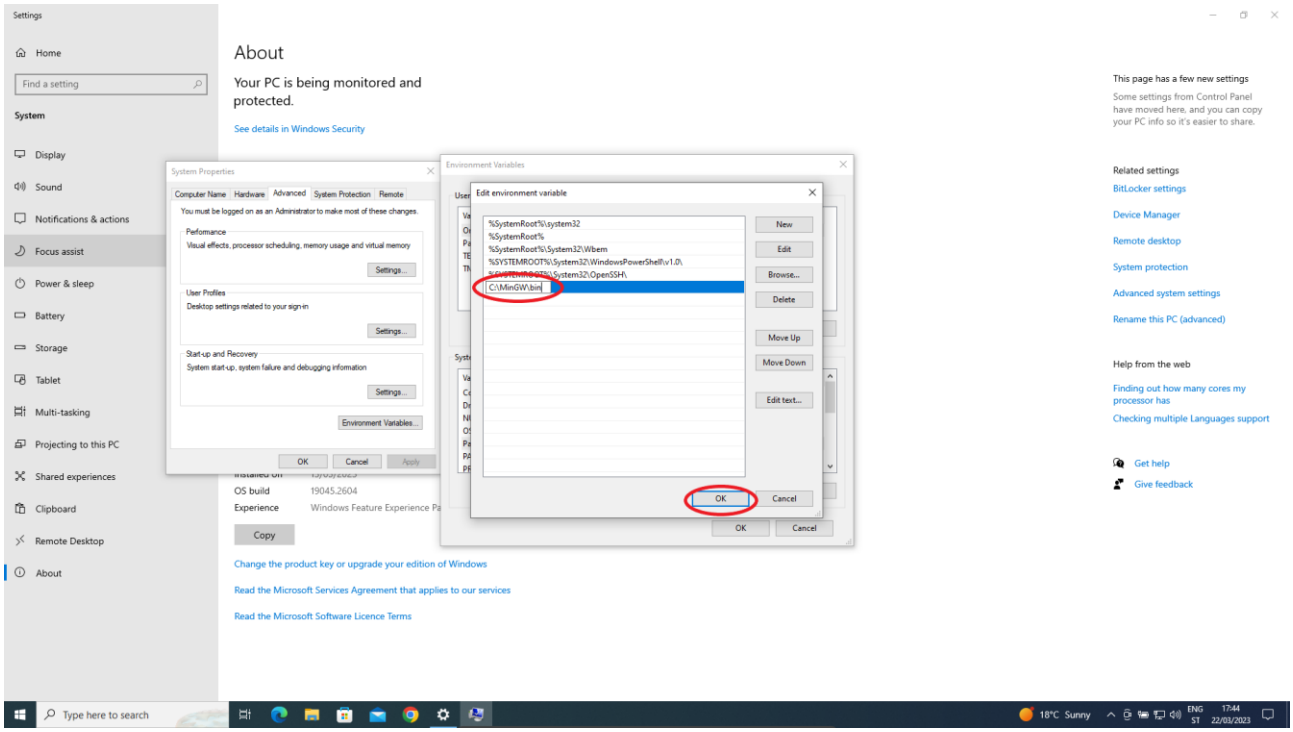

*Slika 1.1.39 Sistemska varijabla Path*

Klikom na Edit… otvara nam se prozor za ažuriranje vrijednosti varijable Path. Kliknut ćemo na gumb New kako bismo varijabli dodali novu vrijednost koja je putanja do podmape bin unutar mape gdje je instaliran kompajler (u jednom od prethodnih koraka tu smo putanju kopirali).



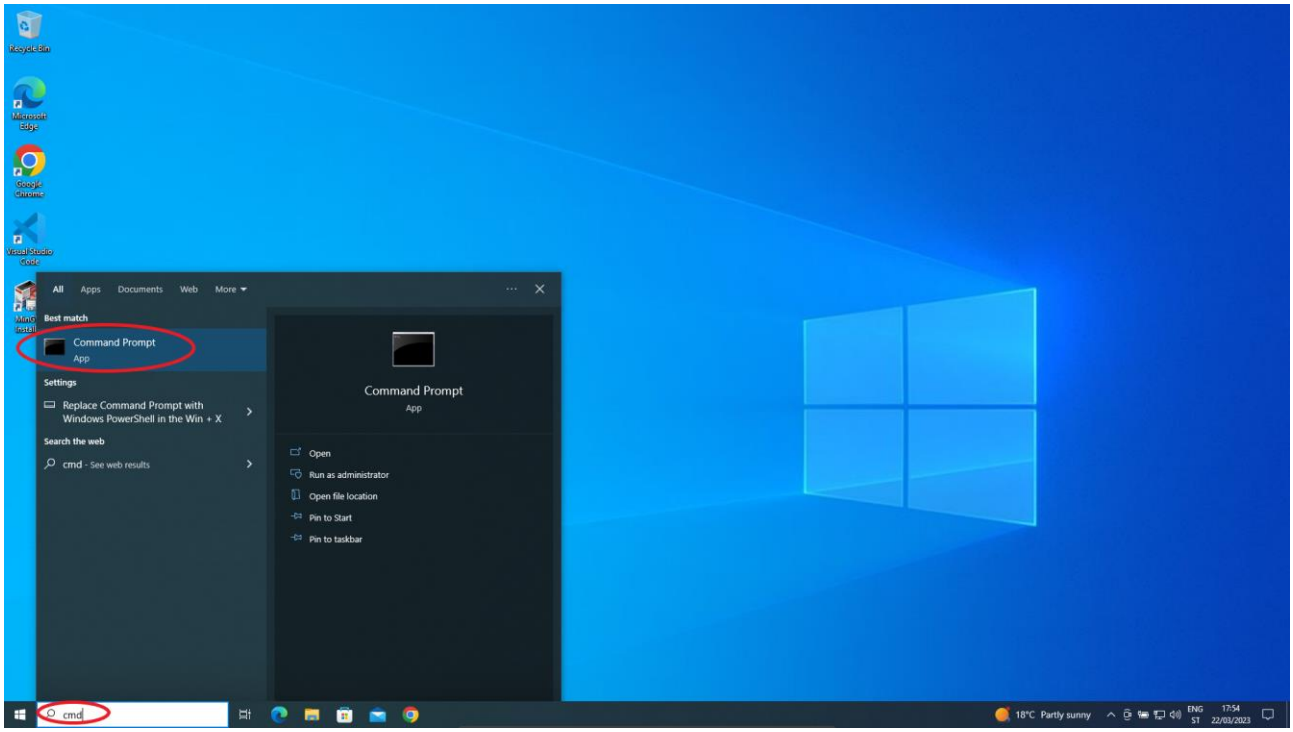

*Slika 1.1.40 Prozor za izmjenu sistemske varijable Path*

Nakon ubacivanja nove putanje potrebno je kliknuti na OK.



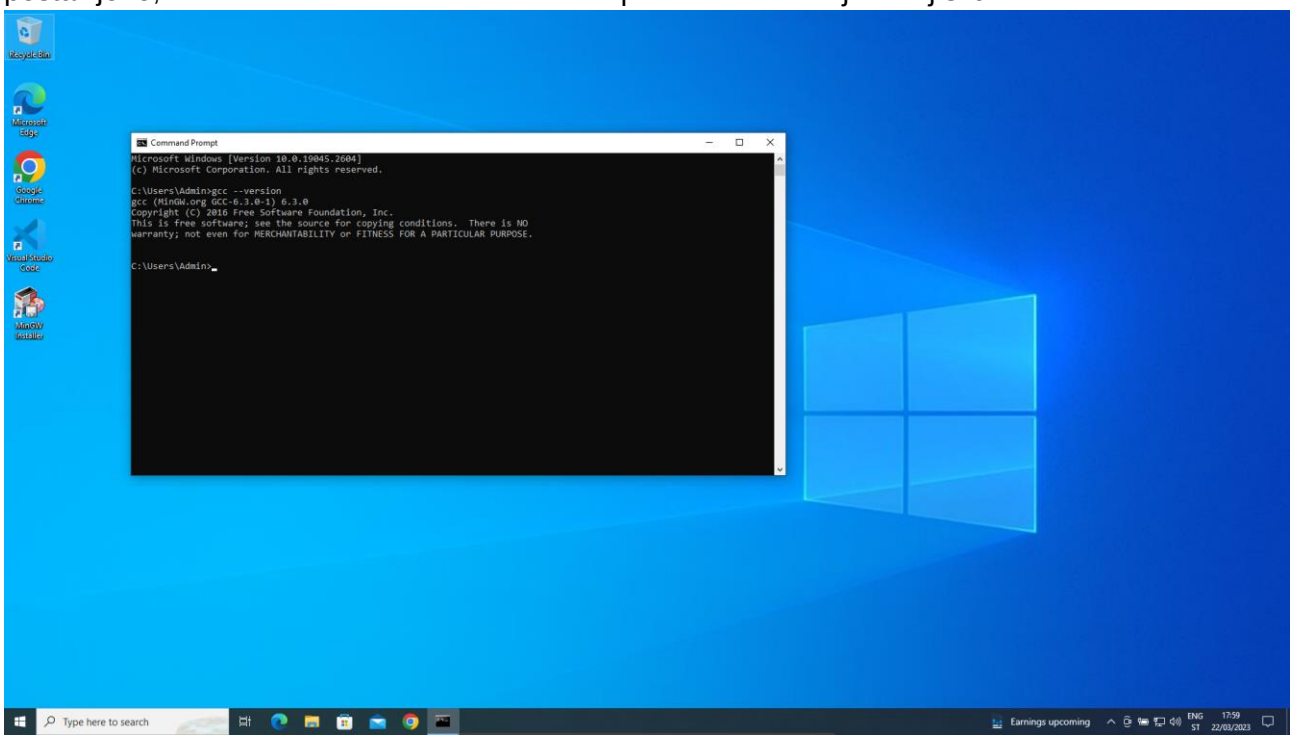

*Slika 1.1.41 Ubacivanje nove putanje u sistemsku varijablu Path*

Nakon što se klikne na OK, potrebno je na gumb OK kliknuti i na prozoru Environment Variables i System Properties (svaki od prozora na koji se klikne na OK zatvara se).
Kako bismo provjerili je li kompajler uspješno postavljen, možemo otvoriti naredbeni redak (engl. *command prompt*) tako da u pretraživaču upišemo cmd i kliknemo na aplikaciju Command Prompt.



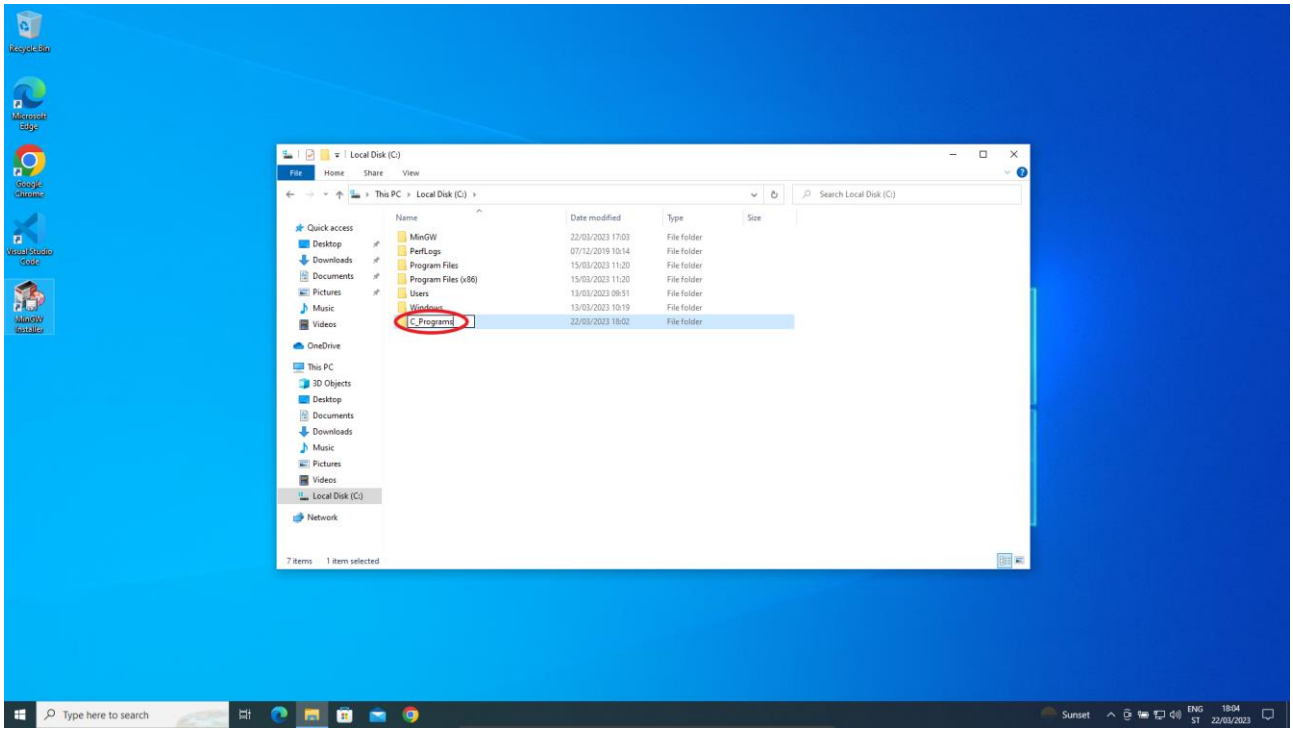

*Slika 1.1.42 Pokretanje naredbenog retka*

U naredbenom retku potrebno je upisati ukucati **gcc --version** te pritisnuti **enter**. Ako je sve uspješno postavljeno, rezultat bi trebao biti sličan ovome prikazanom na sljedećoj slici.



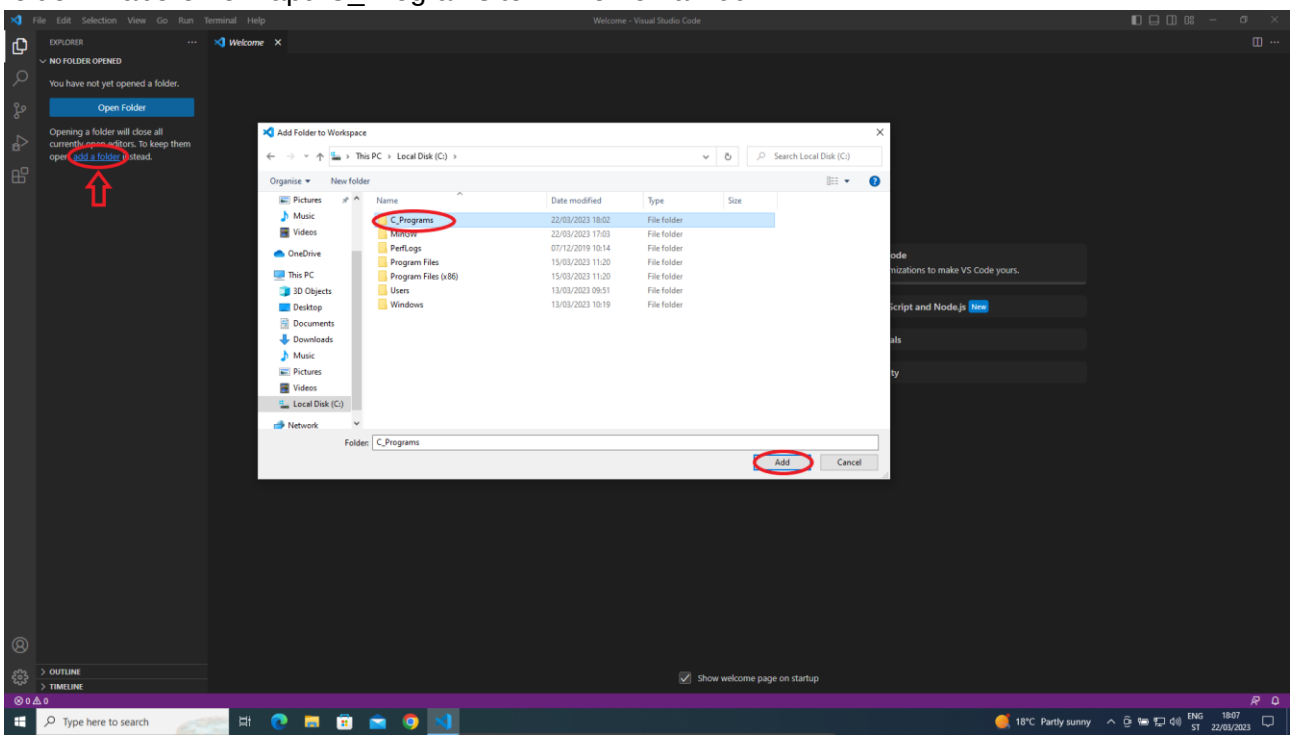

*Slika 1.1.43 Provjera verzije kompajlera*

Nakon uspješno postavljenog kompajlera izradit ćemo mapu u koju ćemo spremati naše programe pisane u programskom jeziku C. Neka se mapa zove C_Programs.



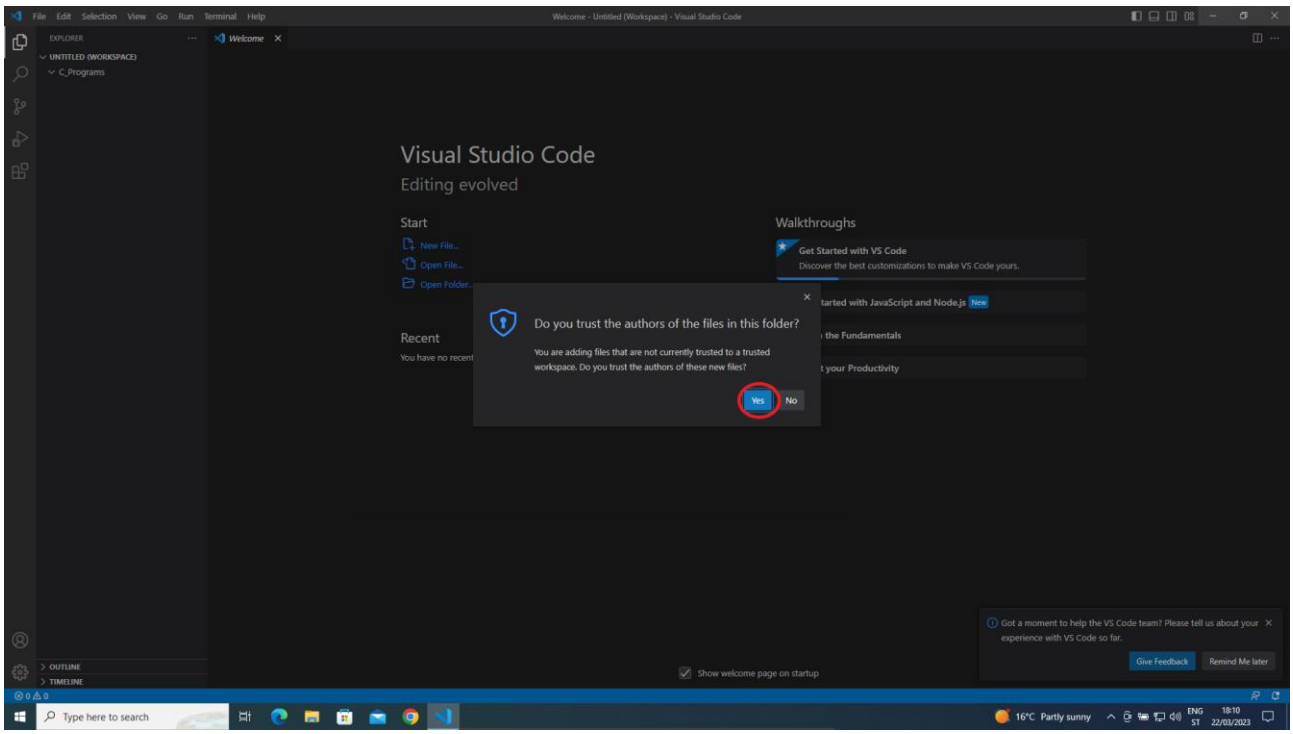

*Slika 1.1.44 Izrada mape C_Programs*

Pokrenut ćemo Visual Studio Code i u njega dodati mapu C_Programs tako da kliknemo na Add folder i izaberemo mapu C_Programs te kliknemo na Add.



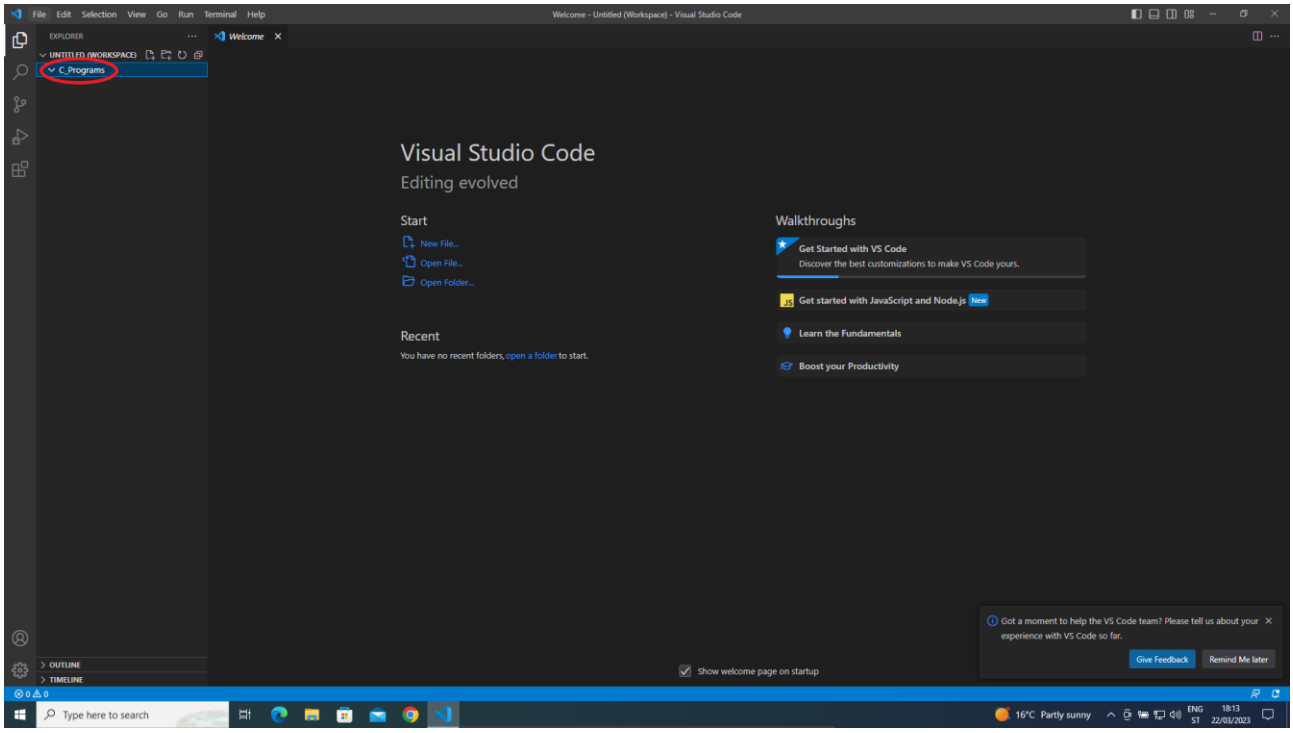

*Slika 1.1.45 Dodavanje mape C_Programs u Visual Studio Code*

Otvara nam se prozor s pitanjem na koje potvrdno odgovaramo.



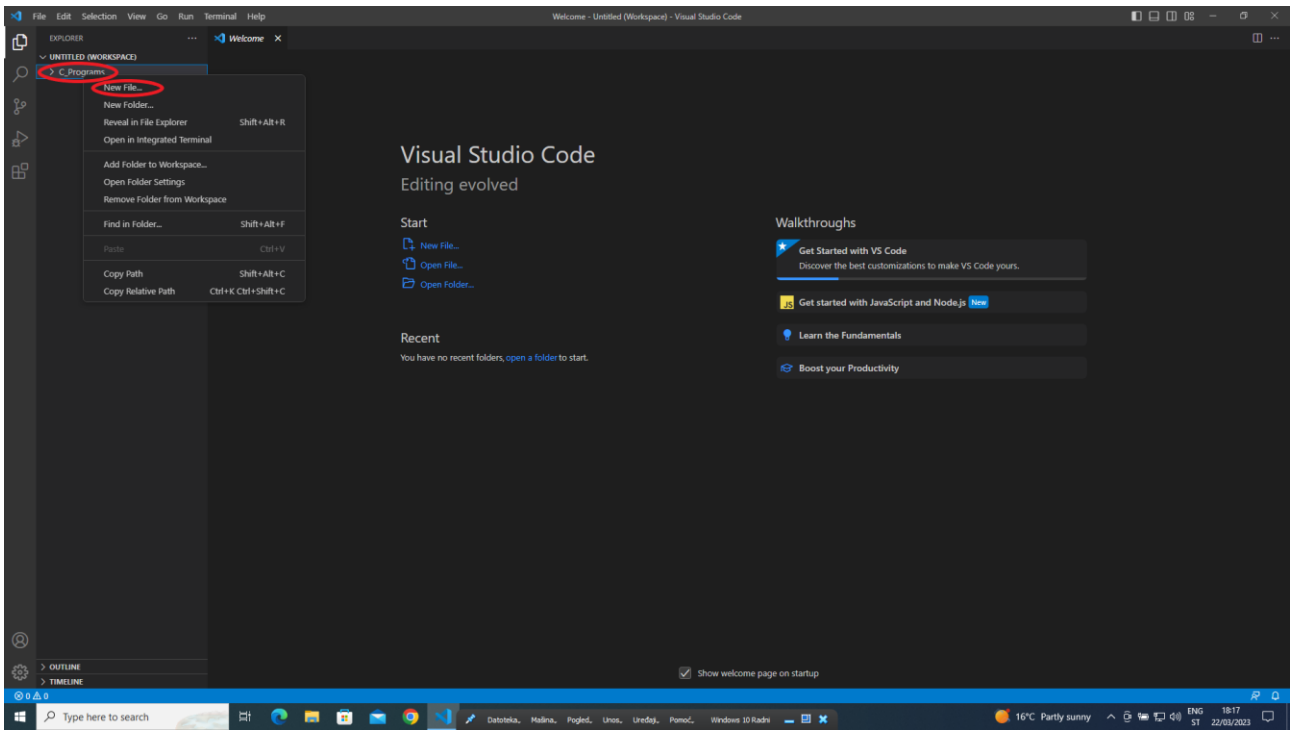

*Slika 1.1.46 Poruka nakon dodavanja mape u Visual Studio Code*

Sada možemo vidjeti da je mapa C_Programs dodana u Visual Studio Code.



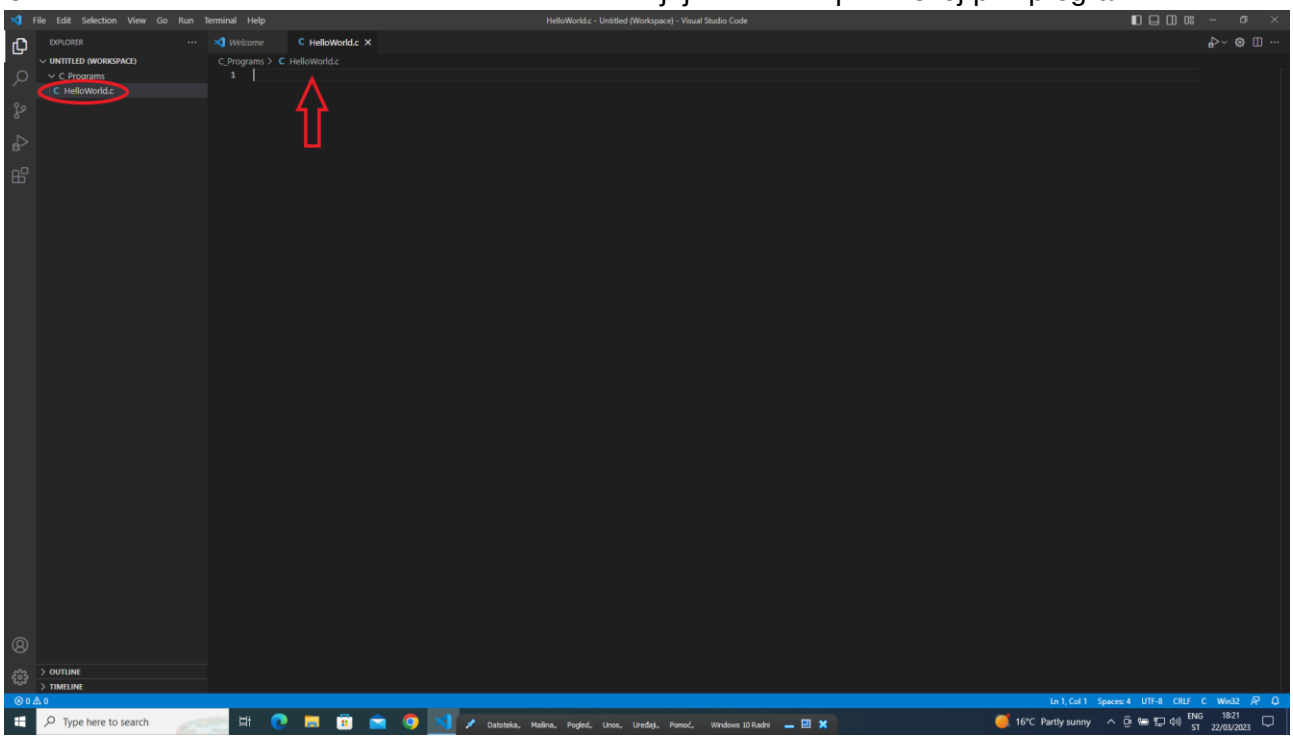

*Slika 1.1.47 Mapa C_Programs dodana u Visual Studio Code*

Kako bismo isprobali mogućnost pisanja programskog koda u programskom jeziku C unutar softvera Visual Studio Code i njegova kompajliranja i pokretanja, izradit ćemo jednu datoteku unutar mape C_Programs koju ćemo nazvati HelloWorld.c. Kliknut ćemo desnim klikom na mapu C_Programs i izabrati New File….




*Slika 1.1.48 Stvaranje nove datoteke unutar mape C_Programs*

Kada se otvori područje za unos naziva datoteke, upisat ćemo HelloWorld.c i pritisnuti enter. Otvorit će se nova kartica imena HelloWorld.c u kojoj možemo upisati svoj prvi program.



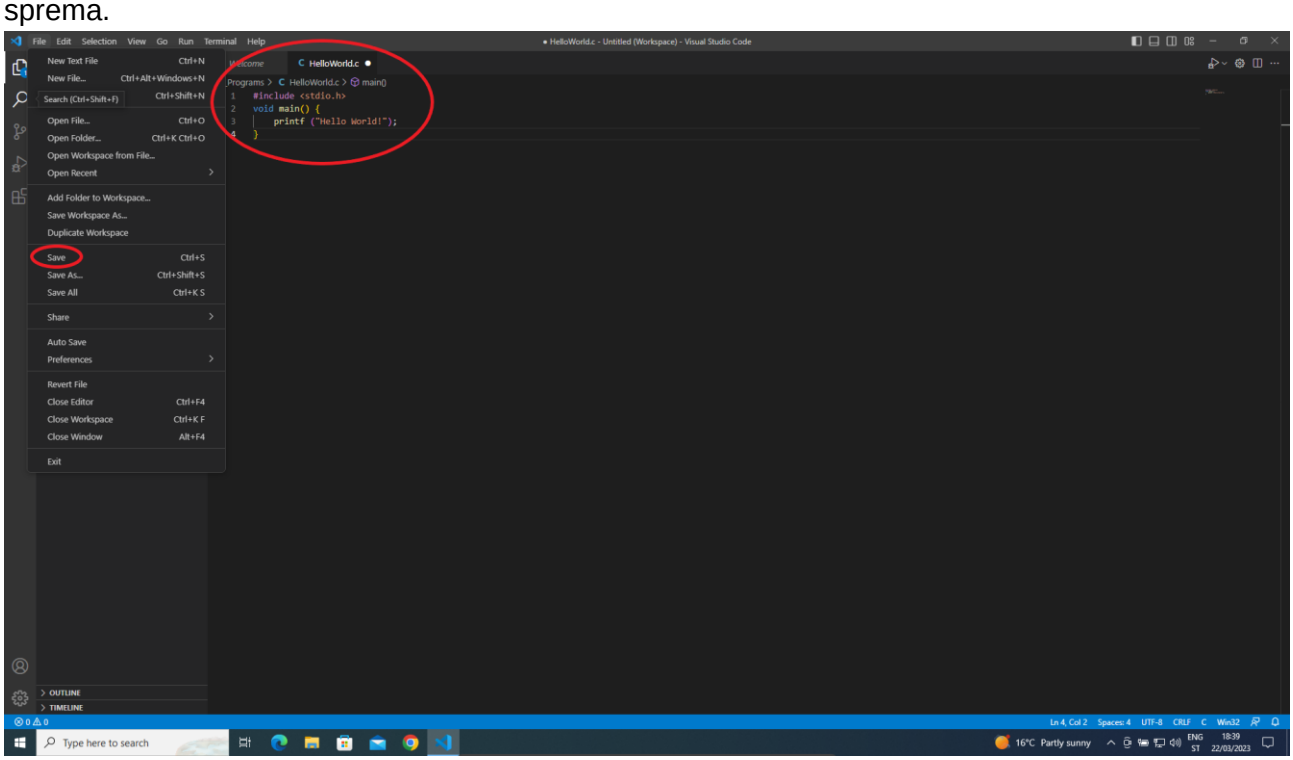
*Slika 1.1.49 Izrađena datoteka HelloWorld.c i spremna za unos programskog koda*

Prvi program koji ćemo napraviti će, kada se pokrene, ispisati pozdravnu poruku Hello World!. Slijedi programski kôd:

```
#include <stdio.h>
void main() {
    printf ("Hello World!");
```

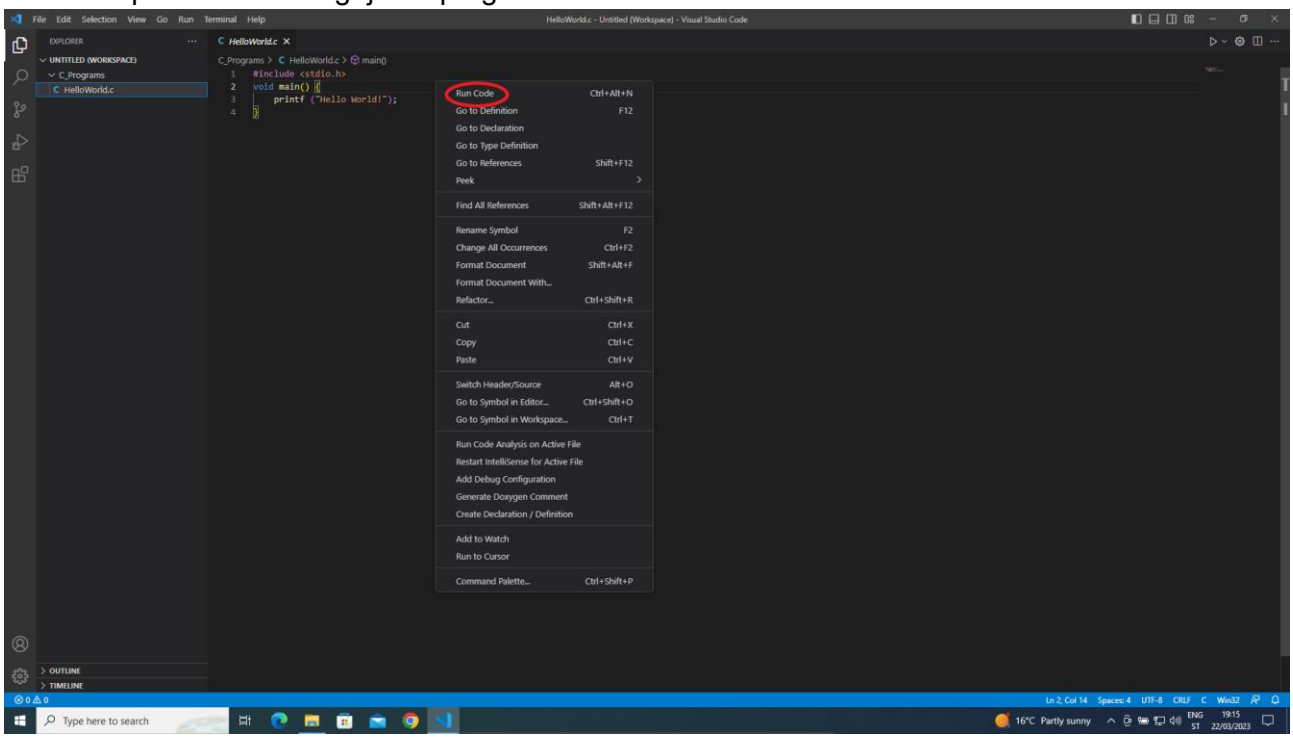
```
}
```

Na sljedećoj slici vidi se kako je programski kôd upisan u datoteku HelloWorld.c i kako se datoteka sprema.



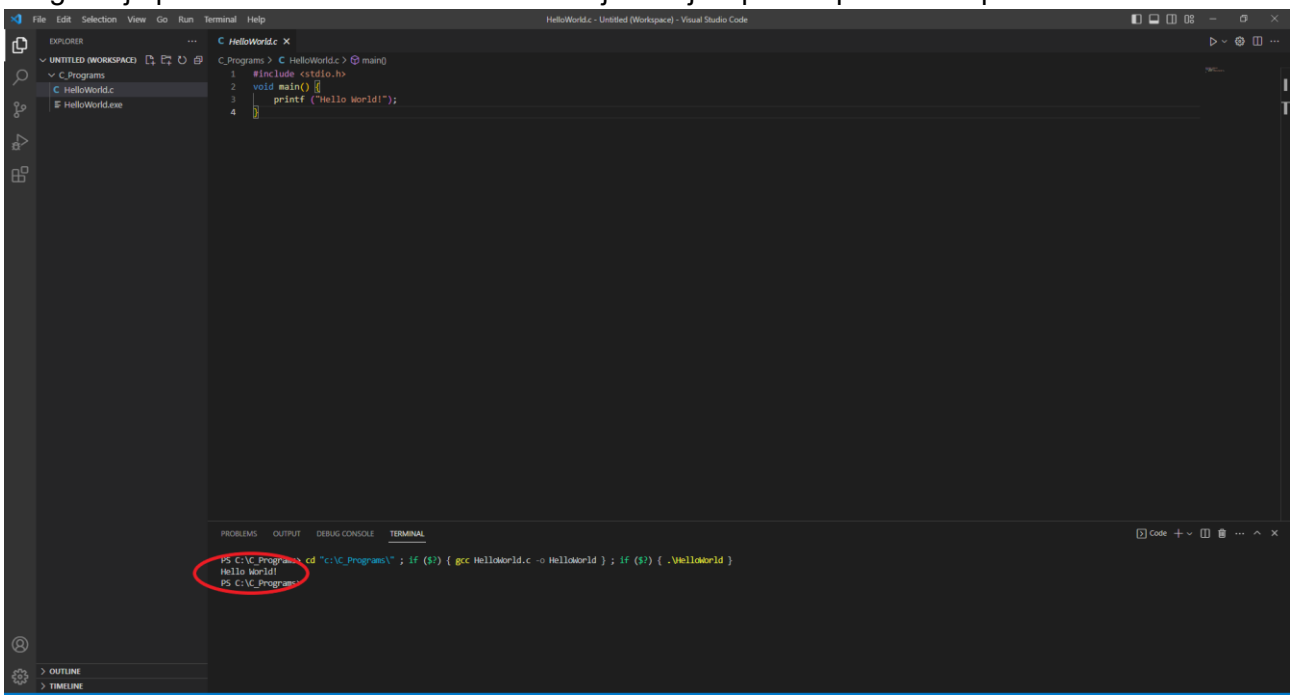

*Slika 1.1.50 Upis programskog koda i spremanje datoteke HelloWorld.c*

Kako bismo mogli pokrenuti program napisan u programskom jeziku C, potrebno je kliknuti desnom tipkom miša bilo gdje na programski kôd i izabrati Run Code.




*Slika 1.1.51 Pokretanje programa*

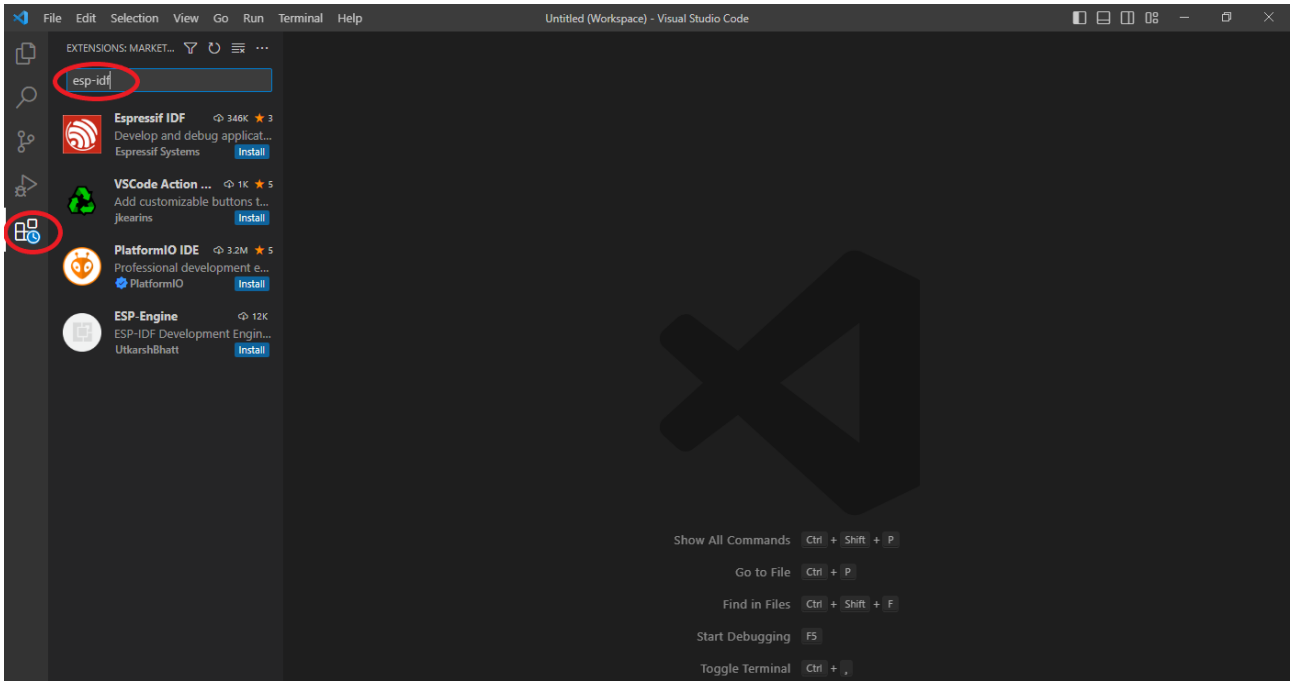
Program je pokrenut u terminalu i može se vidjeti da je ispisana pozdravna poruka Hello World!.




*Slika 1.1.52 Program pokrenut u terminalu*

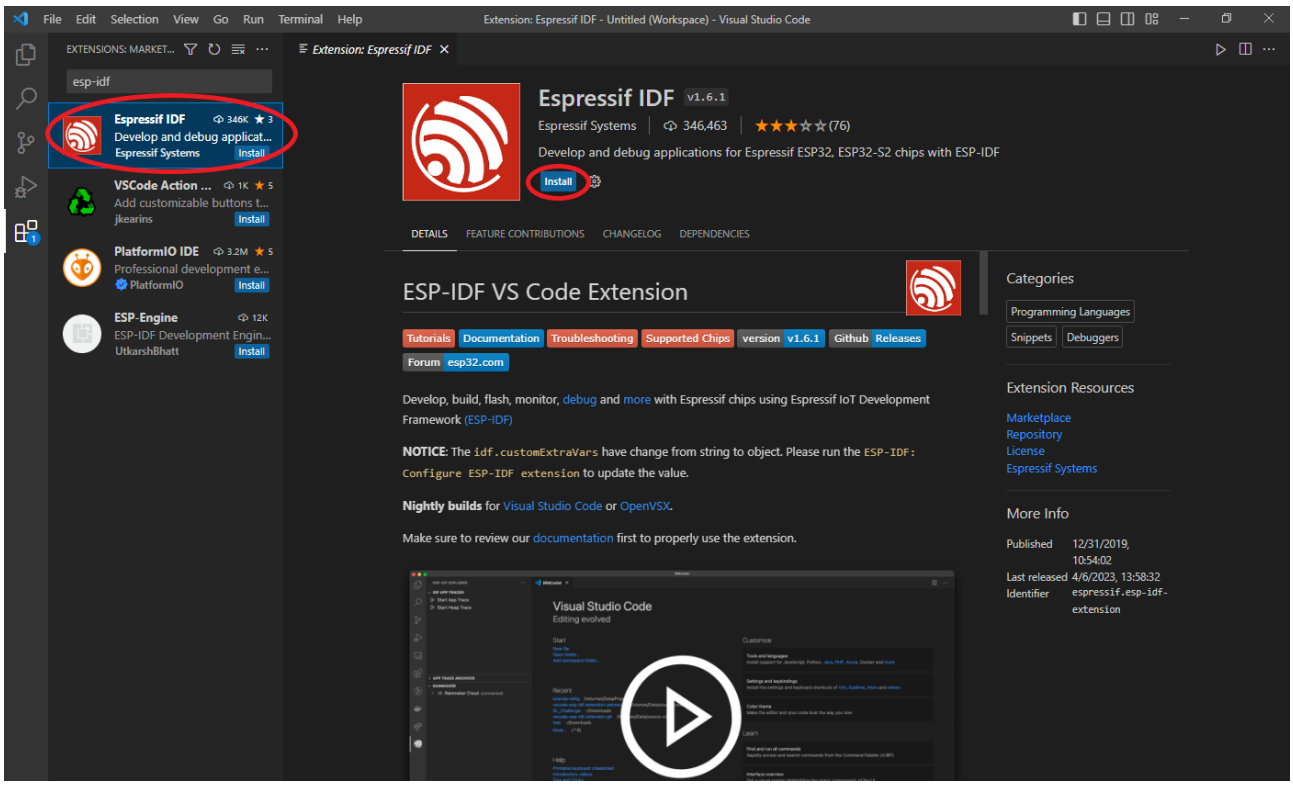
### 1.1.3.4. Preuzimanje i postavljanje proširenja razvojnog okvira ESP-IDF

U svrhu razvoja sustava IoT koji se temelji na razvojnoj pločici ESP32-DevKitC-VIE potreban nam je razvojni okvir ESP-IDF koji će programerima olakšati programiranje različitih dijelova sustava IoT (npr. očitanje vrijednosti temperaturnog senzora). Budući da za programiranje koristimo Visual Studio Code, to je u njemu potrebno preuzeti i instalirati proširenje ESP-IDF. Potrebno je preko izbornika Extensions doći do liste proširenja te napraviti pretraživanje po ključnoj riječi esp-idf.



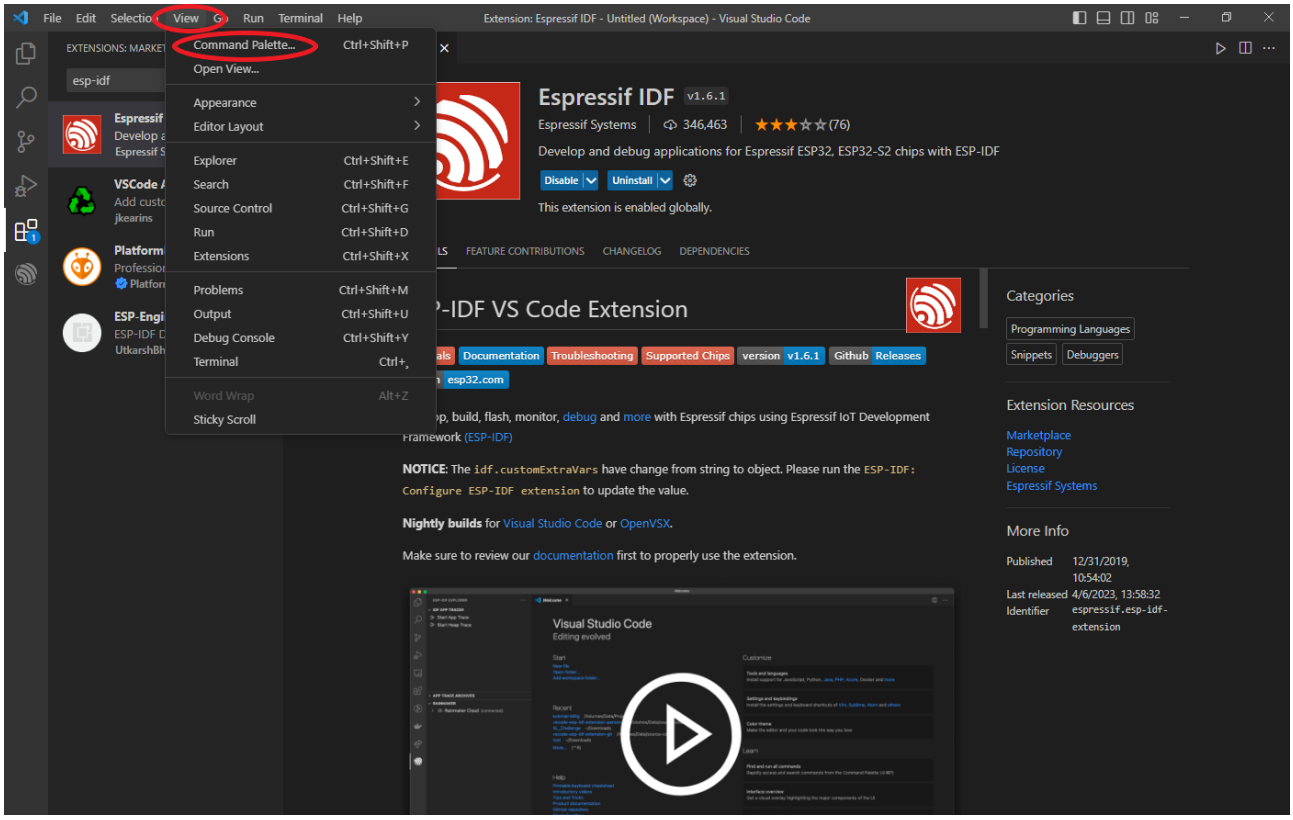

*Slika 1.1.53 Pretraživanje proširenja esp-idf*

Proširenje koje se treba instalirati jest Espressif IDF. U desnom dijelu zaslona odabirom tog proširenja prikazuje se više informacija o njemu te gumb Install, koji je potrebno kliknuti kako bi se započela instalacija proširenja.



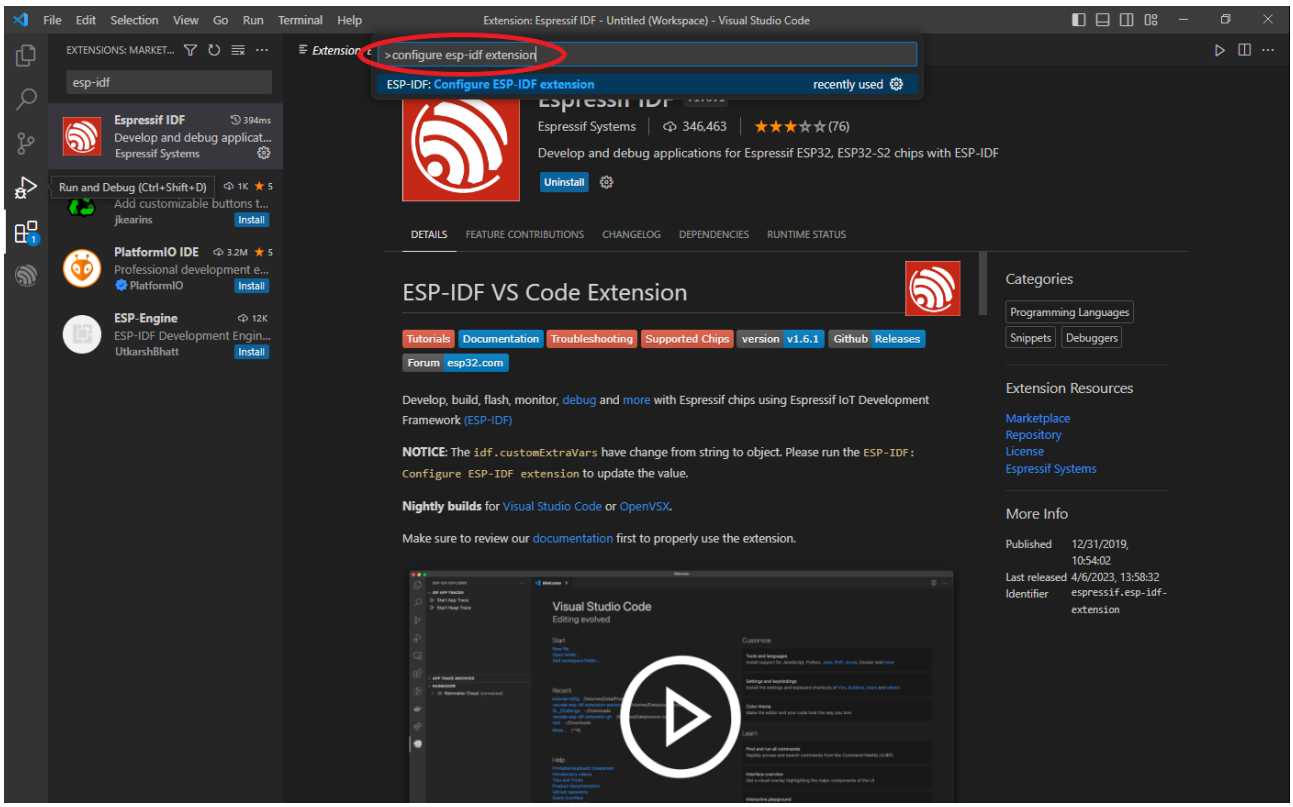

*Slika 1.1.54 Odabir i instalacija proširenja esp-idf*

Nakon preuzimanja i instalacije razvojnog okruženja potrebno ga je postaviti. U tu svrhu potrebno je u naredbenom retku softvera Visual Studio Code upisati configure esp-idf extension. Da bismo došli do naredbenog retka, trebamo izabrati izbornik View i u njemu podizbornik Command Palette….



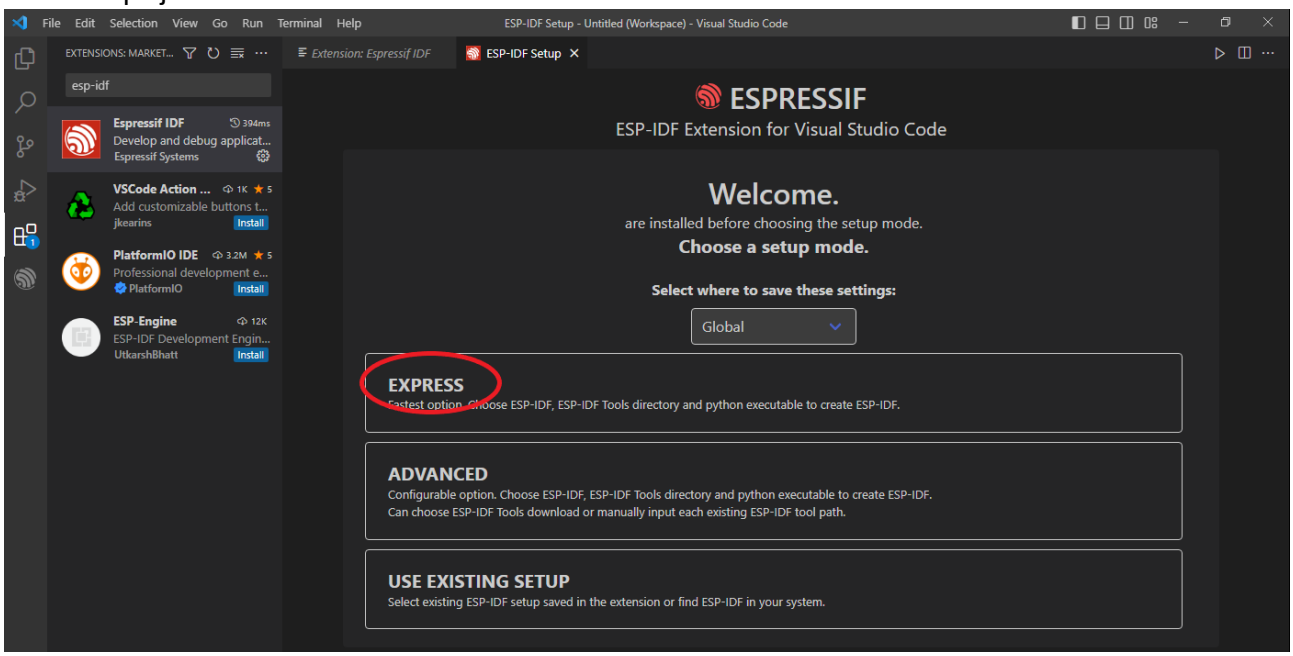

*Slika 1.1.55 Pokretanje naredbenog retka*

U otvorenom naredbenom retku potrebno je upisati configure esp-idf extension i pritisnuti enter.



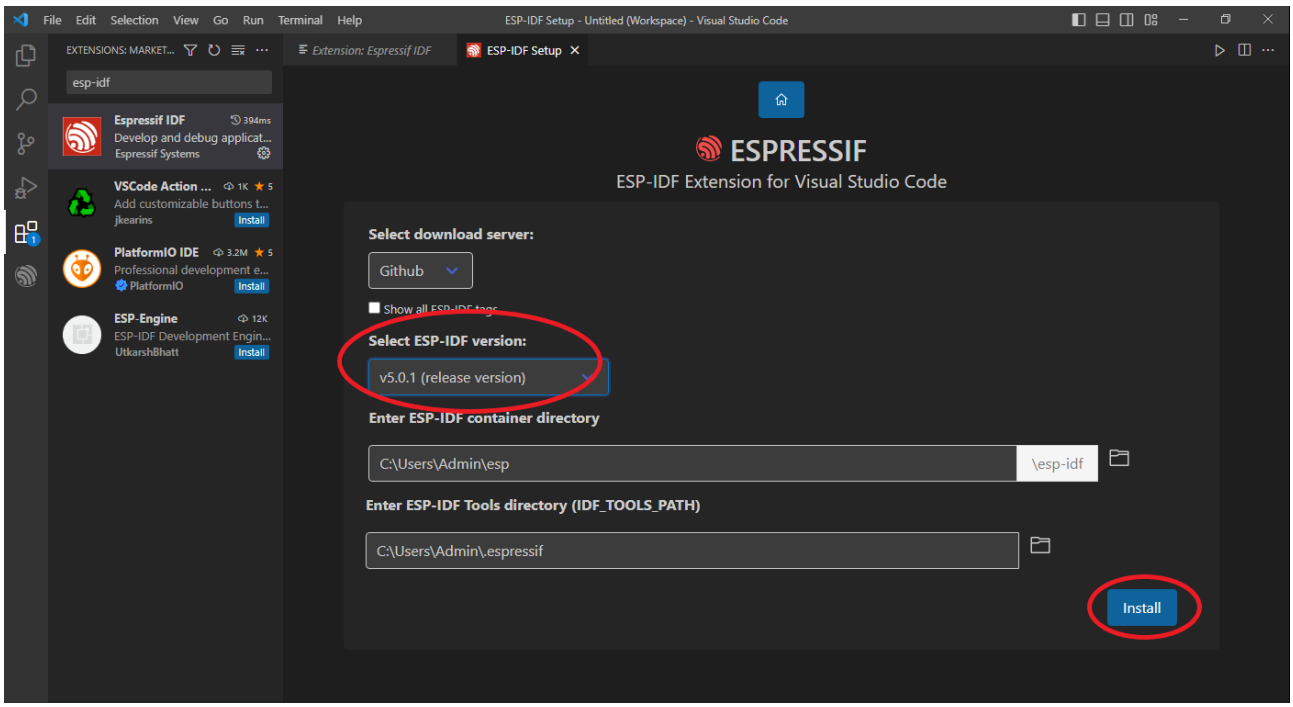

*Slika 1.1.56 Pokretanje naredbe za postavljanje razvojnog okvira*

Nakon pokretanja naredbe otvaraju se postavke razvojnog okvira ESP-IDF u kojima je potrebno izabrati opciju EXPRESS.



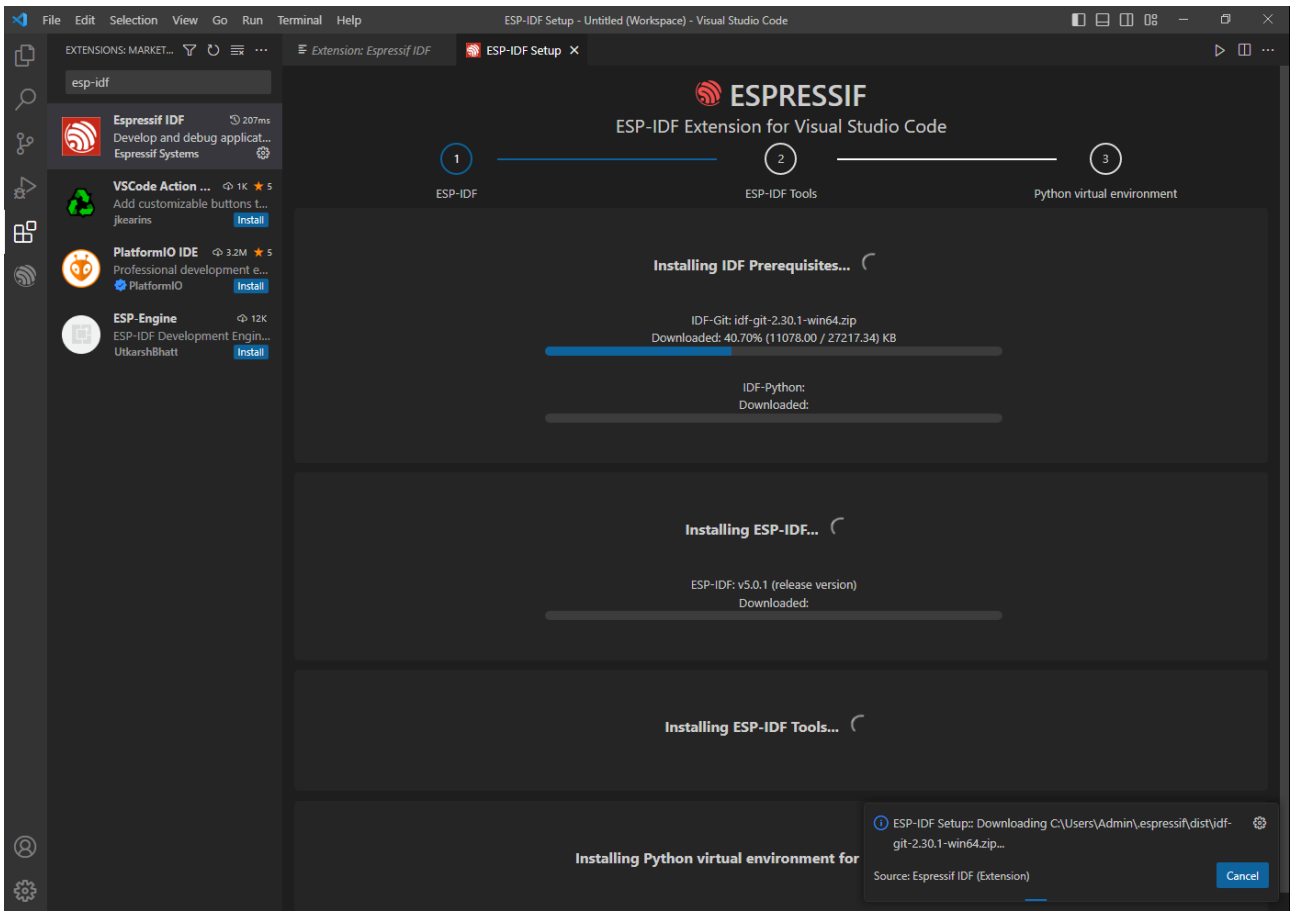

*Slika 1.1.57 Odabir opcije EXPRESS*

Odabirom opcije EXPRESS otvara se prozor na kojem trebamo odabrati verziju razvojnog okvira ESP-IDF (ili ostaviti da ga računalo lokalno pronađe ako je razvojni okvir instaliran ranije). Odabiremo zadnju dostupnu verziju te pokrećemo instalaciju.



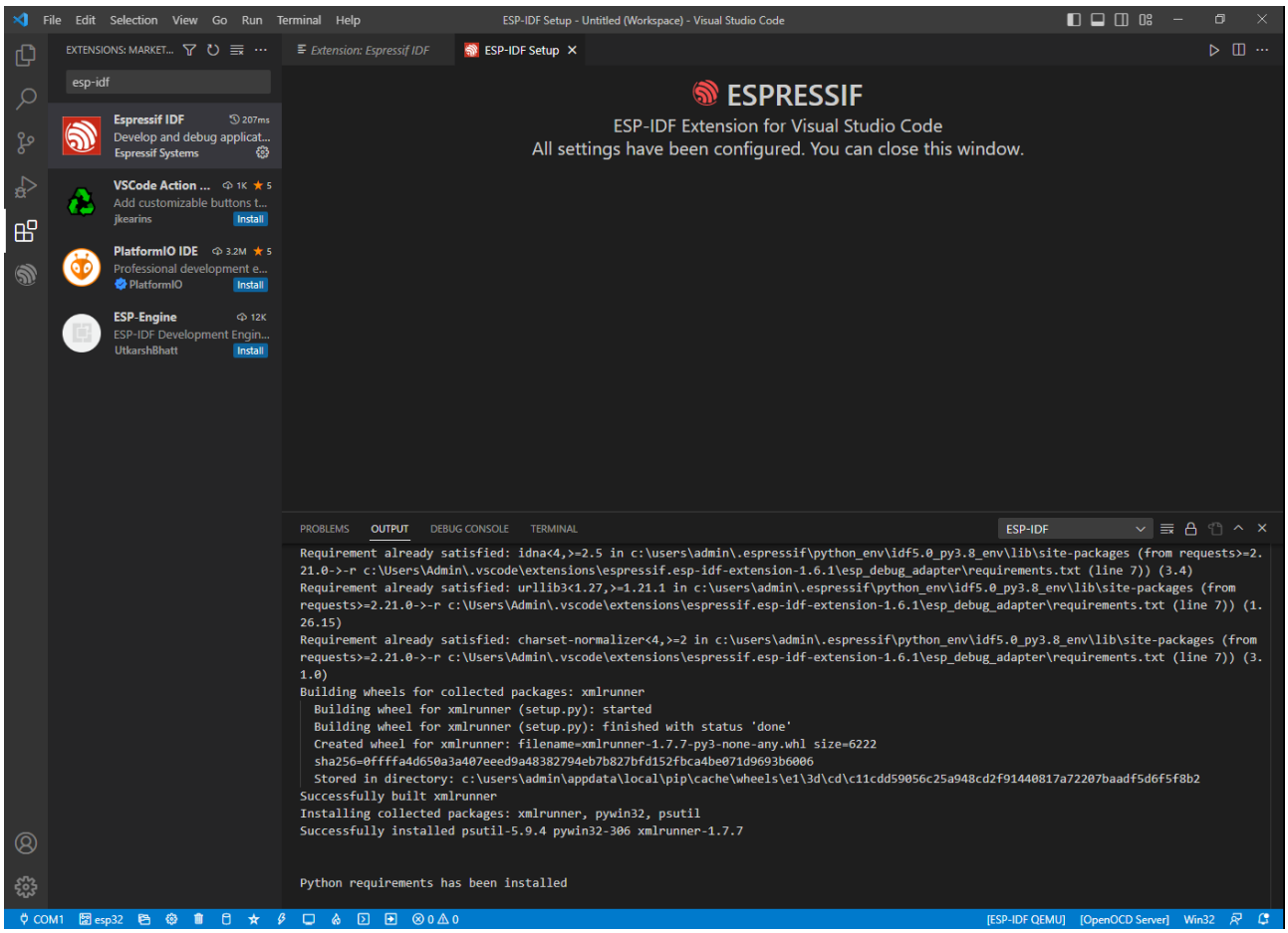

*Slika 1.1.58 Odabir verzije razvojnog okvira ESP-IDF i pokretanje instalacije*

Nakon pokretanja instalacije otvara se prozor koji prikazuje napredak instalacije.




*Slika 1.1.59 Napredak instalacije razvojnog okvira ESP-IDF*

Ako je instalacija uspješno provedena, prikazuje se prozor sa sljedeće slike.




*Slika 1.1.60 Uspješno provedena instalacija razvojnog okvira ESP-IDF*

## 1.1.4. Pitanja i zadaci

1. Visual Studio Code jest integralno razvojno okruženje za pisanje programskog koda. Pronađite i opišite barem tri karakteristike koje ima integralnog razvojno okruženje za pisanje programskog koda.
2. Osim integralnog razvojnog okruženja Visual Studio Code postoji još niz drugih integralnih razvojnih okruženja. Pronađite ih još barem tri te opišite njihove osnovne značajke (npr. kad su se pojavili, tko im je autor, kakvu licencu imaju, za što se najčešće koriste itd.)
3. Za programiranje sustava IoT koji se temelji na razvojnoj pločici ESP32-DevKitC-VIE koristimo Visual Studio Code u koji su uključena dva važna proširenja: proširenje za programiranje u programskom jeziku C i proširenje za razvojni okvir ESP-IDF. Proučite postoje li još koji načini za programiranje sustava IoT (koji se temelje na razvojnoj pločici ESP32-DevKitC-VIE, ali i koji se temelje na nekim drugim mikrokontrolerima, npr. Arduino).

## 1.1.5. Literatura i izvori


1. Espressif - Get Started, https://docs.espressif.com/projects/esp-idf/en/stable/esp32/get-started/index.html
2. How to run a C program in Visual Studio Code?, https://www.javatpoint.com/how-to-run-a-c-program-in-visual-studio-code
3. How to Write And Run C and C++ Code in Visual Studio Code, https://www.freecodecamp.org/news/how-to-write-and-run-c-cpp-code-on-visual-studio-code/
4. Visual Studio Code, https://code.visualstudio.com/

5. VS Code ESP-IDF Extension - Installation, https://github.com/espressif/vscode-esp-idf-extension/blob/master/docs/tutorial/install.md

## 1.2. Deklaracija varijabli i standardni ulaz/izlaz

Jedan od osnovnih koncepata koji je potrebno usvojiti kako bi se stekle kompetencije koje ima programer jesu varijable. Ova nastavna tema opisuje koja je svrha varijable, koje karakteristike ima varijabla, na koji se način varijabla stvara u programskom kodu te kako se koristi. Također je opisano na koji se način od korisnika traži unos nekog podatka koji se zatim sprema u varijablu te na koji se način korisniku prikazuje sadržaj varijable.

**Nakon usvajanja ove nastavne teme čitatelj će moći:**

1. Usvojiti pravila za definiranje identifikatora
2. Razlikovati tipove podataka s obzirom na zauzeće memorije i informacije koje mogu pohraniti
3. Deklarirati i inicijalizirati varijable prema informaciji koju je potrebno pohraniti
4. Pohraniti vrijednost u varijable unesene sa standardnog ulaza te prikaz vrijednosti varijable putem standardnog izlaza

### 1.2.1. Opis radnog zadatka

Potrebno je izraditi računalni program kojim se računa indeks temperature i vlažnosti zraka (engl. *temperature humidity index* – THI). Taj se indeks koristi kako bi se procijenila razina termalne udobnosti ljudi i životinja u nekom prostoru. Formula kojom se računa THI glasi:

$$THI = (1.8 \cdot T + 32) - \big((0.55 - 0.0055 \cdot RH) \cdot (1.8 \cdot T - 26)\big)$$

gdje je:
*THI* – indeks temperature i vlažnosti
*T* – temperatura zraka u celzijusima
*RH* – vlažnost zraka u postocima

Izračunani THI označava razinu termalne udobnosti prema sljedećoj tablici:

| THI | Razina termalne udobnosti |
|---|---|
| < 72 | UGODNO |
| 72 – 79 | PODNOŠLJIVO |
| 80 – 89 | NEUGODNO |
| 90 – 99 | VRLO NEUGODNO |
| > 99 | OPASNO PO ZDRAVLJE |

Korisniku je potrebno prikazati iznos indeksa temperature i vlažnosti kao (THI) te razine termalne udobnosti.

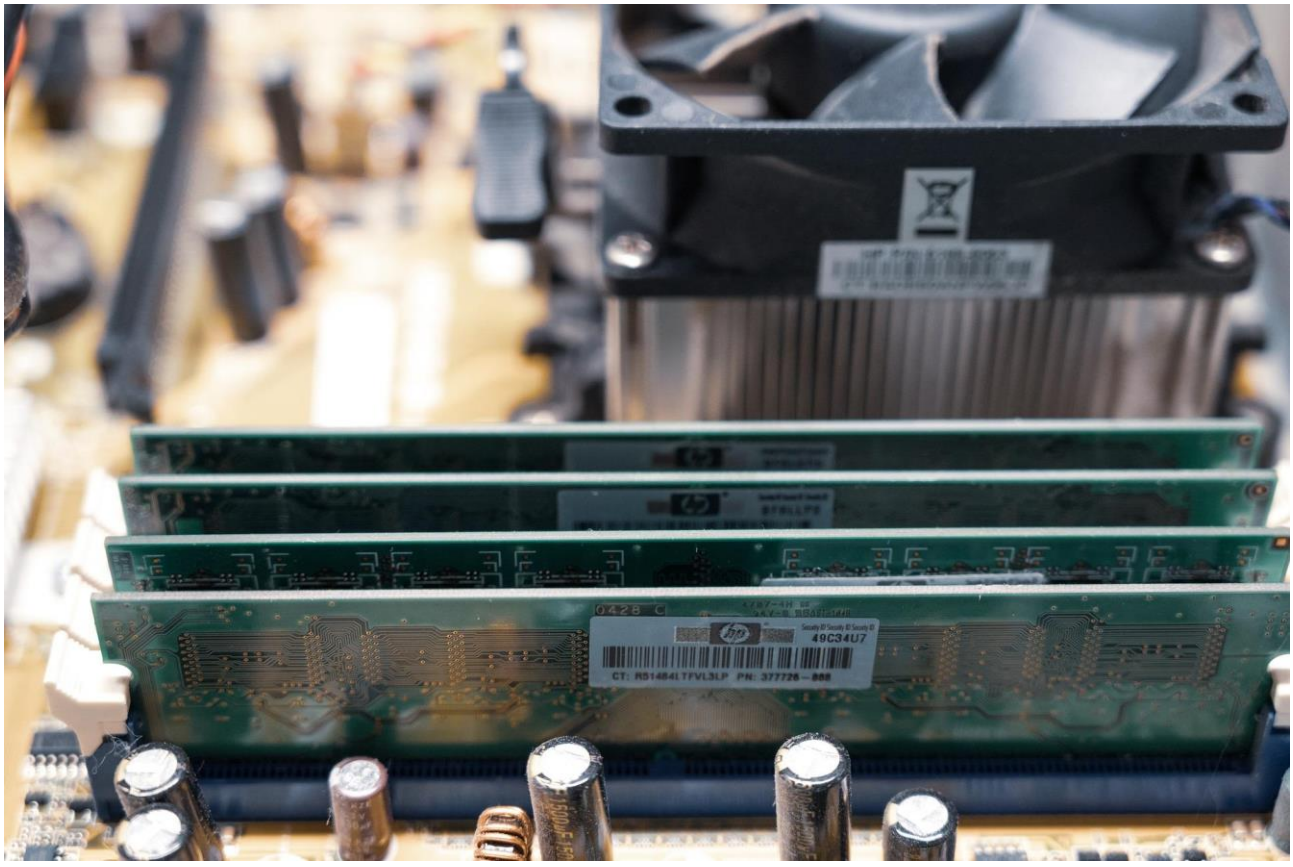
## 1.2.2. Osnovni koncepti

Iz opisa radnog zadatka jasno je da nam za izračun indeksa temperature i vlažnosti zraka trebaju dva podatka: temperatura zraka u celzijusima i vlažnost zraka u postocima. Ta dva podatka računalo treba dobiti od korisnika. Točnije, računalo će korisniku postaviti dva pitanja: „Kolika je temperatura zraka (u celzijusima)?" i „Kolika je vlažnost zraka (u postocima)?" Korisnik će preko tipkovnice unijeti dva tražena podatka koja računalo treba zapamtiti u svojoj memoriji kako bi ih iskoristio u izračunu indeksa temperature i vlažnosti zraka. Potom će računalo korisniku prikazati rezultat te će mu prikazati i tablicu s razinama termalne udobnosti.
Kako bi se izradio traženi program u programskom jeziku C, potrebno je utvrditi kako se računalu naređuje:

1. da korisniku prikaže poruke na zaslonu
2. da od korisnika primi podatke preko tipkovnice
3. da spremi podatke u memoriju
4. da izračuna novi podatak na temelju podataka spremljenih u memoriji računala.

### 1.2.2.1. Varijable

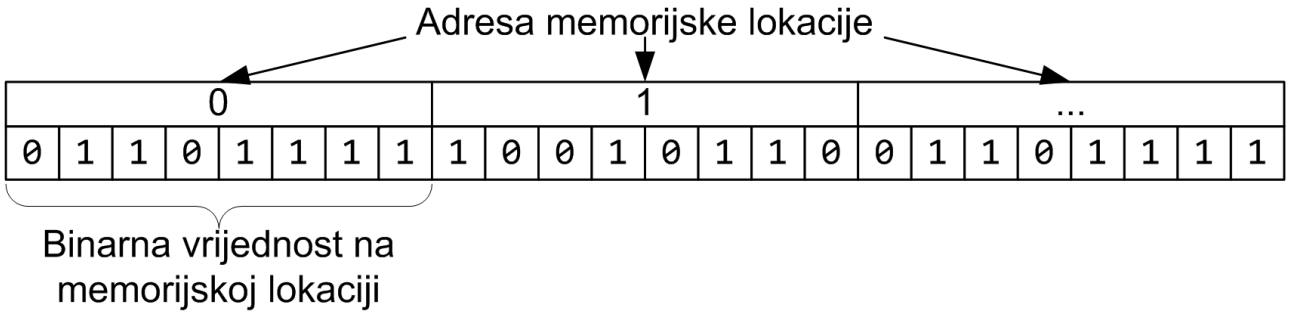
Računalni program sastoji se od niza naredbi koje računalo izvodi i time rješava neki problem. Često računalni programi obrađuju neke podatke pa ih zbog toga računalo treba privremeno spremiti u svoju radnu memoriju. Naredna slika prikazuje kako fizički izgleda radna memorija računala (RAM).



*Slika 1.2.1 Fizički izgled radne memorije računala*

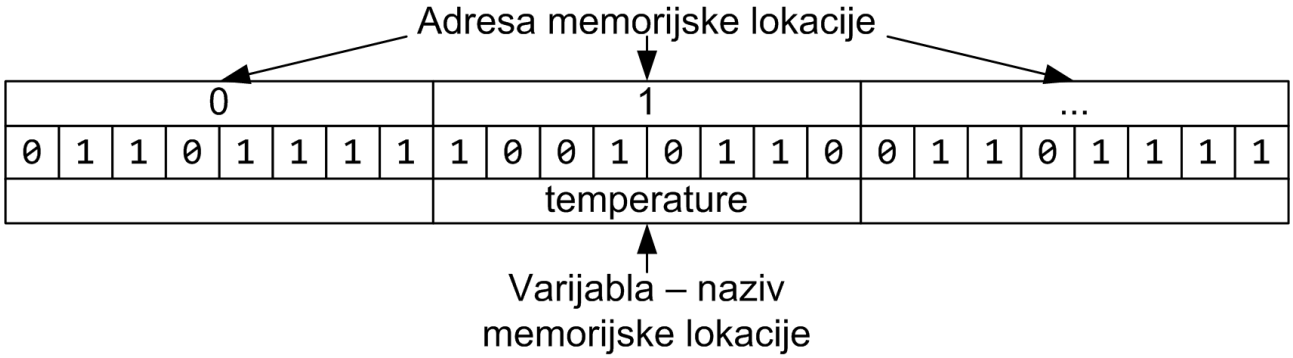
Računalo svoju radnu memoriju vidi kao traku koja je razdijeljena u manje segmente (memorijske lokacije) veličine 8 bitova (8 bit = 1 byte). Računalo sprema podatak na memorijsku lokaciju ili iz memorijske lokacije čita podatak koristeći adresu memorijske lokacije. Adresa memorijske lokacije jest broj koji ide od 0 pa do najvećeg broja koji može stati u adresni registar računala (npr. ako je adresni registar veličine 32 bita, tada se njime može adresirati ukupno $2^{32}$memorijskih lokacija,

odnosno 4.294.967.296 memorijskih lokacija od kojih je svaka velika 1 byte, što znači da se može adresirati ukupno 4 GB RAM memorije).




*Slika 1.2.2 Prikaz radne memorije kako je vidi računalo*

Postoje dva načina na koja se iz računalnog programa može pristupiti nekoj memorijskoj lokaciji i iz nje pročitati podatak ili u nju zapisati podatak. Jedan način jest preko adrese memorijske lokacije. Taj je način zahtjevan jer programeri trebaju znati adresu memorijske lokacije koja je predviđena za neki podatak. Drugi, jednostavniji način jest da programer sam dodijeli naziv nekoj memorijskoj lokaciji i da joj onda pristupa preko tog naziva. I tako dolazimo do pojma varijable. Varijabla je imenovani dio memorije računala (imenovana memorijska lokacija). Naredna sllika prikazuje memoriju računala u kojoj je memorijskoj lokaciji dodijeljen naziv `temperature`.




*Slika 1.2.3 Varijabla pridružena memorijskoj lokaciji*

Radnu memoriju računala koriste istovremeno različiti računalni programi. Zbog toga je računalu potrebno reći da određenu količinu memorijskih lokacija rezervira za naš računalni program, da tim rezerviranim memorijskim lokacijama pridruži određeni naziv te da bitove koji će se tamo čuvati interpretira na određeni način. Sve navedeno računalu se naređuje postupkom deklaracije varijabli. U programskom jeziku C deklaracija varijabli obavezno se radi prije prvog korištenja varijabli, a sastoji se od definiranja tipa podatka varijable i njezina naziva.
Na primjer, nije isto želimo li u memoriji računala čuvati brojeve u rasponu od 0 do 100 ili od 0 do 10000. Također nije isto čuvamo li cijele ili decimalne brojeve (npr. 2 ili 2,54267) ili npr. znak !.
Dakle, prije deklaracije varijabli važno je znati koje vrijednosti podatka očekujemo, a koje će se čuvati u varijabli (odnosno u memorijskoj lokaciji koja je rezervirana za tu varijablu).
O tipu podatka koji će se čuvati u varijabli ovisi i koliko će memorijskih lokacija računalo rezervirati za varijablu (vidjeti tablicu) te na koji će način podatke pretvarati u niz bitova koji će se spremati u rezervirane memorijske lokacije.

| Tip podatka | Veličina memorije | Interval podataka |
|---|---|---|
| `char` | 1 bajt | -128 do 127 |
| `unsigned char` | 1 bajt | 0 do 255 |

| | | |
|---|---|---|
| `short` | 2 bajta | -32,768 do 32,767 |
| `unsigned short` | 2 bajta | 0 do 65,535 |
| `int` | 4 bajta | -2,147,483,648 do 2,147,483,647 |
| `unsigned int` | 4 bajta | 0 do 4,294,967,295 |
| `long` | 8 bajtova | -9223372036854775808 do 9223372036854775807 |
| `unsigned long` | 8 bajtova | 0 do 18446744073709551615 |
| `float` | 4 bajta | 1.2E-38 do 3.4E+38<br>preciznost 6 decimalnih mjesta |
| `double` | 8 bajtova | 2.3E-308 to 1.7E+308<br>preciznost 15 decimalnih mjesta |
| `long double` | 10 bajtova | 3.4E-4932 to 1.1E+4932<br>preciznost 19 decimalnih mjesta |

Opća forma deklaracije varijable u programskom jeziku C glasi:

```
tip_podatka lista_naziva_varijabli;
```

Varijablama se vrijednost pridružuje preko operatora pridruživanja (=) čime se neka vrijednost sprema u memorijsku lokaciju koja je rezervirana za tu varijablu. Varijabli se može pridružiti neka konkretna vrijednost (npr. 22) ili neka izračunana vrijednost (vidjeti formulu u narednom primjeru). Slijedi nekoliko primjera deklaracije varijabli s pridruživanjem vrijednosti.

```
void main() {
    int temperature = 22, humidity = 45;
    float temperatureHumidityIndex;
    char roomLabel;
    roomLabel = 'A';
    temperatureHumidityIndex = (1.8 * temperature + 32) - ((0.55 - 0.0055 *
humidity) * (1.8 * temperature - 26));
}
```

Ako u varijabli trebamo čuvati podatak koji je cijeli broj (-100, -5, 0, 4, 234 itd.), tada možemo koristiti tipove podataka `char`, `int` i/ili `long`.
Ako u varijabli trebamo čuvati podatak koji je nenegativni cijeli broj (0, 1, 100, 233 itd.), tada možemo koristiti tipove podataka unsigned `char`, `unsigned int` i/ili `unsigned long`.
Ako u varijabli trebamo čuvati podatak koji predstavlja neki znak iz tablice ASCII (A, 1, <, !, b, ?, * itd.), tada možemo koristiti tip podatka `char`.
Ako u varijabli trebamo čuvati podatak koji je decimalni broj (-34,567, 0, 3, 6,78 itd.), tada možemo koristiti tipove podataka `float`, `double` i/ili `long double`.
Koji ćemo tip podatka koristiti ovisi i o intervalu mogućih vrijednosti koje podatak može poprimiti. Na primjer, ne možemo vrijednost 1000 spremiti u tip podatka `char` jer u taj tip podatka možemo spremiti samo vrijednosti unurar intervala -128 do 127.
Pri definiranju naziva varijabli vrijede sljedeća pravila:

- u nazivu varijable mogu se nalaziti slova engleske abecede (A – Z, a – z), znamenke (0 – 9) i znak podvlake (underscore _);
- naziv varijable može započeti samo slovom engleske abecede ili znakom podvlake. Ne može započeti znamenkom;

- u nazivu varijable nisu dopušteni razmaci;
- naziv varijable ne smije biti neka ključna riječ unutar programskog jezika C.

Naziv varijable osjetljiv je na veličinu slova (engl. *case-sensitive*) što znači da ako je varijabla deklarirana nazivom `temperature`, tada se ona ne može koristiti kao npr. `Temperature`.
Pri definiranju naziva varijabli poželjno je da počinju malim početnim slovom, da jasno označavaju smisao varijabli te da budu na engleskom jeziku.
Postoji vrsta varijabli kojoj se u računalnom programu vrijednost može pridružiti samo jednom (vrijednost joj se nakon toga više ne može mijenjati). Takve vrste varijabli nazivaju se konstante i dobra je praksa da im je naziv sastavljen od velikih slova.

| Ispravni nazivi varijabli | Neispravni nazivi varijabli |
|---|---|
| `firstNameLastName`<br><br>`FirstNameLastName`<br>(za naziv varijable poželjno da započinje malim slovo)<br><br>`email_1`<br><br>`MSG_NO_DATA`<br>(dobra je praksa jednu vrstu varijabli – konstante – nazivati sve velikim slovima) | `first Name Last Name`<br>(u nazivu razmak)<br><br>`firstName-LastName`<br>(u nazivu znak -)<br><br>`1.email`<br>(početak naziva znamenkom)<br><br>umnožak<br>(u nazivu znak ž)<br><br>`switch`<br>(ključna riječ u C-u) |

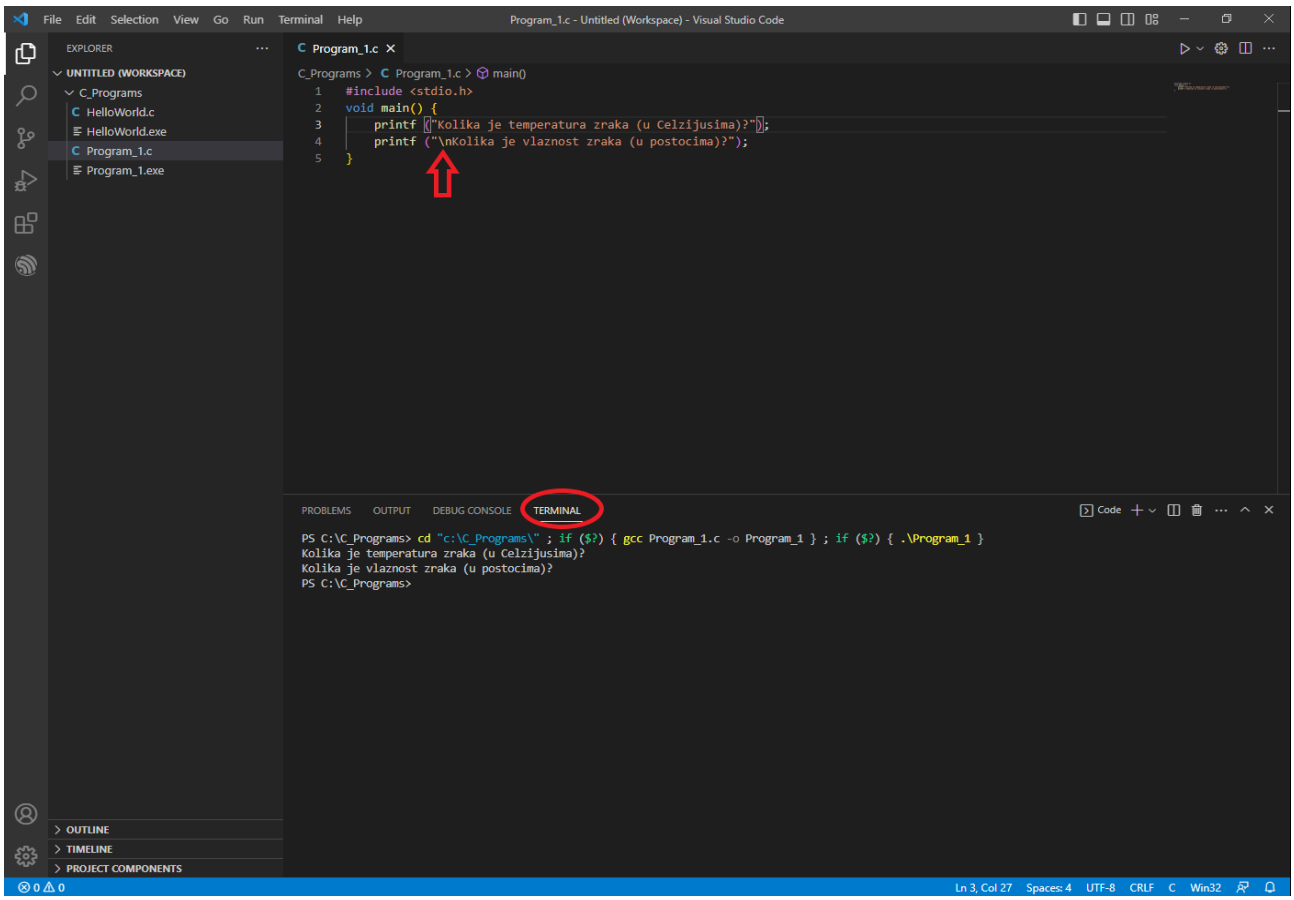
Može se primijetiti da sve dosada spomenute varijable mogu u istom trenutku čuvati samo jednu vrijednost podatka. Na primjer, varijabla `temperature` ne može u istom trenutku čuvati vrijednosti 19 i 22. Varijable koje u istom trenutku mogu čuvati samo jednu vrijednost zovu se jednostavne varijable.

### 1.2.2.2. Prikaz poruka na zaslonu

U programskom jeziku C naredba koja se koristi kako bi se prikazala poruka na zaslonu jest `printf`. Sljedeći računalni program ispisuje dvije poruke: “Kolika je temperatura zraka (u celzijusima)?” i “Kolika je vlaznost zraka (u postocima)?”:

```
#include <stdio.h>
void main() {
    printf ("Kolika je temperatura zraka (u Celzijusima)?");
    printf ("\nKolika je vlaznost zraka (u postocima)?");
}
```

Naredna slika prikazuje rezultat nakon pokretanja računalnog programa.




*Slika 1.2.4 Rezultat pokretanja računalnog programa s prikazom jednostavne poruke*

Kao što se može primijetiti, poruka koja se želi prikazati na zaslonu treba se nalaziti unutar dvostrukih navodnika (“). Također treba primijetiti i poseban znak \n koji označava prelazak u novi redak kod prikaza poruke.

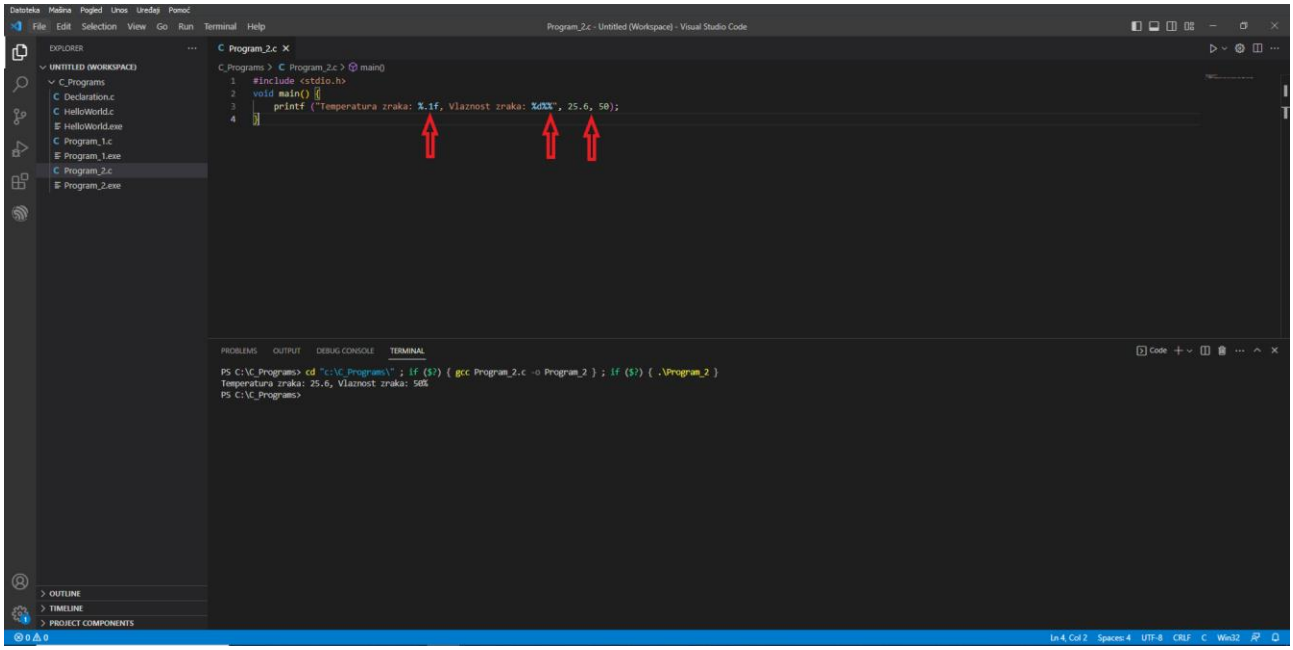
Poruka koja se prikazuje na zaslonu može na određenim mjestima sadržavati i neke podatke. Posebnim znakom % definira se na kojem se mjestu u poruci treba nalaziti koji podatak. Iza znaka % stavlja se znak koji odgovara tipu podatka koji se treba prikazati (konverzijski znak).

| Tip podatka | Konverzijski znak |
|---|---|
| char | %c |
| int | %d ili %i |
| unsigned int | %u |
| long | %ld ili %li |
| unsigned long | %lu |
| float | %f |
| double | %lf |

Sljedeći računalni program prikazuje poruku “Temperatura zraka: 25,6, Vlaznost zraka: 50%”:

```c
#include <stdio.h>
void main() {
    printf ("Temperatura zraka: %.1f, Vlaznost zraka: %d%%", 25.6, 50);
}
```

Naredna slika prikazuje rezultat ispisa podataka.



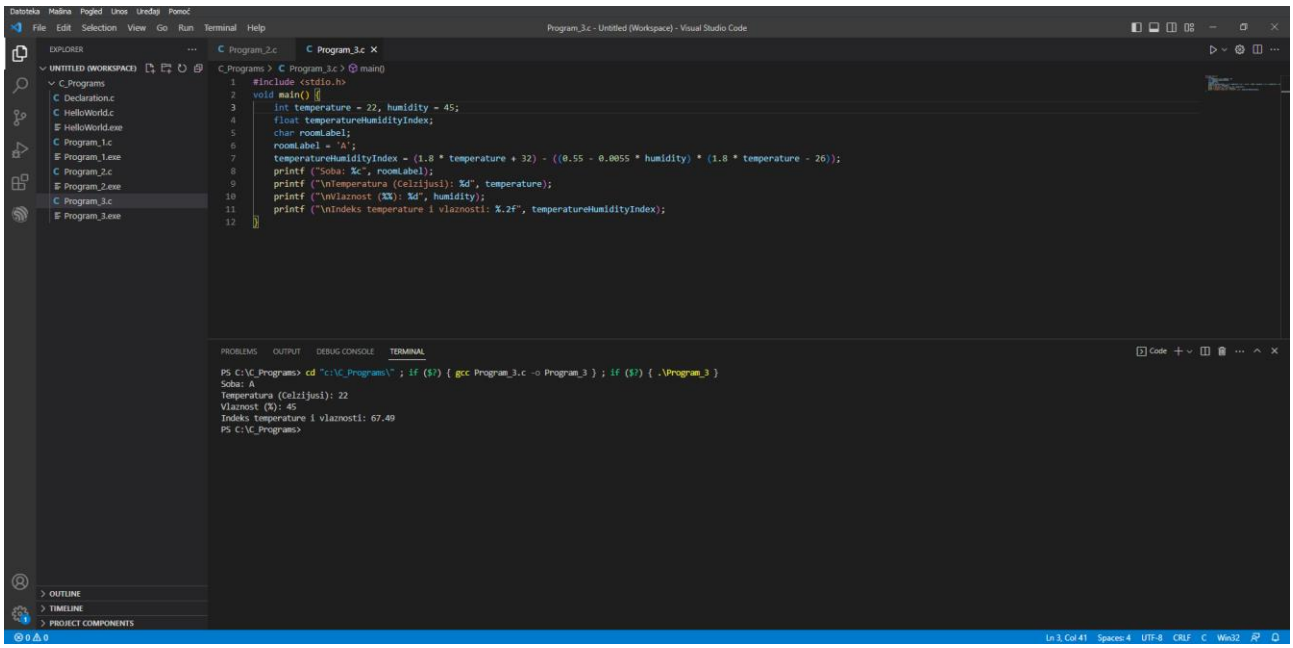

*Slika 1.2.5 Rezultat ispisa podataka*

Na prethodnoj slici može se primijetiti konverzijski znak `%.1f` koji označava da je potrebno prikazati decimalni podatak, ali samo s jednom znamenkom iza decimalne točke. Također se može primijetiti poseban znak `%%` koji označava da se želi ispisati znak %. Na kraju, iza poruke, mogu se vidjeti konkretne vrijednosti (25.6 i 50) koje će se ispisati na odgovarajućim mjestima unutar poruke.
Osim konkretnih vrijednosti, mogu se ispisati i vrijednosti koje su pridružene nekim varijablama. Sljedeći program prikazuje ispis:

```
Soba: A
Temperatura (Celzijusi): 22
Vlaznost (%): 45
Indeks temperature i vlaznosti: 67.49
```

```c
#include <stdio.h>
void main() {
    int temperature = 22, humidity = 45;
    float temperatureHumidityIndex;
    char roomLabel;
    roomLabel = 'A';
    temperatureHumidityIndex = (1.8 * temperature + 32) - ((0.55 - 0.0055 *
humidity) * (1.8 * temperature - 26));
    printf ("Soba: %c", roomLabel);
    printf ("\nTemperatura (Celzijusi): %d", temperature);
    printf ("\nVlaznost (%%): %d", humidity);
    printf ("\nIndeks temperature i vlaznosti: %.2f", temperatureHumidityIndex);
}
```

Naredna slika prikazuje rezultat izvođenja programa kojim se ispisuju vrijednosti varijabli koje su različitog tipa.




*Slika 1.2.6 Ispis podataka iz varijabli*

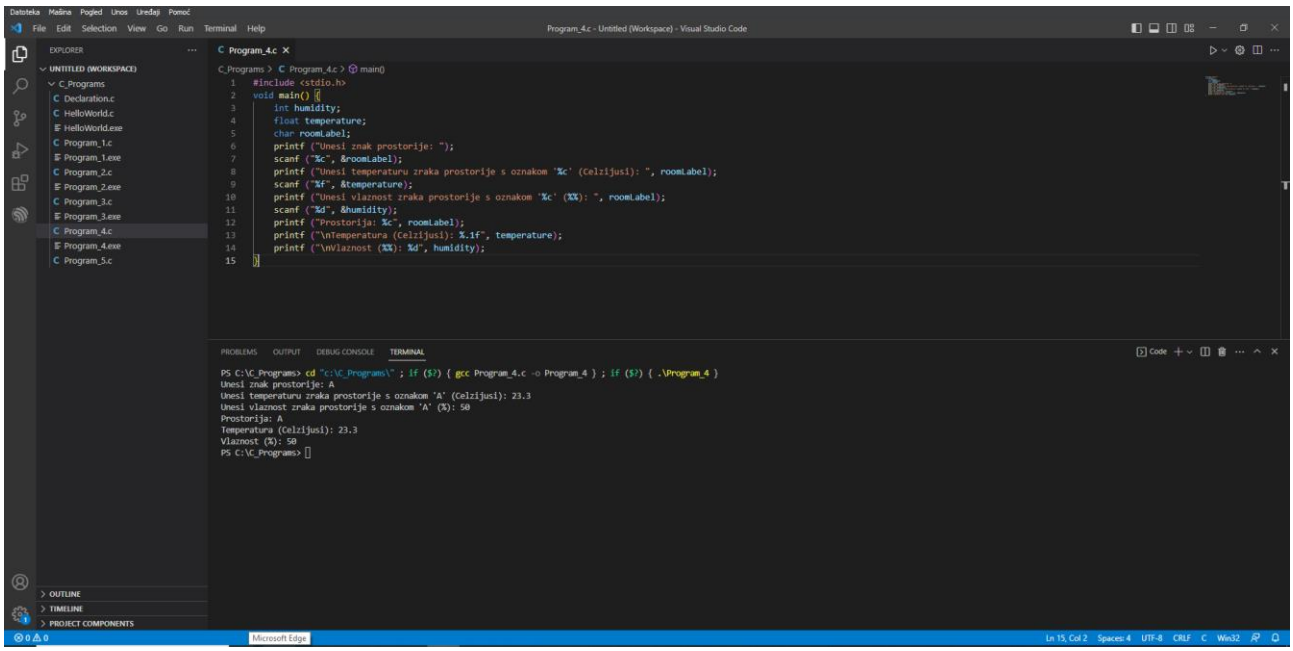
Na prethodnoj slici mogu se primijetiti različiti konverzijski znakovi koji se koriste kako bi se ispisale vrijednosti varijabli koje su različitog tipa podatka – `%c`, `%d`, `%.2f`.

### 1.2.2.3. Unos podataka preko tipkovnice

U programskom jeziku C naredba koja se koristi kako bi se korisniku omogućio unos podataka preko tipkovnice jest `scanf`. Sljedeći računalni program od korisnika traži unos oznake, temperaturu i vlažnost zraka neke prostorije. Potom se ispisuju uneseni podaci.

```
#include <stdio.h>
void main() {
    int humidity;
    float temperature;
    char roomLabel;
    printf ("Unesi znak prostorije: ");
    scanf ("%c", &roomLabel);
    printf ("Unesi temperaturu zraka prostorije s oznakom '%c' (Celzijusi): ",
roomLabel);
    scanf ("%f", &temperature);
    printf ("Unesi vlaznost zraka prostorije s oznakom '%c' (%%): ", roomLabel);
    scanf ("%d", &humidity);
    printf ("Prostorija: %c", roomLabel);
    printf ("\nTemperatura (Celzijusi): %.1f", temperature);
    printf ("\nVlaznost (%%): %d", humidity);
}
```

Naredna slika prikazuje rezultat izvođenja ovog programa.




*Slika 1.2.7 Unos podataka preko tipkovnice*

Iz programskog koda i prethodne slike može se uočiti da se naredbi `scanf` trebaju dostaviti dva podatka. Prvi podatak jest konverzijski znak koji treba odgovarati tipu podatka koji očekujemo od korisnika i koji će se spremiti u varijablu (u našem primjeru to su `%c`, `%f` i `%d`). Drugi podatak jest adresa varijable u koju želimo da računalo spremi podatak. Do adrese varijable dolazi se preko adresnog operatora & koji se nalazi ispred naziva varijable (u našem primjeru `&roomLabel`, `&temperature` i `&humidity`).

Također se može primijetiti da je u poruci za unos temperature i vlažnosti zraka, koja je na zaslonu prikazana preko printf naredbe, navedena i oznaka prostorije, odnosno ispisana je i vrijednost varijable `roomLabel`.

## 1.2.3. Rješenje radnog zadatka

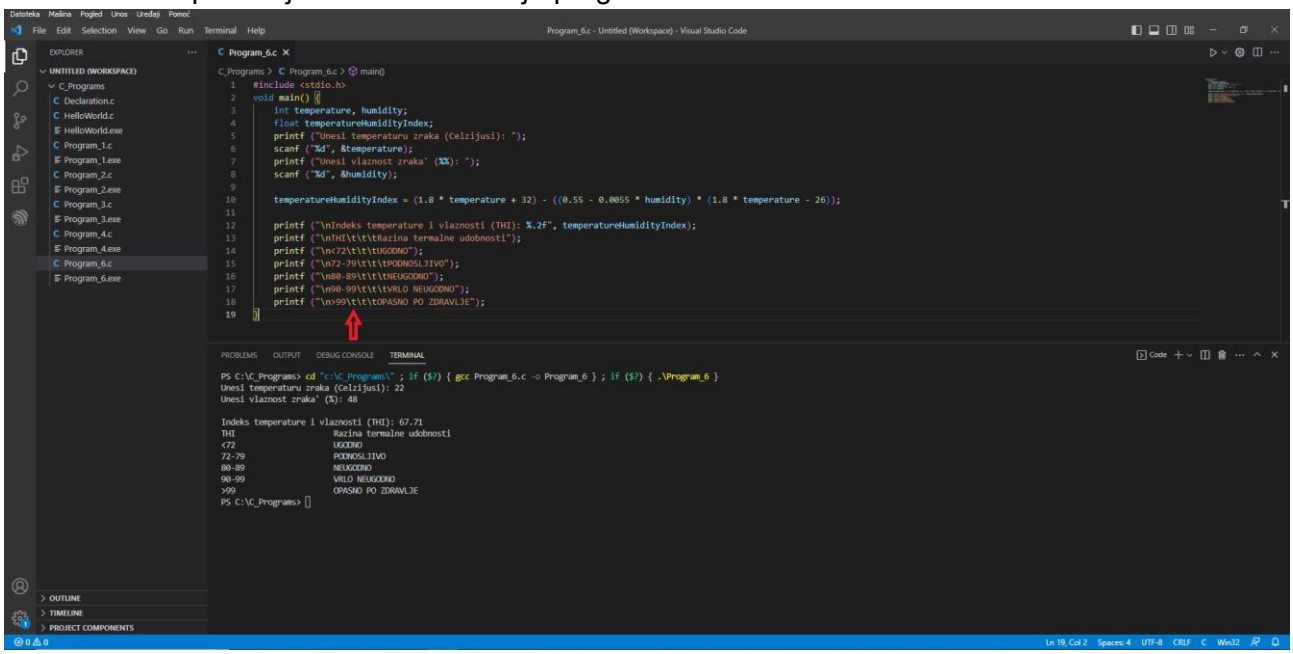
Iz opisa radnog zadatka mogu se identificirati koraci koje računalo treba izvesti kako bi se realizirao računalni program:

1. deklarirati varijable koje su potrebne u računalnom programu – potrebne su varijable u kojima će se čuvati iznos temperature i vlažnost zraka koje korisnik unosi preko tipkovnice te varijabla u kojoj će se čuvati izračunani indeks temperature i vlažnosti zraka. Za temperaturu i vlažnost zraka korisnik unosi cijeli pozitivan broj. Prema formuli za izračun indeksa temperature i vlažnosti zraka može se zaključiti da će rezultat biti decimalni broj;
2. korisniku prikazati odgovarajuću poruku (pitanje) na koju će dati odgovor unosom preko tipkovnice. Prvo će biti prikazana poruka “`Unesi temperaturu zraka prostorije (Celzijusi): `” nakon koje će računalo čekati da korisnik unese vrijednost. Potom će biti prikazana poruka “`Unesi vlaznost zraka prostorije (%): `” te će računalo opet čekati unos podatka preko tipkovnice. Obje unesene vrijednosti spremaju se u za to predviđene varijable;
3. izračunati indeks temperature i vlažnosti zraka prema formuli koristeći vrijednosti koje su sačuvane u odgovarajućim varijablama;
4. korisniku prikazati poruku “`Indeks temperature i vlaznosti (THI): `” te na dvije decimale prikazati izračunani indeks;
5. korisniku prikazati tablicu s razinama termalne udobnosti.

Na temelju tih identificiranih koraka i prethodno opisanih osnovnih koncepata koji se odnose na varijable, prikaz poruka na zaslonu i unos podataka preko tipkovnice, može se izraditi računalni program.

```
#include <stdio.h>
void main() {
    int temperature, humidity;
    float temperatureHumidityIndex;
    printf ("Unesi temperaturu zraka (Celzijusi): ");
    scanf ("%d", &temperature);
    printf ("Unesi vlaznost zraka' (%%): ");
    scanf ("%d", &humidity);

    temperatureHumidityIndex = (1.8 * temperature + 32) - ((0.55 - 0.0055 *
humidity) * (1.8 * temperature - 26));

    printf ("\nIndeks temperature i vlaznosti (THI): %.2f",
temperatureHumidityIndex);
    printf ("\nTHI\t\t\tRazina termalne udobnosti");
    printf ("\n<72\t\t\tUGODNO");
    printf ("\n72-79\t\t\tPODNOSLJIVO");
    printf ("\n80-89\t\t\tNEUGODNO");
    printf ("\n90-99\t\t\tVRLO NEUGODNO");
    printf ("\n>99\t\t\tOPASNO PO ZDRAVLJE");
}
```

Naredna slika prikazuje rezultat izvođenja programa.




*Slika 1.2.8 Rješenje radnog zadatka izračuna indeksa temperature i vlažnosti zraka*

Može se uočiti da su deklarirane tri varijable: `temperature`, `humidity` i `temperatureHumidityIndex`. Prve su dvije varijable tipa podatka `int` i koriste se za spremanje vrijednosti koje će korisnik unijeti preko tipkovnice. Treća varijabla jest tipa podatka `float` i koristi se za spremanje izračunane vrijednosti indeksa temperature i vlažnosti zraka.
Kod ispisa tablice razine termalne udobnosti može se uočiti poseban znak `\t` koji označava ispis tabulatornih razmaka.

### 1.2.4. Pitanja i zadaci

1. Potrebno je napisati računalni program kojim se od korisnika traži unos decimalne vrijednosti duljine u metrima. Nakon toga program na zaslonu prikazuje vrijednost duljine u milimetrima, centimetrima, metrima i kilometrima na dvije decimale.

2. Potrebno je napisati računalni program kojim se od korisnika traži unos vrijednosti duljine kao cijeloga pozitivnog broja u milimetrima, centimetrima i metrima. Nakon toga program na zaslonu prikazuje vrijednost duljine u metrima prikazan na dvije decimale. Paziti na efekt cjelobrojnog dijeljenja.

3. Potrebno je napisati računalni program kojim se od korisnika traži unos decimalne vrijednosti vremena u satima. Nakon toga program na zaslonu prikazuje vrijednost vremena u sekundama, minutama i satima na dvije decimale.

4. Potrebno je napisati računalni program kojim se od korisnika traži unos vrijednosti vremena kao cijelog broja u sekundama, minutama i satima. Nakon toga program na zaslonu prikazuje vrijednost vremena u satima prikazan na dvije decimale. Paziti na efekt cjelobrojnog dijeljenja.

5. Potrebno je napisati računalni program kojim se radi konverzija unesene decimalne vrijednosti temperature izražene u celzijusima u vrijednost izraženu u fahrenheitima.

6. U varijablu se treba spremiti sljedeći podatak: #. Kojeg tipa podatka treba biti ta varijabla?
    a. main
    b. float
    c. char
    d. double
    e. long

7. Podatak # može se spremiti u varijablu koja je tipa podatka [main, float, char, double, long].

8. U varijablu se treba spremiti sljedeći podatak: 1000. Kojeg tipa podatka može biti ta varijabla?
    a. int
    b. float
    c. char
    d. double
    e. scanf

9. U varijablu se treba spremiti sljedeći podatak: -2670. Kojeg tipa podatka može biti ta varijabla?
    a. unsigned int
    b. float
    c. char
    d. printf
    e. long

10. U varijablu se treba spremiti sljedeći podatak: 100,245. Kojeg tipa podatka može biti ta varijabla?
    a. int

b. float
c. char
d. printf
e. long

11. Označite ispravno imenovane varijable.
    a. int
    b. float
    c. year
    d. user_selection
    e. 1user

12. Označite ispravno imenovane varijable.
    a. printf
    b. scanf
    c. page_1
    d. income-outcome
    e. _user

13. Označite ispravno imenovane varijable.
    a. PAGE_NUM
    b. scanf
    c. Temp
    d. primaryEmail
    e. size_1*size_2

14. Označite tipove podataka koji zahtijevaju veličinu memorije od 1 bajt.
    a. printf
    b. int
    c. char
    d. short
    e. unsigned char

15. Označite tipove podataka koji zahtijevaju veličinu memorije od 2 bajta.
    a. scanf
    b. int
    c. char
    d. short
    e. unsigned short

16. Označite tipove podataka koji zahtijevaju veličinu memorije od 4 bajta.
    a. printf
    b. int
    c. char
    d. float
    e. unsigned int

17. Označite tipove podataka koji zahtijevaju veličinu memorije od 8 bajtova.

a. printf
b. double
c. long
d. float
e. unsigned long

18. Koju veličinu memorije zahtijeva određeni tip podatka?
    a. 1 bajt zahtijeva tip podatka [int, char, short, double].
    b. 2 bajta zahtijevaju tip podatka [int, char, short, double].
    c. 4 bajta zahtijevaju tip podatka [int, char, short, double].
    d. 8 bajtova zahitjeva tip podatka [int, char, short, double].

19. U programskom jeziku C dvije varijable koje imaju isti naziv, ali pisan različitom veličinom slova u biti su [iste, različite] varijable.

20. U programskom jeziku C varijabla naziva page i varijabla naziva Page u biti su iste varijable.
    a. Točno
    b. Netočno

21. Deklaracijom varijable u programskom jeziku C definiraju se njezin naziv i tip podatka.
    a. Točno
    b. Netočno

22. Deklaracijom varijable u programskom jeziku C postavlja se njezina vrijednost.
    a. Točno
    b. Netočno

23. Postupak kojim se u programskom jeziku C definira tip podatka i naziv varijable zove se [programiranje varijable, deklaracija varijable, ispis varijable, postavljanje varijable].

24. Je li u programskom jeziku C ovakva dodjela vrijednosti varijable ispravna: 26 = page;?
    a. Da
    b. Ne

25. Je li u programskom jeziku C ovakva dodjela vrijednosti varijable ispravna: size = 300;?
    a. Da
    b. Ne

26. Potrebno je deklarirati varijablu naziva sensorNumber čija će vrijednost biti neki pozitivni cijeli broj (broj veći od 0 ili jednak 0). Označite koja je deklaracija varijable ispravna:
    a. int sensorNumber;
    b. float sensorNumber;
    c. unsigned int sensorNumbe;
    d. long sensorNumber;
    e. short sensorNumber;

27. Potrebno je deklarirati varijablu naziva sensorValue čija će vrijednost biti neki decimalni broj. Označite koja je deklaracija varijable ispravna:

a. int sensorNumber;
b. float sensorNumber;
c. unsigned int sensorNumbe;
d. long sensorNumber;
e. short sensorNumber;

28. U varijablu average potrebno je spremiti izračunani prosjek vrijednosti koje se čuvaju u varijablama value1, value2 i value3. Označite koji je izračun ispravan:
    a. average = value1 + value2 + value3 / 3;
    b. average = (value1 + value2) + value3 / 3;
    c. average = (value1 + value2 + value3) / 3;
    d. average = value1 + (value2 + value3) / 3;
    e. average = value1 + value2 + (value3) / 3;

29. Naredba za prikaz poruke na zaslonu jest?
    a. scanf
    b. int
    c. printf
    d. char
    e. double

30. Koji se konverzijski znak koristi za koji tip podatka u ispisu na zaslon?
    a. %c za tip podatka [int, char, float]
    b. %d za tip podatka [int, char, float]
    c. %f za tip podatka [int, char, float]

31. Za ispis tipa podatka char na zaslon se koristi sljedeći konverzijski znak [%c, %d, %f, %g].

32. Za ispis tipa podatka int na zaslon se koristi sljedeći konverzijski znak [%c, %d, %f, %g].

33. Za ispis tipa podatka float na zaslon se koristi sljedeći konverzijski znak [%c, %d, %f, %g].

34. Na zaslon je potrebno prikazati poruku “Primjer sustava IoT.”. Označite koja je naredba ispravna:
    a. printf (“”,Primjer sustava IoT.);
    b. printf (Primjer sustava IoT.);
    c. printf (“Primjer sustava IoT.”);
    d. printf (Primjer sustava IoT., “”);
    e. printf (“”, “Primjer sustava IoT.”);

35. Na zaslon je potrebno prikazati poruku „Izmjerena temperatura: 22.6”. Vrijednost 22.6 nalazi se u varijabli tempValue. Označite koja je naredba ispravna:
    a. printf (“Izmjerena temperatura %.1d”, tempValue);
    b. printf (“Izmjerena temperatura: %f “, tempValue);
    c. printf (“Izmjerena temperatura: %.1f “, tempValue);
    d. printf (“Izmjerena temperatura: %.1c “, tempValue);
    e. printf (“Izmjerena temperatura: %%.1f “, tempValue);

36. Na zaslon je potrebno prikazati poruku “Zgrada: A”. Vrijednost A nalazi se u varijabli buildingCode. Označite koja je naredba ispravna:
    a. printf (“Zgrada: %d”, buildingCode);
    b. printf (“Zgrada: %f”, buildingCode);
    c. printf (“Zgrada: %k”, buildingCode);
    d. printf (“Zgrada: %c”, buildingCode);
    e. printf (“Zgrada: %h”, buildingCode);

37. Za prelazak u novi red kod prikaza poruka na zaslonu se koristi poseban znak [\t, \w, \n, %n, %t, %w].

38. Na zaslonu je potrebno prikazati poruku “Kamatna stopa: 10%”. Vrijednost 10 nalazi se u varijabli percentage. Označite koja je naredba ispravna:
    a. printf (“Kamatna stopa: %d%”, percentage);
    b. printf (“Kamatna stopa: %d%%”, percentage);
    c. printf (“Kamatna stopa: %%d%”, percentage);
    d. printf (“Kamatna stopa: d%”, percentage);
    e. printf (“Kamatna stopa: %%d”, percentage);

39. Na zaslonu je potrebno prikazati poruku “Cijena: 100 eur, PDV: 25 eur.”. Vrijednost 100 nalazi se u varijabli price, a vrijednost 25 u varijabli vat. Označite koja je naredba ispravna:
    a. printf (“Cijena: %d eur, PDV: %d eur.”, vat, price);
    b. printf (“Cijena: %d eur, PDV: %d eur.”, price, vat);
    c. printf (“Cijena: d% eur, PDV: d% eur.”, vat, price);
    d. printf (“Cijena: d% eur, PDV: d% eur.”, price, vat);
    e. printf (“Cijena: %%d eur, PDV: %%d eur.”, price, vat);

40. Naredba za unos podataka preko tipkovnice jest?
    a. scanf
    b. int
    c. printf
    d. char
    e. double

41. Koji se konverzijski znak koristi za koji tip podatka kod unosa preko tipkovnice?
    a. %c za tip podatka [int, char, float].
    b. %d za tip podatka [int, char, float].
    c. %f za tip podatka [int, char, float].

42. Za unos tipa podatka char preko tipkovnice koristi se sljedeći konverzijski znak [%c, %d, %f, %g].

43. Za unos tipa podatka int preko tipkovnice koristi se sljedeći konverzijski znak [%c, %d, %f, %g].

44. Za unos tipa podatka float preko tipkovnice koristi se sljedeći konverzijski znak [%c, %d, %f, %g].

45. Od korisnika se traži unos godine rođenja preko tipkovnice koja se treba spremiti u varijablu year_of_birth. Koja je naredba ispravna?
    a. scanf(“&d”, year_of_birth);
    b. scanf(“%d”, %year_of_birth);
    c. scanf(“%d”, &year_of_birth);
    d. scanf(“&d”, %year_of_birth);
    e. scanf(“%d”, year_of_birth);

### 1.2.5. Literatura i izvori


1. C Programming Language Tutorial, https://www.javatpoint.com/c-programming-language-tutorial
2. C Tutorial, https://www.tutorialspoint.com/cprogramming/index.htm
3. C Tutorial, https://www.w3schools.in/c-tutorial
4. Jakupović, A.; Šuman, S.: Osnove programiranja, https://www.veleri.hr/sites/default/files/2021-07/osnove_programiranja_konacna_verzija_online_verzija.pdf


## 1.3. Upravljanje tijekom programa

Računalo program izvodi slijedno – jednu naredbu za drugom – i to se zove tijek programa. Često u pisanju programa postoji potreba da se računalu naredi da jednu skupinu naredbi izvede samo onda kada su zadovoljeni određeni uvjeti. Tada se govori o uvjetnom izvođenju naredbi, odnosno o upravljanju tijekom programa. Ova nastavna tema opisuje na koji se način u programskom jeziku C implementira uvjetovano izvođenje skupine naredbi. Obradit će se grananje i ponavljanje naredbi u programu.

**Nakon usvajanja ove nastavne teme čitatelj će moći:**
1. upravljati tijekom programa korištenjem naredbi grananja if, if else i if else if
2. upravljati tijekom programa korištenjem naredbe grananja switch case
3. upravljati tijekom programa korištenjem petlji for, while i do while te primijeniti naredbe break i continue
4. organizirati programski kôd korištenjem ugniježđenih petlji

### 1.3.1. Opis radnog zadatka

Potrebno je izraditi računalni program kojim se računa indeks temperature i vlažnosti zraka (engl. *Temperature Humidity Index* – THI). Taj se indeks koristi kako bi se procijenila razina termalne udobnosti ljudi i životinja u nekom prostoru. Formula kojom se računa THI glasi:

$$THI = (1.8 \cdot T + 32) - \big((0.55 - 0.0055 \cdot RH) \cdot (1.8 \cdot T - 26)\big)$$

gdje je:
*THI* – indeks temperature i vlažnosti
*T* – temperatura zraka u celzijusima
*RH* – vlažnost zraka u postocima

Izračunani THI označava razinu termalne udobnosti prema sljedećoj tablici:

| THI | Razina termalne udobnosti |
|---|---|

| | |
|---|---|
| < 72 | UGODNO |
| 72 – 79 | PODNOŠLJIVO |
| 80 – 89 | NEUGODNO |
| 90 – 99 | VRLO NEUGODNO |
| > 99 | OPASNO PO ZDRAVLJE |

Korisnika je potrebno pitati koliko unosa želi napraviti (maksimalni broj unosa jest 10). Nakon toga potrebno ga je toliko puta pitati za unos temperature i vlažnosti zraka. Kod svakog unosa potrebno je provjeriti ispravnost unesenih podataka (temperatura treba biti u intervalu od 0 do 50, a vlažnost zraka u intervalu od 0 do 100). Nakon završetka unosa temperature i vlažnosti zraka potrebno je prikazati iznos indeksa temperature i vlažnosti (THI) te razine termalne udobnosti.

### 1.3.2. Osnovni koncepti

Iz opisa zadatka može se vidjeti da računalo treba provesti provjeru ispravnosti unesenih podataka te u skladu s time ili ispisati poruku o pogrešci (jedna skupina naredbi) i korisnika ponovno pitati za unos ili nastaviti dalje s izvođenjem programa. Znači postoji potreba za grananjem i ponavljanjem naredbi.
Dvije su temeljne naredbe kojima se u programskom jeziku C implementira grananje: `if – else if – else` naredba i `switch – case` naredba.
Opći oblik naredbe `if – else if – else` glasi:

```
...
naredbaX
if (logički_izraz_1) {
      blok_naredbi_1
}
else if (logički_izraz_2){
      blok_naredbi_2
}
...
else if (logički_izraz_n){
      blok_naredbi_n
}
else {
      blok_naredbi_x
}
naredbaY
...
```

U prikazanom općem obliku naredbe `if – else if – else` treba primijetiti logički izraz. Logički izraz, kada ga računalo izračuna, može poprimiti samo dvije vrijednosti: istina (engl. *true*) i neistina (engl. *false*). Logički izrazi grade se iz relacijskih i logičkih operatora.
Relacijski operator ili operator uspoređivanja koristi se kako bi se međusobno usporedile vrijednosti dva podatka. Postoji šest relacijskih operatora u programskom jeziku C:

| **Relacijski operator** | **Značenje** |
|---|---|
| a < b | je li a manje od b |

| | |
|---|---|
| a <= b | je li a manje ili jednako b |
| a > b | je li a veće od b |
| a >= b | je li a veće ili jednako b |
| a == b | je li a jednako b |
| a != b | je li a različito od b |

Posebnu pažnju treba obratiti na operator jednakosti (==) i operator različitosti (!=). Operator jednakosti (==) nije isti kao i operator pridruživanja (=).
Logički operatori koriste se u povezivanu više logičkih izraza. Postoje tri logička operatora u programskom jeziku C: negacija (!), logiki I (&&) i logički ILI (||). Za logičke operatore važne su tablice istine:

**NEGACIJA**

| Logička vrijednost (L) | Negacija logičke vrijednosti (!L) |
|---|---|
| 0 | 1 |
| 1 | 0 |

**LOGIČKI I**

| Logička vrijednost (L1) | Logička vrijednost (L2) | Logički operator I (L1 && L2) |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

**LOGIČKI ILI**

| Logička vrijednost (L1) | Logička vrijednost (L2) | Logički operator ILI (L1 \|\| L2) |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Nakon linije 2 iz općeg oblika naredbe `if – else if – else` izvodi se linija 3. Ako je izračunana vrijednost izraza `logički_izraz_1` istina, tada će računalo izvesti `blok_naredbi_1` i nakon toga liniju 16 (znači preskočit će sve ostale blokove naredbi). Ako je pak `logički_izraz_1` neistina, tada će računalo izračunati `logički_izraz_2` te ako je rezultat istina, bit će izveden `blok_naredbi_2` i nakon njega linija 16 (ponovno se svi ostali blokovi naredbi preskaču). Ako niti jedan logički izraz nije istinit, tada će se izvesti `blok_naredbi_x` i nakon njega linija 16 (treba primijetiti da `else` dio nema logičkog izraza).
Sljedeći program prikazuje primjenu naredbe `if – else if – else` u primjeru provjere je li korisnik unesao ispravnu temperaturu i vlažnost zraka – temperatura (0 – 50), vlažnost zraka (0 – 100).

```
#include <stdio.h>
void main()
{
    int temperature, humidity;
    printf("Unesi temperaturu zraka (Celzijusi): ");
    scanf("%d", &temperature);
    printf("Unesi vlaznost zraka (%%): ");
    scanf("%d", &humidity);
    if (temperature < 0 || temperature > 50)
```

```
    {
        printf("Temperatura zraka treba biti u intervalu od 0 do 50 Celzijusa.");
    }
    else if (humidity < 0 || humidity > 100)
    {
        printf("Vlaznost zraka treba biti u intervalu od 0 do 100 %%.");
    }
    else
    {
        printf("Unos temperature i vlaznosti zraka je ispravan.");
    }
}
```

Rezultat programa prikazuje sljedeća slika.

```
PS C:\C_Programs> cd "c:\C_Programs\" ; if ($?) { gcc Program_7.c -o Program_7 } ; if ($?) { .\Program_7 }
Unesi temperaturu zraka (Celzijusi): -7
Unesi vlaznost zraka' (%): 23
Temperatura zraka treba biti u intervalu od 0 do 50 Celzijusa.
PS C:\C_Programs> cd "c:\C_Programs\" ; if ($?) { gcc Program_7.c -o Program_7 } ; if ($?) { .\Program_7 }
Unesi temperaturu zraka (Celzijusi): 45
Unesi vlaznost zraka' (%): 120
Vlaznost zraka treba biti u intervalu od 0 do 100 %.
PS C:\C_Programs> cd "c:\C_Programs\" ; if ($?) { gcc Program_7.c -o Program_7 } ; if ($?) { .\Program_7 }
Unesi temperaturu zraka (Celzijusi): 45
Unesi vlaznost zraka' (%): 56
Unos temperature i vlaznosti zraka je ispravan.
PS C:\C_Programs>
```

*Slika 1.3.1 Rezultat programa za provjeru unosa temperature i vlažnosti zraka*

Postoji još jedna naredba koja se koristi u implementaciji grananja: `switch – case`. Ta je naredba specifična jer se koristi samo za cjelobrojne tipove podataka (npr. `char` i `integer`). Opći oblik naredbe `switch – case` glasi:

```
1.  ...
2.  naredbaX
3.  switch (izraz){
4.  case konstanta_1:
5.              blok_naredbi_1
6.              break
7.  case konstanta_2:
8.              blok_naredbi_2
9.  ...
10. case konstanta_n:
11.             blok_naredbi_n
12. default:
13.             blok_naredbi_x
14. }
15. naredbaY
16. ...
```

U općem obliku naredbe `switch – case izraz` predstavlja bilo što što ima cjelobrojnu vrijednost (npr. neka varijabla, algebarski izraz, poziv neke funkcije i sl.). Vrijednost podatka `izraz` prvo se uspoređuje s `konstanta_1` (konstanta je konkretna vrijednost, npr: 4, 3, A i sl.). Ako je izraz jednak `konstanta_1`, tada se izvodi `blok_naredbi_1`. Nakon izvođenja `blok_naredbi_1` zbog naredbe `break` (linija 6.) izlazi se iz naredbe `switch – case` te se izvodi linija 15. Ako pak `izraz` nije jednak

konstanta_1, tada se provjerava je li on jednak konstanta_2. Ako jest, izvodi se blok_naredbi_2. Treba primijetiti da iza blok_naredbi_2 nema naredbe break zbog čega će se izvesti naredni blok naredbi (u narednom case) – to se svojstvo zove propadanje. Također treba primijetiti default koji ima sličnu ulogu kao else u naredbi if – else if – else. Znači ako izraz nije jednak niti jednoj konstanti, onda se izvodi blok_naredbi_x.
Sljedeći primjer pokazuje primjenu naredbe switch – case.

```
#include <stdio.h>
void main()
{
    char operation;
    int number_1, number_2;
    printf("Unesite prvi broj: ");
    scanf("%d", &number_1);
    printf("Unesite drugi broj: ");
    scanf("%d", &number_2);
    printf("Racunske operacije su\n(+) zbrajanje\n(-) oduzimanje\n(*)
mnozenje\n(/) djeljenje\n");
    printf("Unesite racunsku operaciju: ");
    scanf(" %c", &operation);
    switch (operation)
    {
    case '+':
        printf("%d + %d = %d", number_1, number_2, number_1 + number_2);
        break;
    case '*':
        printf("%d * %d = %d", number_1, number_2, number_1 * number_2);
        break;
    case '-':
        printf("%d - %d = %d", number_1, number_2, number_1 - number_2);
        break;
    case '/':
        printf("%d / %d = ", number_1, number_2);
        if (number_2 == 0)
            printf("Djeljenje s nulom nije dozvoljeno.");
        else
            printf("%.2f", (float)number_1 / number_2);
        break;
    default:
        printf("Nepostojeca racunska operacija.");
    }
}
```

U programu za provjeru unosa temperature i vlažnosti zraka može se primijetiti da nakon unosa podataka i njihove provjere program završava. Znači računalo korisnika ne pita ponovno za unos podataka koji nisu ispravni. Drugim riječima računalo bi trebalo ponavljati pitanje za unos podataka dokle god unos nije ispravan. Također, želi li se implementirati da broj unosa podataka odgovara željenom broju unosa koje je korisnik specificirao, tada se ponovno treba implementirati ponavljanje naredbi. Za implementaciju ponavljanja koriste se sljedeće naredbe u programskom jeziku C: for, while i do while.
Opći oblik naredbe for glasi:

```
1. …
2. naredbaX
3. for(inicijalizacija; logički_izraz; korak)
4. {
5.       blok_naredbi;
6. }
7. naredbaY
8. ...
```

Za naredbu `for` važna su tri elementa: `inicijalizacija`, `logički_izraz` i `korak`. `Inicijalizacija` predstavlja postavljanje neke varijable na početnu vrijednost. `Logički_izraz` predstavlja izraz u kojemu sudjeluje inicijalizirana varijabla i dokle god je on istinit, ponavlja se izvođenje naredbi u `blok_naredbi`. `Korak` predstavlja način promjene vrijednosti inicijalizirane varijable. Redoslijed izvođenja naredbe `for` jest:

1. inicijaliziraj varijablu
2. izračunaj `logički_izraz`
3. ako je `logički_izraz` istina, onda
    a. izvedi `blok_naredbi`
    b. promijeni vrijednost varijable kako je to definirano s `korak`
    c. ponovi 2.
4. ako je `logički_izraz` neistina, onda izvedi liniju 7 (`naredbaY`).

Sljedeći primjer prikazuje primjenu naredbe `for` u ponavljanju unosa podataka.

```c
#include <stdio.h>
void main()
{
    int temperature, humidity;
    int totalNumberOfEntries, numberOfEntries;
    printf("Koliko unosa zelis: ");
    scanf("%d", &totalNumberOfEntries);
    for (numberOfEntries = 1; numberOfEntries <= totalNumberOfEntries;
numberOfEntries = numberOfEntries + 1)
    {
        printf("Unesi temperaturu zraka (Celzijusi): ");
        scanf("%d", &temperature);
        printf("Unesi vlaznost zraka (%%): ");
        scanf("%d", &humidity);
    }
}
```

Rezultat izvođenja programa prikazan je sljedećom slikom:

```
PS C:\C_Programs> cd "c:\C_Programs\" ; if ($?) { gcc Program_9.c -o Program_9 } ; if ($?) { .\Program_9 }
Koliko unosa zelis: 3
Unesi temperaturu zraka (Celzijusi): 24
Unesi vlaznost zraka' (%): 66
Unesi temperaturu zraka (Celzijusi): 32
Unesi vlaznost zraka' (%): 45
Unesi temperaturu zraka (Celzijusi): 25
Unesi vlaznost zraka' (%): 70
PS C:\C_Programs>
```

*Slika 1.3.2 Izvođenje programa s ponavljanjem upita za unos podataka.*

Treba primijetiti korak promjene: `numberOfEntries = numberOfEntries + 1`. To znači da se vrijednost varijable mijenja tako da se njezinoj prethodnoj vrijednosti doda 1 (znači varijabla će imati vrijednosti 1, 2, 3,... pa sve do vrijednosti koju je korisnik unio i koja je spremljena u varijablu

totalNumberOfEntries). Često se izraz numberOfEntries = numberOfEntries + 1 skraćeno piše numberOfEntries++.
Opći oblik naredbe while glasi:

```
1.  ...
2.  naredbaX
3.  while(logički_izraz)
4.  {
5.        blok_naredbi;
6.  }
7.  naredbaY
8.  ...
```

Ponavljanje naredbe u blok_naredbi izvodi se dok je vrijednost logički_izraz istinit. Redoslijed izvođenja naredbe while jest:

1. izračunaj logički_izraz
2. ako je logički_izraz istina, onda
    a. izvedi blok_naredbi
    b. ponovi 1.
3. ako je logički_izraz neistina, onda izvedi liniju 7 (naredbaY).

Sljedeći primjer prikazuje primjenu naredbe while u ponavljanju neispravnog unosa podataka.

```
#include <stdio.h>
void main()
{
    int temperature, humidity;
    while (1 == 1)
    {
        printf("Unesi temperaturu zraka (Celzijusi): ");
        scanf("%d", &temperature);
        if (temperature < 0 || temperature > 50)
        {
            printf("Temperatura zraka treba biti u intervalu od 0 do 50
Celzijusa.\n");
            continue;
        }
        printf("Unos temperature je ispravan.\n");
        break;
    }
    while (1 == 1)
    {

        printf("Unesi vlaznost zraka (%%): ");
        scanf("%d", &humidity);
        if (humidity < 0 || humidity > 100)
        {
            printf("Vlaznost zraka treba biti u intervalu od 0 do 100 %%.\n");
            continue;
        }
        printf("Unos vlaznosti zraka je ispravan.");
        break;
    }
}
```

Treba primijetiti da je kao logički_izraz stavljen 1==1 što je uvijek istina (ovaj izraz može se i kraće zapisati kao 1). Time je dobivena beskonačna petlja koja se prekida naredbom break, odnosno čije se izvođenje nastavlja s naredbom continue.
Slika prikazuje rezultat izvođenja programa.

```
PS C:\C_Programs> cd "c:\C_Programs\" ; if ($?) { gcc Program_10.c -o Program_10 } ; if ($?) { .\Program_10 }
Unesi temperaturu zraka (Celzijusi): 120
Temperatura zraka treba biti u intervalu od 0 do 50 Celzijusa.
Unesi temperaturu zraka (Celzijusi): 67
Temperatura zraka treba biti u intervalu od 0 do 50 Celzijusa.
Unesi temperaturu zraka (Celzijusi): 45
Unos temperature je ispravan.
Unesi vlaznost zraka' (%): 569
Vlaznost zraka treba biti u intervalu od 0 do 100 %.
Unesi vlaznost zraka' (%): 67
Unos vlaznosti zraka je ispravan.
PS C:\C_Programs>
```

*Slika 1.3.3 Provjera unosa podataka s ponovnim upitom*

Opći oblik naredbe do – while glasi:

```
1.  ...
2.  naredbaX
3.  do {
4.        blok_naredbi;
5.  }
6.  while(logički_izraz)
7.  naredbaY
8.  ...
```

Ponavljanje naredbi u blok_naredbi izvodi se dok je vrijednost logički_izraz istinit – slično kao i kod naredbe while. Glavna razlika s obzirom na naredbu while jest ta što se tu naredbe u blok_naredbi sigurno izvode barem jedanput jer se provjera logičkog izraza radi nakon izvođenja naredbi. Redoslijed izvođenja naredbe do – while jest:

1. izvedi blok_naredbi
2. izračunaj logički_izraz
3. ako je logički_izraz istina, onda
    a. izvedi blok_naredbi
    b. ponovi 2.
4. ako je logički_izraz neistina, onda izvedi liniju 7 (naredbaY).

Sljedeći primjer prikazuje primjenu naredbe do – while u ponavljanju neispravnog unosa podataka.

```c
#include <stdio.h>
void main()
{
    int temperature, humidity;
    do
    {
        printf("Unesi temperaturu zraka (Celzijusi): ");
        scanf("%d", &temperature);
        if (temperature < 0 || temperature > 50)
        {
            printf("Temperatura zraka treba biti u intervalu od 0 do 50
Celzijusa.\n");
            continue;
        }
        printf("Unos temperature je ispravan.\n");
        break;
    } while (1 == 1);
    do
```

```
    {

        printf("Unesi vlaznost zraka (%%): ");
        scanf("%d", &humidity);
        if (humidity < 0 || humidity > 100)
        {
            printf("Vlaznost zraka treba biti u intervalu od 0 do 100 %%.\n");
            continue;
        }
        printf("Unos vlaznosti zraka je ispravan.");
        break;
    } while (1 == 1);
}
```

I tu je kao `logički_izraz` stavljen 1==1 što je uvijek istina čime je dobivena beskonačna petlja koja se prekida naredbom `break`, odnosno čije se izvođenje nastavlja naredbom `continue`.

### 1.3.3. Rješenje radnog zadatka

Iz opisa radnog zadatka vidljivo je da se treba izraditi provjera unosa podataka:

- broj ponavljanja unosa (može biti od 0 do 10)
- temperaturu (može biti od0 do 50)
- vlažnost zraka (može biti od 0 do 100)

Nadalje, za izračunani indeks temperature i vlažnosti zraka treba se ispisati razina termalne udobnosti.

Sljedeći program predstavlja rješenje radnog zadatka.

```
#include <stdio.h>
void main()
{
    int totalNumberOfEntries, numberOfEntries;
    do
    {
        printf("Koliko unosa zelis: ");
        scanf("%d", &totalNumberOfEntries);
        if (totalNumberOfEntries < 0 || totalNumberOfEntries > 10)
        {
            printf("Broj unosa treba biti u intervalu od 0 do 10.\n");
            continue;
        }
        break;
    } while (1 == 1);
    int temperature, humidity;
    float temperatureHumidityIndex;
    for (numberOfEntries = 1; numberOfEntries <= totalNumberOfEntries;
numberOfEntries = numberOfEntries + 1)
    {
        printf("\n\nUNOS %d/%d\n", numberOfEntries, totalNumberOfEntries);
        do
        {
            printf("Unesi temperaturu zraka (Celzijusi): ");
            scanf("%d", &temperature);
            if (temperature < 0 || temperature > 50)
            {
                printf("Temperatura zraka treba biti u intervalu od 0 do 50
Celzijusa.\n");
                continue;
```

```
            }
            break;
        } while (1 == 1);
        do
        {

            printf("Unesi vlaznost zraka (%%): ");
            scanf("%d", &humidity);
            if (humidity < 0 || humidity > 100)
            {
                printf("Vlaznost zraka treba biti u intervalu od 0 do 100
%%.\n");
                continue;
            }
            break;
        } while (1 == 1);
        temperatureHumidityIndex = (1.8 * temperature + 32) - ((0.55 - 0.0055 *
humidity) * (1.8 * temperature - 26));
        printf("\nIndeks temperature i vlaznosti (THI): %.2f",
temperatureHumidityIndex);
        if (temperatureHumidityIndex < 72)
        {
            printf(" - UGODNO");
        }
        else if (temperatureHumidityIndex < 79)
        {
            printf(" - PODNOSLJIVO");
        }
        else if (temperatureHumidityIndex < 89)
        {
            printf(" - NEUGODNO");
        }
        else if (temperatureHumidityIndex < 99)
        {
            printf(" - VRLO NEUGODNO");
        }
        else
        {
            printf(" - OPASNO PO ZDRAVLJE");
        }
    }
}
```

Sljedeća slika prikazuje rezultat izvođenja programa.

```
PS C:\C_Programs> cd "c:\C_Programs\" ; if ($?) { gcc Program_11.c -o Program_11 } ; if ($?) { .\Program_11 }
Koliko unosa zelis: 2


UNOS 1/2
Unesi temperaturu zraka (Celzijusi): 20
Unesi vlaznost zraka (%): 60

Indeks temperature i vlaznosti (THI): 65.80 - UGODNO

UNOS 2/2
Unesi temperaturu zraka (Celzijusi): 23
Unesi vlaznost zraka (%): 90

Indeks temperature i vlaznosti (THI): 72.55 - PODNOSLJIVO
PS C:\C_Programs>
```

*Slika 1.3.4 Rezultat izvođenja programa*

### 1.3.4. Pitanja i zadaci

1. Izmijenite program na način da se koriste druge naredbe za implementaciju ponavljanja.
2. Izmijenite program na način da se nakon unosa provjera podataka radi tako da je logički izraz istinit onda kada su podaci ispravni (sada je logički izraz istinit kada su podaci neispravni, npr. `totalNumberOfEntries < 0 || totalNumberOfEntries > 10`).
3. Kod naredbe if – else if – else ako je jedan logički izraz istinit, tada se izračun logičkih izraza u narednim `else if` dijelovima preskaču.
    a. Točno
    b. Netočno
4. Kod if – else if – else naredbe dio `else` izvodi se bez obzira na to je li neki logički izraz u `if` i `else if` dijelu naredbe istinit.
    a. Točno
    b. Netočno
5. Pojam propadanja koji se odnosi na izvođenje naredbi i u narednom bloku naredbi odnosi se na sljedeću naredbu. Kod naredbe if – else if – else dio `else` izvodi se bez obzira na to je li neki logički izraz u `if` i `else if` dijelu naredbe istinit.
    a. if – else if – else
    b. `switch – case`
    c. `while`
    d. `do – while`
    e. `for`
6. Kod naredbe `for` prvo se izvode naredbe u bloku naredbi pa se nakon toga računa logički izraz.
    a. Točno
    b. Netočno
7. Kod naredbe `while` prvo se izvode naredbe u bloku naredbi pa se nakon toga računa logički izraz.
    a. Točno
    b. Netočno
8. Kod naredbe `do – while` prvo se izvode naredbe u bloku naredbi pa se nakon toga računa logički izraz.
    a. Točno
    b. Netočno

### 1.3.5. Literatura i izvori


1. C Programming Language Tutorial, https://www.javatpoint.com/c-programming-language-tutorial
2. C Tutorial, https://www.tutorialspoint.com/cprogramming/index.htm
3. C Tutorial, https://www.w3schools.in/c-tutorial
4. Jakupović, A.; Šuman, S.: Osnove programiranja, https://www.veleri.hr/sites/default/files/2021-07/osnove_programiranja_konacna_verzija_online_verzija.pdf


## 1.4. Funkcije

Funkcija predstavlja koncept u programiranju koji omogućuje grupiranje niza naredbi koje se nalaze u tijelu funkcije. Taj je niz naredbi je imenovan – to je naziv funkcije – te se izvodi tako da se funkcija pozove preko svoga imena. Koncept takvog grupiranja naredbi, osim funkcija, još se naziva i procedura, rutina i potprogram. Funkciji, odnosno nizu naredbi, mogu se poslati podaci preko argumenata (parametara) funkcije, a ona može i vratiti vrijednost mjestu poziva preko svojega povratnog tipa ili argumenta. Obradit će se deklaracija i definicija funkcije, argumenti i povratni tip.

**Nakon usvajanja ove nastavne teme čitatelj će moći:**
1. izraditi definiciju funkcije u programskom jeziku C
2. izraditi deklaraciju funkcije u programskom jeziku C
3. razlikovati deklaraciju i definiciju funkcije
4. izraditi funkciju bez povratne vrijednosti i bez argumenata
5. izraditi funkciju bez povratne vrijednosti i s argumentima
6. izraditi funkciju s povratnom vrijednošću i bez argumenata
7. izraditi funkciju s povratnom vrijednošću i argumentima

### 1.4.1. Opis radnog zadatka

Potrebno je izraditi računalni program kojim se računa indeks temperature i vlažnosti zraka (engl. *Temperature Humidity Index* – THI). Taj se indeks koristi kako bi se procijenila razina termalne udobnosti ljudi i životinja u nekom prostoru. Formula kojom se računa THI glasi:

$$THI = (1.8 \cdot T + 32) - \big((0.55 - 0.0055 \cdot RH) \cdot (1.8 \cdot T - 26)\big)$$

gdje je:
*THI* – indeks temperature i vlažnosti
*T* – temperatura zraka u celzijusima
*RH* – vlažnost zraka u postocima

Izračunani THI označava razinu termalne udobnosti prema sljedećoj tablici:

| THI | Razina termalne udobnosti |
|---|---|
| < 72 | UGODNO |
| 72 – 79 | PODNOŠLJIVO |
| 80 – 89 | NEUGODNO |
| 90 – 99 | VRLO NEUGODNO |

| | |
|---|---|
| > 99 | OPASNO PO ZDRAVLJE |

Korisnika je potrebno pitati koliko unosa želi napraviti (maksimalni broj unosa jest 10). Nakon toga potrebno ga je toliko puta pitati za unos temperature i vlažnosti zraka. Kod svakog unosa potrebno je provjeriti ispravnost unesenih podataka (temperatura treba biti u intervalu od 0 do 50, a vlažnost zraka u intervalu od 0 do 100). Nakon završetka unosa temperature i vlažnosti zraka potrebno je prikazati iznos indeksa temperature i vlažnosti (THI) te razine termalne udobnosti.
Program treba strukturirati tako da se sastoji od četiri funkcije:

- printAppTitle, koja ne vraća vrijednost i nema argumenata. Njezina je svrha da ispiše naslov aplikacije
- getNumberOfEntries, koja vraća broj unosa koje je korisnik specificirao. Funkcija nema argumenata
- validateData, koja vraća 1 ili 0 u ovisnosti o tome je li podatak ispravan. Funkcija ima dva argumenta: argData i argDataPurpose (E – entries, T – temperature, H – humidity)
- printTHI, koja ne vraća vrijednost, ali prima dva argumenta (argTemperature i argHumidity). Funkcija ispisuje vrijednost THI i razinu termalne udobnosti.

### 1.4.2. Osnovni koncepti

Funkcija (procedura, potprogram, rutina) predstavlja imenovani skup naredbi kojima se rješava neki manji problem. Skup naredbi nalazi se u tijelu funkcije, a izvodi se pozivom njegova imena – i to se naziv poziv funkcije. Funkciji, odnosno naredbama u njezinom tijelu moguće je poslati podatke preko njezinih argumenata (parametara). Funkcija može i vratiti podatke mjestu poziva preko povratnog tipa ili kroz argumente.
Opći oblik definicije funkcije u programskom jeziku C glasi:

```
1.  povratni_tip naziv_funkcije( const tip_argumenta1 naziv_argumenta1,
                                 tip_argumenta2 naziv_argumenta2,
                                 ...
                                 tip_argumentan naziv_argumentan){
2.      naredba1
3.      return povratna_vrijednost;
4.      naredba2
5.  }
6.  neka_funkcija(){
7.      naredbaX
8.      vrijednost = naziv_funkcije (  konkretna_vrijednost,
                                       varijabla1,
                                       ...
                                       varijablan);
9.      naredbaY
10.     ...
11. }
```

Prikazani su sljedeći dijelovi definicije funkcije: povratni tip, naziv funkcije, popis argumenata i tijelo funkcije.
Povratni tip (`povratni_tip`) jest tip podatka koji funkcija vraća. Ako je naveden, u tijelu funkcije mora postojati naredba `return` i vrijednost koja se vraća. Ako funkcija ne vraća vrijednost kroz povratni tip, tada se kao povratni tip stavlja `void`.

Naziv funkcije (`naziv_funkcije`) jest naziv preko kojeg će se funkcija pozivati s nekog mjesta poziva. Funkcija treba biti definirana prije prvog poziva funkcije. U primjeru opće definicije funkcija `naziv_funkcije` definirana je prije svojeg poziva unutar funkcije `neka_funkcija`.
Popis argumenata funkcije predstavlja deklaraciju varijabli preko kojih će podaci ulaziti u tijelo funkcije. Redoslijed deklaracije varijabli, odnosno popisa argumenata važan je jer ako je neki argument namijenjen nekom određenom podatku, tada se taj podatak funkciji mora proslijediti upravo kroz taj argument.
Tijelo funkcije jest blok naredbi omeđen vitičastim zagradama koji će se izvesti onda kada se funkcija pozove. Ako funkcija posjeduje argumente, tada se i oni trebaju pojaviti u tijelu funkcije. Ako je specificiran povratni tip funkcije, onda se u tijelu funkcije mora nalaziti naredba `return` koja vraća podatak.
Imenovanje funkcije i njezinih argumenata provodi se po pravilima imenovanja varijabli.
Sljedeći primjer prikazuje definiciju funkcije printAppTitle koja ispisuje naziv aplikacije i nema povratni tip niti argumenata.

```c
#include <stdio.h>

void printAppTitle()
{
    printf("_____________________________________\n\n");
    printf("IZRACUN RAZINE TERMALNE UDOBNOSTI\n");
    printf("_____________________________________\n");
}

void main()
{
    printAppTitle();
}
```

Može se primijetiti poziv funkcije `printAppTitle` iz tijela funkcije `main`. Također se može primijetiti koje se naredbe nalaze u tijelu funkcije `printAppTitle`, a koje će se izvesti kada se funkcija pozove.
Sljedeća slika prikazuje rezultat izvođenja navedenog programa.

```
PS C:\C_Programs> cd "c:\C_Programs\" ; if ($?) { gcc Program_12.c -o Program_12 } ; if ($?) { .\Program_12 }
_____________________________

IZRACUN RAZINE TERMALNE UDOBNOSTI

_____________________________
PS C:\C_Programs>
```

*Slika 1.4.1 Rezultat izvođenja programa za ispis naslova aplikacije*

Sljedeći program pokazuje definiciju funkcije `getNumberOfEntries` koja vraća broj unosa koje je korisnik specificirao.

```c
#include <stdio.h>

int getNumberOfEntries()
{
    int totalNumberOfEntries;
    do
    {
        printf("Koliko unosa zelis: ");
        scanf("%d", &totalNumberOfEntries);
        if (totalNumberOfEntries < 0 || totalNumberOfEntries > 10)
        {
            printf("Broj unosa treba biti u intervalu od 0 do 10.\n");
            continue;
```

```
        }
        break;
    } while (1);

    return totalNumberOfEntries;
}

void main()
{
    int usersTotalNumberOfEntries;

    usersTotalNumberOfEntries = getNumberOfEntries();

    printf("\nKorisnik zeli %d unosa.", usersTotalNumberOfEntries);
}
```

U ovom primjeru može se uočiti da se u tijelu funkcije `getNumberOfEntries` nalazi složeniji program koji se osim pitanja korisniku za broj unosa sastoji i od provjere unosa. Također treba primijetiti da funkcija posjeduje povratni tip `int` te naredbu `return`.

U funkciji `main` poziva se funkcija `getNumberOfEntries`. Treba primijetiti kako se operatorom pridruživanja (=) povratna vrijednost funkcije sprema u varijablu `usersTotalNumberOfEntries` koja se dalje koristi u ispisu poruke.

Rezultat izvođenja navedenog programa prikazan je sljedećom slikom.

```
PS C:\C_Programs> cd "c:\C_Programs\" ; if ($?) { gcc Program_13.c -o Program_13 } ; if ($?) { .\Program_13 }
Koliko unosa zelis: 23
Broj unosa treba biti u intervalu od 0 do 10.
Koliko unosa zelis: -4
Broj unosa treba biti u intervalu od 0 do 10.
Koliko unosa zelis: 5

Korisnik zeli 5 unosa.
PS C:\C_Programs>
```

*Slika 1.4.2 Rezultat izvođenja programa za specifikaciju broja unosa*

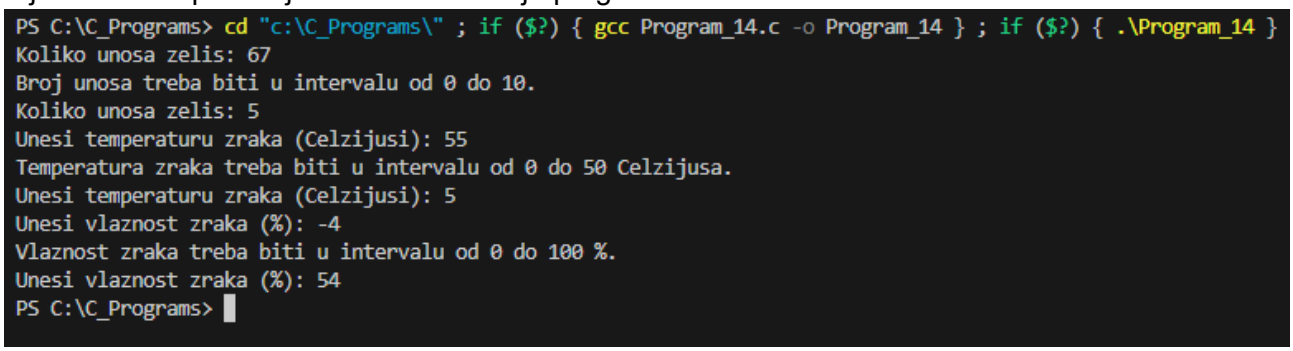
Sljedeći program prikazuje funkciju `validateData`. Ona vraća 1 ili 0 u ovisnosti o tome je li podatak ispravan. Preko argumenta `argData` funkciji se dostavlja podatak koji se treba provjeriti. Preko argumenta `argDataPurpose` dostavlja se svrha podatka (E – entries, T – temperature, H – humidity).

```
#include <stdio.h>

int validateData(int agData, char argDataPurpose)
{
    switch (argDataPurpose)
    {
    case 'E':
        if (agData < 0 || agData > 10)
        {
            return 0;
        }
        return 1;

    case 'T':
        if (agData < 0 || agData > 50)
        {
            return 0;
        }
        return 1;
```

```
    case 'H':
        if (agData < 0 || agData > 100)
        {
            return 0;
        }
        return 1;
    }
    return 0;
}

void main()
{
    int totalNumberOfEntries;
    int temperature, humidity;
    do
    {
        printf("Koliko unosa zelis: ");
        scanf("%d", &totalNumberOfEntries);
        if (validateData(totalNumberOfEntries, 'E') == 0)
        {
            printf("Broj unosa treba biti u intervalu od 0 do 10.\n");
            continue;
        }
        break;
    } while (1 == 1);
    do
    {
        printf("Unesi temperaturu zraka (Celzijusi): ");
        scanf("%d", &temperature);
        if (validateData(temperature, 'T') == 0)
        {
            printf("Temperatura zraka treba biti u intervalu od 0 do 50
Celzijusa.\n");
            continue;
        }
        break;
    } while (1 == 1);
    do
    {

        printf("Unesi vlaznost zraka (%%): ");
        scanf("%d", &humidity);
        if (validateData(humidity, 'H') == 0)
        {
            printf("Vlaznost zraka treba biti u intervalu od 0 do 100 %%.\n");
            continue;
        }
        break;
    } while (1 == 1);
}
```

Funkcija `validateData` ima povrati tip `int` i dva argumenta (jedan tipa podatka `int`, a drugi tipa podatka `char`). Tijelo funkcije sastoji se od naredbe `switch – case` preko koje se sukladno vrijednosti argumenta `argDataPurpose` izvodi neka od provjera vrijednosti argumenta `argData`. Treba primijetiti naredbu `return` u pojedinom `case` i nepostojanje naredbe `break` – naredba `break` u ovom slučaju nije potrebna jer se naredbom `return` izlazi iz tijela funkcije.

U glavnom dijelu programa funkcija `validateData` poziva se unutar naredbe `if` gdje se provjerava vrijednost koju funkcija vraća. Treba primijeniti način na koji se funkciji kroz poziv šalju podaci preko njezina dva argumenta (kroz prvi argument šalje se vrijednost varijable – `totalNumberOfEntries`, `temperature`, `humidity` – a kroz drugi konkretna vrijednost – znak E, T ili H).
Sljedeća slika prikazuje rezultat izvođenja programa.




*Slika 1.4.3 Rezultat izvođenja programa kojim se provjeravaju unesene vrijednosti*

Sljedeći program prikazuje funkciju `printTHI` koja ne vraća vrijednost, ali ima dva argumenta.

```
#include <stdio.h>

void printTHI(int argTemperature, int argHumidity)
{
    float temperatureHumidityIndex = (1.8 * argTemperature + 32) - ((0.55 -
0.0055 * argHumidity) * (1.8 * argTemperature - 26));
    printf("\nIndeks temperature i vlaznosti (THI): %.2f",
temperatureHumidityIndex);
    if (temperatureHumidityIndex < 72)
    {
        printf(" - UGODNO");
    }
    else if (temperatureHumidityIndex < 79)
    {
        printf(" - PODNOSLJIVO");
    }
    else if (temperatureHumidityIndex < 89)
    {
        printf(" - NEUGODNO");
    }
    else if (temperatureHumidityIndex < 99)
    {
        printf(" - VRLO NEUGODNO");
    }
    else
    {
        printf(" - OPASNO PO ZDRAVLJE");
    }
}

void main()
{
    int temperature, humidity;

    do
    {
        printf("Unesi temperaturu zraka (Celzijusi): ");
        scanf("%d", &temperature);
```

```
        if (temperature < 0 || temperature > 50)
        {
            printf("Temperatura zraka treba biti u intervalu od 0 do 50
Celzijusa.\n");
            continue;
        }
        break;
    } while (1 == 1);
    do
    {

        printf("Unesi vlaznost zraka (%%): ");
        scanf("%d", &humidity);
        if (humidity < 0 || humidity > 100)
        {
            printf("Vlaznost zraka treba biti u intervalu od 0 do 100 %%.\n");
            continue;
        }
        break;
    } while (1 == 1);

    printTHI(temperature, humidity);
}
```

Pozivom funkcije `printTHI` preko njezinih argumenata šalju joj se dva podatka koji se nalaze u varijablama `temperature` i `humidity`. U tijelu funkcije nalazi se izračun THI-ja te njezin ispis i ispis razine termalne udobnosti.
Sljedeća slika prikazuje rezultat izvođenja programa.

```
PS C:\C_Programs> cd "c:\C_Programs\" ; if ($?) { gcc Program_15.c -o Program_15 } ; if ($?) { .\Program_15 }
Unesi temperaturu zraka (Celzijusi): 22
Unesi vlaznost zraka (%): 69

Indeks temperature i vlaznosti (THI): 69.28 - UGODNO
PS C:\C_Programs>
```

*Slika 1.4.4 Rezultat izvođenja programa izračuna i ispisa THI-ja i razine termalne udobnosti*

### 1.4.3. Rješenje radnog zadatka

Iz opisa radnog zadatka proizlazi potreba izrade četiri funkcije:

- printAppTitle, koja ne vraća vrijednost i nema argumenata. Njezina je svrha ispisati naslov aplikacije
- getNumberOfEntries, koja vraća broj unosa koje je korisnik specificirao. Funkcija nema argumenata
- validateData, koja vraća 1 ili 0 u ovisnosti o tome je li podataka ispravan. Funkcija ima dva argumenta: argData i argDataPurpose (E – entries, T – temperature, H – humidity);
- printTHI koja ne vraća vrijednost, ali prima dva argumenta (argTemperature i argHumidity). Funkcija ispisuje vrijednost THI i razinu termalne udobnosti.

Sve navedene funkcije prikazane su u prethodnom poglavlju te ih je sada potrebno uključiti u jedan program. Sljedeći program predstavlja rješenje radnog zadatka.

```
#include <stdio.h>

void printAppTitle()
{
```

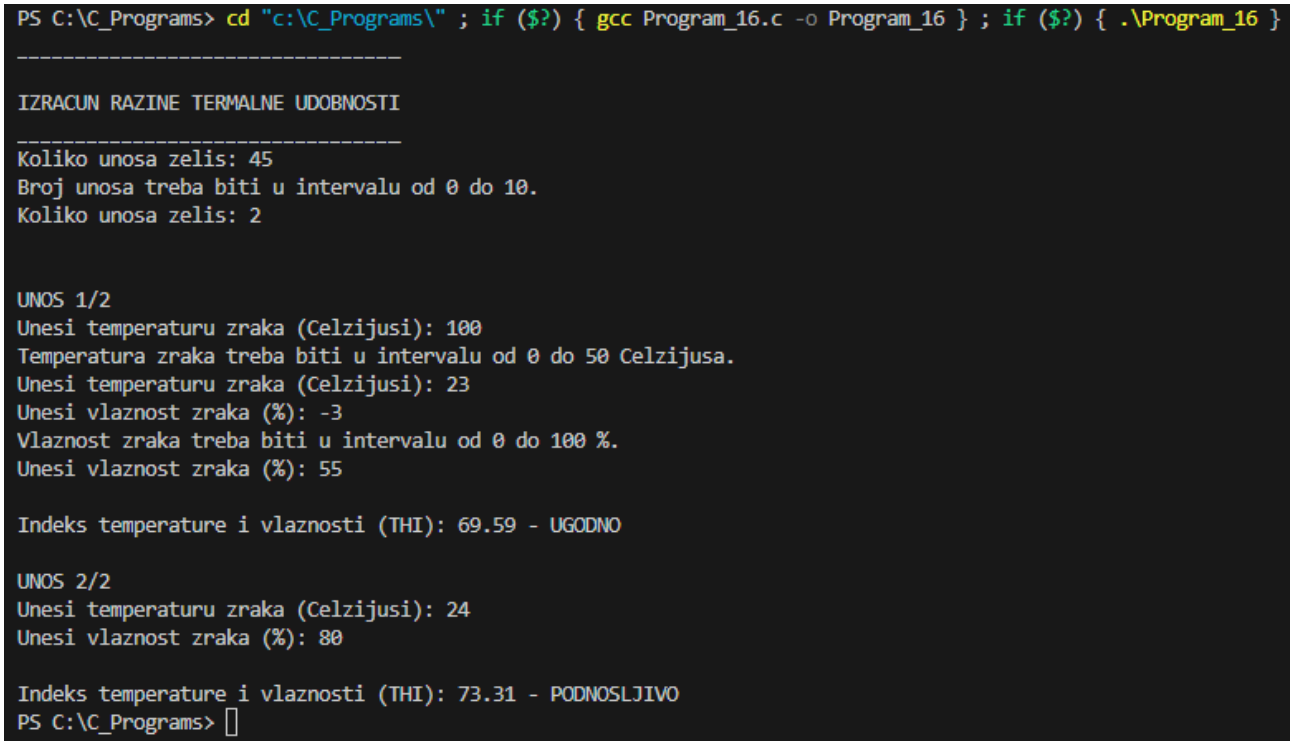
```
    printf("_____\n\n");
    printf("IZRACUN RAZINE TERMALNE UDOBNOSTI\n");
    printf("_____\n");
}

int validateData(int agData, char argDataPurpose)
{
    switch (argDataPurpose)
    {
    case 'E':
        if (agData < 0 || agData > 10)
        {
            return 0;
        }
        return 1;

    case 'T':
        if (agData < 0 || agData > 50)
        {
            return 0;
        }
        return 1;
    case 'H':
        if (agData < 0 || agData > 100)
        {
            return 0;
        }
        return 1;
    }
    return 0;
}

int getNumberOfEntries()
{
    int totalNumberOfEntries;
    do
    {
        printf("Koliko unosa zelis: ");
        scanf("%d", &totalNumberOfEntries);
        if (validateData(totalNumberOfEntries, 'E') == 0)
        {
            printf("Broj unosa treba biti u intervalu od 0 do 10.\n");
            continue;
        }
        break;
    } while (1);

    return totalNumberOfEntries;
}

void printTHI(int argTemperature, int argHumidity)
{
    float temperatureHumidityIndex = (1.8 * argTemperature + 32) - ((0.55 -
0.0055 * argHumidity) * (1.8 * argTemperature - 26));
    printf("\nIndeks temperature i vlaznosti (THI): %.2f",
temperatureHumidityIndex);
    if (temperatureHumidityIndex < 72)
    {
```

```
        printf(" - UGODNO");
    }
    else if (temperatureHumidityIndex < 79)
    {
        printf(" - PODNOSLJIVO");
    }
    else if (temperatureHumidityIndex < 89)
    {
        printf(" - NEUGODNO");
    }
    else if (temperatureHumidityIndex < 99)
    {
        printf(" - VRLO NEUGODNO");
    }
    else
    {
        printf(" - OPASNO PO ZDRAVLJE");
    }
}

void main()
{
    int userTotalNumberOfEntries, numberOfEntries;
    printAppTitle();
    userTotalNumberOfEntries = getNumberOfEntries();

    int temperature, humidity;
    for (numberOfEntries = 1; numberOfEntries <= userTotalNumberOfEntries;
numberOfEntries = numberOfEntries + 1)
    {
        printf("\n\nUNOS %d/%d\n", numberOfEntries, userTotalNumberOfEntries);
        do
        {
            printf("Unesi temperaturu zraka (Celzijusi): ");
            scanf("%d", &temperature);
            if (validateData(temperature, 'T') == 0)
            {
                printf("Temperatura zraka treba biti u intervalu od 0 do 50
Celzijusa.\n");
                continue;
            }
            break;
        } while (1 == 1);
        do
        {

            printf("Unesi vlaznost zraka (%%): ");
            scanf("%d", &humidity);
            if (validateData(humidity, 'H') == 0)
            {
                printf("Vlaznost zraka treba biti u intervalu od 0 do 100
%%.\n");
                continue;
            }
            break;
        } while (1 == 1);
        printTHI(temperature, humidity);
```

```
    }
}
```

Treba primijetiti da je funkcija `validateData` definirana prije funkcije `getNumberOfEntries` jer se u funkciji `getNumberOfEntries` ona poziva radi provjere unesenih podataka. Ostala razmatranja slična su onima iz prethodnog poglavlja.

Sljedeća slika prikazuje rezultat izvođenja programa.




*Slika 1.4.5 Rezultat izvođenja programa za unos temperature i vlažnosti zraka i izračun THI-ja*

## 1.4.4. Pitanja i zadaci

1. Izmijenite program tako da se za temperaturu i vlažnost zraka koristi tip podatka `float`, a za svrhu podatka u funkciji `validateData` da se koristi tip podatka `int`.
2. Izmijenite program tako da se unos temperature radi u jednoj funkciji (`getTemperature`), a unos važnosti zraka u drugoj funkciji (`getHumidity`).
3. Ako funkcija ima povratni tip, tada se u njezinom tijelu treba nalaziti naredba:
    a. break
    b. continue
    c. return
    d. exit
    e. goto

4. Funkcija koja kroz svoj povratni tip ne vraća vrijednost treba biti definirana s:
    a. int
    b. float
    c. void
    d. char
    e. double

5. Funkcija obavezno mora imati argumente.
    a. Točno

b. Netočno

6. Funkcija u svojem tijelu može pozvati drugu funkciju.
   a. Točno
   b. Netočno

7. Vrijednost koju funkcija vraća naredbom return mora odgovarati povratnom tipu podatka funkcije.
   a. Točno
   b. Netočno

### 1.4.5. Literatura i izvori


1. C Programming Language Tutorial, https://www.javatpoint.com/c-programming-language-tutorial
2. C Tutorial, https://www.tutorialspoint.com/cprogramming/indeks.htm
3. C Tutorial, https://www.w3schools.in/c-tutorial
4. Jakupović, A.; Šuman, S.: Osnove programiranja, https://www.veleri.hr/sites/default/files/2021-07/osnove_programiranja_konacna_verzija_online_verzija.pdf


## 1.5. Polja podataka

Varijable se u računalnom programu koriste kako bi se u memoriji računala mogli čuvati podaci. Koji se podaci čuvaju peko varijable u memoriji računala ovisi o tipu podatka kojim je varijabla deklarirana (npr. znak, cijeli broj, decimalni broj itd.). Jedna posebna vrsta varijabli jest polje podataka. Polje podataka omogućuje istovremeno čuvanje više vrijednosti podataka koji su svi istoga tipa. U nastavku će biti opisana deklaracija jednodimenzionalnog i višedimenzionalnog polja, rad s poljima, deklaracija znakovnog polja i rad sa znakovnim poljem.

**Nakon usvajanja ove nastavne teme čitatelj će moći:**
1. deklarirati i inicijalizirati jednodimenzionalno polje
2. dohvatiti član jednodimenzionalnog polja i pridružiti mu vrijednost
3. deklarirati i inicijalizirati dvodimenzionalno ili višedimenzionalno polje
4. dohvatiti član dvodimenzionalnog ili višedimenzionalnog polja i pridružiti mu vrijednost
5. deklarirati i inicijalizirati znakovno polje
6. dohvatiti član znakovnog polja i pridružiti mu vrijednost

### 1.5.1. Opis radnog zadatka

Potrebno je izraditi računalni program kojim se računa indeks temperature i vlažnosti zraka (engl. *Temperature Humidity Index* – THI). Taj se indeks koristi kako bi se procijenila razina termalne udobnosti ljudi i životinja u nekom prostoru. Formula kojom se računa THI glasi:

$$THI = (1.8 \cdot T + 32) - \big((0.55 - 0.0055 \cdot RH) \cdot (1.8 \cdot T - 26)\big)$$

gdje je:
*THI* – indeks temperature i vlažnosti
*T* – temperatura zraka u celzijusima
*RH* – vlažnost zraka u postocima

Izračunani THI označava razinu termalne udobnosti prema sljedećoj tablici:

| THI | Razina termalne udobnosti |
|---|---|
| < 72 | UGODNO |
| 72 – 79 | PODNOŠLJIVO |
| 80 – 89 | NEUGODNO |
| 90 – 99 | VRLO NEUGODNO |
| > 99 | OPASNO PO ZDRAVLJE |

Korisnika je potrebno pitati koliko unosa želi napraviti (maksimalni broj unosa jest 10). Nakon toga potrebno ga je toliko puta pitati za unos naziva prostorije, temperature i vlažnosti zraka. Kod svakog unosa potrebno je provjeriti ispravnost unesenih podataka (temperatura treba biti u intervalu od 0 do 50, a vlažnost zraka u intervalu od 0 do 100). Nakon završetka unosa temperature i vlažnosti zraka potrebno je prikazati iznos indeksa temperature i vlažnosti (THI) te razine termalne udobnosti.
Program treba strukturirati tako da se sastoji od tri polja:

- roomName, dvodimenzionalno polje u kojemu se za svaku prostoriju čuva njezin naziv
- temperatures i humidities, dva jednodimenzionalna polja u kojima se čuvaju unesene temperature i vlažnosti zraka
- temperatureHumidityIndexes, dvodimenzionalno polje koje u sebi čuva podatak sljedećeg JSON formata: `{„Prostorija”: vrijednost, „Temperatura”: vrijednost, „Vlaznost”: vrijednost, „Indeks”: vrijednost}.`

## 1.5.2. Osnovni koncepti

Polje podataka posebna je vrsta varijabli koja omogućuje istovremeno čuvanje više podataka, ali koji su svi istoga tipa podatka. Deklaracijom polja podataka specificira se koji će se tip podatka čuvati u polju, kako se polje (varijabla) zove te koliko će se maksimalno elemenata (podataka) moći čuvati u polju. Računalo će s obzirom na tip podatka i broj elemenata polja rezervirati odgovarajući broj memorijskih lokacija. Primjera radi ako je deklaracijom specificirano 10 elemenata i tip podatka jest `int` (koji zahtijeva 32 bita u memoriji), tada će računalo rezervirati 10 * 32 = 320 bitova memorije. Ako se kao tip podatka u deklaraciji poja koristi `char`, tada se govori o znakovnom polju ili nizu znakova (`string`).
Do pojedinog elementa u polju dolazi se preko njegova indeksa s time da je prvi element polja na indeksu 0, drugi na indeksu 1 itd. U primjeru polja unutar kojega se može čuvati 10 elemenata, prvi element nalazi se na indeksu 0, a zadnji na indeksu 9. Polja u kojima se koristi samo jedan indeks za dolazak do elementa polja zove se jednodimenzionalno polje. Ako se do elementa polja dolazi tako da se specificiraju dva indeksa, tada se govori o dvodimenzionalnom polju itd. Definicija dimenzije polja radi se preko njegove deklaracije.
Opći oblik deklaracije polja glasi:

```
tip_podatka ime_Varijable[broj_elemenata1,broj_elemenata2,..., broj_elemenatan];
```

Može se primijetiti da je deklaracija polja slična deklaraciji varijable s razlikom što se nakon specifikacije njezina imena u uglatim zagradama specificira dimenzija polja, odnosno broj elemenata koje će polje moći čuvati. Ako se specificira samo `broj_elemenata1`, tada se govori o jednodimenzionalnom polju. Specifikacijom `broj_elemenata1` i `broj_elemenata2` dobiva se dvodimenzionalno polje itd.

U sljedećem primjeru prikazana je primjena dvodimenzionalnog polja u kojemu se čuvaju nazivi prostorija.

```
#include <stdio.h>
void main()
{
    int totalNumberOfEntries, numberOfEntries;
    printf("Koliko unosa zelis: ");
    scanf("%d", &totalNumberOfEntries);
    char roomNames[totalNumberOfEntries][100 + 1];
    for (numberOfEntries = 1; numberOfEntries <= totalNumberOfEntries;
numberOfEntries++)
    {
        printf("Unesi naziv %d. prostorije: ", numberOfEntries);
        scanf("%s", &roomNames[numberOfEntries - 1]);
    }

    printf("\n\nPOPIS PROSTORIJA\n");
    int indeks;
    for (indeks = 0; indeks < totalNumberOfEntries; indeks++)
    {
        printf("%d.\t%s\n", indeks + 1, roomNames[indeks]);
    }
}
```

Može se primijetiti deklaracija dvodimenzionalnog polja znakova `roomNames`. Deklaracija je provedena nakon što se od korisnika dobila informacija o željenom broju unosa i time se specificirala prva dimenzija polja. Druga dimenzija polja jest 100 + 1 (odnosno 101) – ovo je navedeno ovako kako bi se ukazalo na činjenicu da se u polju znakova uvijek treba rezervirati jedan dodatni znak (`null` znak – /0) koji računalo automatski stavlja na kraj kako bi se znalo gdje niz znakova završava. Npr. ako se u polje znakova veličine 100 + 1 čuva riječ „DVORANA", tada se na indeksu 0 nalazi znak „D", na indeksu 1 znak „V" itd. sve do indeksa 6 gdje se nalazi znak „A". No računalo ne zna da iza indeksa 6 više nema zakova koji pripadaju ovoj riječi. Stoga ono dodaje i specijalni `null` znak (\0) na indeks 7. Time riječ „DVORANA" u polju ne zauzima 7 pozicija, nego 8. To dalje znači da u polje znakova od 100 + 1 element može biti sačuvana riječ duljine do 100 znakova.
Sljedeća slika prikazuje rezultat izvođenja programa.

```
PS C:\C_Programs> cd "c:\C_Programs\" ; if ($?) { gcc Program_17.c -o Program_17 } ; if ($?) { .\Program_17 }
Koliko unosa zelis: 5
Unesi naziv 1. prostorije: Dvorana_1
Unesi naziv 2. prostorije: Dvorana_5/1
Unesi naziv 3. prostorije: Kabinet_2
Unesi naziv 4. prostorije: Racunalni_kabinet
Unesi naziv 5. prostorije: Laboratorij


POPIS PROSTORIJA
1.      Dvorana_1
2.      Dvorana_5/1
3.      Kabinet_2
4.      Racunalni_kabinet
5.      Laboratorij
PS C:\C_Programs>
```

*Slika 1.5.1 Izvođenje programa za unos naziva prostorija*

Sljedeći primjer prikazuje dva jednodimenzionalna polja u kojima se čuvaju temperature i vlažnosti zraka.

```c
#include <stdio.h>
void main()
{
    int totalNumberOfEntries, numberOfEntries;
    printf("Koliko unosa zelis: ");
    scanf("%d", &totalNumberOfEntries);
    int temperatures[totalNumberOfEntries], humidities[totalNumberOfEntries];
    for (numberOfEntries = 1; numberOfEntries <= totalNumberOfEntries;
numberOfEntries++)
    {
        printf("Unesi temperaturu %d. prostorije: ", numberOfEntries);
        scanf("%d", &temperatures[numberOfEntries - 1]);
        printf("Unesi vlaznost zraka %d. prostorije: ", numberOfEntries);
        scanf("%d", &humidities[numberOfEntries - 1]);
    }

    printf("\n\nTEMPERATURA I VLAZNOST\n");
    int indeks;
    for (indeks = 0; indeks < totalNumberOfEntries; indeks++)
    {
        printf("%d.\t%dC\t%d%%\n", indeks + 1, temperatures[indeks],
humidities[indeks]);
    }
}
```

I u ovom primjeru nalazi se deklaracija polja podataka napravljena nakon dobivanja informacije o broju unosa. Broj elemenata polja jednak je specificiranom broju unosa jer za polje koje nije tipa podatka `char` nije potreban dodatni element polja za `null` znak.
Sljedeća slika prikazuje rezultat izvođenja programa.

```
PS C:\C_Programs> cd "c:\C_Programs\" ; if ($?) { gcc Program_18.c -o Program_18 } ; if ($?) { .\Program_18 }
Koliko unosa zelis: 3
Unesi temperaturu 1. prostorije: 21
Unesi vlaznost zraka 1. prostorije: 60
Unesi temperaturu 2. prostorije: 19
Unesi vlaznost zraka 2. prostorije: 55
Unesi temperaturu 3. prostorije: 22
Unesi vlaznost zraka 3. prostorije: 65


TEMPERATURA I VLAZNOST
1.      21C     60%
2.      19C     55%
3.      22C     65%
PS C:\C_Programs>
```

*Slika 1.5.2 Rezultat izvođenja programa s jednodimenzionalnim poljima za temperaturu i vlažnost zraka*

Sljedeći program prikazuje dvodimenzionalno polje znakova u koje se formatirane vrijednosti podataka ubacuju preko naredbe `sprinf`.

```c
#include <stdio.h>
void main()
{
    int totalNumberOfEntries, numberOfEntries;
    printf("Koliko unosa zelis: ");
    scanf("%d", &totalNumberOfEntries);
    int temperatures[totalNumberOfEntries], humidities[totalNumberOfEntries];
    char temperatureHumidityIndexes[totalNumberOfEntries][200 + 1];
    for (numberOfEntries = 1; numberOfEntries <= totalNumberOfEntries;
numberOfEntries++)
```

```
    {
        printf("Unesi temperaturu %d. prostorije: ", numberOfEntries);
        scanf("%d", &temperatures[numberOfEntries - 1]);
        printf("Unesi vlaznost zraka %d. prostorije: ", numberOfEntries);
        scanf("%d", &humidities[numberOfEntries - 1]);
    }

    int indeks;
    float temperatureHumidityIndex;
    for (indeks = 0; indeks < totalNumberOfEntries; indeks++)
    {
        temperatureHumidityIndex = (1.8 * temperatures[indeks] + 32) - ((0.55 -
0.0055 * humidities[indeks]) * (1.8 * temperatures[indeks] - 26));
        sprintf(temperatureHumidityIndexes[indeks],
                "{\"Temperatura\": %d, \"Vlaznost\": %d, \"Indeks\": %.1f}",
                temperatures[indeks], humidities[indeks],
temperatureHumidityIndex);
    }
    printf("\n\nJSON FORMAT ZAPISA\n");
    for (indeks = 0; indeks < totalNumberOfEntries; indeks++)
    {
        printf("%d. %s\n", indeks + 1, temperatureHumidityIndexes[indeks]);
    }
}
```

Naredba `sprintf` omogućuje da se u polje znakova (`string`) spremi formatirana vrijednost. U ovom primjeru podatak se želi prikazati u sljedećem formatu {„Temperatura”: vrijednost, „Vlaznost”: vrijednost, „Indeks”: vrijednost} – što je tzv. format podataka JSON.
Sljedeća slika prikazuje rezultat izvođenja programa.

```
PS C:\C_Programs> cd "c:\C_Programs\" ; if ($?) { gcc Program_19.c -o Program_19 } ; if ($?) { .\Program_19 }
Koliko unosa zelis: 3
Unesi temperaturu 1. prostorije: 23
Unesi vlaznost zraka 1. prostorije: 55
Unesi temperaturu 2. prostorije: 18
Unesi vlaznost zraka 2. prostorije: 65
Unesi temperaturu 3. prostorije: 22
Unesi vlaznost zraka 3. prostorije: 67


JSON FORMAT ZAPISA
1. {"Temperatura": 23, "Vlaznost": 55, "Indeks": 69.6}
2. {"Temperatura": 18, "Vlaznost": 65, "Indeks": 63.2}
3. {"Temperatura": 22, "Vlaznost": 67, "Indeks": 69.1}
PS C:\C_Programs>
```

*Slika 1.5.3 Izvođenje programa kojim se stvara format podatka JSON*

### 1.5.3. Rješenje radnog zadatka

Računalni program treba se sastojati od četiri polja podataka:

- roomName, dvodimenzionalno polje u kojemu se za svaku prostoriju čuva njezin naziv
- temperatures i humidities, dva jednodimenzionalna polja u kojima se čuvaju unesene temperature i vlažnosti zraka
- temperatureHumidityIndexes – dvodimenzionalno polje koje u sebi čuva podatak sljedećeg JSON formata: {„Prostorija: vrijednost, „Temperatura“: vrijednost, „Vlaznost“: vrijednost, „Indeks“: vrijednost}

Slijedi rješenje radnog zadatka.

```c
#include <stdio.h>
void main()
{
    int totalNumberOfEntries, numberOfEntries;
    do
    {
        printf("Koliko unosa zelis: ");
        scanf("%d", &totalNumberOfEntries);
        if (totalNumberOfEntries < 0 || totalNumberOfEntries > 10)
        {
            printf("Broj unosa treba biti u intervalu od 0 do 10.\n");
            continue;
        }
        break;
    } while (1 == 1);
    char roomNames[totalNumberOfEntries][100 + 1];
    int temperatures[totalNumberOfEntries], humidities[totalNumberOfEntries];
    for (numberOfEntries = 1; numberOfEntries <= totalNumberOfEntries;
numberOfEntries = numberOfEntries + 1)
    {
        printf("\n\nUNOS %d/%d\n", numberOfEntries, totalNumberOfEntries);
        printf("Unesi naziv prostorije: ", numberOfEntries);
        scanf("%s", &roomNames[numberOfEntries - 1]);
        do
        {
            printf("Unesi temperaturu zraka (Celzijusi): ");
            scanf("%d", &temperatures[numberOfEntries - 1]);
            if (temperatures[numberOfEntries - 1] < 0 ||
temperatures[numberOfEntries - 1] > 50)
            {
                printf("Temperatura zraka treba biti u intervalu od 0 do 50
Celzijusa.\n");
                continue;
            }
            break;
        } while (1 == 1);
        do
        {

            printf("Unesi vlaznost zraka (%%): ");
            scanf("%d", &humidities[numberOfEntries - 1]);
            if (humidities[numberOfEntries - 1] < 0 || humidities[numberOfEntries
- 1] > 100)
            {
                printf("Vlaznost zraka treba biti u intervalu od 0 do 100
%%.\n");
                continue;
            }
            break;
        } while (1 == 1);
    }
    char temperatureHumidityIndexes[totalNumberOfEntries][300 + 1];
    int indeks;
    float temperatureHumidityIndex;
    for (indeks = 0; indeks < totalNumberOfEntries; indeks++)
    {
        temperatureHumidityIndex = (1.8 * temperatures[indeks] + 32) - ((0.55 -
0.0055 * humidities[indeks]) * (1.8 * temperatures[indeks] - 26));
```

```
        sprintf(temperatureHumidityIndexes[indeks],
                "{\"Prostorija\": %s, \"Temperatura\": %d, \"Vlaznost\": %d,
\"Indeks\": %.1f}",
                roomNames[indeks], temperatures[indeks], humidities[indeks],
temperatureHumidityIndex);
    }
    printf("\n\nJSON FORMAT ZAPISA\n");
    for (indeks = 0; indeks < totalNumberOfEntries; indeks++)
    {
        printf("%d. %s\n", indeks + 1, temperatureHumidityIndexes[indeks]);
    }
}
```

Sljedeća slika prikazuje rezultat izvođenja programa.

```
PS C:\C_Programs> cd "c:\C_Programs\" ; if ($?) { gcc Program_20.c -o Program_20 } ; if ($?) { .\Program_20 }
Koliko unosa zelis: 3


UNOS 1/3
Unesi naziv prostorije: Dvorana_1
Unesi temperaturu zraka (Celzijusi): 22
Unesi vlaznost zraka (%): 60


UNOS 2/3
Unesi naziv prostorije: Racunalni_kabinet
Unesi temperaturu zraka (Celzijusi): 23
Unesi vlaznost zraka (%): 65


UNOS 3/3
Unesi naziv prostorije: Laboratorij
Unesi temperaturu zraka (Celzijusi): 20
Unesi vlaznost zraka (%): 50


JSON FORMAT ZAPISA
1. {"Prostorija": Dvorana_1, "Temperatura": 22, "Vlaznost": 60, "Indeks": 68.6}
2. {"Prostorija": Racunalni_kabinet, "Temperatura": 23, "Vlaznost": 65, "Indeks": 70.4}
3. {"Prostorija": Laboratorij, "Temperatura": 20, "Vlaznost": 50, "Indeks": 65.3}
PS C:\C_Programs>
```

*Slika 1.5.4 Rezultat izvođenja programa za unos naziva prostorije te unos njezine temperature i vlažnosti zraka*

## 1.5.4. Pitanja i zadaci

1. Izmijenite program tako da se za temperaturu i vlažnost zraka koristi tip podatka `float`.
2. Izmijenite program tako da se temperatura i vlažnost zraka čuvaju u jednom dvodimenzionalnom polju.
3. Polje podataka omogućuje čuvanje više podataka, ali koji svi moraju biti istog tipa podatka.
    a. Točno
    b. Netočno
4. Polje znakova jest polje podataka koje je deklarirano tipom podatka `char`.
    a. Točno
    b. Netočno
5. Ako je polje podataka deklarirano kao `int numbers[5]`, tada se radi o sljedećem:
    a. jednodimenzionalnom polju
    b. dvodimenzionalnom polju

c. trodimenzionalnom polju
d. četverodimenzionalnom polju
e. peterodimenzionalnom polju

6. Ako je polje podataka deklarirano kao `float numbers[5]`, tada se do njegova drugog elementa dolazi na sljedeći način:
   a. numbers[0]
   b. numbers[1]
   c. numbers[2]
   d. numbers[3]
   e. numbers[4]
7. Ako je polje podataka deklarirano kao `float numbers[5]`, maksimalni broj elemenata koji se u polje može spremiti jest četiri.
   a. Točno
   b. Netočno
8. Ako je polje podataka deklarirano kao `char word[10]`, u polje se može spremiti riječ duljine do 10 znakova.
   a. Točno
   b. Netočno

### 1.5.5. Literatura i izvori


1. C Programming Language Tutorial, https://www.javatpoint.com/c-programming-language-tutorial
2. C Tutorial, https://www.tutorialspoint.com/cprogramming/index.htm
3. C Tutorial, https://www.w3schools.in/c-tutorial
4. Jakupović, A.; Šuman, S.: Osnove programiranja, https://www.veleri.hr/sites/default/files/2021-07/osnove_programiranja_konacna_verzija_online_verzija.pdf


## 1.6. Pokazivači

Varijabla u programiranju predstavlja koncept koji omogućuje uporabu memorije računala. Tijekom uporabe računalo za neku varijablu rezervira memorijsku lokaciju u kojoj će se čuvati neka vrijednost (o veličini memorijske lokacije kao i o svojstvima vrijednosti koje će se čuvati govori tip podatka varijable). Do rezervirane memorijske lokacije, odnosno do vrijednost koja se u njoj čuva može se doći na dva načina: preko identifikatora (naziva) varijable ili preko adrese memorijske lokacije koja je dodijeljena varijabli. Koncept pokazivača omogućuje primjenu adresa memorijske lokacije (adresa varijable). Obradit će se deklaracija pokazivača, adresni operator, operator dereferenciranja ili indirekcije te osnovni rad s pokazivačima.

**Nakon usvajanja ove nastavne teme čitatelj će moći:**
1. deklarirati i inicijalizirati pokazivač na proizvoljni tip podatka
2. pristupiti vrijednosti memorijskog prostora na koji pokazuje pokazivač
3. koristiti pokazivače u funkcijama za prijenos jedne vrijednosti ili polja podataka
4. dinamički alocirati memoriju za jedan podatak ili polje podataka
5. ispravno upravljati dinamički alociranom memorijom

### 1.6.1. Opis radnog zadatka

Potrebno je izraditi računalni program kojim se računa indeks temperature i vlažnosti zraka (engl. *Temperature Humidity Index* – THI). Taj se indeks koristi kako bi se procijenila razina termalne udobnosti ljudi i životinja u nekom prostoru. Formula kojom se računa THI glasi:

$$THI = (1.8 \cdot T + 32) - \big((0.55 - 0.0055 \cdot RH) \cdot (1.8 \cdot T - 26)\big)$$

gdje je:
*THI* – indeks temperature i vlažnosti
*T* – temperatura zraka u celzijusima
*RH* – vlažnost zraka u postocima

Izračunati THI označava razinu termalne udobnosti prema sljedećoj tablici:

| THI | Razina termalne udobnosti |
|---|---|
| < 72 | UGODNO |
| 72 – 79 | PODNOŠLJIVO |
| 80 – 89 | NEUGODNO |
| 90 – 99 | VRLO NEUGODNO |
| > 99 | OPASNO PO ZDRAVLJE |

Korisnika je potrebno pitati koliko unosa želi napraviti (maksimalni broj unosa jest 10). Nakon toga potrebno ga je toliko puta pitati za unos naziva prostorije, temperature i vlažnosti zraka. Kod svakog unosa potrebno je provjeriti ispravnost unesenih podataka (temperatura treba biti u intervalu od 0 do 50, a vlažnost zraka u intervalu od 0 do 100). Nakon završetka unosa temperature i vlažnosti zraka potrebno je prikazati iznos indeksa temperature i vlažnosti (THI) te razine termalne udobnosti.
Program treba strukturirati tako da se sastoji od četiri funkcije:

- getRoomNames – kroz dvodimenzionalno polje vraća unesene nazive prostorija
- getTemperatures – kroz jednodimenzionalno polje vraća vrijednost unesenih temperatura
- getHumidities – kroz jednodimenzionalno polje vraća vrijednost unesenih vlažnosti zraka
- getTemperatureHumidityIndexes – kroz dvodimenzionalno polje vraća podatak u formatu JSON: `{„Prostorija”: vrijednost, „Temperatura”: vrijednost, „Vlaznost”: vrijednost, „Indeks”: vrijednost}`

Vrijednost broja unosa treba se čuvati u pokazivaču `totalNumberOfEntries`.

### 1.6.2. Osnovni koncepti

Svaka varijabla deklarirana u računalnom programu ima dodijeljenu memorijsku lokaciju, a svaka memorijska lokacija ima svoju adresu. U dodijeljenoj memorijskoj lokaciji čuvaju se vrijednosti koje se kroz program dodjeljuju varijabli. Kakve se vrijednosti varijabli mogu dodijeliti ovisi o tipu podatka kojim je varijabla deklarirana. Postoji posebna vrsta varijabli, pokazivači, koje kao vrijednost čuvaju adresu memorijske lokacije neke druge varijable (i pokazivači imaju svoju memorijsku lokaciju i adresu). Oni preko adrese pokazuju na memorijsku lokaciju koja čuva stvarnu vrijednost podatka s kojom se dalje u programu obavlja procesiranje.
Opći oblik deklaracije varijable koja je pokazivač glasi:

```
tip_podatka *ime_Varijable;
```

Može se zaključiti da je deklaracija pokazivača vrlo slična deklaraciji bilo koje druge varijable s razlikom što se ispred naziva varijable stavlja znak *.
Sljedeći primjer prikazuje program kojim se od korisnika traži željeni broj unos te se taj uneseni broj ispisuje. Pokazivač se koristi za čuvanje unesene vrijednosti.

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
    int *totalNumberOfEntries;
    totalNumberOfEntries = (int *)malloc(sizeof(int));
    do
    {
        printf("Koliko unosa zelis: ");
        scanf("%d", totalNumberOfEntries);
        if (*totalNumberOfEntries < 0 || *totalNumberOfEntries > 10)
        {
            printf("Broj unosa treba biti u intervalu od 0 do 10.\n");
            continue;
        }
        break;
    } while (1 == 1);
    printf("\nBroj unosa je: %d", *totalNumberOfEntries);
}
```

Treba primijetiti uključivanje dodatne vanjske datoteke `stdlib.h` koja je potrebna zbog dinamičke alokacije memorije. Naime, kada se deklarira varijabla (npr. `int number`), tada računalo (odnosno operacijski sustav) rezervira memorijsku lokaciju za tu varijablu čija veličina i način interpretacije bitova na toj memorijskoj lokaciji odgovaraju tipu podatka deklaracije varijable. No ako se deklarira pokazivač (npr. `int *number`), tada računalo rezervira memorijsku lokaciju u kojoj će se čuvati adresa, ali ne i memorijsku lokaciju na kojoj će se nalaziti stvarni podaci koji su potrebni programu za procesiranje. Zato je potrebno računalu narediti rezervaciju nove memorijske lokacije, a pri tome treba navesti koja je veličina memorijske lokacije potrebna te kako će se bitovi na toj memorijskoj lokaciji interpretirati. Taj se postupak naziva dinamička alokacija memorije i provodi se preko naredbi `malloc` i `sizeof`. Naredba `malloc` vraća adresu nove memorijske lokacije koja se može spremiti u varijablu tipa pokazivač (vidjeti `totalNumberOfEntries = (int *)malloc(sizeof(int));`).
Budući da se u samom pokazivaču nalazi adresa memorijske lokacije na kojoj se treba spremiti neki podatak, treba primijetiti da u naredbi `scanf` ispred varijable nema adresnog operatora & (taj operator vraća adresu memorijske lokacije koja je rezervirana za varijablu). Nadalje, u provjeri valjanosti podatka koji se nalazi na adresi memorijske lokacije koja je spremljena kao vrijednost u pokazivaču koristi se operator dereferenciranja ili indirekcije *. Njime se preko adrese memorijske lokacije dolazi do vrijednosti koja je u njoj zapisana (vidjeti npr. `(*totalNumberOfEntries < 0)`.
Sljedeća slika prikazuje rezultat izvođenja programa.

```
PS C:\C_Programs> cd "c:\C_Programs\" ; if ($?) { gcc Program_21.c -o Program_21 } ; if ($?) { .\Program_21 }
Koliko unosa zelis: 45
Broj unosa treba biti u intervalu od 0 do 10.
Koliko unosa zelis: 5

Broj unosa je: 5
PS C:\C_Programs>
```

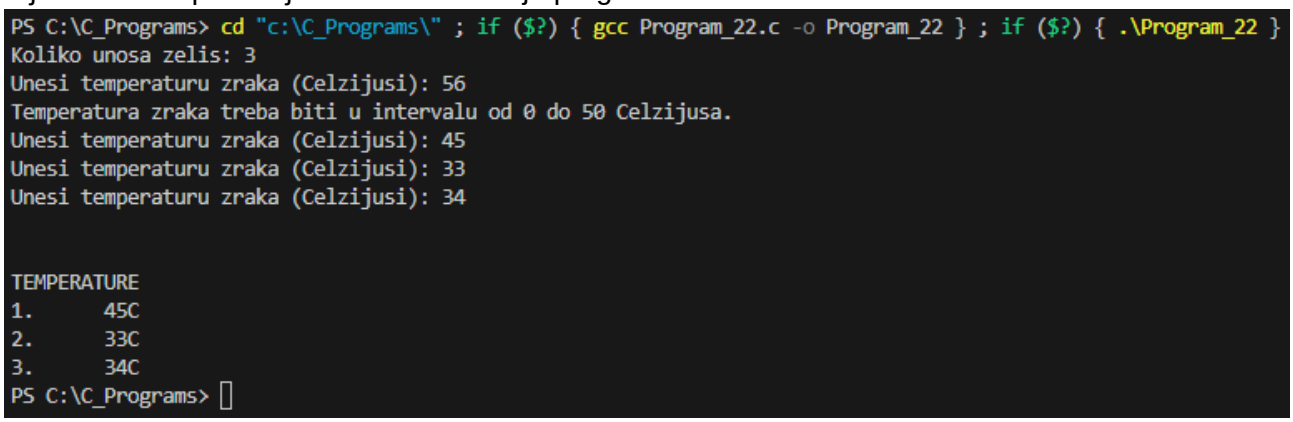
*Slika 1.6.1 Izvođenje programa za specifikaciju broja unosa podataka primjenom pokazivača*

Sljedeći primjer prikazuje unos temperature u jednodimenzionalno polje koje se obavlja u funkciji i koja mjestu poziva vraća ispunjeno polje.

```c
#include <stdio.h>
#include <stdlib.h>

void getTemperatures(int argTemperatures[], int argTotalNumberOfEntries)
{
    int numberOfEntries;
    for (numberOfEntries = 1; numberOfEntries <= argTotalNumberOfEntries;
numberOfEntries++)
    {
        do
        {
            printf("Unesi temperaturu zraka (Celzijusi): ");
            scanf("%d", &argTemperatures[numberOfEntries - 1]);
            if (argTemperatures[numberOfEntries - 1] < 0 ||
argTemperatures[numberOfEntries - 1] > 50)
            {
                printf("Temperatura zraka treba biti u intervalu od 0 do 50
Celzijusa.\n");
                continue;
            }
            break;
        } while (1 == 1);
    }
}

void main()
{
    int *totalNumberOfEntries;
    totalNumberOfEntries = (int *)malloc(sizeof(int));
    do
    {
        printf("Koliko unosa zelis: ");
        scanf("%d", totalNumberOfEntries);
        if (*totalNumberOfEntries < 0 || *totalNumberOfEntries > 10)
        {
            printf("Broj unosa treba biti u intervalu od 0 do 10.\n");
            continue;
        }
        break;
    } while (1 == 1);

    int temperatures[*totalNumberOfEntries];

    getTemperatures(temperatures, *totalNumberOfEntries);
    printf("\n\nTEMPERATURE\n");
    int indeks;
    for (indeks = 0; indeks < *totalNumberOfEntries; indeks++)
    {
        printf("%d.\t%dC\n", indeks + 1, temperatures[indeks]);
    }
}
```

Treba primijeniti argument funkcije `getTemperatures` – `int argTemperatures[]`. To je način na koji se deklarira argument kroz koji će se poslati jednodimenzionalno polje u koje će se ubaciti podaci i koje će se vratiti mjestu poziva. Također treba primijetiti način na koji se funkcija `getTemperatures` poziva, odnosno na koji joj se način šalju argumenti. Prvi je argument polje `temperatures`, a drugi vrijednost koja se čuva na adresi koja se nalazi u pokazivaču `totalNumberOfEntries` – zato se tu koristi operator dereferenciranja ili indirekcije *.

Sljedeća slika prikazuje rezultat izvođenja programa.




*Slika 1.6.2 Izvođenje programa unosa temperature u jednodimenzionalno polje poslano kao funkciji kao argument*

Sljedeći primjer prikazuje unos naziva prostorije u dvodimenzionalno polje koje se obavlja u funkciji i koja mjestu poziva vraća ispunjeno polje.

```
#include <stdio.h>
#include <stdlib.h>

void getRoomNames(char argRoomNames[][100 + 1], int argTotalNumberOfEntries)
{
    int numberOfEntries;
    for (numberOfEntries = 1; numberOfEntries <= argTotalNumberOfEntries;
numberOfEntries++)
    {
        printf("Unesi naziv %d. prostorije: ", numberOfEntries);
        scanf("%s", &argRoomNames[numberOfEntries - 1]);
    }
}

void main()
{
    int *totalNumberOfEntries;
    totalNumberOfEntries = (int *)malloc(sizeof(int));
    do
    {
        printf("Koliko unosa zelis: ");
        scanf("%d", totalNumberOfEntries);
        if (*totalNumberOfEntries < 0 || *totalNumberOfEntries > 10)
        {
            printf("Broj unosa treba biti u intervalu od 0 do 10.\n");
            continue;
        }
        break;
    } while (1 == 1);

    char roomNames[*totalNumberOfEntries][100 + 1];

    getRoomNames(roomNames, *totalNumberOfEntries);
    printf("\n\nPOPIS PROSTORIJA\n");
    int indeks;
    for (indeks = 0; indeks < *totalNumberOfEntries; indeks++)
    {
        printf("%d.\t%s\n", indeks + 1, roomNames[indeks]);
```

```
    }
}
```

Deklaracija argumenta kroz koji funkcija prima dvodimenzionalno polje jest `char argRoomNames[][100 + 1]`. Treba primijeniti da se prva dimenzija ne treba specificirati, dok se sve ostale dimenzije moraju navesti.

Sljedeća slika prikazuje rezultat izvođenja programa.

```
PS C:\C_Programs> cd "c:\C_Programs\" ; if ($?) { gcc Program_23.c -o Program_23 } ; if ($?) { .\Program_23 }
Koliko unosa zelis: 4
Unesi naziv 1. prostorije: Dvorana_1
Unesi naziv 2. prostorije: Kabinet_2
Unesi naziv 3. prostorije: Laboratorij_1
Unesi naziv 4. prostorije: Laboratorij_2


POPIS PROSTORIJA
1.      Dvorana_1
2.      Kabinet_2
3.      Laboratorij_1
4.      Laboratorij_2
PS C:\C_Programs>
```

*Slika 1.6.3 Izvođenje programa unosa naziva prostorije preko funkcije kojoj se dostavlja dvodimenzionalno polje*

## 1.6.3. Rješenje radnog zadatka

Program koji se treba izraditi mora biti strukturiran tako da se sastoji od sljedećih funkcija:

- getRoomNames – kroz dvodimenzionalno polje vraća unesene nazive prostorija
- getTemperatures – kroz jednodimenzionalno polje vraća vrijednost unesenih temperatura
- getHumidities – kroz jednodimenzionalno polje vraća vrijednost unesenih vlažnosti zraka
- getTemperatureHumidityIndexes – kroz dvodimenzionalno polje vraća podatak u formatu JSON: `{„Prostorija”: vrijednost, „Temperatura”: vrijednost, „Vlaznost”: vrijednost, „Indeks”: vrijednost}`

Također se vrijednost broja unosa treba čuvati u pokazivaču `totalNumberOfEntries`.

Slijedi rješenje radnog zadatka.

```
#include <stdio.h>
#include <stdlib.h>

void getRoomNames(char argRoomNames[][100 + 1], int argTotalNumberOfEntries)
{
    int numberOfEntries;
    for (numberOfEntries = 1; numberOfEntries <= argTotalNumberOfEntries;
numberOfEntries++)
    {
        printf("Unesi naziv %d. prostorije: ", numberOfEntries);
        scanf("%s", &argRoomNames[numberOfEntries - 1]);
    }
}

void getTemperatures(int argTemperatures[], int argTotalNumberOfEntries)
{
    int numberOfEntries;
    for (numberOfEntries = 1; numberOfEntries <= argTotalNumberOfEntries;
numberOfEntries++)
    {
```

```
        printf("\nUNOS %d/%d\n", numberOfEntries, argTotalNumberOfEntries);
        do
        {
            printf("Unesi temperaturu zraka (Celzijusi): ");
            scanf("%d", &argTemperatures[numberOfEntries - 1]);
            if (argTemperatures[numberOfEntries - 1] < 0 ||
argTemperatures[numberOfEntries - 1] > 50)
            {
                printf("Temperatura zraka treba biti u intervalu od 0 do 50
Celzijusa.\n");
                continue;
            }
            break;
        } while (1 == 1);
    }
}
void getHumidities(int argHumidities[], int argTotalNumberOfEntries)
{
    int numberOfEntries;
    for (numberOfEntries = 1; numberOfEntries <= argTotalNumberOfEntries;
numberOfEntries++)
    {
        printf("\nUNOS %d/%d\n", numberOfEntries, argTotalNumberOfEntries);
        do
        {

            printf("Unesi vlaznost zraka (%%): ");
            scanf("%d", &argHumidities[numberOfEntries - 1]);
            if (argHumidities[numberOfEntries - 1] < 0 ||
argHumidities[numberOfEntries - 1] > 100)
            {
                printf("Vlaznost zraka treba biti u intervalu od 0 do 100
%%.\n");
                continue;
            }
            break;
        } while (1 == 1);
    }
}

void getTemperatureHumidityIndexes(char argTemperatureHumidityIndexes[][300 + 1],
                                   char argRoomNames[][100 + 1],
                                   int argTemperatures[],
                                   int argHumidities[],
                                   int argTotalNumberOfEntries)
{
    int index;
    float temperatureHumidityIndex;
    for (index = 0; index < argTotalNumberOfEntries; index++)
    {
        temperatureHumidityIndex = (1.8 * argTemperatures[index] + 32) - ((0.55 -
0.0055 * argHumidities[index]) * (1.8 * argTemperatures[index] - 26));
        sprintf(argTemperatureHumidityIndexes[index],
                "{\"Prostorija\": %s, \"Temperatura\": %d, \"Vlaznost\": %d,
\"Indeks\": %.1f}",
                argRoomNames[index], argTemperatures[index],
argHumidities[index], temperatureHumidityIndex);
    }
```

```
}

void main()
{
    int *totalNumberOfEntries;
    totalNumberOfEntries = (int *)malloc(sizeof(int));
    do
    {
        printf("Koliko unosa zelis: ");
        scanf("%d", totalNumberOfEntries);
        if (*totalNumberOfEntries < 0 || *totalNumberOfEntries > 10)
        {
            printf("Broj unosa treba biti u intervalu od 0 do 10.\n");
            continue;
        }
        break;
    } while (1 == 1);

    char roomNames[*totalNumberOfEntries][100 + 1];
    int temperatures[*totalNumberOfEntries], humidities[*totalNumberOfEntries];
    char temperatureHumidityIndexes[*totalNumberOfEntries][300 + 1];

    getRoomNames(roomNames, *totalNumberOfEntries);
    getTemperatures(temperatures, *totalNumberOfEntries);
    getHumidities(humidities, *totalNumberOfEntries);
    getTemperatureHumidityIndexes(temperatureHumidityIndexes, roomNames,
temperatures, humidities, *totalNumberOfEntries);

    int index;
    printf("\n\nJSON FORMAT ZAPISA\n");
    for (index = 0; index < *totalNumberOfEntries; index++)
    {
        printf("%d. %s\n", index + 1, temperatureHumidityIndexes[index]);
    }
}
```

Sljedeća slika prikazuje rezultat izvođenja programa.

```
PS C:\C_Programs> cd "c:\C_Programs\" ; if ($?) { gcc Program_24.c -o Program_24 } ; if ($?) { .\Program_24 }
Koliko unosa zelis: 4
Unesi naziv 1. prostorije: Dvorana_1
Unesi naziv 2. prostorije: Kabinet_1
Unesi naziv 3. prostorije: Kabinet_2
Unesi naziv 4. prostorije: Laboratorij

UNOS 1/4
Unesi temperaturu zraka (Celzijusi): 20

UNOS 2/4
Unesi temperaturu zraka (Celzijusi): 21

UNOS 3/4
Unesi temperaturu zraka (Celzijusi): 22

UNOS 4/4
Unesi temperaturu zraka (Celzijusi): 23

UNOS 1/4
Unesi vlaznost zraka (%): 55

UNOS 2/4
Unesi vlaznost zraka (%): 50

UNOS 3/4
Unesi vlaznost zraka (%): 66

UNOS 4/4
Unesi vlaznost zraka (%): 56


JSON FORMAT ZAPISA
1. {"Prostorija": Dvorana_1, "Temperatura": 20, "Vlaznost": 55, "Indeks": 65.5}
2. {"Prostorija": Kabinet_1, "Temperatura": 21, "Vlaznost": 50, "Indeks": 66.6}
3. {"Prostorija": Kabinet_2, "Temperatura": 22, "Vlaznost": 66, "Indeks": 69.1}
4. {"Prostorija": Laboratorij, "Temperatura": 23, "Vlaznost": 56, "Indeks": 69.7}
PS C:\C_Programs>
```

*Slika 1.6.4 Izvođenje programa za izračun THI-ja preko slanja polja podataka funkcijama*

## 1.6.4. Pitanja i zadaci

1. Izmijenite program tako da se za temperaturu i vlažnost zraka koriste polja tipa podatka `float`.
2. Izmijenite program tako da se umjesto dvije funkcije `getTemperatures` i `getHumidities` koristi samo jedna.`getTemperaturesAndHumidities`.
3. Izmijenite program tako da se temperatura i vlažnost zraka čuvaju u jednom dvodimenzionalnom polju koje se šalje jednoj funkciji kao argument.
4. Ispravna deklaracija pokazivača glasi:
    a. *int number;
    b. int *number;
    c. int number*;
    d. int* *number;
    e. *int *number;
5. Vrijednost pokazivača jest adresa neke memorijske lokacije.
    a. Točno
    b. Netočno
6. Do vrijednosti koja se čuva na memorijskoj lokaciji na koju pokazuje pokazivač dolazi se preko adresnog operatora.

    a. Točno
    b. Netočno
7. Adresni operator koristi se kako bi se doznala adresa memorijske lokacije koja je rezervirana za neku varijablu.
    a. Točno
    b. Netočno
8. Adresni operator jest:
    a. *
    b. $
    c. ?
    d. &
    e. %
9. Operator dereferenciranja ili indirekcije jest:
    a. *
    b. $
    c. ?
    d. &
    e. %
10. Operator dereferenciranja ili indirekcije koristi se kako bi se došlo do vrijednosti koja se čuva na memorijskoj lokaciji čija je adresa spremljena u pokazivač.
    a. Točno
    b. Netočno

### 1.6.5. Literatura i izvori


1. C Programming Language Tutorial, https://www.javatpoint.com/c-programming-language-tutorial
2. C Tutorial, https://www.tutorialspoint.com/cprogramming/index.htm
3. C Tutorial, https://www.w3schools.in/c-tutorial
4. Jakupović, A.; Šuman, S.: Osnove programiranja, https://www.veleri.hr/sites/default/files/2021-07/osnove_programiranja_konacna_verzija_online_verzija.pdf

# 2. PROGRAMIRANJE MIKROKONTROLERA ESP32

U ovoj nastavnoj cjelini obrađuju se sljedeći koncepti koji se odnose na programiranje mikrokontrolera: povezivanje mikrokontrolera na razvojno računalo, spajanje periferije na ulazno-izlazne priključke mikrokontrolera, programiranje jednostavnijih programa kojima se upravlja ulazno-izlaznim priključcima mikrokontrolera, povezivanje mikrokontrolera na bežičnu mrežu, izrada i poziv *web*-aplikacijskoga programskog sučelja te izrada baze podataka. Svi opisani koncepti završavaju opisom provedbe konkretnoga radnog zadatka. Na kraju se nalaze dodatna pitanja i zadaci te popis literature i drugih izvora.
Cilj je ove nastavne jedinice osposobiti čitatelja za spajanje jednostavnije periferije na mikrokontroler te za programiranje jednostavnijih programa za upravljanje tom periferijom.

## 2.1. Povezivanje mikrokontrolera ESP32 s razvojnim računalom

Mikrokontroler ESP32 potrebno je povezati s razvojnim računalom s pomoću USB kabela. Potrebno je na razvojno računalo instalirati odgovarajuće upravljačke programe.

**Nakon usvajanja ove nastavne teme čitatelj će moći:**
1. Instalirati, konfigurirati odabrane programske potpore za razvoj vlastitih rješenja zasnovanih na internetu stvari
2. Proširiti osnovnu funkcionalnost programske potpore pomoću javno dostupnih biblioteka

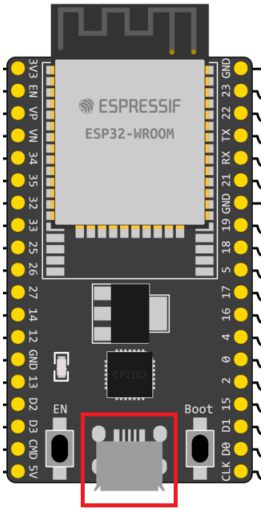
### 2.1.1. Osnovni koncepti

Mikrokontroler je malo računalo na jednom integriranom krugu koje ima funkciju centralne procesorske jedinice, memorije i ulazno/izlaznih sučelja. Mikrokontroleri se uvelike koriste u tzv. ugradbenim sustavima (hardversko-softverskim sustavima koji obavljaju neku funkciju bilo kao neovisni sustav ili kao dio nekog većeg sustava), npr. u kućanskim aparatima, automobilskim sustavima, medicinskim uređajima i industrijskim kontrolnim sustavima. Također se koriste u proizvodima potrošačke elektronike kao što su sustavi za igre, digitalne kamere i audiosustavi.
Obično se mikrokontroler sastoji od procesora, postojane (trajne) i nepostojane (privremene) memorije, ulazno/izlaznih sučelja i komunikacijskog sučelja. Procesor je zadužen za izvođenje programskih instrukcija i kontrolu ostalih dijelova mikrokontrolera. Memorija se koristi za čuvanje podataka i programskog koda, dok se ulazno/izlazno i komunikacijsko sučelje koristi za povezivanje mikrokontrolera s okolinom (npr. LED, tipkalo, senzori LCD zaslon i sl.).
Mikrokontroler se može programirati u nekom programskom jeziku (npr. C) te je u tu svrhu potrebno uspostaviti razvojno okruženje koje se sastoji od razvojnog računala i na njemu instaliranoga potrebnog softvera (npr. Visual Studio Code, dodaci za programiranje u C, razvojni okvir, upravljački programi i sl.)
Jedan od mikrokontrolera koji je razvio Espressif jest ESP32. Njegove osnovne značajke jesu:

- niska cijena
- niska potrošnja električne energije
- mogućnost povezivanja putem Wi-Fija
- mogućnost Bluetootha
- dvojezgreni procesor
- bogata ulazno/izlazna sučelja (npr. analogno-digitalni konverter – ADC, digitalno-analogni konverter – DAC, pulsno-širinska modulacija – PWM, serijska komunikacija – UART, I2C komunikacija, SPI komunikacija itd.)
- kompatibilan je s programskim jezikom Arduino

- kompatibilan je s MicroPythonom – primjena programskog jezika Python

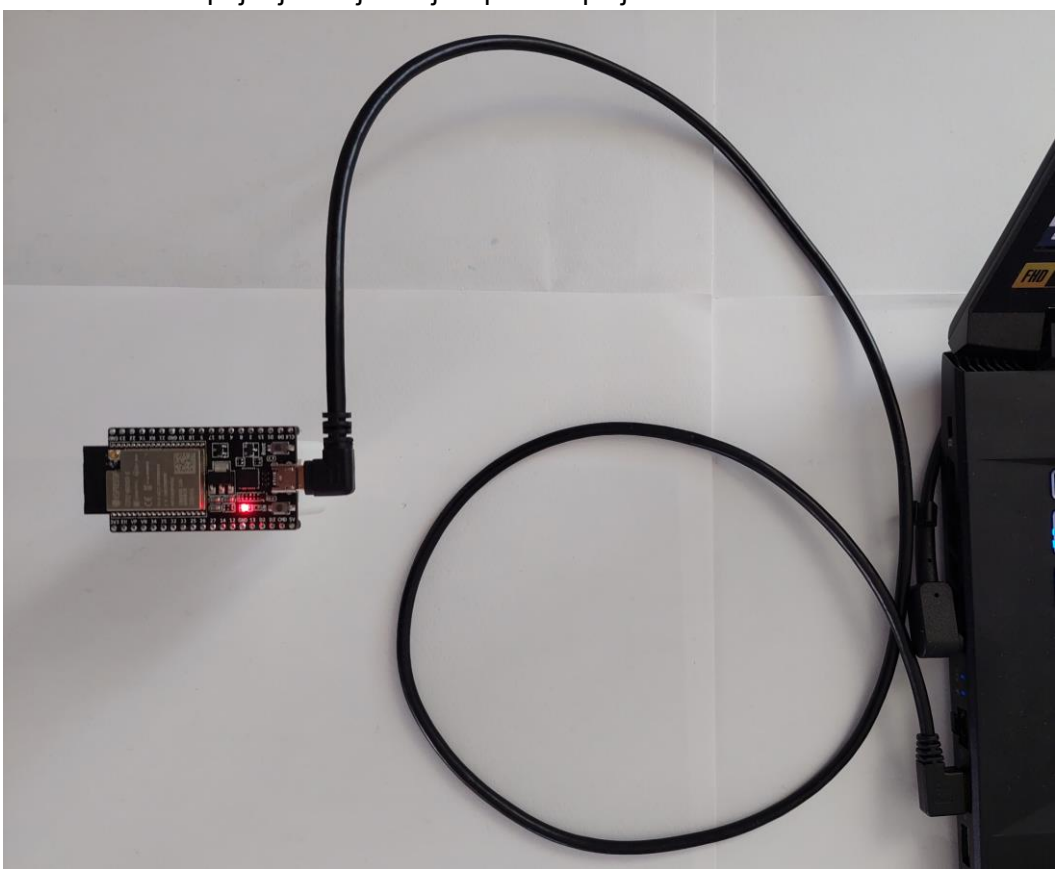
Mikrokontroler ESP32 s razvojnim se računalom povezuje preko USB kabela. Naredna slika prikazuje USB priključak preko kojega se mikrokontroler povezuje s razvojnim računalom. Preko USB kabela mikrokontroler dobiva napajanje (5 V) te se preko istog kabela na mikrokontroler prebacuje izrađeni program.




*Slika 2.1.1USB priključak mikrokontrolera ESP32*

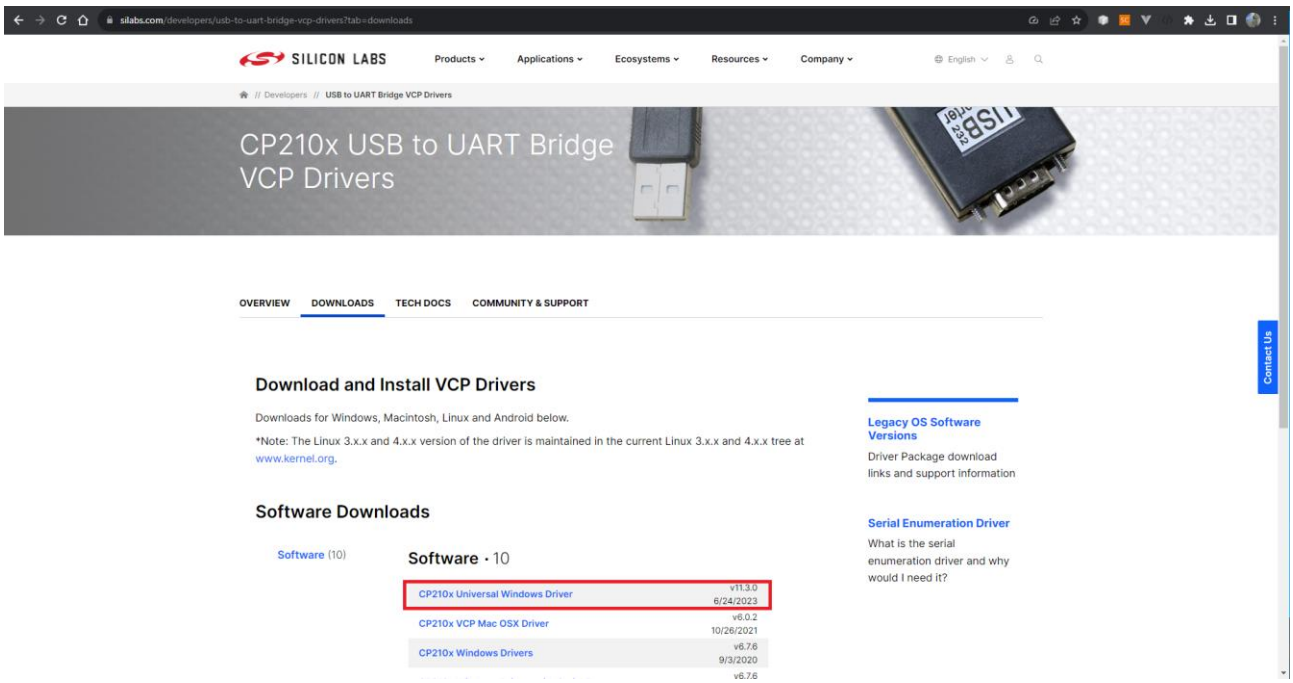
## 2.1.2. Rješenje radnog zadatka

Na razvojno računalo potrebno je povezati mikrokontroler ESP32 s pomoću USB kabela. Na taj način mikrokontroler dobiva napajanje što je vidljivo preko upaljene crvene LED.



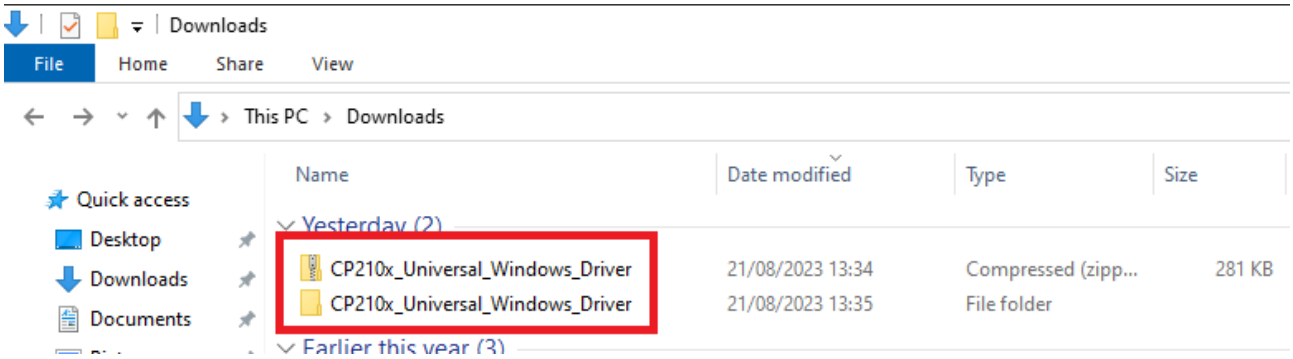
*Slika 2.1.2 Mikrokontroler ESP32 povezan s razvojnim računalom preko USB kabela*

Sa stranice https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers?tab=downloads potrebno je na razvojno računalo preuzeti upravljački program CP210x Universal Windows Driver (vidjeti sliku).



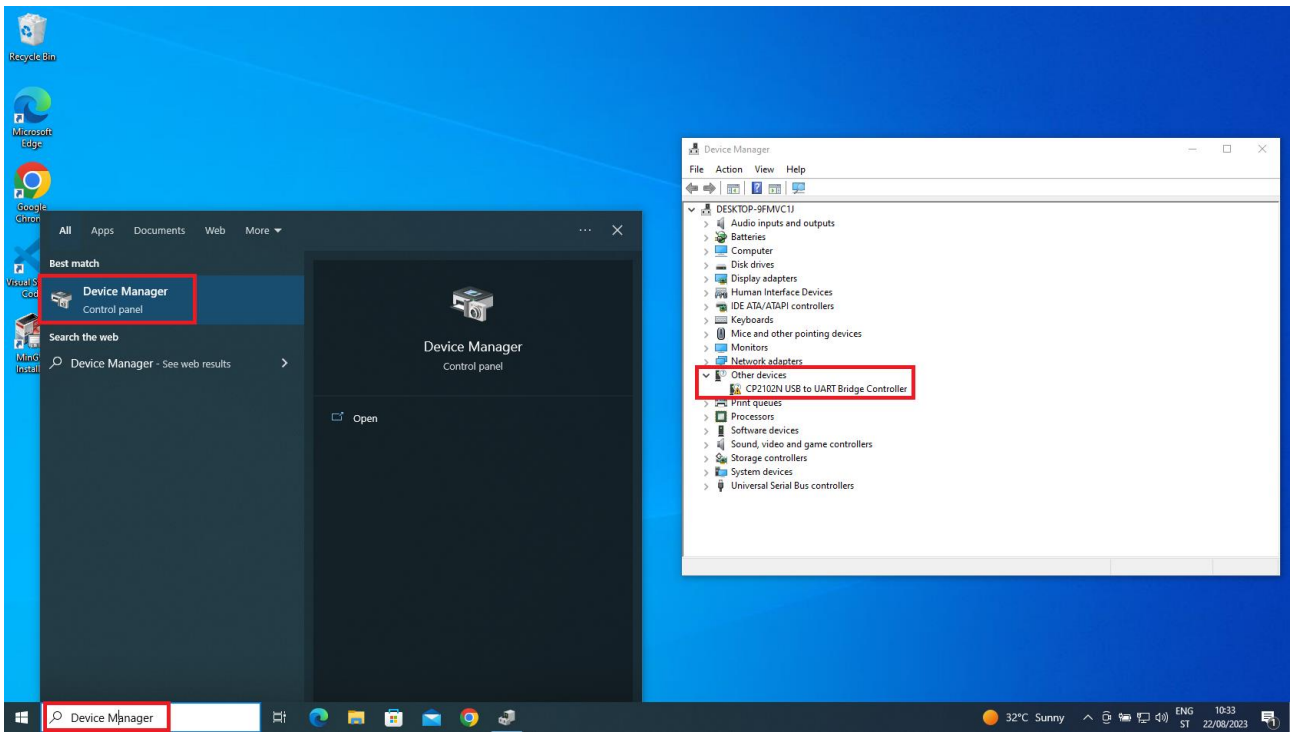

*Slika 2.1.3 Preuzimanje odgovarajućeg drivera*

Nakon preuzimanja datoteku je potrebno raspakirati.



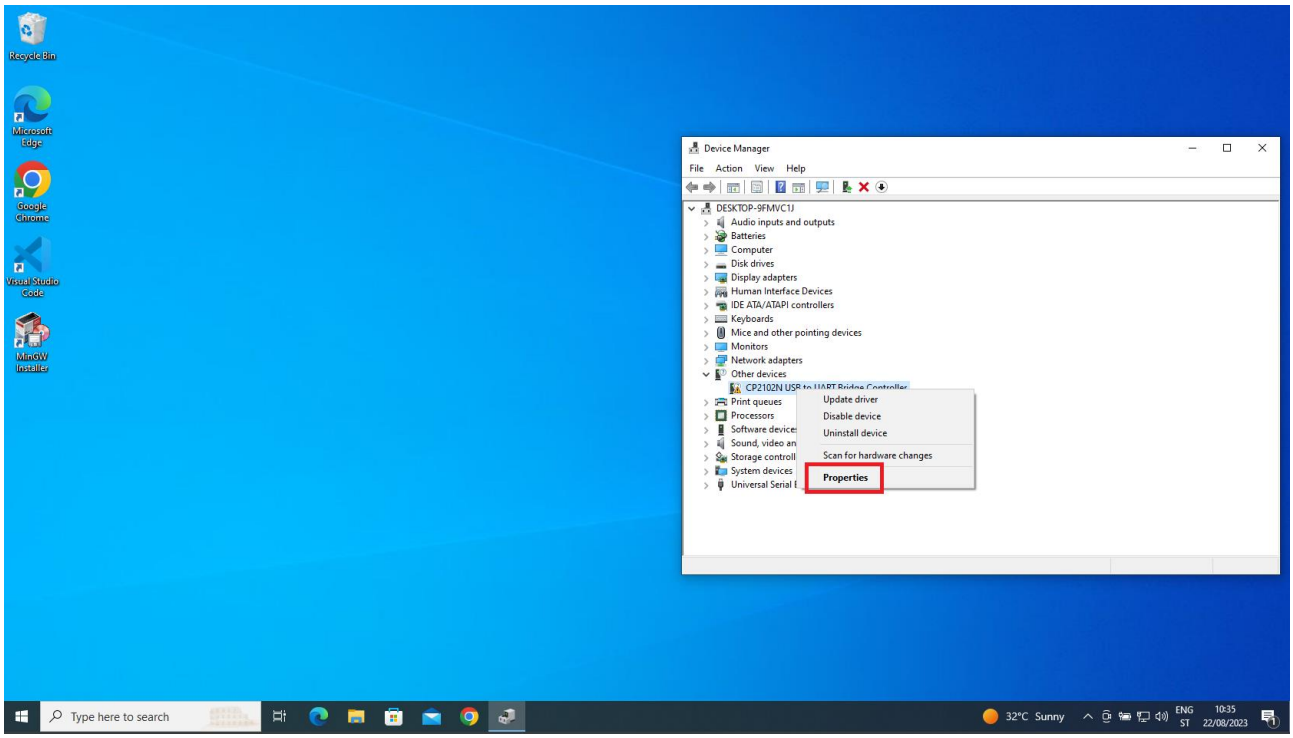

*Slika 2.1.4 Preuzeta i raspakirana datoteka*

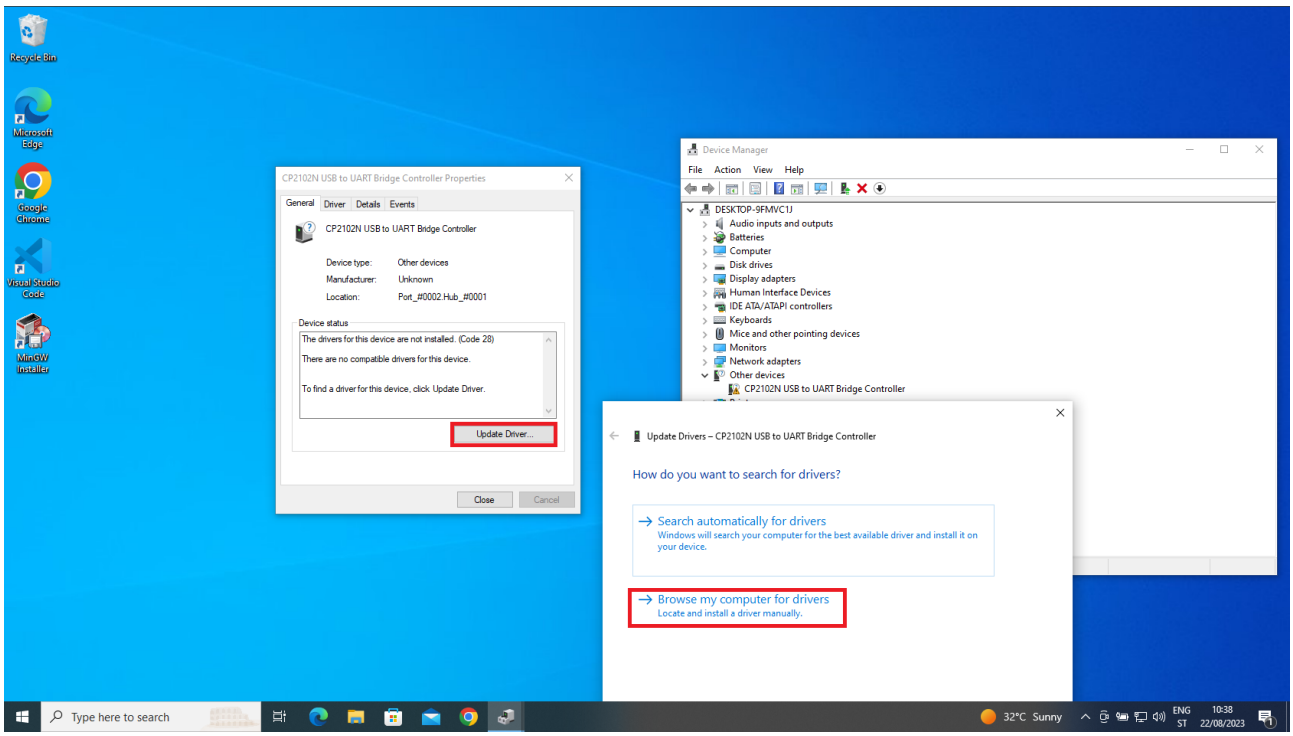
Na razvojnom računalu potrebno je pokrenuti Device Manager kako bi se instalirao preuzeti upravljački program.




*Slika 2.1.5 Pokretanje Device Managera i pronalazak priključenog mikrokontrolera*

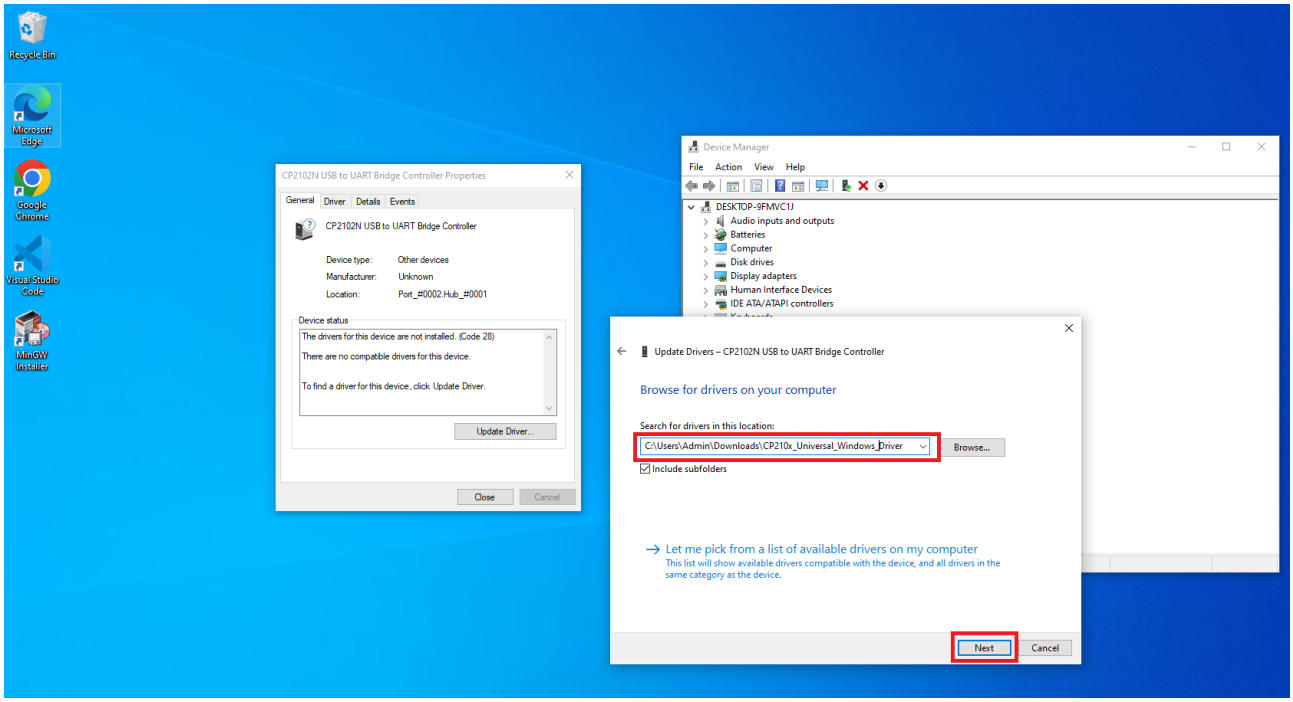
Na priključeni mikrokontroler potrebno jekliknuti desnim gumbom miša i izabrati opciju Properties. Na otvorenom prozoru potrebno je kliknuti na gumb Update Driver… nakon toga te odabrati opciju Browse my computer for drivers.





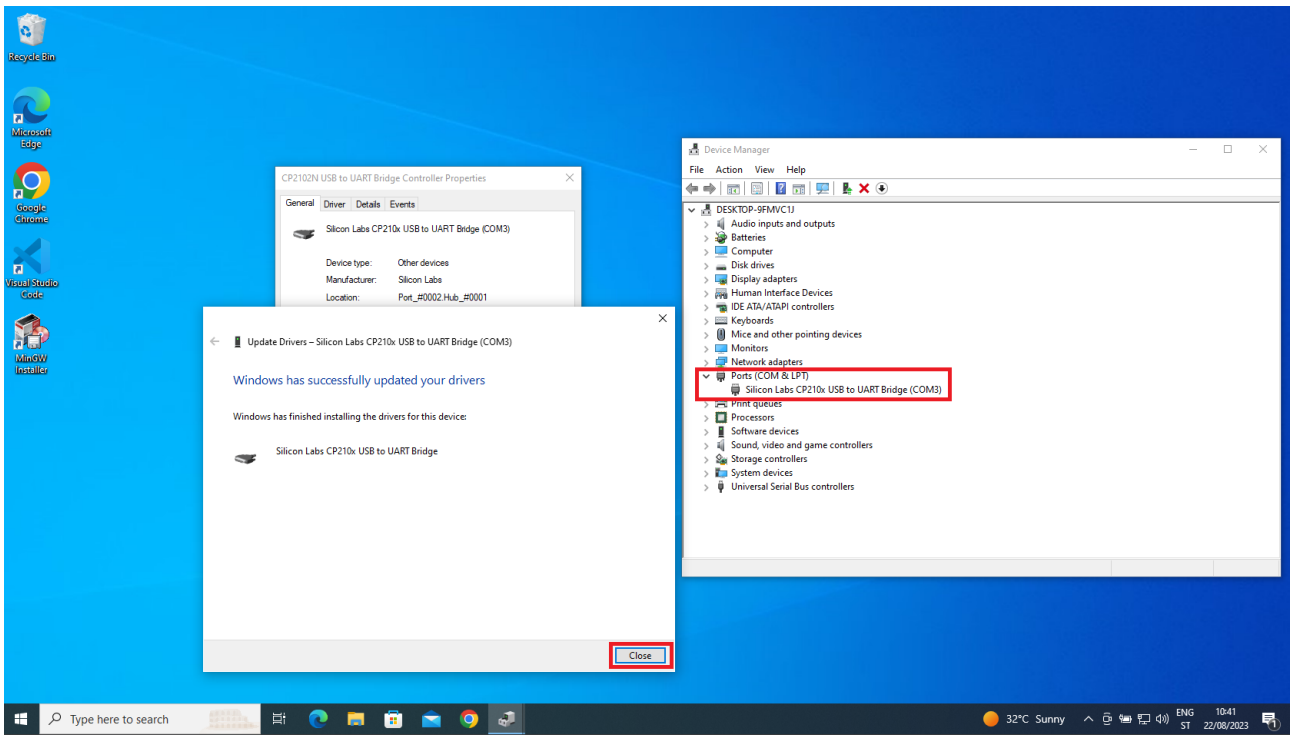

*Slika 2.1.6 Priprema za instalaciju upravljačkog programa*

Potrebno je odabrati putanju do mape u kojoj je otpakirana preuzeta datoteka (upravljački program).




*Slika 2.1.7 Odabir putanje do upravljačkog programa*

Nakon uspješne instalacije upravljačkog programa može se vidjeti da je priključenom mikrokontroleru dodijeljen port COM3 (taj je port potrebno zapamtiti jer će se kasnije programi slati na njega i zbog toga će oni biti prebačeni na mikrokontroler).




*Slika 2.1.8 Uspješno završena instalacija upravljačkog programa*

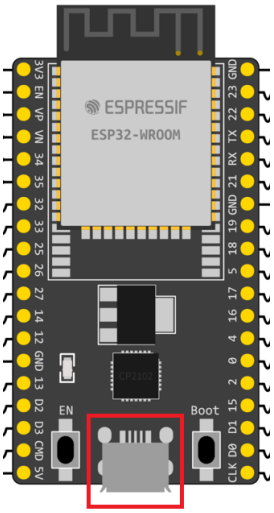
## 2.1.3. Pitanja i zadaci

1. Koji se kabel koristi za povezivanje mikrokontrolera ESP32 s razvojnim računalom?
    a. PLC
    b. VGA
    c. HMI
    d. USB
    e. RGB

2. USB kabel koristi se za povezivanje mikrokontrolera ESP32 na napajanje.
    a. Točno
    b. Netočno

3. Razvojno računalo šalje program na mikrokontroler ESP32 preko USB kabela.
    a. Točno
    b. Netočno

4. Na sljedećoj slici zaokružen je priključak napajanja mikrokontrolera ESP32.

5. 

    a. Točno
    b. Netočno

6. Kako se zove program na razvojnom računalu preko kojega možemo vidjeti port na kojem je priključen mikrokontroler ESP32?
    a. Desktop Manager
    b. Folder Manager
    c. File Manager
    d. Device Manager
    e. Network Manager

7. Dio mikrokontrolera je procesor.
    a. Točno
    b. Netočno

8. Mikrokontroler posjeduje dvije vrste memorije.
    a. Točno
    b. Netočno

9. Koje vrste memorije posjeduje mikrokontoler?
    a. Trajnu
    b. Vanjsku
    c. Privremenu
    d. Masivnu
    e. U oblaku

10. Mikrokontroler ESP32 posjeduje mali broj ulazno/izlaznih sučelja.
    a. Točno
    b. Netočno

11. Izvor napajanja treba omogućiti veliku snagu jer mikrokontroler ESP32 ima visoku potrošnju električne energije.
    a. Točno
    b. Netočno

12. Mikrokontroler ESP32 može se povezati na bežičnu računalnu mrežu.

a. Točno
b. Netočno

13. Mikrokontroler ESP32 ne može se povezati Bluethoothom.
    a. Točno
    b. Netočno

14. Tko je u mikrokontroleru zadužen za izvođenje programskog koda?
    a. Memorija
    b. Ulazno sučelje
    c. Izlazno sučelje
    d. Procesor
    e. Komunikacijsko sučelje

15. Programski kôd u mikrokontroleru spremljen je u [procesoru, memoriji, ulazno/izlaznom sučelju, komunikacijskom sučelju].

16. Mikrokontroleri se često koriste u kućanskim aparatima.
    a. Točno
    b. Netočno

17. Mikrokontroleri se često koriste u automobilskim sustavima.
    a. Točno
    b. Netočno

18. Mikrokontroleri se često koriste u medicinskim uređajima.
    a. Točno
    b. Netočno

19. Mikrokontroleri se često koriste u sustavima za implementaciju usluga u oblaku.
    a. Točno
    b. Netočno

20. Mikrokontroleri se često koriste u sustavima za implementaciju računalne mreže.
    a. Točno
    b. Netočno

21. Ulazno/izlazna sučelja mikrokontrolera koriste se samo kako bi se on povezao na računalnu mrežu.
    a. Točno
    b. Netočno

### 2.1.4. Literatura i izvori


1. CP210x USB to UART Bridge VCP Drivers, https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers?tab=downloads
2. ESP32-DevKitC V4 Getting Started Guide, https://docs.espressif.com/projects/esp-idf/en/latest/esp32/hw-reference/esp32/get-started-devkitc.html#
3. Establish Serial Connection with ESP32, https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/establish-serial-connection.html

4. Getting Started with the ESP32 Development Board, https://randomnerdtutorials.com/getting-started-with-esp32/
5. Microcontroller and its Types, https://www.geeksforgeeks.org/microcontroller-and-its-types/

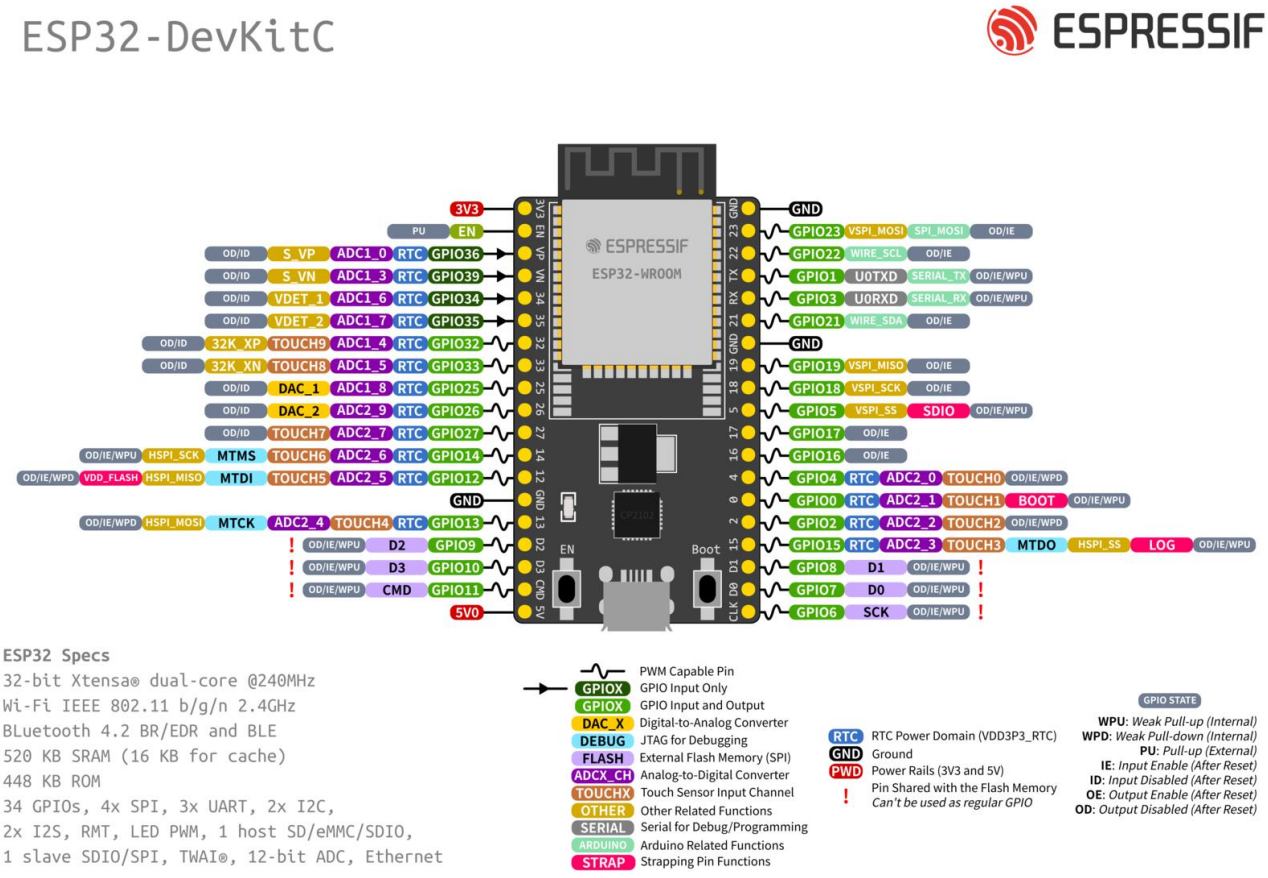
## 2.2. Izrada programa kojim se periodički aktivira digitalni izlaz mikrokontrolera ESP32

Potrebno je izraditi program kojim se periodički (svake sekunde) aktivira digitalni izlaz GPIO0 mikrokontrolera ESP32. Koristeći LED provjeriti je li digitalni izlaz aktiviran.

**Nakon usvajanja ove nastavne teme čitatelj će moći:**
1. Programirati mikroupravljač koristeći odabranu programsku potporu
2. Povezati aktuator sa sklopovskom potporom
3. Programirati sklopovsku opremu za rad s priključenim aktuatorom
4. Izraditi programsku potporu za vlastiti sustav interneta stvari

### 2.2.1. Osnovi koncepti

Mikrokontroler ESP32 posjeduje bogata ulazno/izlazna sučelja. Naredna slika prikazuje sve priključke mikrokontrolera ESP32.




*Slika 2.2.1 Priključci mikrokontrolera ESP32*

Jedna vrsta izlaznog priključka jest digitalni izlaz (npr. GPIO0) koji može poslati dvije vrste signala – 0 (0V) i 1 (3.3V). Da bi se kontrolirala njegova aktivacija, potrebno je izraditi program. Za izradu programa koristi se razvoji okvir ESP-IDF pa je potrebno pratiti njegovu proceduru razvoja koja se sastoji od sljedećih koraka:

1. izraditi mapu u kojoj će se nalaziti projekt
2. preko softvera Visual Studio Code i naredbe ESP-IDF: Show Examples Projects dobiti popis primjera projekata
3. iz popisa primjera projekata odabrati onaj koji naviše odgovara ili odabrati sample_project za prazan projekt
4. stvoriti izabrani projekt unutar izrađene mape projekta (1. korak)

Jednom kada je program izrađen, procedura njegova pokretanja na mikorkontroleru jest sljedeća:

1. kompajlirati program (naredba ESP-IDF: Build your project)
2. program poslati na mikrokontroler (naredba ESP-IDF: Flash your project)
3. opcionalno pratiti rad programa preko konzole (naredba: Monitor your device).

Postoji naredba koja odradi sva tri koraka: ESP-IDF: Build, Flash and start monitor on your device.

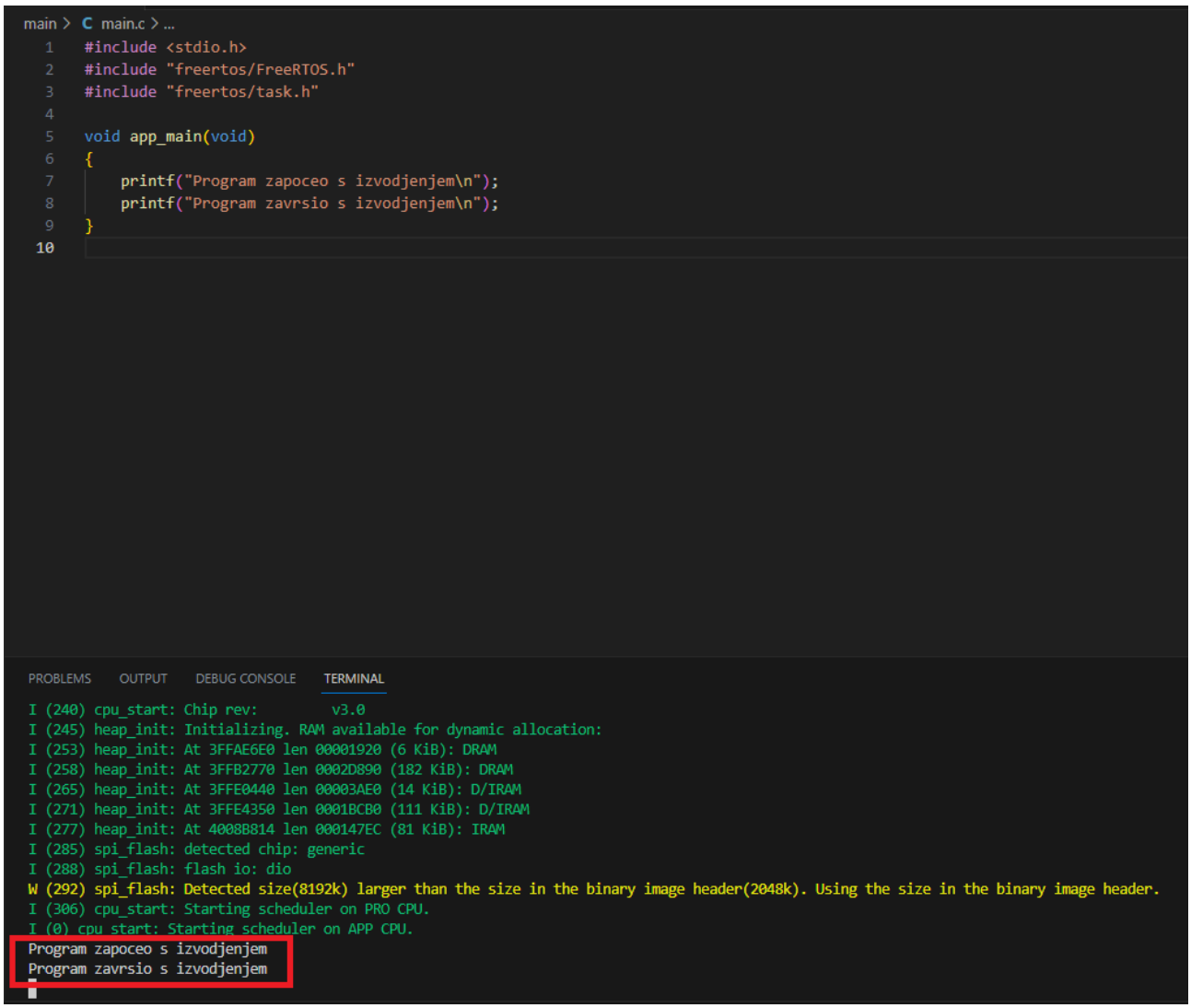
Ako se odabere prazan projekt (sample project), tada se u mapi projekta izradi (pored ostaloga) i datoteka main.c koja ima sljedeći sadržaj:

```
#include <stdio.h>

void app_main(void)
{

}
```

Može se primijetiti naziv glavne funkcije app_main koja se prva pokreće kada mikrokontroler pokrene program.
Budući da se digitalni izlaz može implementirati samo kroz višenamjenske priključke mikrokontrolera (npr. GPIO0), prvo je potrebno specificirati koju će namjenu imati priključak. Naredba kojom se to radi jest gpio_set_direction. Kroz prvi argument dostavlja joj se identifikator priključka (npr. GPIO_NUM_0), a kroz drugi identifikator načina rada priključka. Na primjer, naredba gpio_set_direction(GPIO_NUM_0, GPIO_MODE_OUTPUT) znači da se priključak GPIO0 stavlja u način rada izlaz.
Nakon definiranja načina rada priključka njegovo stanje mijenja se preko naredbe gpio_set_level. Kroz prvi argument dostavlja se identifikator priključka, a kroz drugi vrijednost koja se želi poslati na izlaz (0 ili 1). Na primjer, gpio_set_level(GPIO_NUM_0, 1) na priključak GPIO_NUM_0 šalje vrijednost 1.
Kada bi se program mikrokontrolera sastojao samo od niza naredbi za aktiviranje digitalnih izlaza unutar funkcije app_main, dolaskom do zadnje naredbe mikrokontroler bi završio s izvođenjem programa (ne bi se vratio na ponovno izvođenje programa).
Slijedi primjer jednog programa čije izvođenje završava nakon zadnje naredbe.

```
#include <stdio.h>

void app_main(void)
{
    printf("Program zapoceo s izvodjenjem\n");
    printf("Program zavrsio s izvodjenjem\n");
}
```

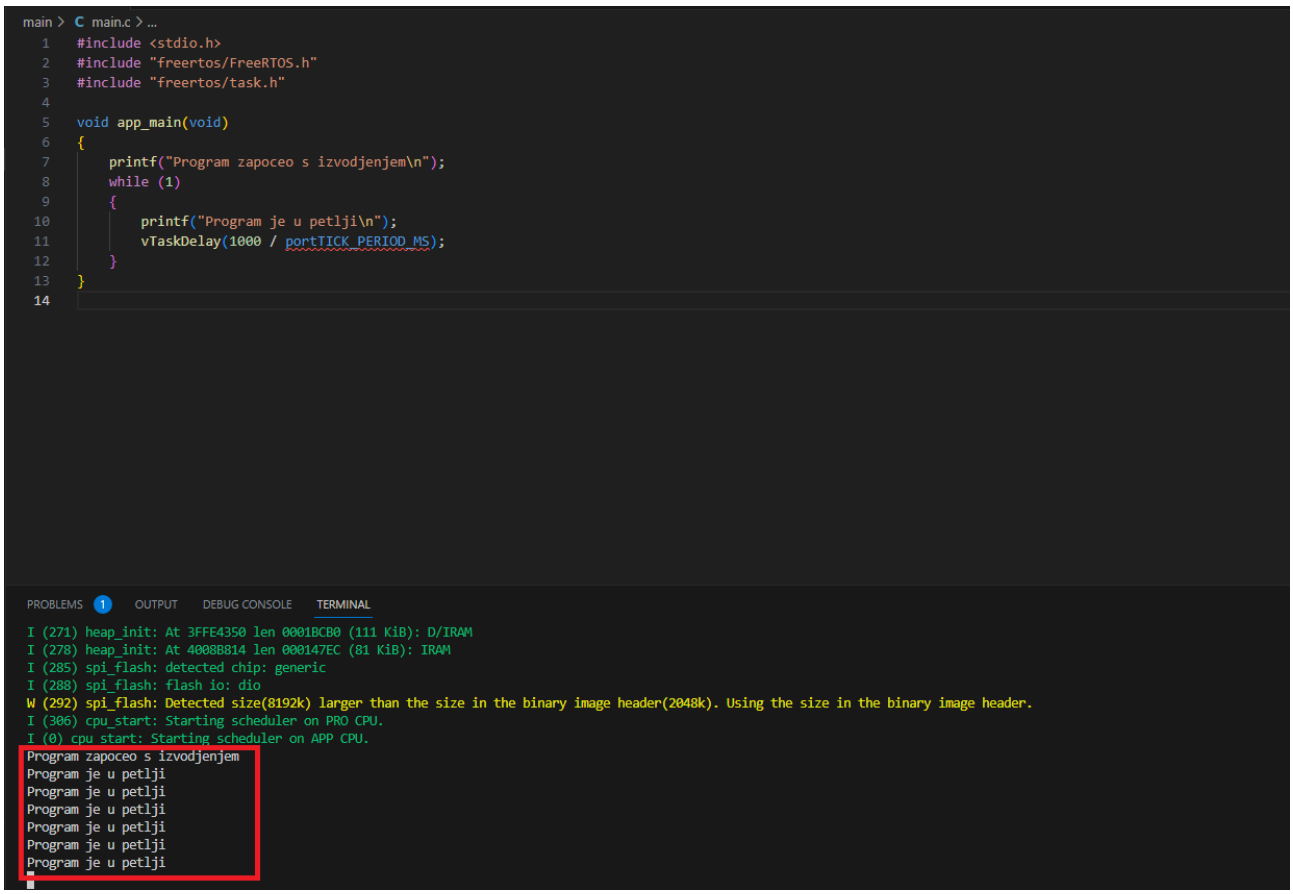
Naredna slika prikazuje rezultat izvođenja ovoga programa.




*Slika 2.2.2 Primjer programa koji završava svoje izvođenje*

Zbog toga se niz naredbi koje mikrokontroler treba stalno izvoditi mora staviti u beskonačnu petlju – npr. naredbom while(1). Brzina ponavljanja naredbi u tijelu petlje može se ograničiti naredbom vTaskDelay koja kao argument prima milisekunde. Kako bi se ta naredba koristila, potrebno je u program uključiti dvije zaglavne FreeRTOS.h i task.h.
Naredni program svake sekunde ispisuje u konzoli poruku „Program je u petlji”. Ovaj program ne završava.

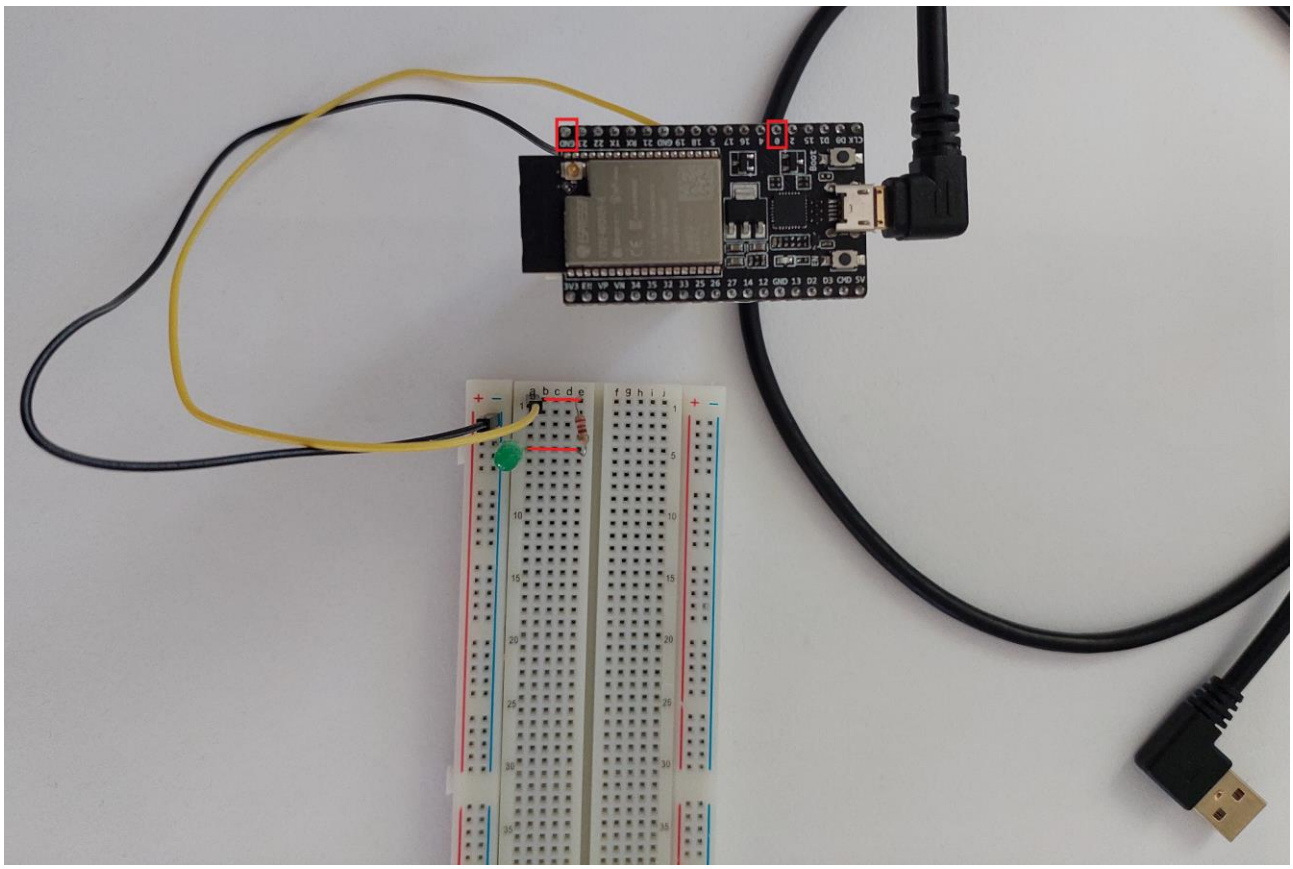
```
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"

void app_main(void)
{
    printf("Program zapoceo s izvodjenjem\n");
    while (1)
    {
        printf("Program je u petlji\n");
        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
}
```




*Slika 2.2.3 Primjer programa koji ne završava sa svojim izvođenjem*

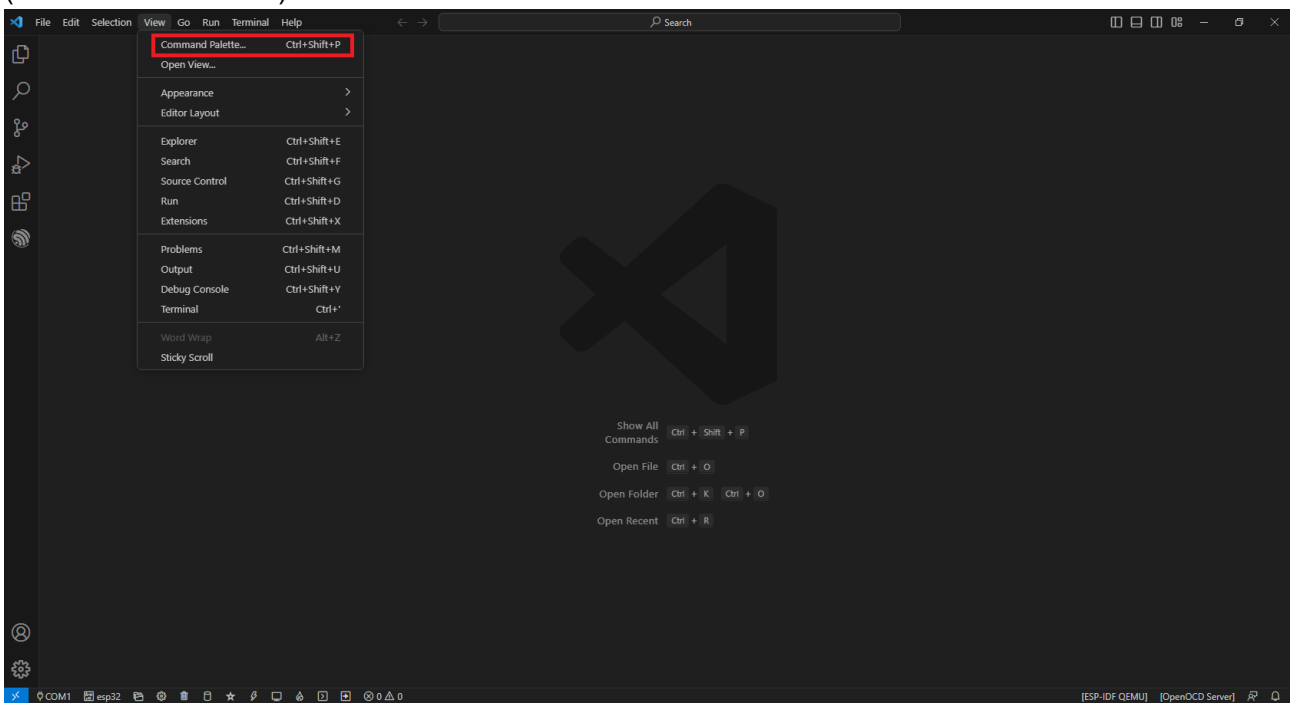
## 2.2.2. Rješenje radnog zadatka

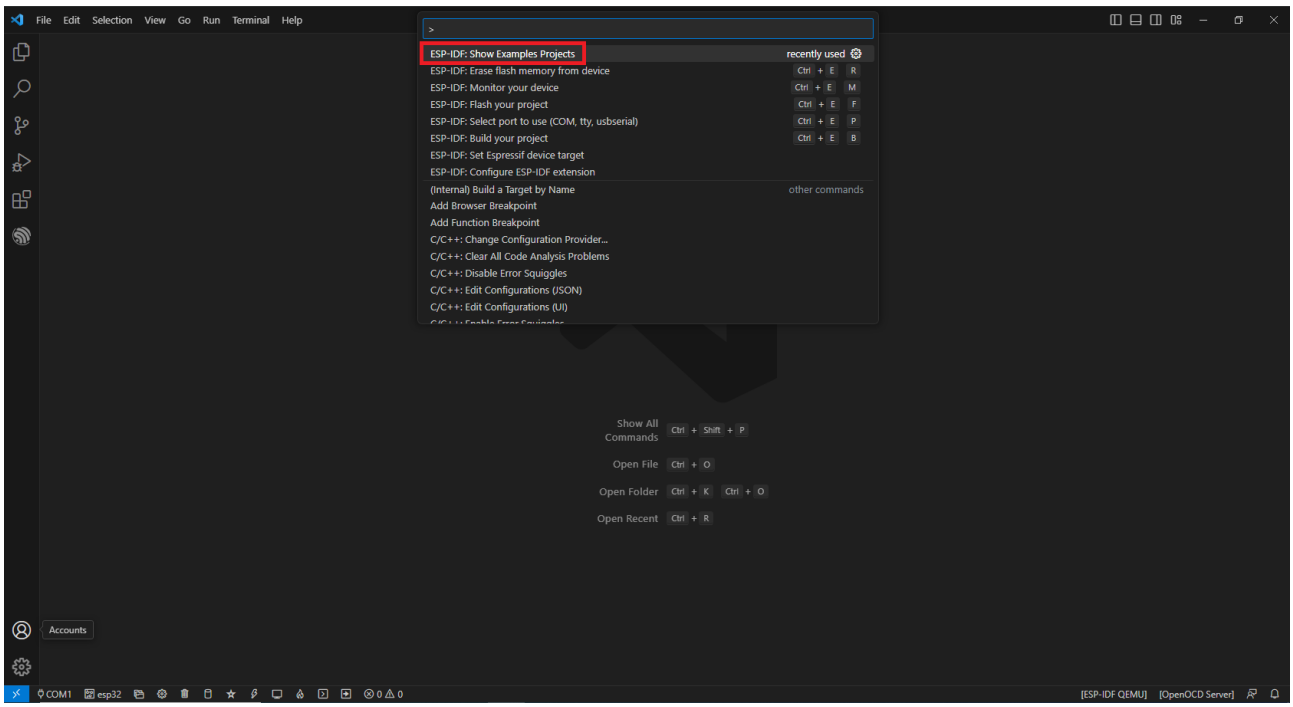
Mikrokontroler je potrebno povezati s razvojnim računalom. Na njegov priključak GPIO0 potrebno je spojiti otpornik kako bi se ograničila struja koja prolazi kroz LED (pogodna vrijednost otpornika jest od 200 do 400 ohma) – u zadatku je korišten otpornik od 220 ohma. Anodu LED (+) potrebno je spojiti na otpornik, a katodu LED (-) na priključak mikrokontrolera GND.



*Slika 2.2.4 Povezivanje mikrokontrolera s razvojnim računalom i povezivanje LED na priključak GPIO1 preko otpornika i GND*

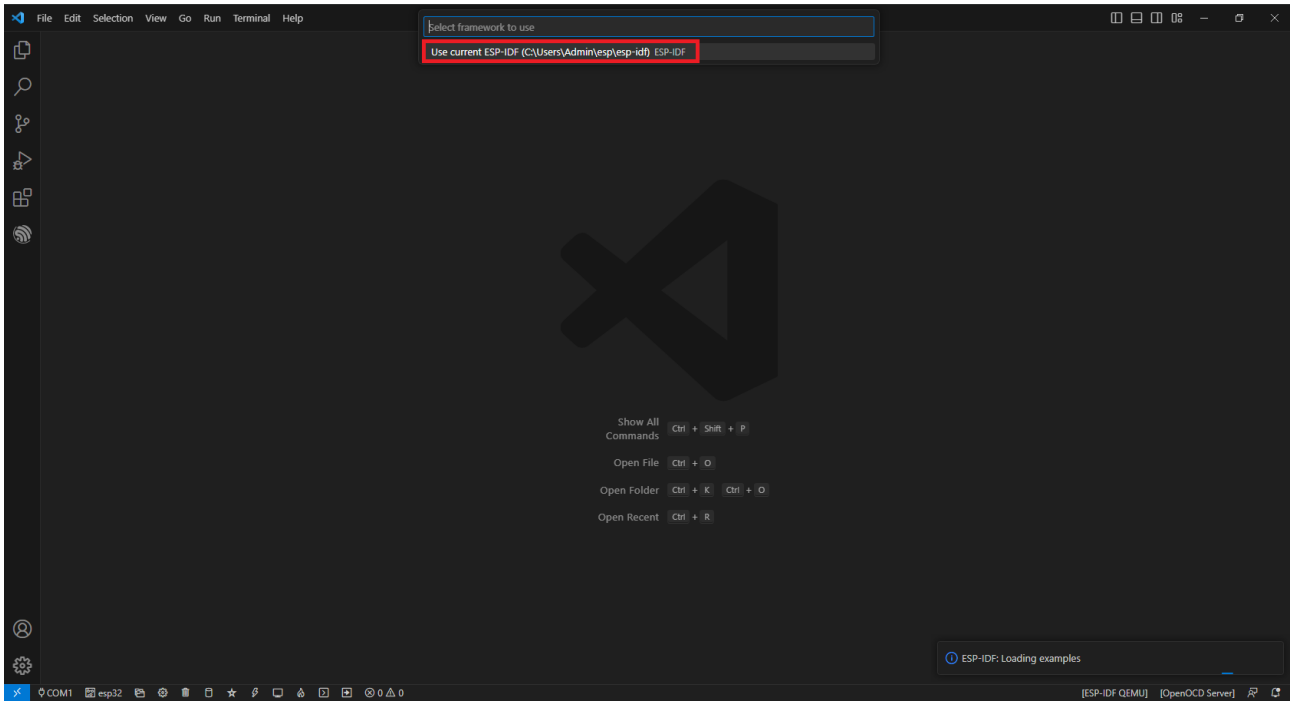
Kroz Visual Studio Code potrebno je izraditi prazni projekt koristeći naredbu ESP-IDF: Show Examples Projects. Tu naredbu (kao i sve druge naredbe) potrebno je upisati u naredbeni redak (Command Palette…) softvera Visual Studio Code.

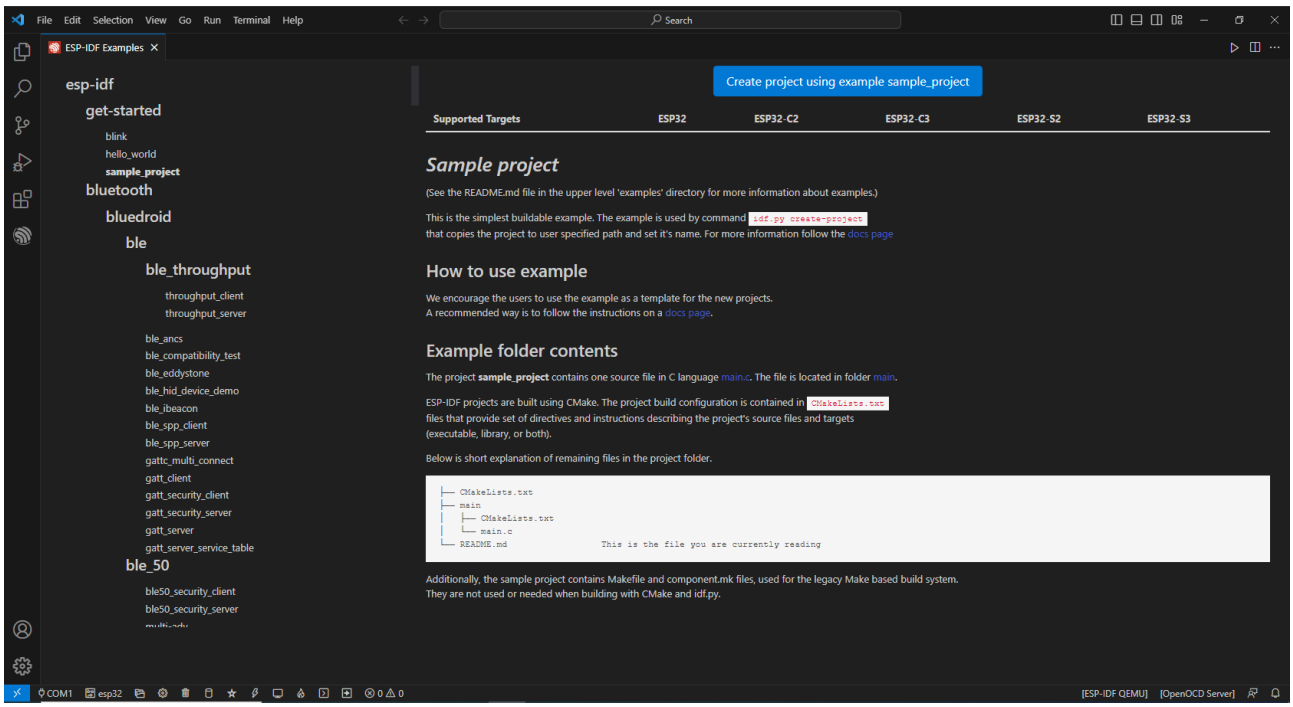





*Slika 2.2.5 Naredbeni redak Visual Studio Code-a i naredba ESP-IDF: Show Examples Projects*

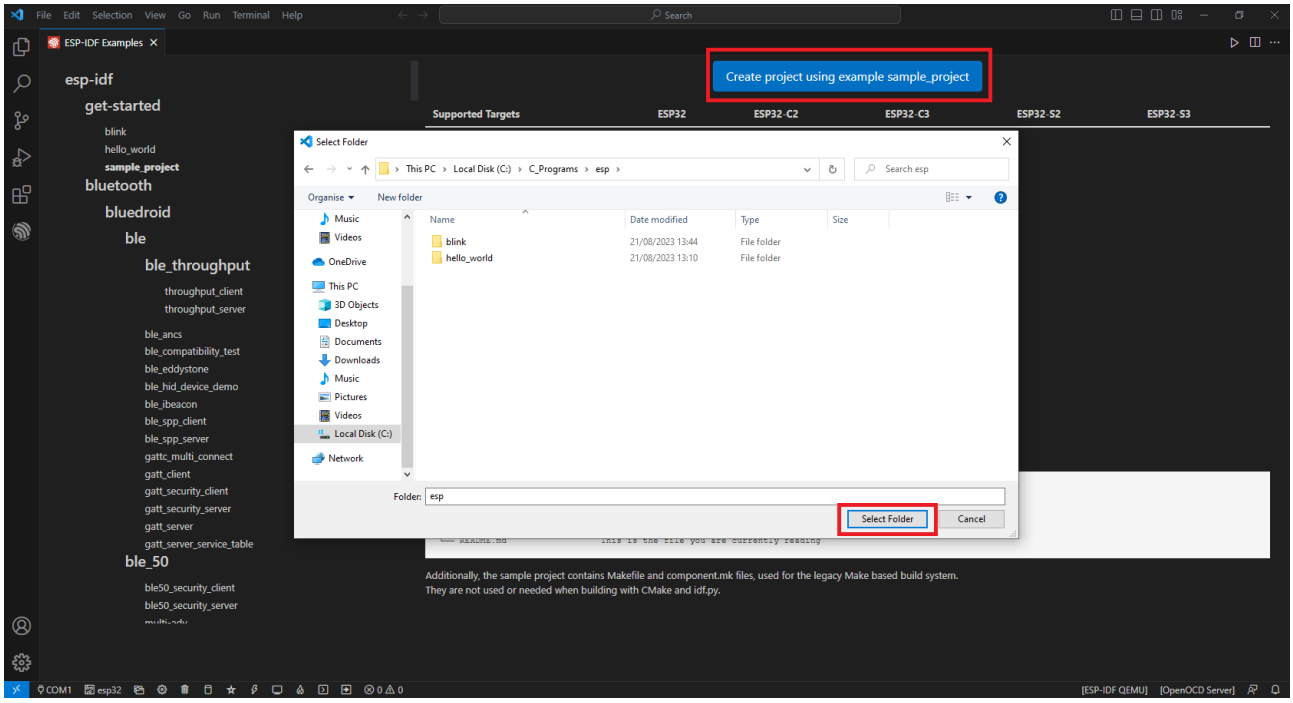
Nakon pokretanja naredbe ESP-IDF: Show Examples Projects i izbora razvojnog okruženja dobiva se popis dostupnih primjera projekata.





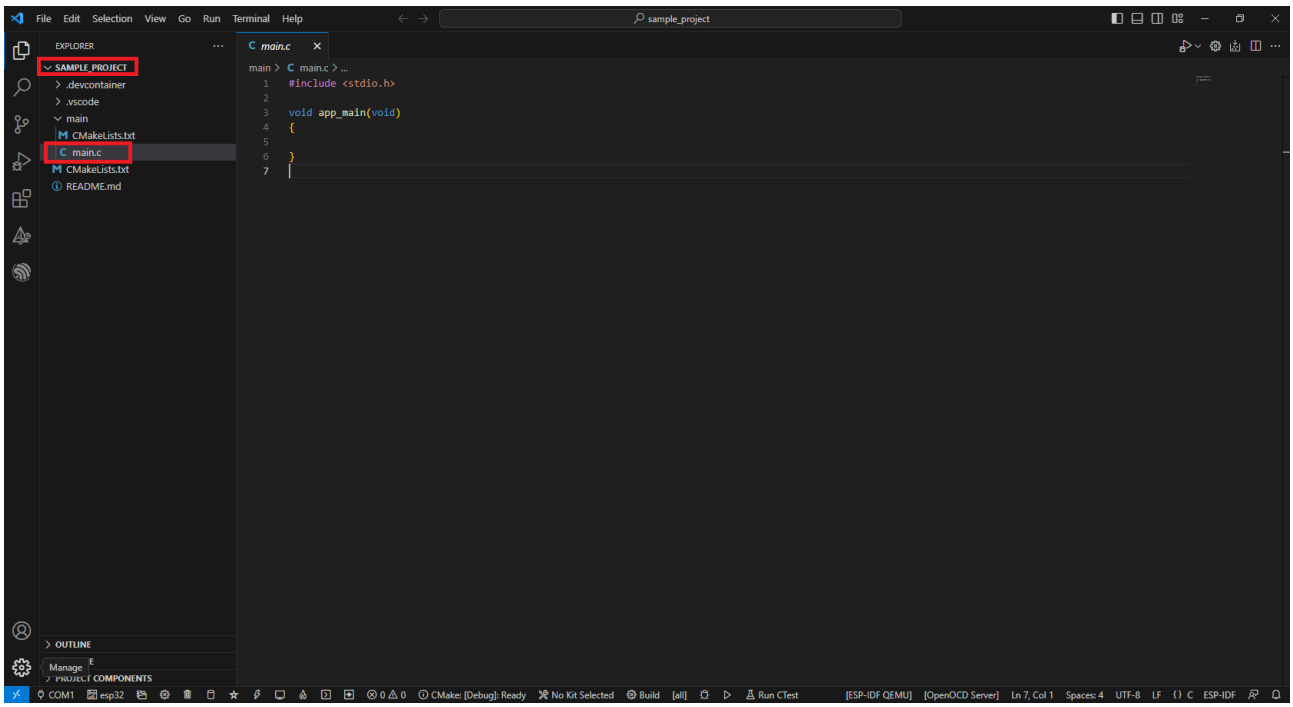

*Slika 2.2.6 Popis dostupnih projekata*

Projekt koji će se izabrati je sample_project. Kako bi se on kreirao potrebno je kliknuti na gumb “Create project using example sample_project”. Potrebno je izabrati mapu u kojoj će se projekt kreirati.



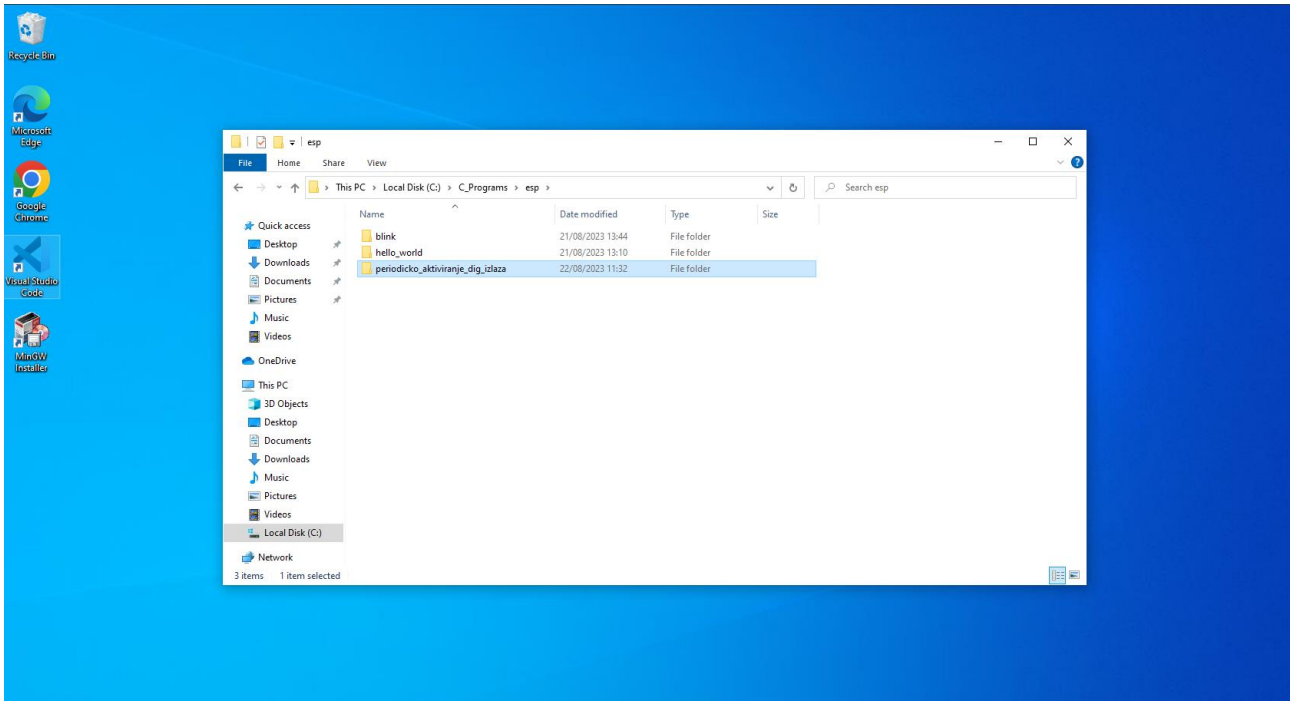

*Slika 2.2.7 Pokretanje procesa izrade projekta prema predlošku*

Nakon završetka procesa izrade stvoren je novi projekt sa svim potrebnim mapama i datotekama. Program će se pisati u datoteci main.c



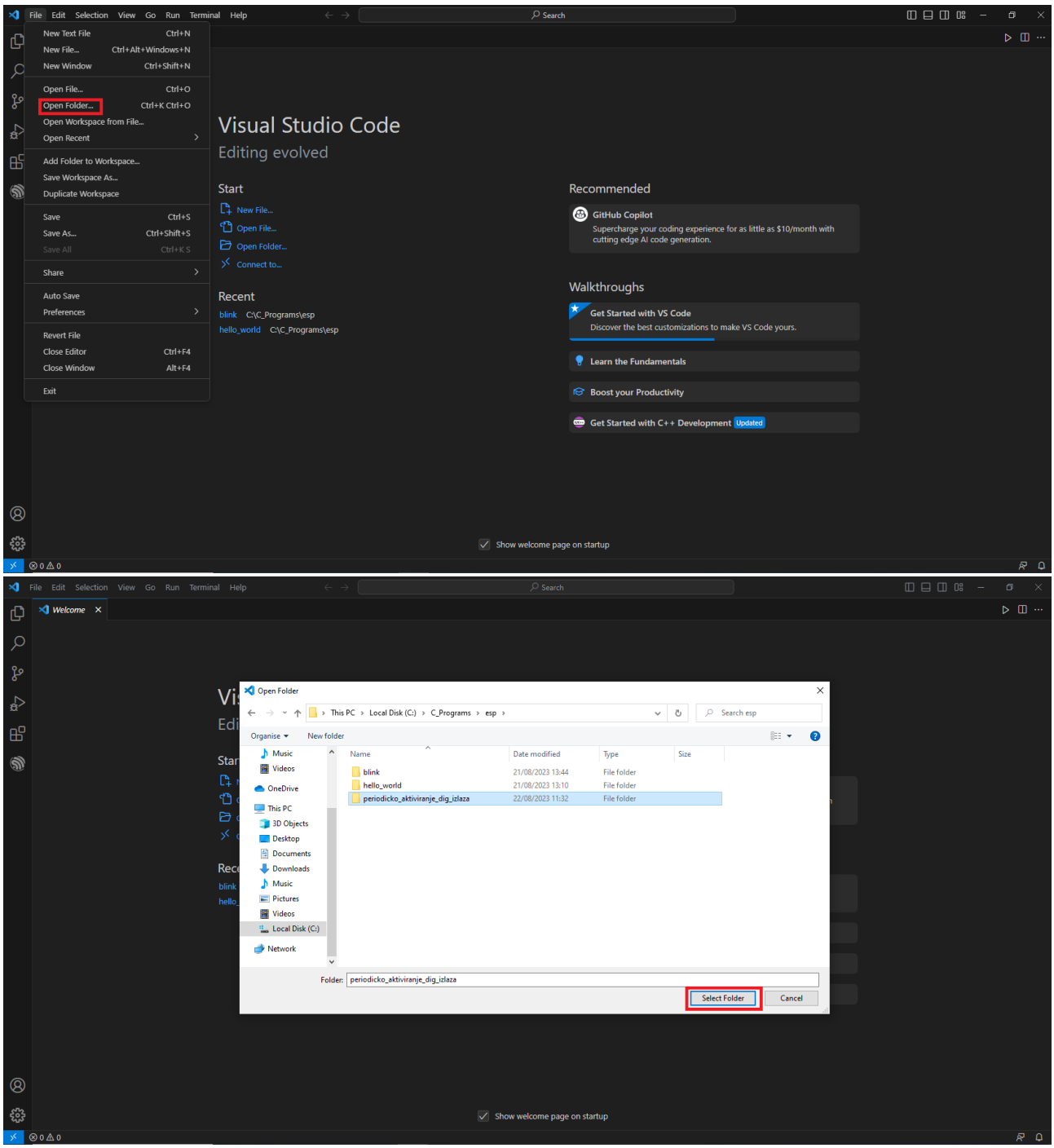

*Slika 2.2.8 Izrada projekta*

Preko programa File Explorer može se izmijeniti naziv mape projekta u periodicko_aktiviranje_dig_izlaza (prije toga potrebno je zatvoriti Visual Studio Code).

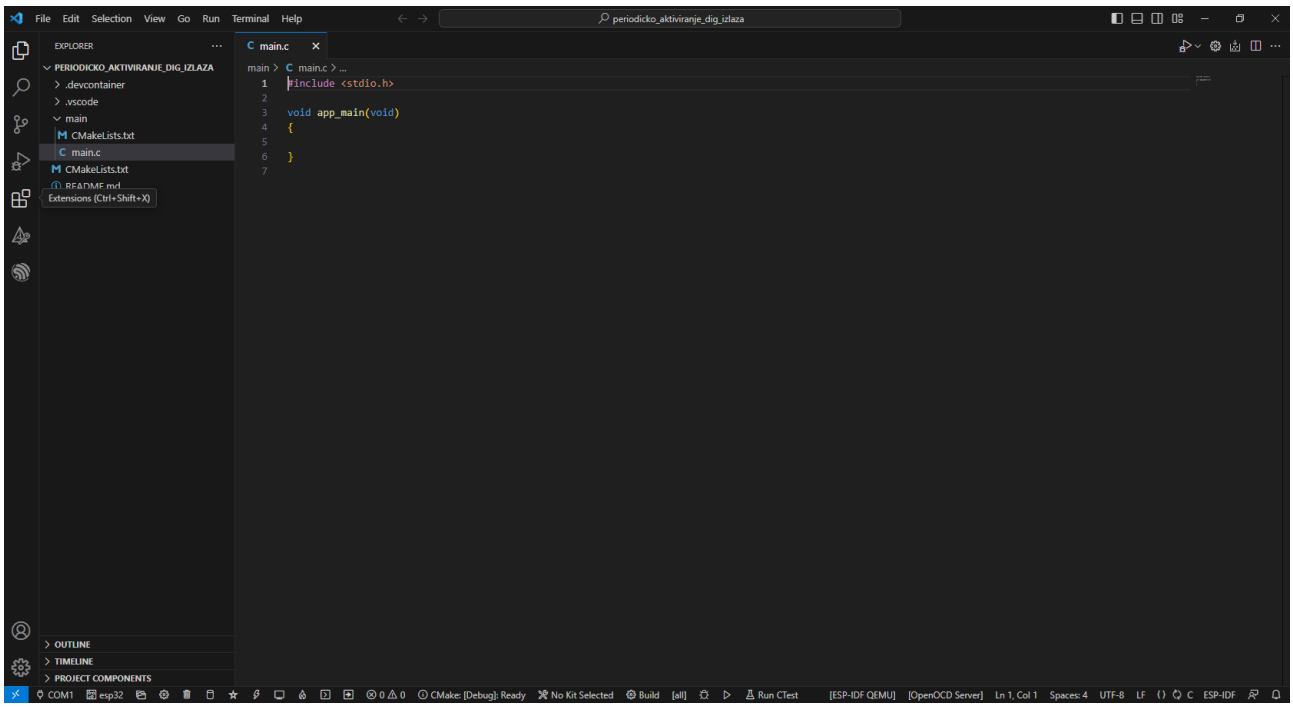



*Slika 2.2.9 Izmjena naziva mape projekta*

Ponovnim otvaranjem softvera Visual Studio Code i otvaranjem mape projekta ponovno se prikazuje cjelokupna struktura projekta.


File Edit Selection View Go Run Terminal Help
Search
New Text File Ctrl+N
New File... Ctrl+Alt+Windows+N
New Window Ctrl+Shift+N
Open File... Ctrl+O
Open Folder... Ctrl+K Ctrl+O
Open Workspace from File...
Open Recent
Add Folder to Workspace...
Save Workspace As...
Duplicate Workspace
Save Ctrl+S
Save As... Ctrl+Shift+S
Save All Ctrl+K S
Share
Auto Save
Preferences
Revert File
Close Editor Ctrl+F4
Close Window Alt+F4
Exit
Visual Studio Code
Editing evolved
Start
New File...
Open File...
Open Folder...
Connect to...
Recent
blink C:\C_Programs\esp
hello_world C:\C_Programs\esp
Recommended
GitHub Copilot
Supercharge your coding experience for as little as $10/month with cutting edge AI code generation.
Walkthroughs
Get Started with VS Code
Discover the best customizations to make VS Code yours.
Learn the Fundamentals
Boost your Productivity
Get Started with C++ Development
Updated
Show welcome page on startup


Welcome
Open Folder
This PC › Local Disk (C:) › C_Programs › esp ›
Search esp
Organise
New folder
Music
Videos
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
Network
Name
Date modified
Type
Size
blink 21/08/2023 13:44 File folder
hello_world 21/08/2023 13:10 File folder
periodicko_aktiviranje_dig_izlaza 22/08/2023 11:32 File folder
Folder: periodicko_aktiviranje_dig_izlaza
Select Folder
Cancel
Show welcome page on startup



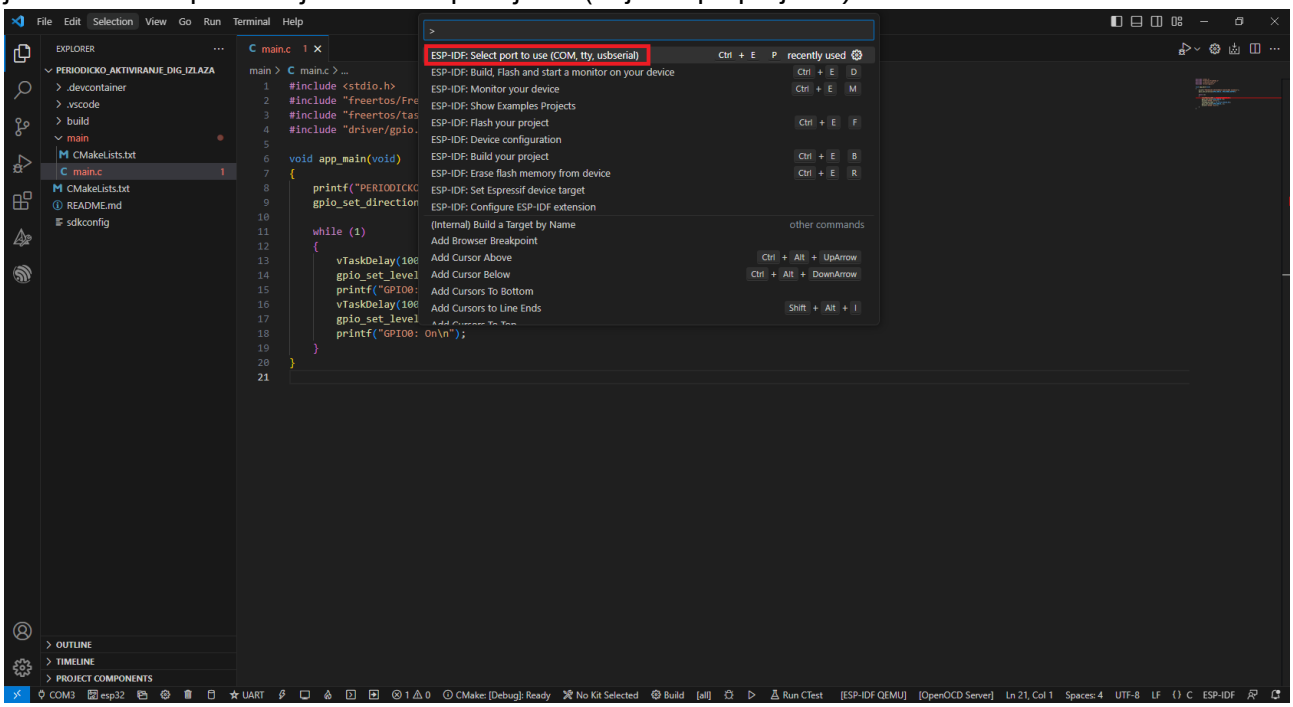

*Slika 2.2.10 Otvaranje mape s projektom u Visual Studio Code*

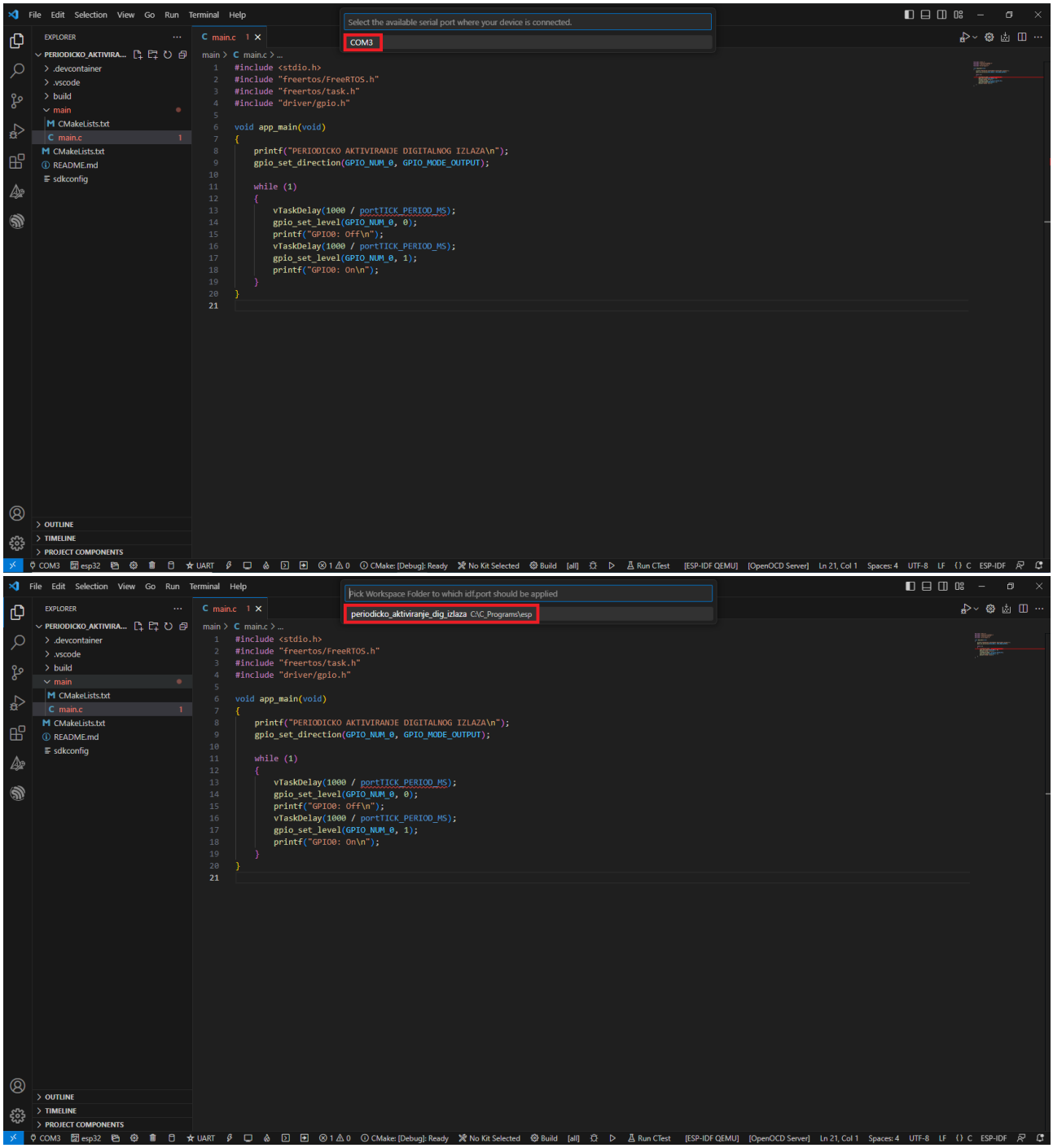
Sada se u datoteci main.c može napisati program koji će periodički paliti i gasiti digitalni izlaz GPIO0 mikrokontrolera.

```
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/gpio.h"

void app_main(void)
{
    printf("PERIODICKO AKTIVIRANJE DIGITALNOG IZLAZA\n");
    gpio_set_direction(GPIO_NUM_0, GPIO_MODE_OUTPUT);

    while (1)
    {
        vTaskDelay(1000 / portTICK_PERIOD_MS);
        gpio_set_level(GPIO_NUM_0, 0);
        printf("GPIO0: Off\n");
        vTaskDelay(1000 / portTICK_PERIOD_MS);
        gpio_set_level(GPIO_NUM_0, 1);
        printf("GPIO0: On\n");
    }
}
```

Na početku programa naredbom #include specificiraju se vanjske datoteke koje su potrebne kako bi se programski kôd mogao izvesti. Te datoteke u sebi sadrže definiciju funkcija koje se koriste u programu (npr. definicija funkcija gpio_set_direction i gpio_set_level nalazi se unutar datoteke gpio.h).
Kako bi se upravljalo nekim od izlaza (ili ulaza) mikrokontrolera, potrebno ih je prvo konfigurirati jer velika većina priključaka mikrokontrolera može obavljati više različitih funkcija. Zbog toga glavni dio

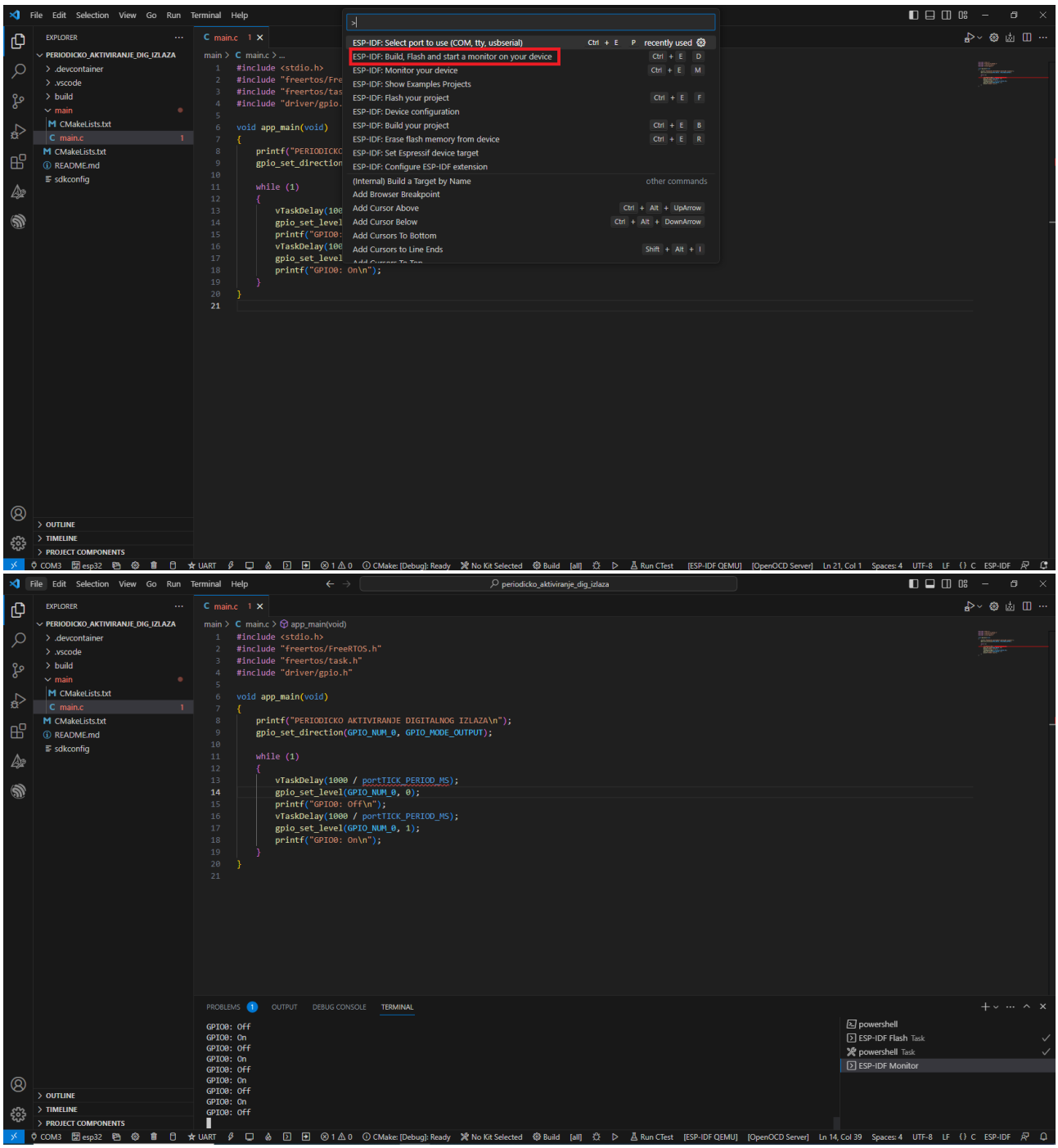
programa (funkcija app_main) započinje s konfiguracijom priključka GPIO0 kao izlaznog priključka (gpio_set_direction funkcija).
Nakon završetka konfiguracije priključka slijedi beskonačna petlja (while(1)) u kojoj se naizmjenično izlaz GPIO0 postavlja na vrijednost, odnosno 0 (funkcija gpio_set_level). Postavljena vrijednost zadržava se jednu sekundu (vTaskDelay(1000 / portTICK_PERIOD_MS) – 1000 ms) prije nego što se promijeni u drugu vrijednost (1). Također se u konzoli ispisuje poruka “GPIO0: Off” odnosno “GPIO0: On”.
Kada se izlaz GPI0 postavi na vrijednost 1, on dobiva naponsku razinu od 3,3 V zbog čega se pali LED. Kada se izlaz GPIO0 postavi na vrijednost 0, naponska razina pada na 0V i LED se ugasi.
Program se sada može kompajlirati i poslati na mikrokontroler. Mikrokontroler treba biti povezan s razvojnim računalom. U naredbenom retku softvera Visual Studio Code preko naredbe ESP-IDF:Select port to use (COM, tty, usbserial) potrebno je odabrati priključak preko kojega je mikrokontroler spojen na razvojno računalo (na slici je to COM3). Nakon izbora priključka potrebno je odabrati mapu za koju će se on primijeniti (to je mapa projekta).





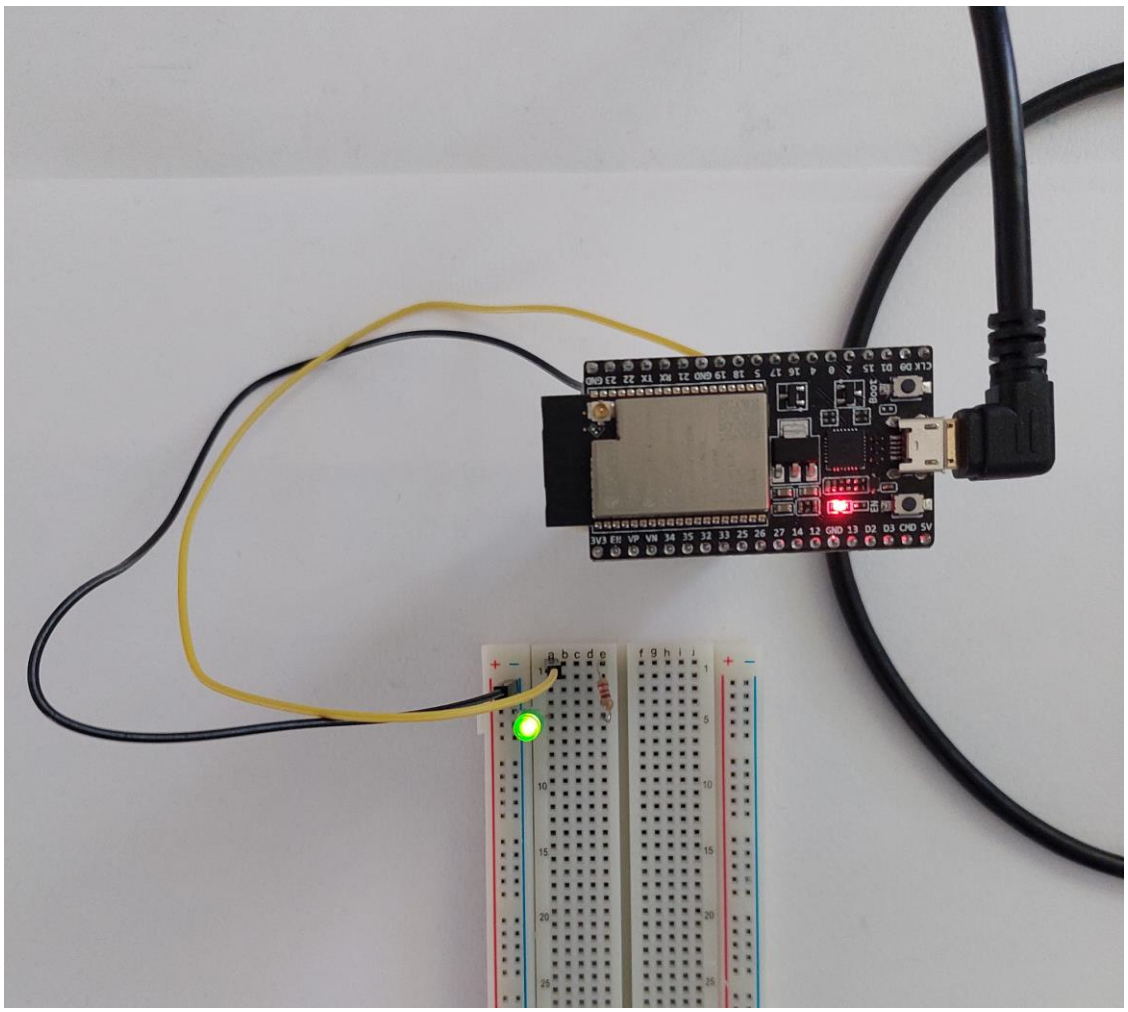

*Slika 2.2.11 Odabir priključka preko kojega je mikrokontroler povezan s razvojnim računalom*

Nakon specifikacije priključka slijedi kompajliranje i prebacivanje programa na mikrokontroler. U naredbenom retku softvera Visual Studio Code može se upisati naredba ESP-IDF: Build, Flash and start monitor on your device koja će osim kompajliranja i slanja programa na mikrokontroler pokrenuti i praćenje rada mikrokontrolera. Rad se prati preko terminala.




*Slika 2.2.12 Pokretanje naredbe za kompajliranje, slanje programa na mikrokontroler i praćenje njegova rada*

Nakon pokretanja programa LED se naizmjenično pali i gasi – trajanje periode jest 1000 ms, odnosno 1 s.



*Slika 2.2.13 Gotov sustav periodičkog aktiviranja digitalnog izlaza GPIO0*

### 2.2.3. Pitanja i zadaci

1. Izraditi program koji će periodički aktivirati druge digitalne izlaze mikrokontrolera te kojim će se promijeniti period aktiviranja LED.

2. Ispravno poredajte korake u razvoju programa primjenom razvojnog okvira ESP-IDF:
    a. Odabir primjera projekta
    b. Prikaz popisa primjera projekata
    c. Izrada mape projekta
    d. Izrada projekta iz primjera

3. Ispravno poredajte korake koji se trebaju provesti prilikom slanja programa na mikrokontroler:
    a. Praćenje rada programa preko konzole
    b. Slanje programa na mikorkontroler
    c. Kompajliranje programa

4. Naredba ESP-IDF: Show Examples Projects koristi se kako bi se:
    a. Izradila mapa projekta
    b. Prikazao popis primjera projekata
    c. Pratio rad mikrokontrolera
    d. Poslao program na mikrokontroler
    e. Kompajlirao program

5. Naredba ESP-IDF: Build your project koristi se kako bi se:
    a. Izradila mapa projekta
    b. Prikazao popisa primjera projekata
    c. Prati rad mikrokontrolera
    d. Poslao program na mikrokontroler
    e. Kompajlirao program

6. Naredba ESP-IDF: Flash your project koristi se kako bi se:
    a. Izradila mapa projekta
    b. Prikazao popis primjera projekata
    c. Pratio rad mikrokontrolera
    d. Poslao program na mikrokontroler
    e. Kompajlirao program

7. Naredba ESP-IDF: Monitor your device koristi se kako bi se:
    a. Izradila mapa projekta
    b. Prikazao popis primjera projekata
    c. Pratio rad mikrokontrolera
    d. Poslao program na mikrokontroler
    e. Kompajlirao program

8. Naziv glavne funkcije programa za mikrokontroler jest:
    a. main_app
    b. app_main
    c. main-app
    d. app-main
    e. main

9. Naredba za definiranje načina primjene višenamjenskih priključaka mikrokontrolera jest:
    a. gpio_get_direction
    b. gpio_set_direction
    c. gpio_set_level
    d. gpio_get_level
    e. gpio_direction_set

10. Naredba za definiranje stanja izlaznog priključka mikrokontrolera jest:
    a. gpio_get_direction
    b. gpio_set_direction
    c. gpio_set_level
    d. gpio_get_level
    e. gpio_direction_set

11. Naredba vTaskDelay koristi se kako bi se kontrolirala brzina ponavljanja naredbi u tijelu petlje.
    a. Točno
    b. Netočno

12. Argument funkcije vTaskDelay predstavlja:

a. Mikrosekunde
b. Pikosekunde
c. Nanosekunde
d. Milisekunde
e. Sekunde

13. Prvi argument funkcije gpio_set_direction predstavlja:
    a. Stanje
    b. Način uporabe
    c. Priključak
    d. Brzinu
    e. Vrijeme

14. Drugi argument funkcije gpio_set_direction predstavlja:
    a. Stanje
    b. Način uporabe
    c. Priključak
    d. Brzinu
    e. Vrijeme

15. Prvi argument funkcije gpio_set_level predstavlja:
    a. Stanje
    b. Način uporabe
    c. Priključak
    d. Brzinu
    e. Vrijeme

16. Drugi argument funkcije gpio_set_level predstavlja:
    a. Stanje
    b. Način uporabe
    c. Priključak
    d. Brzinu
    e. Vrijeme

### 2.2.4. Literatura i izvori


1. Blink Example, https://github.com/espressif/esp-idf/tree/32472536715d674c160a2565795895e0273f8cde/examples/get-started/blink
2. Example: GPIO, https://github.com/espressif/esp-idf/tree/32472536715d674c160a2565795895e0273f8cde/examples/peripherals/gpio/generic_gpio
3. GPIO & RTC GPIO, https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/gpio.html
4. Hello World Example, https://github.com/espressif/esp-idf/tree/32472536715d674c160a2565795895e0273f8cde/examples/get-started/hello_world

## 2.3. Izrada programa kojim se preko tipkala postavlja perioda aktiviranja digitalnog izlaza mikrokontrolera ESP32

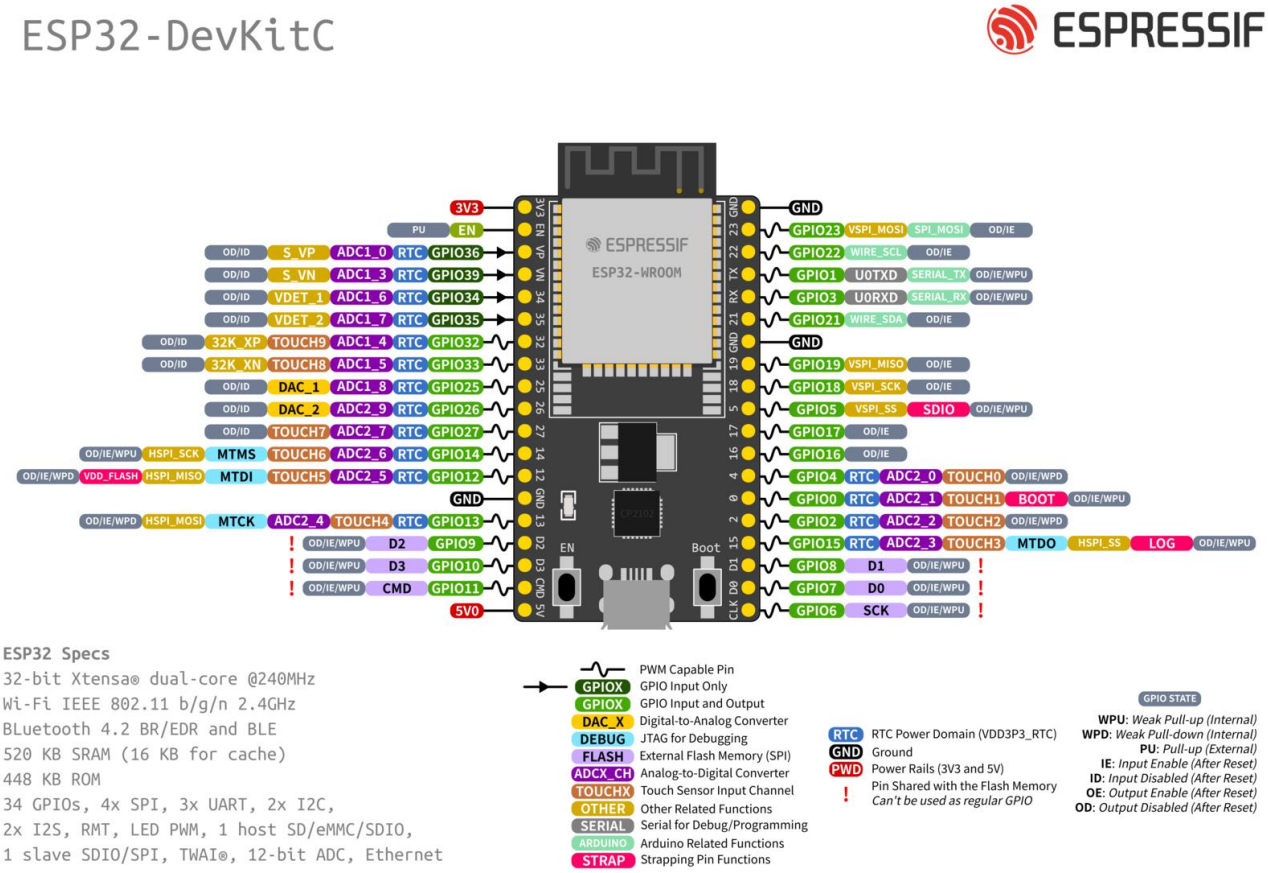
Preko tipkala postavlja se jedan od tri moguće periode aktiviranja digitalnog izlaza mikrokontrolera (1s, 2s i 3s) kao ulaz koristiti priključak GPIO32. Izlaz koji se aktivira jest GPIO2. Preko LED vidjeti aktiviranje izlaza GPIO2.

**Nakon usvajanja ove nastavne teme čitatelj će moći:**

1. Programirati mikroupravljač koristeći odabranu programsku potporu
2. Povezati senzor sa sklopovskom potporom
3. Programirati sklopovsku opremu za rad s priključenim senzorom
4. Povezati aktuator sa sklopovskom potporom
5. Programirati sklopovsku opremu za rad s priključenim aktuatorom
6. Izraditi programsku potporu za vlastiti sustav interneta stvari

### 2.3.1. Osnovni koncepti

Mikrokontroler ESP32 posjeduje i digitalne ulaze koji se mogu implementirati ili preko priključaka koji omogućuju samo digitalni ulaz (npr. GPIO34) ili preko priključaka koji omogućuju digitalni ulaz/izlaz (npr. GPIO0).




*Slika 2.3.1 Priključci mikrokontrolera ESP32*

Procedura upravljanja digitalnim ulazima slična je kao i s digitalnim izlazima. To znači da se kod ulazno/izlaznih priključaka prvo treba definirati namjena – naredba gpio_set_direction. Kod samo ulaznih priključaka taj korak nije potrebno raditi.

Naredba kojom se čita digitalni ulaz jest gpio_get_level koja kao argument treba dobiti identifikator ulaza, a kao rezultat vraća vrijednost 0 ili 1 (vrijednost koja se očitala na ulazu).

Mikrokontroler u ovom zadatku treba neovisno upravljati dvama zadacima:

1. uključivanje i isključivanje digitalnog izlaza (vidljivo preko LED svjetla)
2. očitavanje tipkala

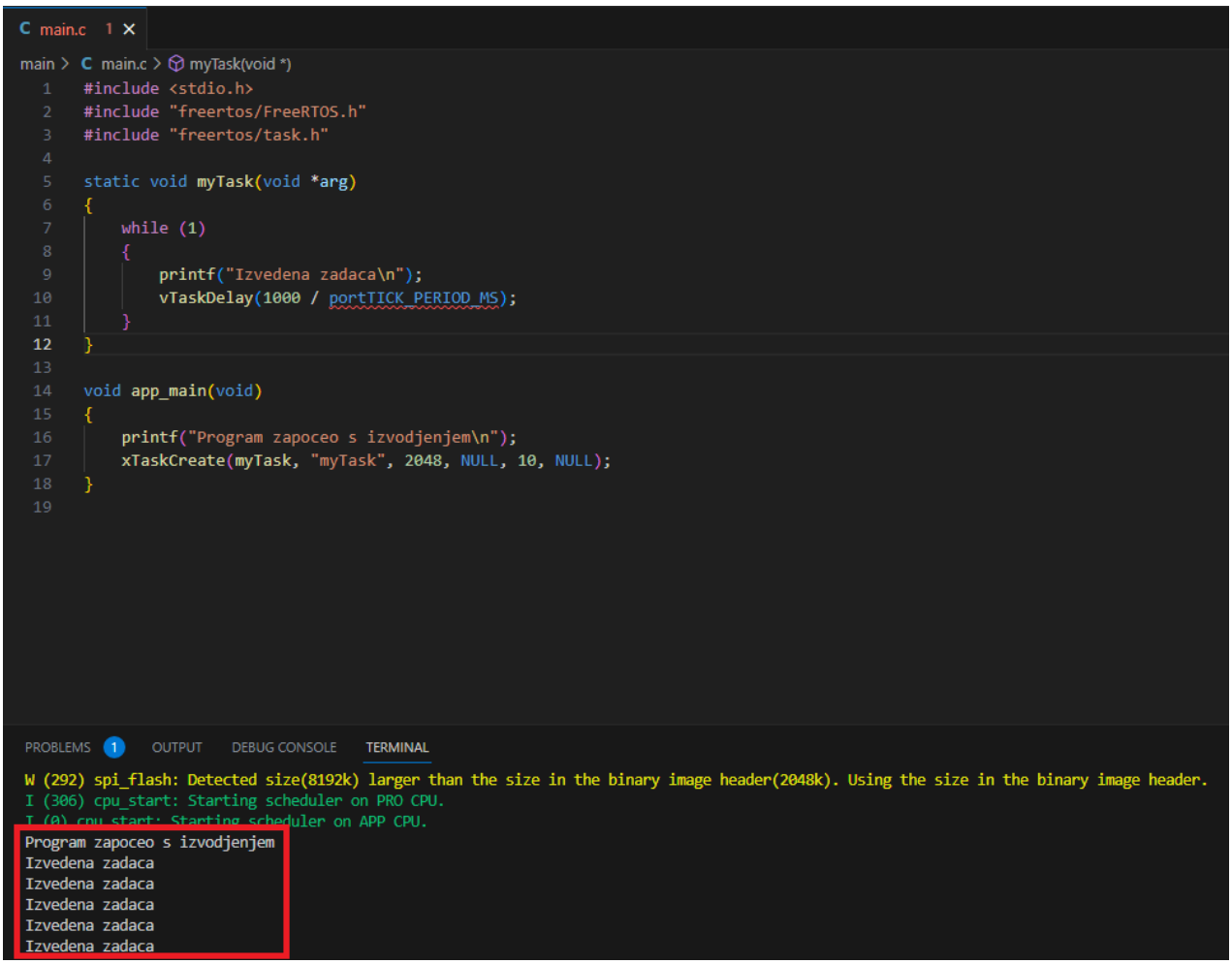
Iako je praćenje ta dva zadatka neovisno, oni ipak trebaju utjecati jedan na drugoga – tipkalo na uključivanje i isključivanje digitalnog izlaza.

Neovisno izvođenje zadataka (odnosno niza naredbi programskog koda) provodi se tako da se koristeći naredbu xTaskCreate izradi nova zadaća (novi *task*) te se specificira koja se to funkcija izvodi u novoj zadaći. Mikrokontroler će svoju „pažnju" naizmjenično davati zadaćama, odnosno izvodit će naredbe koje se nalaze u funkcijama koje se trebaju izvesti. Te funkcije ne smiju završiti, odnosno u njima se uvijek treba nalaziti beskonačna petlja. Ponovno se kontrola brzine ponavljanja naredbi u petlji radi koristeći naredbu vTaskDelay.

Sljedeći program svake sekunde ispisuje poruku „Izvedena zadaca".

```
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"

static void myTask(void *arg)
{
    while (1)
    {
        printf("Izvedena zadaca\n");
        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
}

void app_main(void)
{
    printf("Program zapoceo s izvodjenjem\n");
    xTaskCreate(myTask, "myTask", 2048, NULL, 10, NULL);
}
```

Naredna slika prikazuje rezultat izvođenja programa.



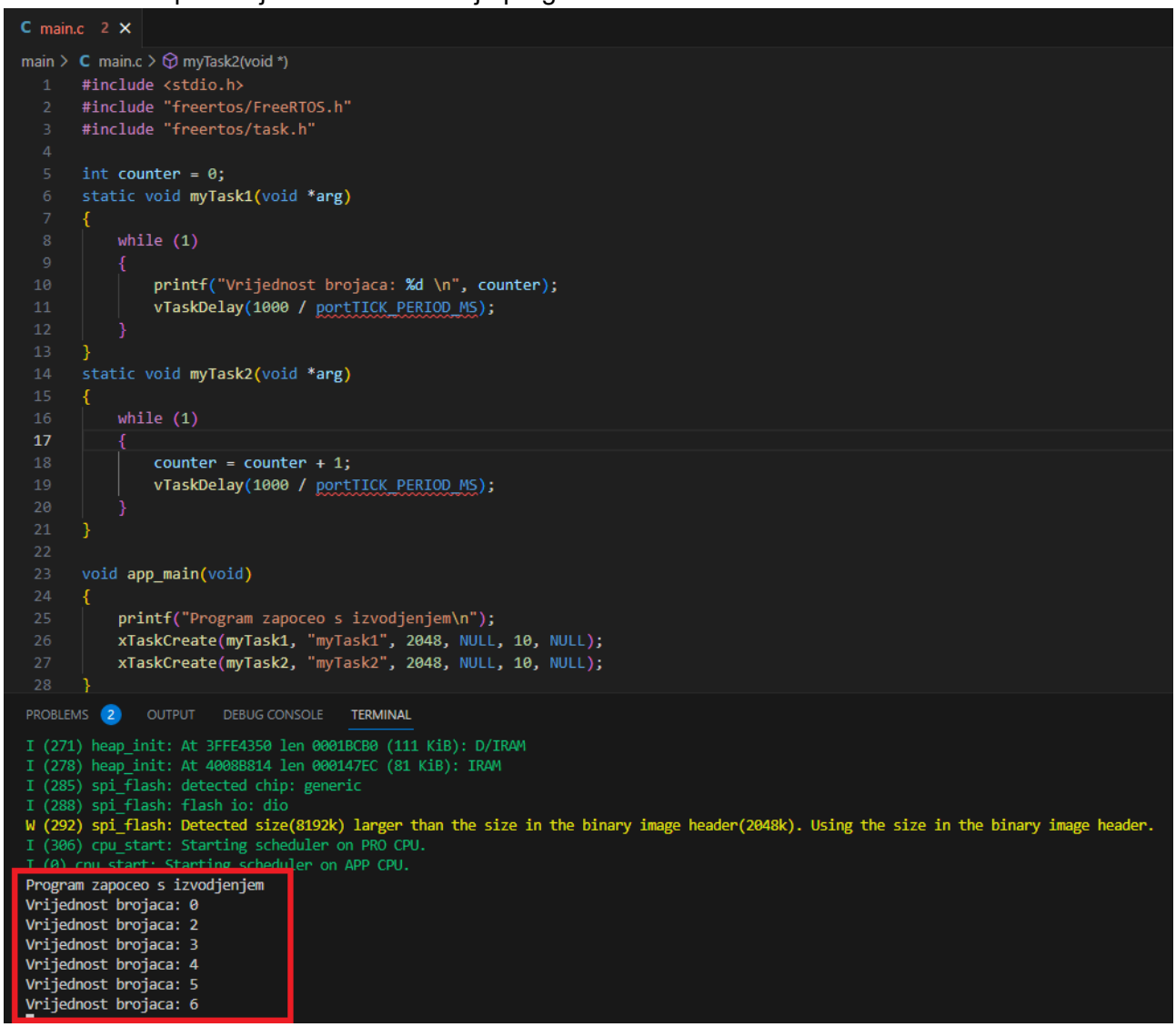

*Slika 2.3.2 Izvođenje funkcije u zadaći (task)*

Postoji više načina da različite zadaće međusobno dijele podatke. Jedan od načina jest deklariranje varijable koja je vidljiva u svim funkcijama – globalna varijabla.
Sljedeći program u jednoj zadaći mijenja vrijednost varijable svake sekunde, a u drugoj zadaći ispisuje vrijednost te varijable.

```
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"

int counter = 0;
static void myTask1(void *arg)
{
    while (1)
    {
        printf("Vrijednost brojaca: %d \n", counter);
        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
}
static void myTask2(void *arg)
{
    while (1)
    {
        counter = counter + 1;
        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
```

```
}

void app_main(void)
{
    printf("Program zapoceo s izvodjenjem\n");
    xTaskCreate(myTask1, "myTask1", 2048, NULL, 10, NULL);
    xTaskCreate(myTask2, "myTask2", 2048, NULL, 10, NULL);
}
```

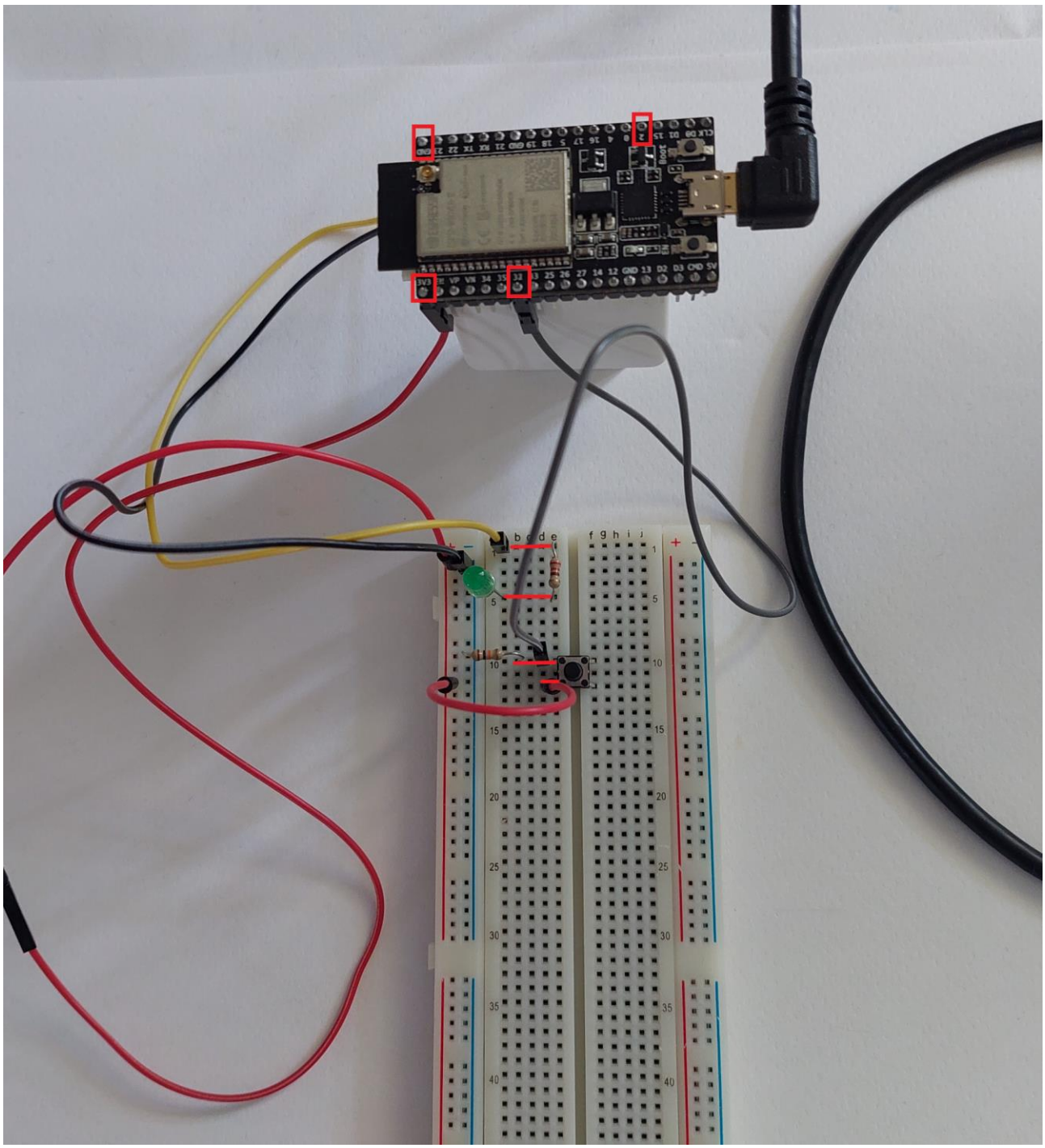
Naredna slika prikazuje rezultat izvođenja programa.




*Slika 2.3.3 Rezultat izvođenja programa s dvije zadaće koje razmjenjuju podatak*

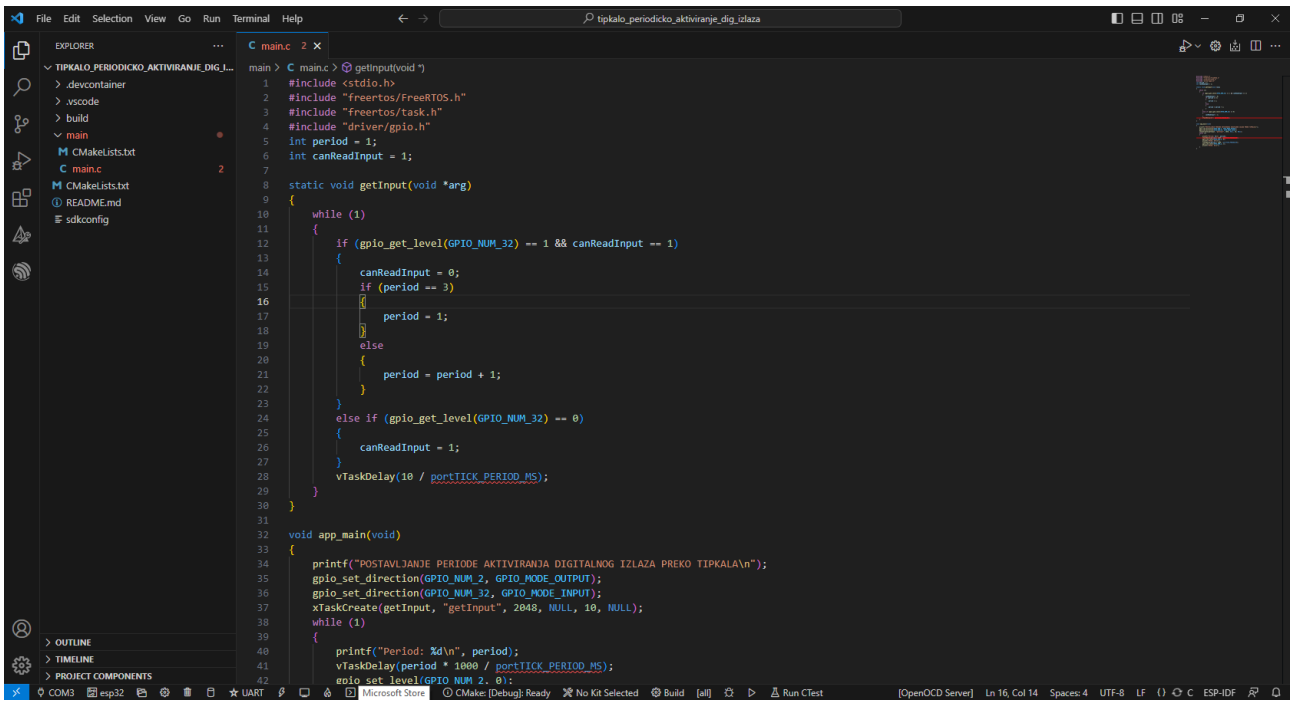
## 2.3.2. Rješenje radnog zadatka

Na mikrokontroler je potrebno povezati tipkalo na način da se preko njega dovodi napajanje (+ pol – 3V3 priključak mikrokontrolera) na digitalni ulaz GPIO32. Prema priključku GND stavlja se *pull-down* otpornik od 10 kOhm. Povezivanje LED indikatora s izlazom GPIO2 isto je kao i u prethodnom zadatku.



*Slika 2.3.4 Povezivanje tipkala i LED na priključke GPIO32 i GPIO2.*

U softveru Visual Studio Code potrebno je izraditi novi projekt iz primjera projekta sample_projects (naredba ESP-IDF: Show Examples Projects). Nakon što se projekt stvori i Visual Studio Code zatvori, potrebno mu je preko programa File Explorer promijeniti naziv (promijeniti naziv mape) u tipkalo_periodicko_aktiviranje_dig_izlaza. Potom je potrebno tu novu projektnu mapu otvoriti u softveru Visual Studio Code i u datoteci main.c napisati program.




*Slika 2.3.5 Izrađen novi projekt tipkalo_periodicko_aktiviranje_dig_izlaza*

Sljedeći program upravlja periodom aktiviranja digitalnog izlaza GPIO2 peko tipkala spojenog na digitalni ulaz GPIO32.

```
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/gpio.h"
unsigned char period = 1;
unsigned char canReadInput = 1;

static void getInput(void *arg)
{
    while (1)
    {
        if (gpio_get_level(GPIO_NUM_32) == 1 && canReadInput == 1)
        {
            canReadInput = 0;
            if (period == 3)
            {
                period = 1;
            }
            else
            {
                period = period + 1;
            }
        }
        else if (gpio_get_level(GPIO_NUM_32) == 0)
        {
            canReadInput = 1;
        }
        vTaskDelay(10 / portTICK_PERIOD_MS);
```

```
    }
}

void app_main(void)
{
    printf("POSTAVLJANJE PERIODE AKTIVIRANJA DIGITALNOG IZLAZA PREKO TIPKALA\n");
    gpio_set_direction(GPIO_NUM_2, GPIO_MODE_OUTPUT);
    gpio_set_direction(GPIO_NUM_32, GPIO_MODE_INPUT);
    xTaskCreate(getInput, "getInput", 2048, NULL, 10, NULL);
    while (1)
    {
        printf("Period: %d\n", period);
        vTaskDelay(period * 1000 / portTICK_PERIOD_MS);
        gpio_set_level(GPIO_NUM_2, 0);
        printf("GPIO2: Off\n");
        vTaskDelay(period * 1000 / portTICK_PERIOD_MS);
        gpio_set_level(GPIO_NUM_2, 1);
        printf("GPIO2: On\n");
    }
}
```

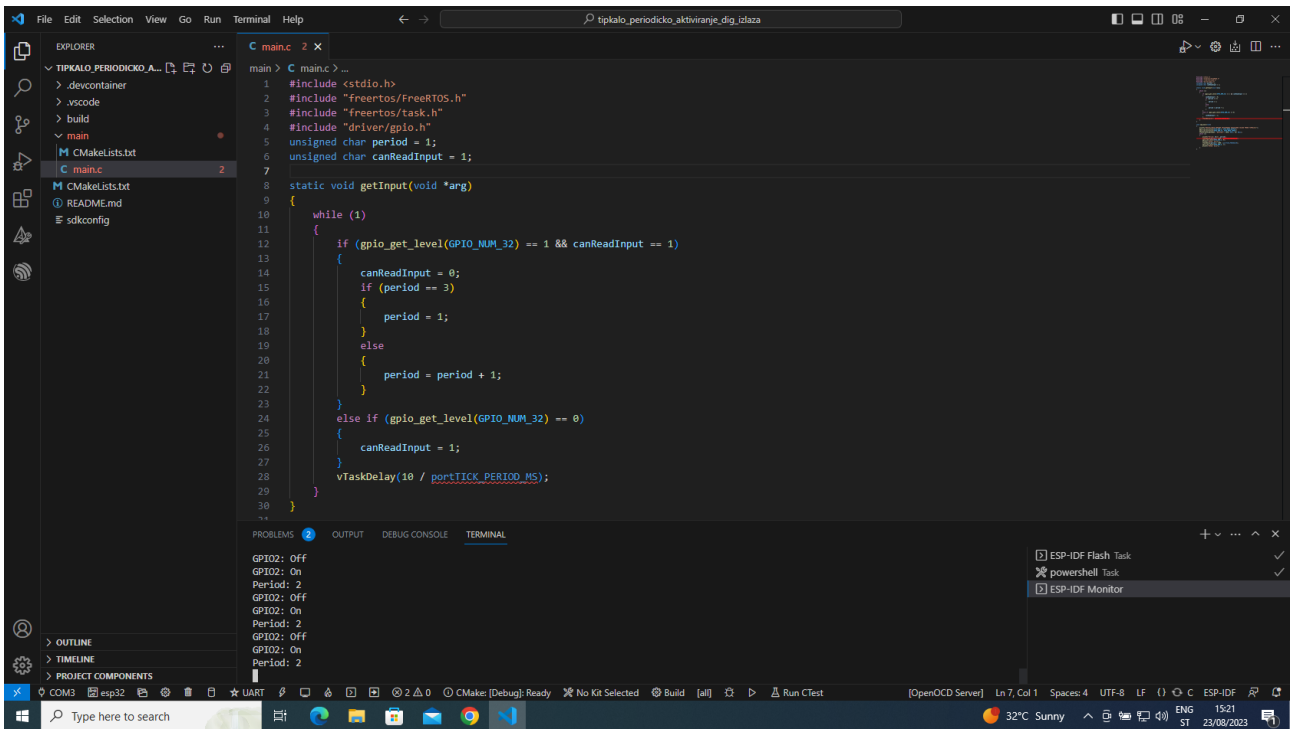
U programu se deklariraju dvije varijable tipa podatka unsigned char (period i canReadInput). Varijabla period može imati samo tri vrijednosti (1, 2 i 3), a koristi se za definiranje perioda aktivacije digitalnog izlaza (1 s, 2s i 3s). Varijabla canReadInput može imati samo dvije vrijednosti (0 i 1), a koristi se kako bi se osiguralo da pritisak i otpuštanje tipkala predstavljaju promjenu periode (a ne samo pritisak).

U glavnom dijelu programa prvo se konfiguriraju priključci tako da GPIO2 bude izlazni, a GPIO32 ulazni priključak. Nakon toga izradi se novi zadatak (*Task*) što znači da će periodički mikrokontroler pokretati funkciju koja je dodiljenjena novoj zadaći (funkcija je getInput). U toj funkciji promatra se stanje ulaza GPIO32 (preko funkcije gpio_get_level) i ako je to stanje 1 i omogućeno je čitanje (varijabla canReanInput), tada se varijabla period povećava za 1, odnosno ako ima vrijednost 3, tada se ona postavlja na vrijednost 1. Time se postiglo da aktiviranje tipkala (pali i gasi) povećava period s 1 na 2 i potom na 3. Aktiviranje tipkala nakon 3 vraća vrijednost na 1. Aplikativno to znači da se period aktiviranja izlaza mijenja na sljedeći način 1s > 2s > 3s > 1s > 2s …

Treba posebnu pažnju obratiti na varijablu canReadInput koja onemogućuje višestruko čitanje izlaza, odnosno situaciju da se tipkalo drži pritisnuto (ne i otpušteno) i da se cijelo to vrijeme mijenja varijabla period. Ono što je postignuto jest da tek kada se jednom pritisnuto tipkalo otpusti, tek tada dolazi do mogućnosti očitanja njegova ponovnog pritiska.

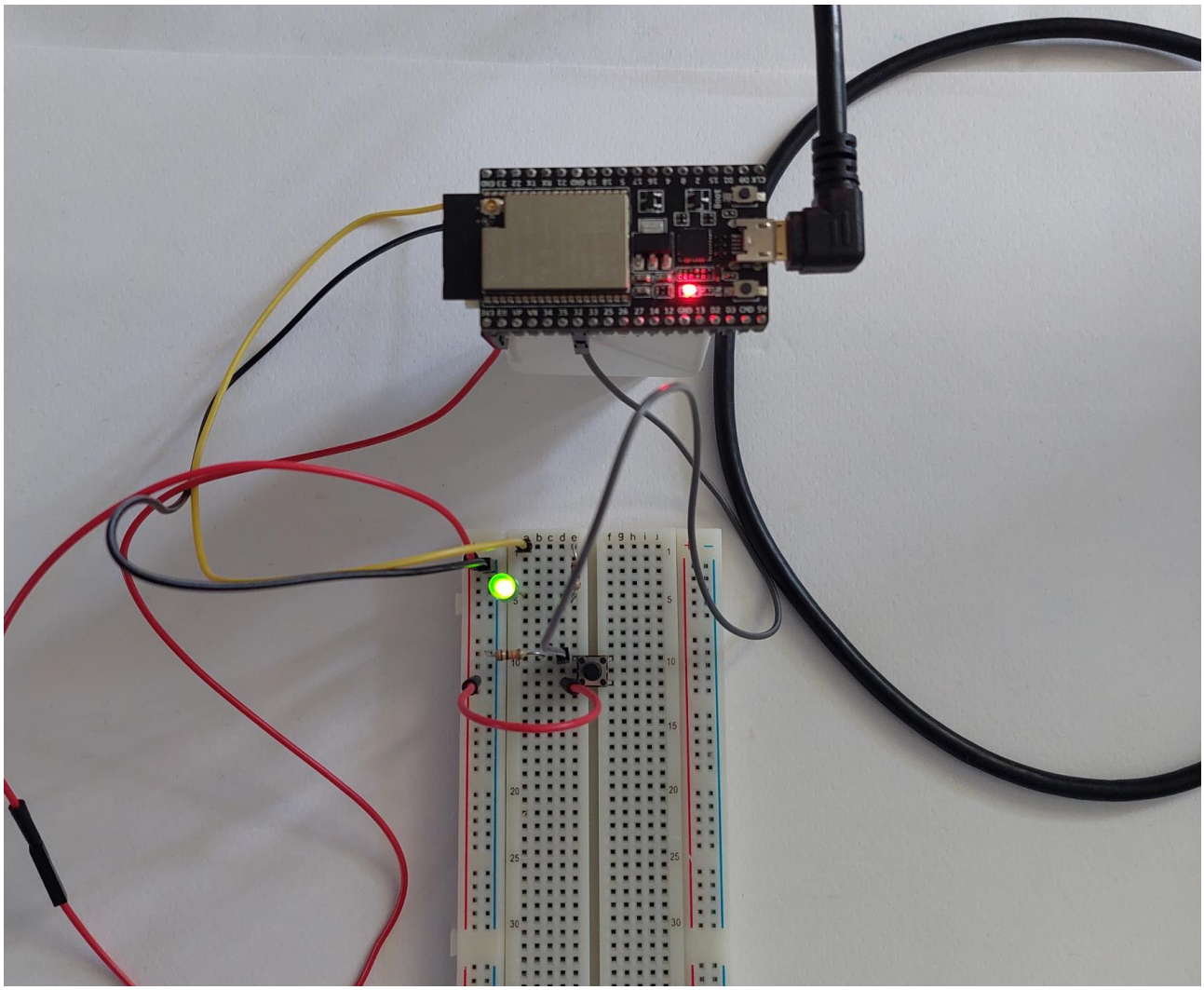
U glavnom dijelu programa (app_main) nalazi se programski kôd kojim se aktivira digitalni izlaz GPIO2 u određenim periodima (slično kao kod prethodnog radnog zadatka). Jedina razlika jest ta što se vrijeme kašnjenja izračunava kao period*1000 (što je vrijednost u ms) čime se dobiva period aktivacije od 1, 2 i 3 s.

Program se sada može kompajlirati, poslati na mikrokontroler i promatrati rezultat kroz konzolu.




*Slika 2.3.6 Program pokrenut na mikrokontroleru*

Gotov sustav prikazuje naredna slika.



*Slika 2.3.7 Sustav kontrole periode aktiviranja digitalnog izlaza preko tipkala*

### 2.3.3. Pitanja i zadaci

1. Izraditi isti program, ali uz korištenje drugih priključaka mikrokontrolera
2. Prikazani program izmijeniti tako da se omogući i gašenje digitalnog izlaza
3. Prikazani program izmijeniti tako da se omogući više perioda aktiviranja digitalnog izlaza
4. Mikrokontroler ESP32 posjeduje priključke koji mogu biti i ulazni i izlazni (ovisno o njihovoj konfiguraciji u programu).
    a. Točno
    b. Netočno

5. Mikrokontroler ESP32 posjeduje priključke koji mogu biti samo ulazni.
    a. Točno
    b. Netočno

6. Naredba za čitanje stanja ulaznog priključka mikrokontrolera jest:
    a. gpio_get_direction
    b. gpio_set_direction
    c. gpio_set_level
    d. gpio_get_level
    e. gpio_level_get

7. Argument funkcije gpio_get_level predstavlja:
    a. Stanje
    b. Način uporabe
    c. Priključak
    d. Brzina
    e. Vrijeme

8. Mikrokontroler ESP32 može neovisno upravljati više zadaća koje su predstavljene funkcijom.
    a. Točno
    b. Netočno

9. Naredba kojom se definira nova zadaća jest:
    a. vCreateDelay
    b. xCreateDelay
    c. xTaskCreate
    d. xCreateTask
    e. xCreate

10. Izrađene zadaće (*task*) u programu mogu međusobno dijeliti podatke.
    a. Točno
    b. Netočno

11. Kreirane zadaće (*task*) u programu mogu dijeliti podatke preko globalno deklariranih varijabli.
    a. Točno
    b. Netočno

### 2.3.4. Literatura i izvori


1. Controlling ESP32 GPIO with ESP-IDF, https://embeddedexplorer.com/esp32-gpio-tutorial/#:~:text=To%20configure%20a%20GPIO%20pin,pin%200%2C%20pass%20in%20GPIO_NUM_0%20
2. ESP32 Push Button with ESP-IDF (Digital Input), https://esp32tutorials.com/esp32-push-button-esp-idf-digital-input/#google_vignette
3. Example: GPIO, https://github.com/espressif/esp-idf/tree/32472536715d674c160a2565795895e0273f8cde/examples/peripherals/gpio/generic_gpio
4. FreeRTOS (ESP-IDF), https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/freertos_idf.html


## 2.4. Izrada programa kojim se preko potenciometra postavlja perioda aktiviranja digitalnog izlaza mikrokontrolera ESP32

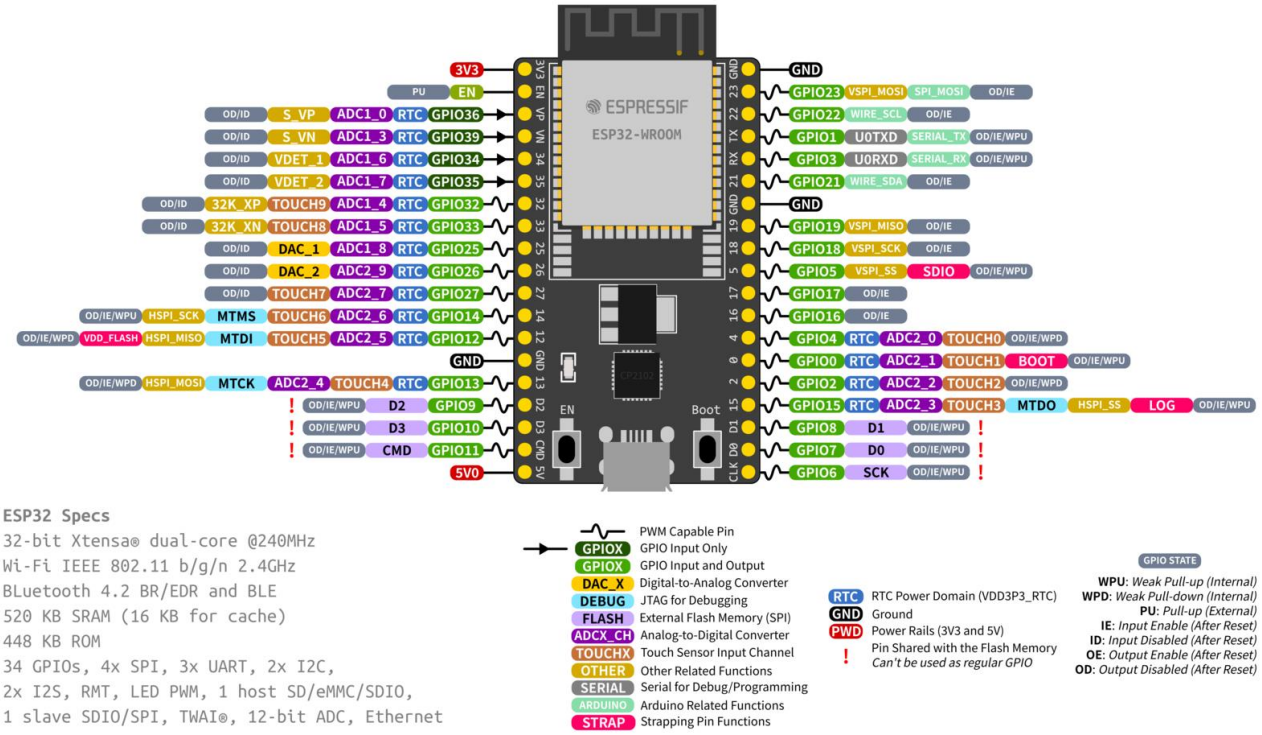
Preko potenciometra postavlja se perioda aktiviranja digitalnog izlaza mikrokontrolera (raspon je od 0,2 do 2 s) – kao ulaz koristiti priključak GPIO32. Izlaz koji se aktivira jest GPIO2. Preko LED indikatora vidjeti aktiviranje izlaza GPIO2.

**Nakon usvajanja ove nastavne teme čitatelj će moći:**

1. Programirati mikroupravljač koristeći odabranu programsku potporu
2. Povezati senzor sa sklopovskom potporom
3. Programirati sklopovsku opremu za rad s priključenim senzorom
4. Povezati aktuator sa sklopovskom potporom
5. Programirati sklopovsku opremu za rad s priključenim aktuatorom
6. Izraditi programsku potporu za vlastiti sustav interneta stvari

### 2.4.1. Osnovi koncepti

Mikrokontroler ESP32 ima niz različitih ulaznih i ulazno/izlaznih priključaka što se može vidjeti na narednoj slici.




*Slika 2.4.1 ADC ulazi mikrokontrolera*

Jedna od vrsta ulaza jest analogno-digitalni pretvornik (Analog to Digital Converter – ADC, npr. GPIO32) koji promjene dovedenoga analognog signala (promjene njegove naponske razine) pretvara u broj. U primjeru zadatka postavljanja perioda aktiviranja digitalnog izlaza preko potenciometra na ADC ulaz dovodi se analogni signal čija se naponska razina nalazi u rasponu od 0 V do 3,3 V (ovisno o poziciji potenciometra). Rezolucija pretvorbe analognog signala u digitalni (koliko se male promjene naponske razine mogu prikazati brojem) ovisi o broju bitova koji se koriste za prezentaciju naponske razine. U mikrokontroleru ESP32 standardno se koristi 12 bitova što znači da na raspolaganju postoji ukupno $2^{12}$ = 4096 brojeva s time da 0 predstavlja 0 V, a 4095 predstavlja 3,3 V.

Ulazi ADC mikrokontrolera ESP32 grupirani su u dvije jedinice: ADC_UNIT_1 i ADC_UNIT_2. U svakoj jedinici nalaze se ulazi ADC koji se identificiraju kao ADC_CHANNEL_0 do ADC_CHANNEL_1. Primjera radi ulaz GPIO 32 pripada jedinici ADC_UNIT_1 i identificira se kao ADC_CHANNEL_4.

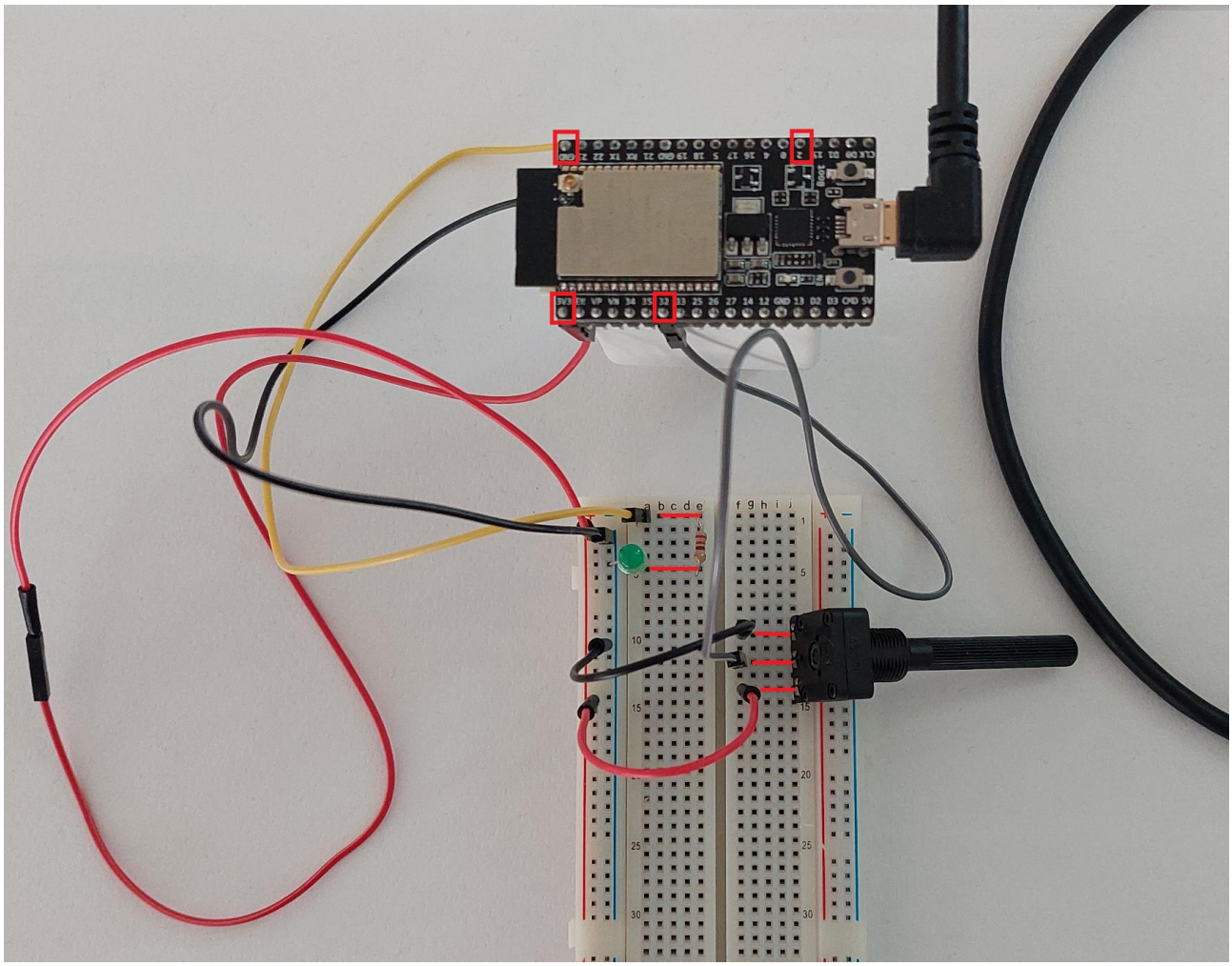
Procedura primjene ADC ulaza jest sljedeća:

1. ulaz ADC potrebno je inicijalizirati naredbom adc_oneshot_new_unit. Inicijalizacijom se postavlja jedinica kojoj ulaz ADC pripada (npr. za GPIO32 to je ADC_UNIT_1);
2. ulaz ADC potrebno je konfigurirati naredbom adc_oneshot_config_channel. Konfiguracija se sastoji od definiranja broja bitova koji će se koristiti u reprezentaciji naponske razine (npr. ADC_BITWIDTH_DEFAULT za standardni broj bitova ili npr. ADC_BITWIDTH_10 za 10 bitova) i od definiranja prigušenja ulaznog napona (npr. ADC_ATTEN_DB_11 znači da je prigušenje 11 dB, odnosno naponske su razine u intervalu od 150 mV do 2450 mV; ADC_ATTEN_DB_6 označava prigušenje od 6 dB, odnosno naponske razine u intervalu od 150 mV do 1750 mV itd. – prigušenje pokazuje da ESP32 pouzdano radi samo na određenom intervalu naponskih razina);

3. očitanje vrijednosti pretvorenoga analognog signala u digitalni radi se preko funkcije adc_oneshot_read. Njoj se kroz treći argument dostavlja adresa varijable u kojoj će se spremiti broj koji predstavlja naponsku razinu analognog signala.

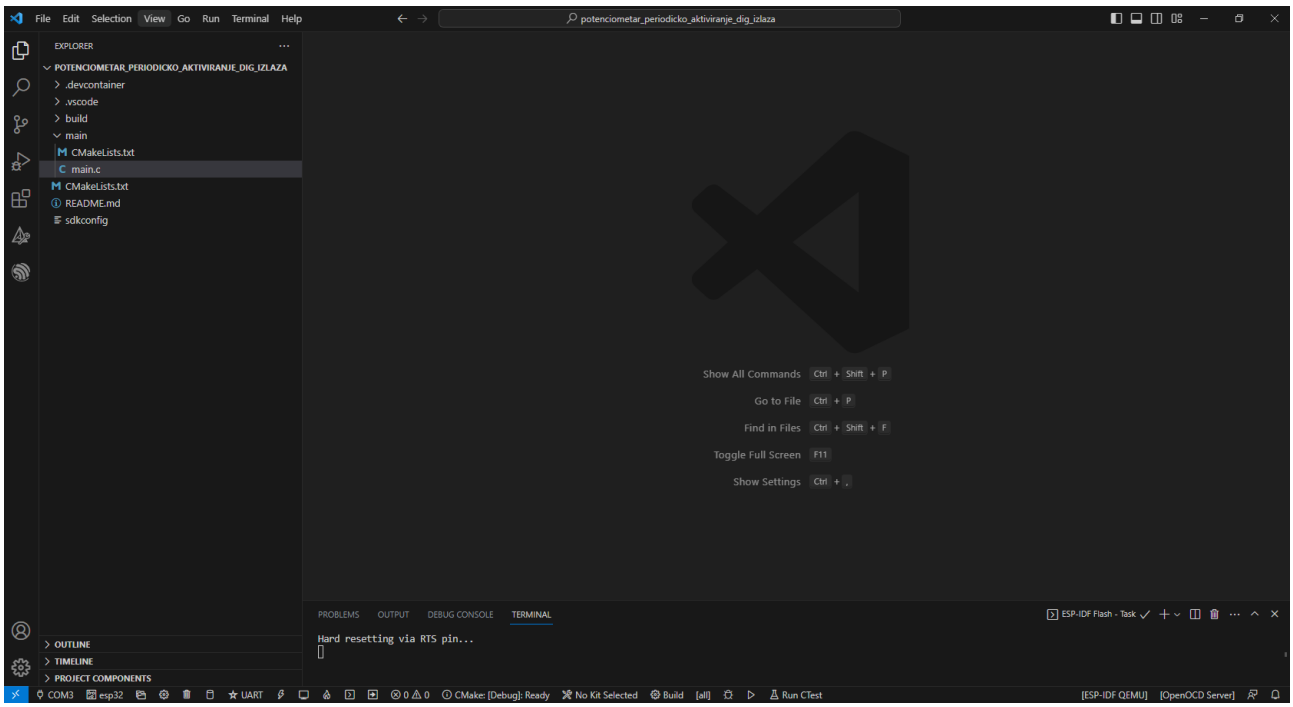
### 2.4.2. Rješenje radnog zadatka

Na mikrokontroler je potrebno povezati potenciometar na način da se njegovi krajnji priključci povežu na napajanje (+ pol i GND). Srednji priključak treba se povezati na ulaz ADC GPIO32. Potenciometar ima vrijednost 0..10 kOhm. Povezivanje LED indikatora s izlazom GPIO2 isto je kao i kod prethodnog zadatka.



*Slika 2.4.2 Povezivanje potenciometra i LED indikatora na priključke GPIO32 i GPIO2.*

U softveru Visual Studio Code potrebno je izraditi projekt potenciometar_periodicko_aktiviranje_dig_izlaza iz primjera projekta naziva sample_projects.




*Slika 2.4.3 Izrađen projekt potenciometar_periodicko_aktiviranje_dig_izlaza*

U datoteku main.c potrebno je napisati programski kôd.

```
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/gpio.h"
#include "esp_adc/adc_oneshot.h"

float period = 1;

static void getInput(void *arg)
{
    // -- -- -- -- -- -- -ADC1 Init-- -- -- -- -- -- - //
    adc_oneshot_unit_handle_t adc_handle;
    adc_oneshot_unit_init_cfg_t init_config = {
        .unit_id = ADC_UNIT_1,
    };
    adc_oneshot_new_unit(&init_config, &adc_handle);

    //-------------ADC1 Config---------------//
    adc_oneshot_chan_cfg_t config = {
        .bitwidth = ADC_BITWIDTH_DEFAULT,
        .atten = ADC_ATTEN_DB_11,
    };
    adc_oneshot_config_channel(adc_handle, ADC_CHANNEL_4, &config);

    int adc_value;
    while (1)
    {
        adc_oneshot_read(adc_handle, ADC_CHANNEL_4, &adc_value);
```

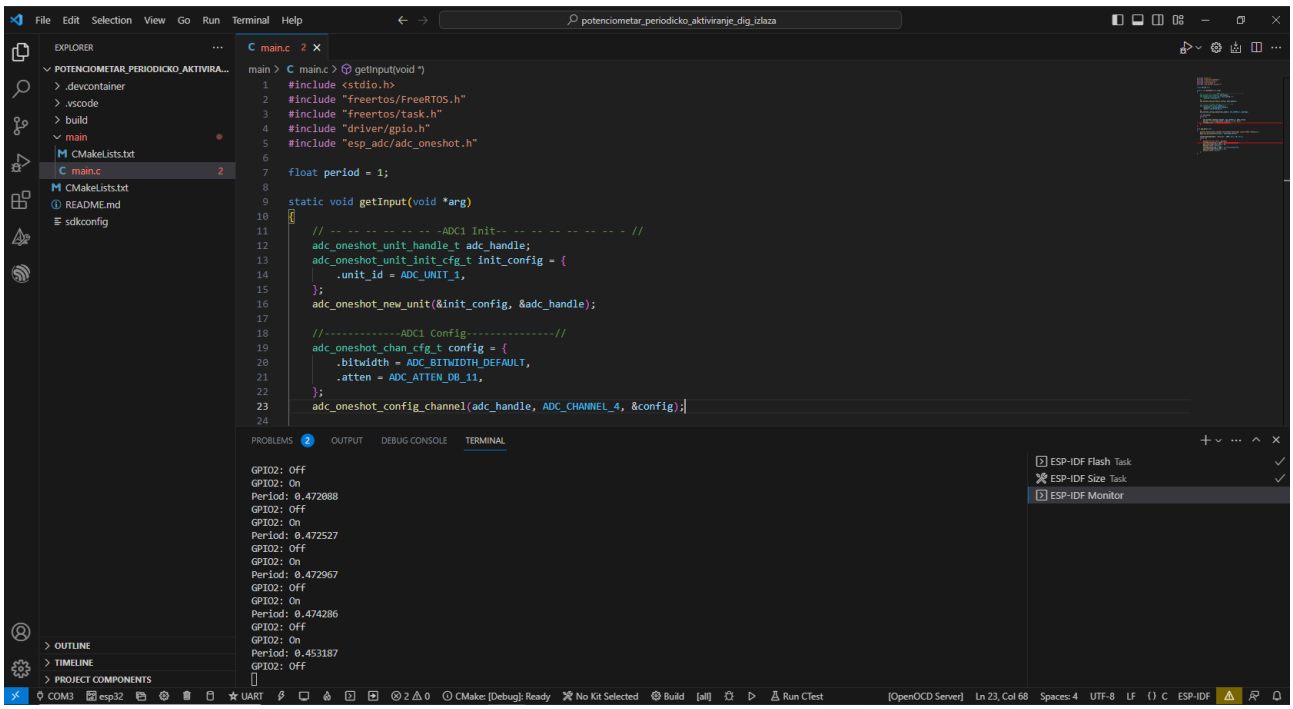
```
        period = (float)(adc_value / 4095.0 * 1.8 + 0.2);
        vTaskDelay(10 / portTICK_PERIOD_MS);
    }
}

void app_main(void)
{
    printf("POSTAVLJANJE PERIODE AKTIVIRANJA DIGITALNOG IZLAZA PREKO TIPKALA\n");
    gpio_set_direction(GPIO_NUM_2, GPIO_MODE_OUTPUT);

    xTaskCreate(getInput, "getInput", 2048, NULL, 10, NULL);
    while (1)
    {
        printf("Period: %f\n", period);
        vTaskDelay(period * 1000 / portTICK_PERIOD_MS);
        gpio_set_level(GPIO_NUM_2, 0);
        printf("GPIO2: Off\n");
        vTaskDelay(period * 1000 / portTICK_PERIOD_MS);
        gpio_set_level(GPIO_NUM_2, 1);
        printf("GPIO2: On\n");
    }
}
```

U glavnom dijelu programa nalazi se konfiguracija priključka GPIO2 (rad kao izlaz). Potom se izrađuje novi zadatak (*task*) za pozivanje funkcije getInput. Zatim se u beskonačnoj petlji naizmjenično pali i gasi izlazni digitalni priključak GPIO2, odnosno LED, u periodi koja je definirana unutar varijable period koja se postavlja u funkciji getInput.
U funkciji getInput inicijalizira se i konfigurira ulaz ADC ADC_CHANNEL_4, koji predstavlja priključak 32 mikrokontrolera. Nakon toga slijedi beskonačna petlja u kojoj se obavlja očitanje ulaza ADC (funkcija adc_oneshot_read). Vrijednost očitanja sprema se u varijablu adc_value. Analogni signal koji se dovodi na priključak 32 ima naponske razine od 0 do 3,3 V (ovisi o zakretanju potenciometra). Tu analognu vrijednost ADC pretvara u digitalnu vrijednost u rasponu od 0 do 4095 (12-bitna rezolucija).
Dobivenu vrijednost unutar intervala [0, 4095] potrebno je pretvoriti u vrijednost unutar intervala [0,2, 2] – to će biti vrijednost varijable period i predstavlja sekunde. To se radi tako da se prva vrijednost normalizira (dovede u interval [0, 1]) i zatim se ta normalizirana vrijednost skalira (dovodi u interval [0,2, 2].
Linija koda koja to radi jest:

```
period = (float)(adc_value / 4095.0 * 1.8 + 0.2);
```

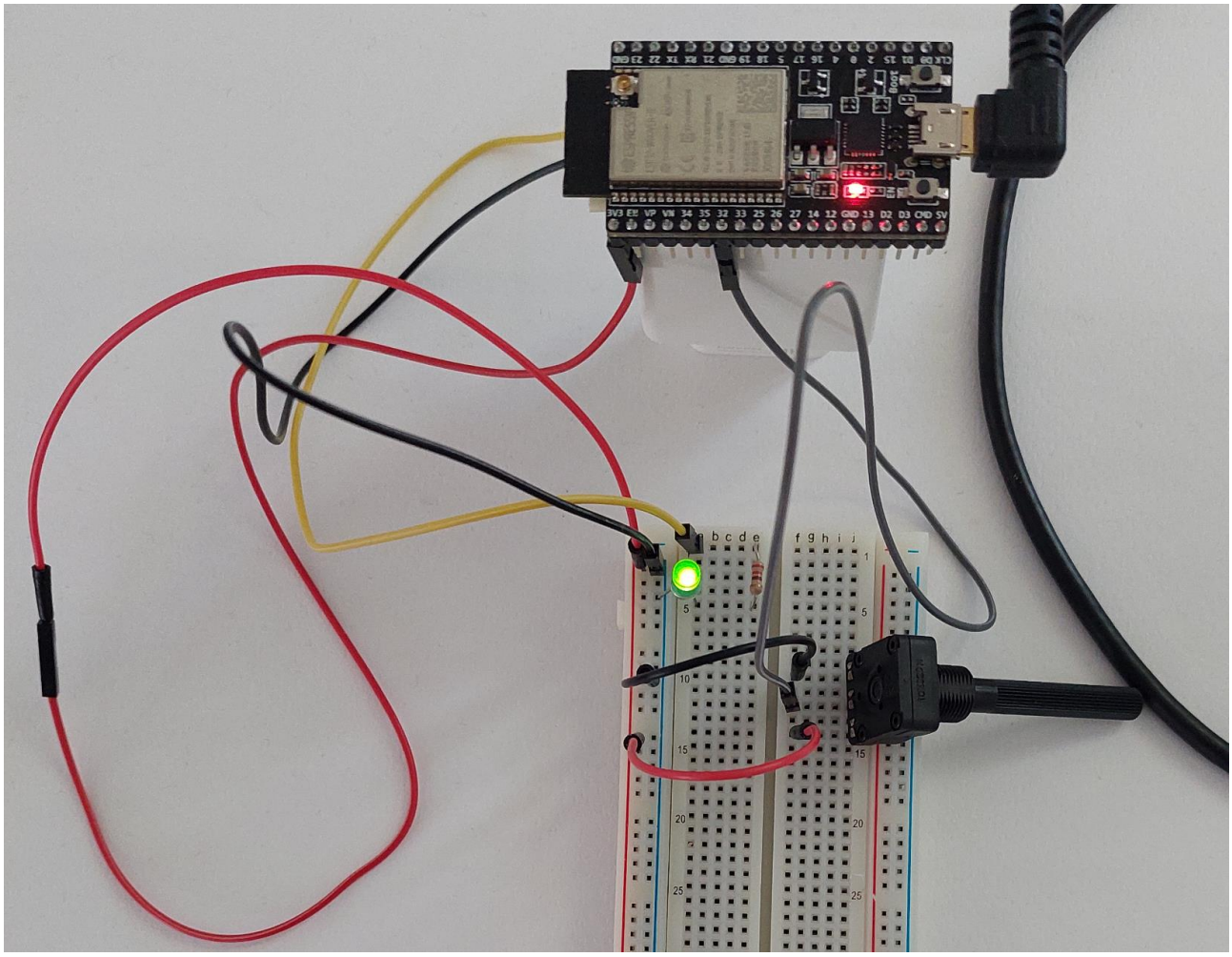
gdje je `adc_value / 4095.0` normalizacija a ostatak formule skaliranje.

Nakon što je program napisan, može ga se kompajlirati, poslati na mikrokontroler i pratiti u konzoli.




*Slika 2.4.4 Program pokrenut na mikrokontroleru*

Konačni izgled sustava za kontrolu periode aktiviranja digitalnog izlaza preko potenciometraprikazan je na narednoj slici.



*Slika 2.4.5 Gotov sustav za kontrolu periode aktiviranja digitalnog izlaza preko potenciometra*

### 2.4.3. Pitanja i zadaci

1. Izraditi isti program, ali uz korištenje drugih priključaka mikrokontrolera.
2. Prikazani program izmijeniti tako da se omogući i gašenje digitalnog izlaza.
3. Mikrokontroler ESP32 posjeduje samo jedan ADC priključak.
    a. Točno
    b. Netočno

4. ADC priključak mikrokontrolera ESP32 koristi se za pretvorbu:
    a. analognog signala u analogni signal manje naponske razine
    b. analognog signala u digitalni signal
    c. digitalnog signala u analogni signal
    d. digitalnog signala u digitalni signal s manje bitova

5. Rezolucija pretvorbe naponske razine u broj ovisi o sljedećem:
    a. naponskoj razini signala
    b. naponskoj razini na ulazu mikrokontrolera
    c. broju bitova za reprezentaciju naponske razine
    d. jedinici kojoj ulaz pripada
    e. kanalu kojem ulaz pripada

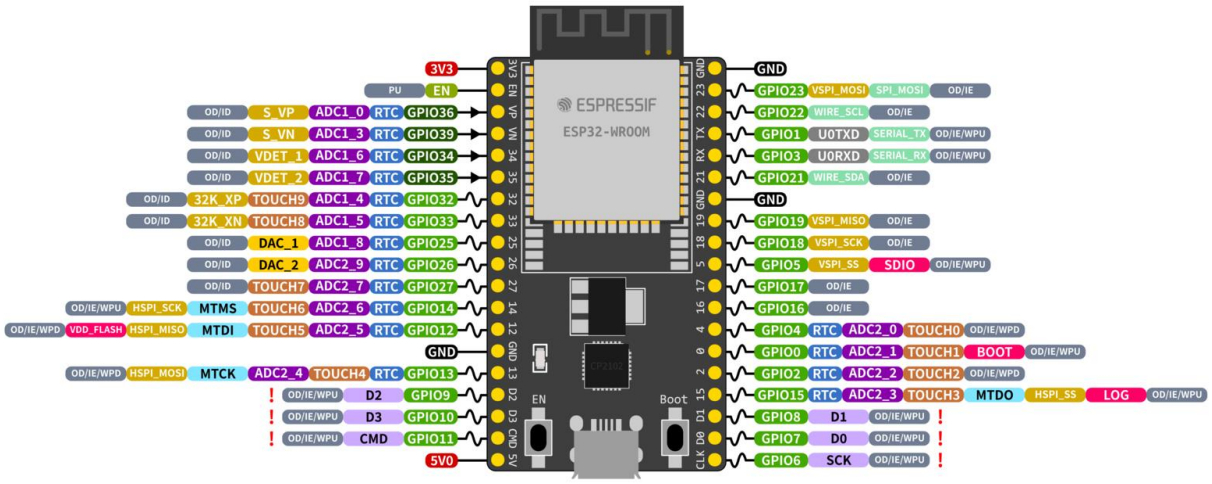
6. Potrebno je ispravno poredati korake kod rada s ulazom ADC:
    a. očitanje vrijednosti analognog signala
    b. konfiguriranje ulaza ADC
    c. inicijalizacija ulaza ADC

7. Kojom se naredbom inicijalizira ADC ulaz?
    a. adc_oneshot_new_channel
    b. adc_oneshot_read
    c. adc_oneshot_config_unit
    d. adc_oneshot_config_channel
    e. adc_oneshot_new_unit

8. Kojom se naredbom konfigurira ulaz ADC?
    a. adc_oneshot_new_channel
    b. adc_oneshot_read
    c. adc_oneshot_config_unit
    d. adc_oneshot_config_channel
    e. adc_oneshot_new_unit

9. Kojom se naredbom očitava ulaz ADC?
    a. adc_oneshot_new_channel
    b. adc_oneshot_read
    c. adc_oneshot_config_unit
    d. adc_oneshot_config_channel
    e. adc_oneshot_new_unit

10. Za konfiguriranje ulaza ADC GPIO14 koji su ispravni identifikatori jedinice i kanala:




a. ADC_UNIT_1 i ADC_CHANNEL_14
b. ADC_UNIT_2 i ADC_CHANNEL_14
c. ADC_UNIT_1 i ADC_CHANNEL_6
d. ADC_UNIT_2 i ADC_CHANNEL_6
e. ADC_UNIT_2 i ADC_CHANNEL_14_6

### 2.4.4. Literatura i izvori


1. Analog to Digital Converter (ADC) Oneshot Mode Driver, https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/adc_oneshot.html
2. ESP32 ADC – Read Analog Values with Arduino IDE, https://randomnerdtutorials.com/esp32-adc-analog-read-arduino-ide/
3. ESP32 ADC with ESP-IDF Measure Analog Inputs, https://esp32tutorials.com/esp32-adc-esp-idf/

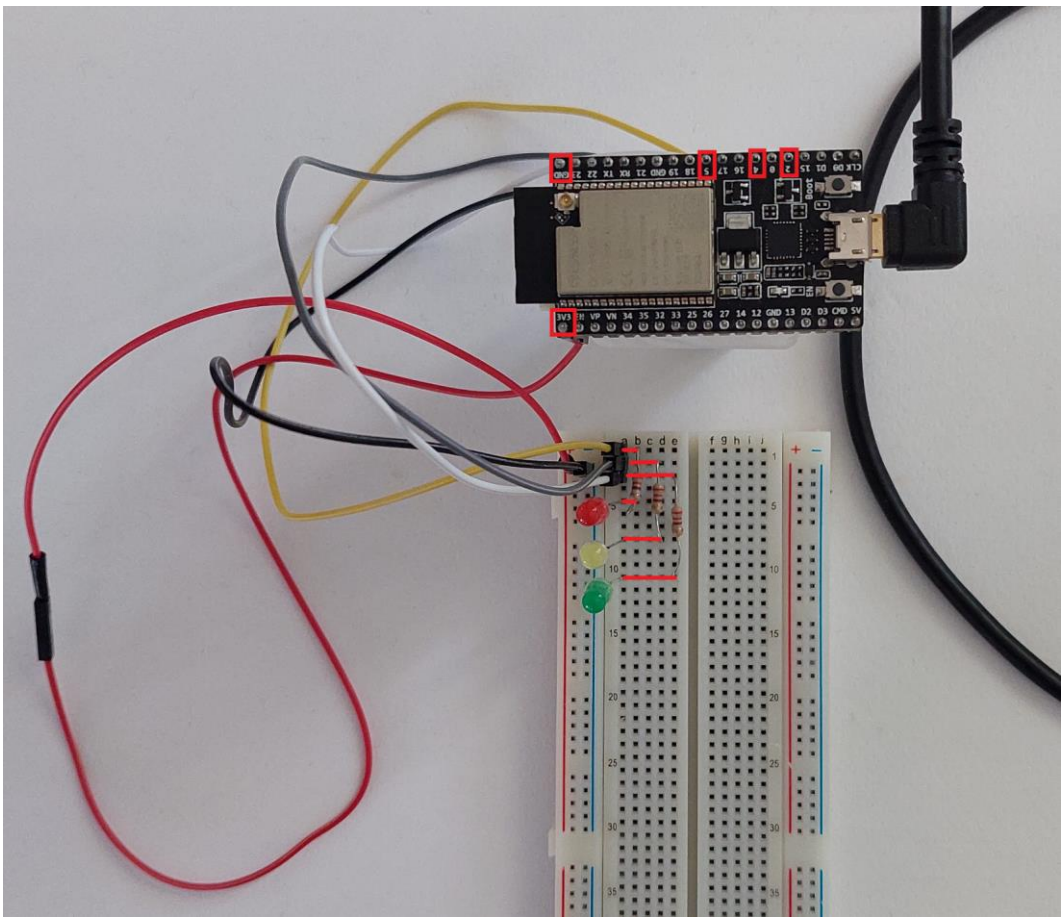

## 2.5. Izrada programa kojim se u istoj periodi naizmjenično aktiviraju tri digitalna izlaza mikrokontrolera ESP32

Na mikrokontroleru se u periodi od 500 ms naizmjenično aktiviraju tri digitalna izlaza na koja su spojene tri LED. Izlazi koje treba koristiti jesu GPIO2, GPIO4 i GPIO5.

**Nakon usvajanja ove nastavne teme čitatelj će moći:**
1. Programirati mikroupravljač koristeći odabranu programsku potporu
2. Povezati aktuator sa sklopovskom potporom
3. Programirati sklopovsku opremu za rad s priključenim aktuatorom
4. Izraditi programsku potporu za vlastiti sustav interneta stvari

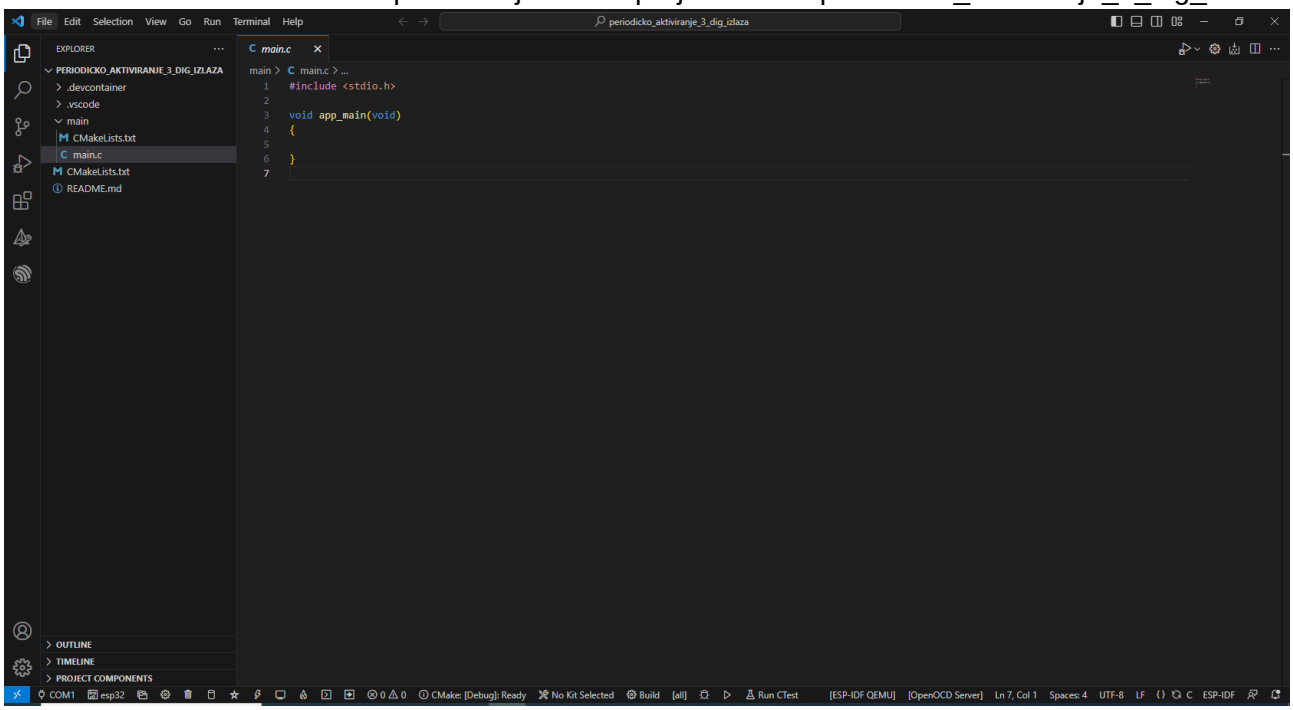
### 2.5.1. Rješenje radnog zadatka

Tri LED indikatora potrebno je povezati na način da je svaki povezan s katodom na GND priključak napajanja, a anoda na jedan od digitalnih priključaka (GPIO2, GPIO4 ili GPIO5) – paziti da se postavi otpornik od barem 200 ohm između LED indikatora i digitalnog priključka.



*Slika 2.5.1 Povezivanje tri LED indikatora s tri digitalna izlaza (GPIO2, GPIO3 i GPIO4)*

U softveru Visual Studio Code potrebno je izraditi projekt naziva periodicko_aktiviranje_3_dig_izlaza.



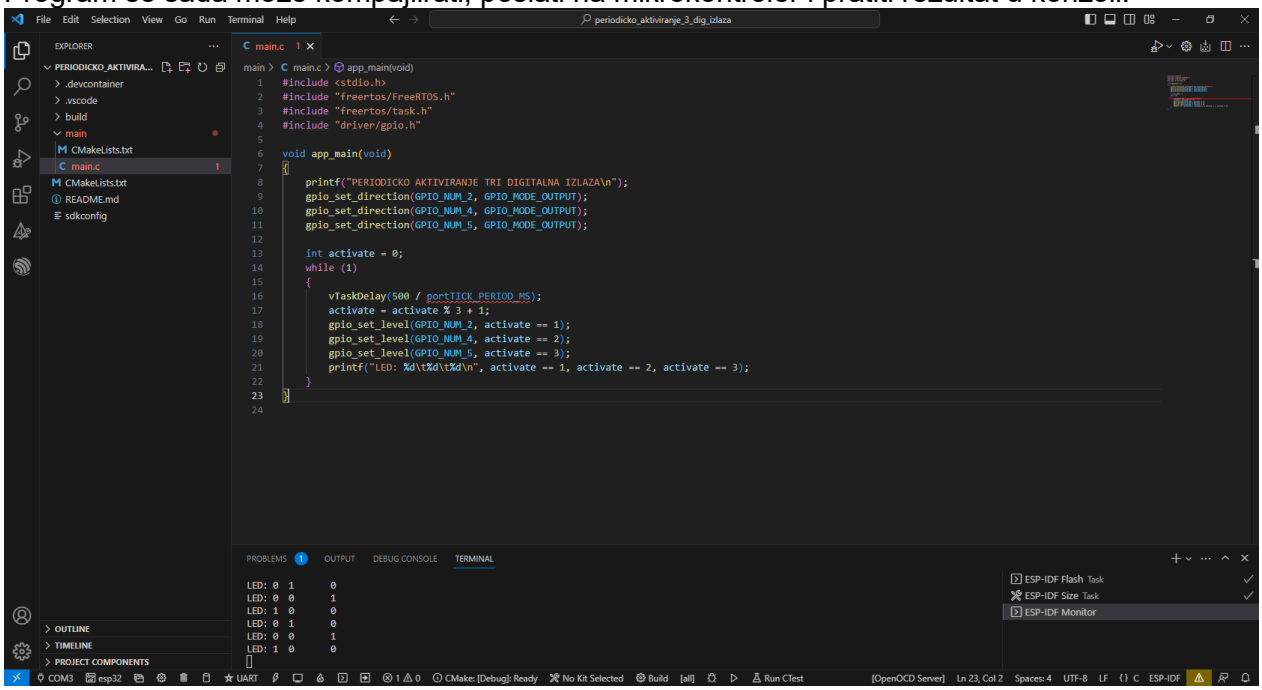

*Slika 2.5.2 Izrađen projekt periodicko_aktiviranje_3_dig_izlaza*

Sada se u datoteci main.c može napisati program koji će periodički paliti i gasiti tri digitalna izlaza mikrokontrolera.

```c
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/gpio.h"

void app_main(void)
{
    printf("PERIODICKO AKTIVIRANJE TRI DIGITALNA IZLAZA\n");
    gpio_set_direction(GPIO_NUM_2, GPIO_MODE_OUTPUT);
    gpio_set_direction(GPIO_NUM_4, GPIO_MODE_OUTPUT);
    gpio_set_direction(GPIO_NUM_5, GPIO_MODE_OUTPUT);

    int activate = 0;
    while (1)
    {
        vTaskDelay(500 / portTICK_PERIOD_MS);
        activate = activate % 3 + 1;
        gpio_set_level(GPIO_NUM_2, activate == 1);
        gpio_set_level(GPIO_NUM_4, activate == 2);
        gpio_set_level(GPIO_NUM_5, activate == 3);
        printf("LED: %d\t%d\t%d\n", activate == 1, activate == 2, activate == 3);
    }
}
```

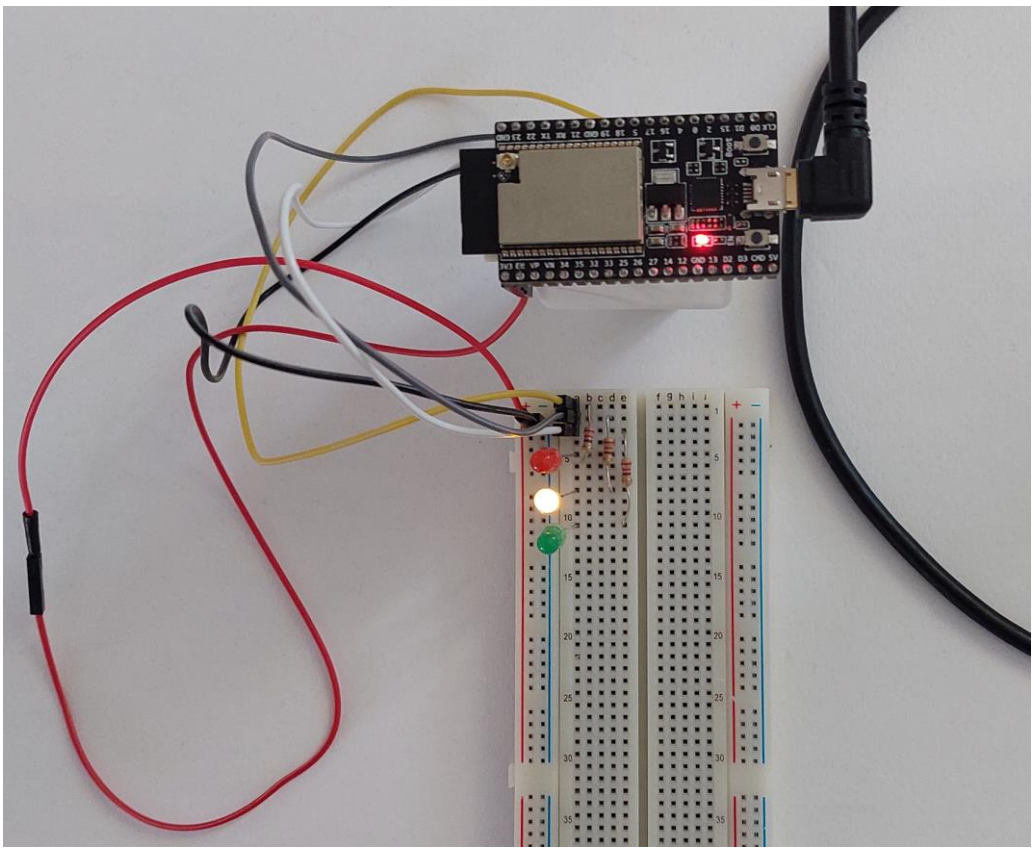
Program započinje konfiguracijom priključaka GPIO2, GPIO4 i GPIO5 kao digitalnih izlaza. Nakon toga slijedi beskonačna petlja u kojoj se svakih 500 ms mijenjanju razine digitalnih izlaza. Treba obratiti pažnju na izvedbu brojača (varijabla activate) koja će poprimati vrijednosti u intervalu [1, 3], znači 1, 2, 3, 1, 2, 3, 1… Razina koju treba poprimiti izlaz definirana je preko logičkog izlaza (activate==1, activate==2 i activate==3). U konzoli se ispisuje koji je LED indikator aktiviran.

Program se sada može kompajlirati, poslati na mikrokontroler i pratiti rezultat u konzoli.




*Slika 2.5.3 Pokrenuti projekt periodicko_aktiviranje_3_dig_izlaza*

Gotov sustav za naizmjenično aktiviranje tri digitalna izlaza prikazuje naredna slika.



*Slika 2.5.4 Gotov sustav za naizmjenično aktiviranje tri digitalna izlaza*

### 2.5.2. Pitanja i zadaci

1. Izraditi isti program, ali uz korištenje drugih priključaka mikrokontrolera.
2. Izraditi program kojim se perioda naizmjeničnog aktiviranja digitalnog izlaza mijenja preko tipkala.
3. Izraditi program kojim se perioda naizmjeničnog aktiviranja digitalnog izlaza mijenja preko potenciometra.
4. Prikazani program izmijeniti tako da se omogući paljenje/gašenje procesa naizmjeničnog aktiviranja digitalnog izlaza preko tipkala.

### 2.5.3. Literatura i izvori


1. Blink Example, https://github.com/espressif/esp-idf/tree/32472536715d674c160a2565795895e0273f8cde/examples/get-started/blink
2. Example: GPIO, https://github.com/espressif/esp-idf/tree/32472536715d674c160a2565795895e0273f8cde/examples/peripherals/gpio/generic_gpio
3. GPIO & RTC GPI, https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/gpio.html
4. Hello World Example, https://github.com/espressif/esp-idf/tree/32472536715d674c160a2565795895e0273f8cde/examples/get-started/hello_world

## 2.6. Izrada programa kojim se preko tipkala naizmjenično aktiviraju tri digitalna izlaza mikrokontrolera ESP32

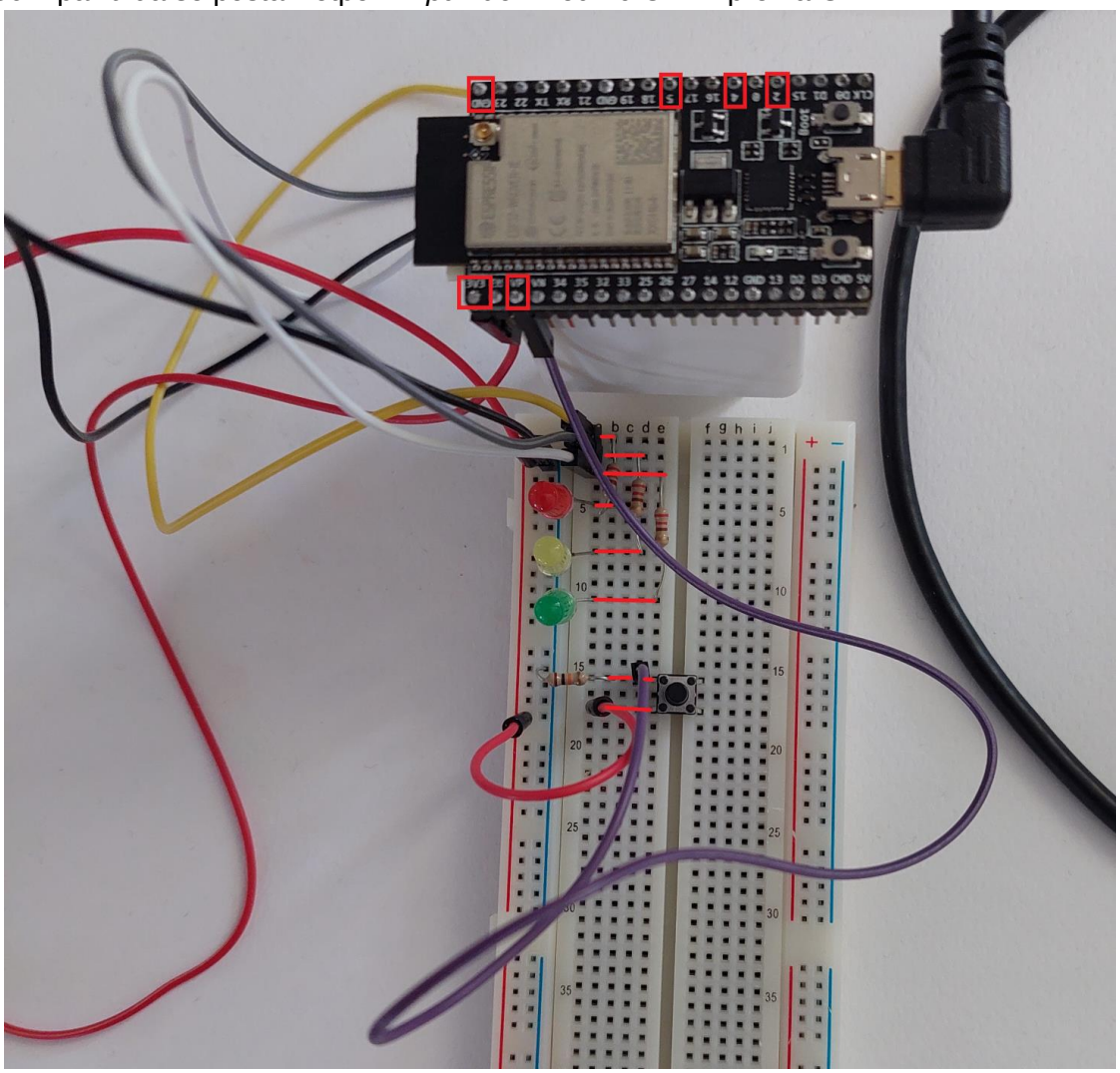
Preko tipkala naizmjenično se aktiviraju digitalni izlazi GPIO2, GPIO4 i GPIO5. Na njih se trebaju spojiti tri LED indikatora. Tipkalo je potrebno povezati s digitalnim ulazom GPIO36.

**Nakon usvajanja ove nastavne teme čitatelj će moći:**

1. Programirati mikroupravljač koristeći odabranu programsku potporu
2. Povezati senzor sa sklopovskom potporom
3. Programirati sklopovsku opremu za rad s priključenim senzorom
4. Povezati aktuator sa sklopovskom potporom
5. Programirati sklopovsku opremu za rad s priključenim aktuatorom
6. Izraditi programsku potporu za vlastiti sustav interneta stvari

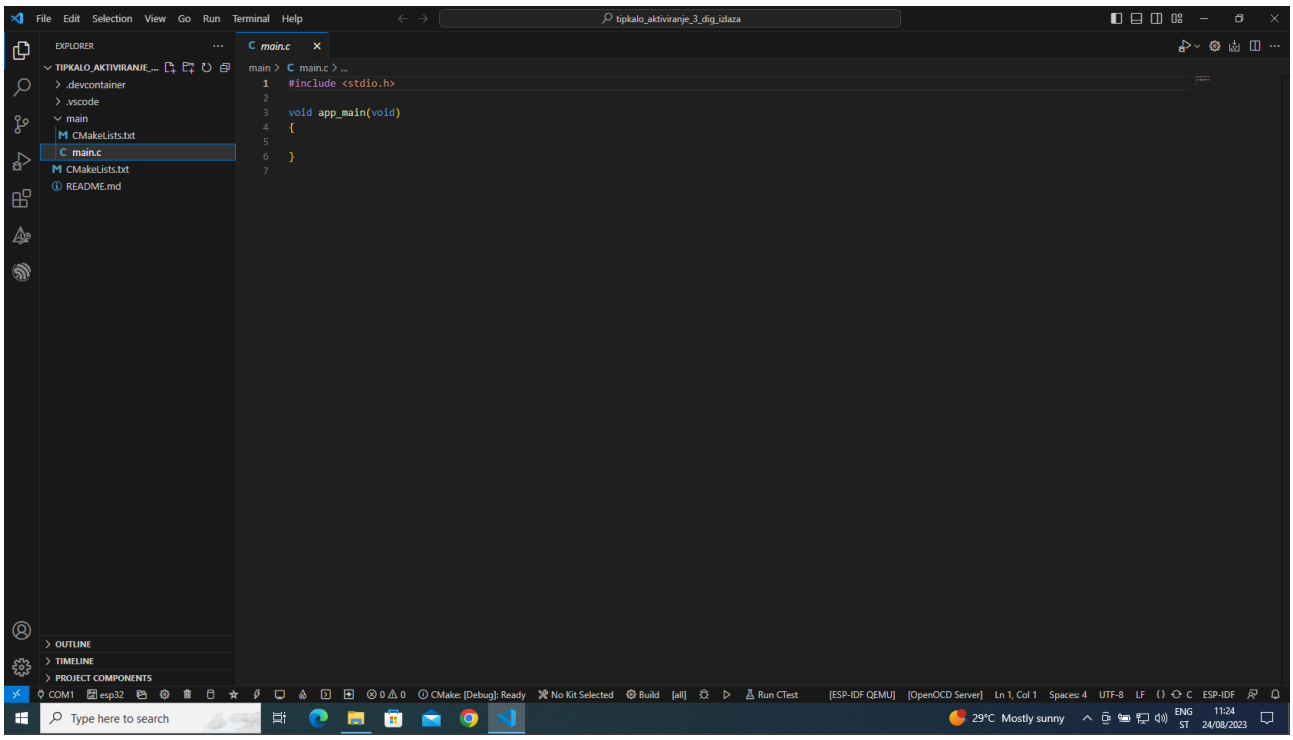
### 2.6.1. Rješenje radnog zadatka

Tri LED indikatora potrebno je povezati na način kako je to ranije objašnjeno u prethodnom zadatku – paziti na potrebni otpornik od minimalno 200 ohm. Tipkalo je potrebno povezati s digitalnim ulazom GPIO36 – paziti da se postavi otpornik *pull-down* od 10 Okhm prema GND.



*Slika 2.6.1 Povezivanje tipkala i tri LED*

U softveru Visual Studio Code potrebno je izraditi projekt naziva tipkalo_aktiviranje_3_dig_izlaza.



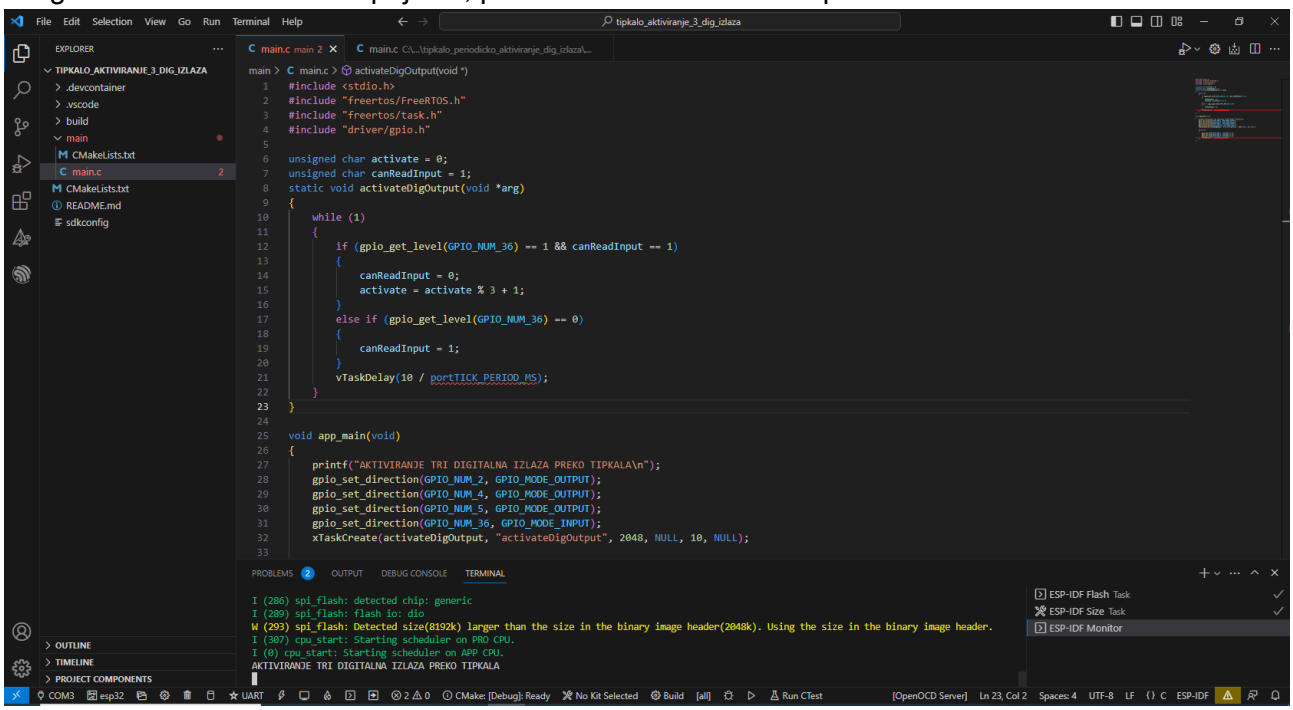

*Slika 2.6.2 Izrađen projekt tipkalo_aktiviranje_3_dig_izlaza*

U datoteku main.c može se upisati program za kontrolu tri digitalna izlaza preko tipkala.

```
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/gpio.h"

unsigned char activate = 0;
unsigned char canReadInput = 1;
static void activateDigOutput(void *arg)
{
    while (1)
    {
        if (gpio_get_level(GPIO_NUM_36) == 1 && canReadInput == 1)
        {
            canReadInput = 0;
            activate = activate % 3 + 1;
        }
        else if (gpio_get_level(GPIO_NUM_36) == 0)
        {
            canReadInput = 1;
        }
        vTaskDelay(10 / portTICK_PERIOD_MS);
    }
}

void app_main(void)
{
    printf("AKTIVIRANJE TRI DIGITALNA IZLAZA PREKO TIPKALA\n");
    gpio_set_direction(GPIO_NUM_2, GPIO_MODE_OUTPUT);
```

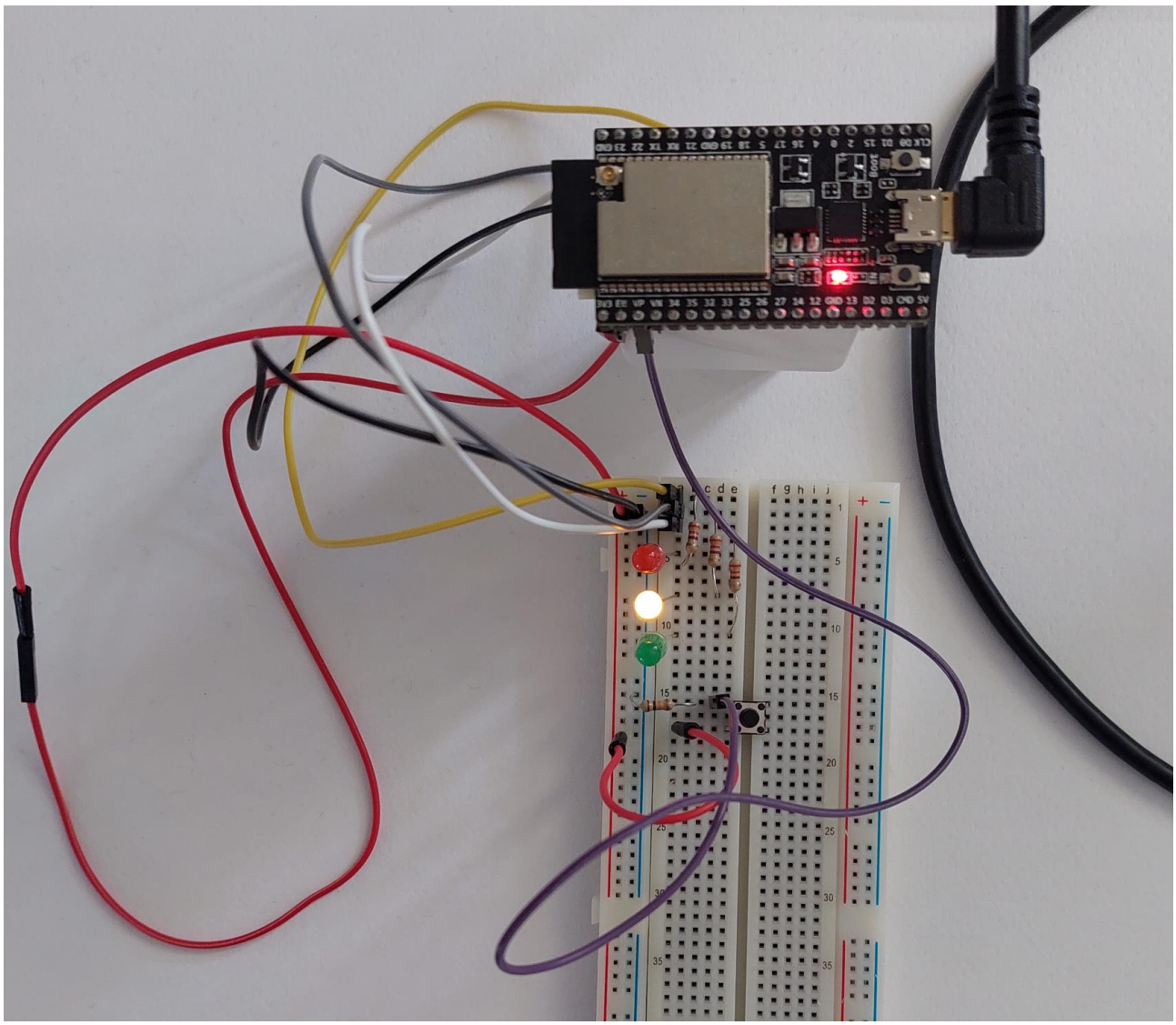
```
    gpio_set_direction(GPIO_NUM_4, GPIO_MODE_OUTPUT);
    gpio_set_direction(GPIO_NUM_5, GPIO_MODE_OUTPUT);
    gpio_set_direction(GPIO_NUM_36, GPIO_MODE_INPUT);
    xTaskCreate(activateDigOutput, "activateDigOutput", 2048, NULL, 10, NULL);

    while (1)
    {
        gpio_set_level(GPIO_NUM_2, activate == 1);
        gpio_set_level(GPIO_NUM_4, activate == 2);
        gpio_set_level(GPIO_NUM_5, activate == 3);
        vTaskDelay(10 / portTICK_PERIOD_MS);
    }
}
```

U glavnom dijelu programa konfiguriraju se četiri priključka (izlazi: GPIO2, GPIO4 i GPIO5; izlaz: GPIO36). Nakon toga izrađuje se zadaća (*task*) za funkciju activateDigOutput i potom se u beskonačnoj petlji aktiviraju digitalni izlazi ovisno o vrijednosti varijable activate.
Varijabla activate postavlja se u funkciji activateDigOutput tek kada se pritisne tipkalo koje izaziva pojavu razine 1 na digitalnom ulazu GPIO36. Varijabla activate jest ciklički brojač koji poprima vrijednost u intervalu [1, 3] – 1, 2, 3, 1, 2, 3, 1…
Varijabla canReadInput koristi se kao statusna varijabla kojom se onemogućuje pojava višestrukog očitanja pritisnutog tipkala (odnosno vrijednosti 1 na ulazi GPIO36).
Program se sada može kompajlirati, poslati na mikrokontroler i pratiti kroz konzolu.




*Slika 2.6.3 Pokrenut program za naizmjenično aktiviranje tri digitalna izlaza preko jednog tipkala*

Naredna slika prikazuje potpun sustav za naizmjenično aktiviranje tri digitalna izlaza preko jednog tipkala u radu.



*Slika 2.6.4 Potpun sustav za naizmjenično aktiviranje tri digitalna izlaza preko jednog tipkala u radu*

## 2.6.2. Pitanja i zadaci

1. Izraditi isti program, ali uz korištenje drugih priključaka mikrokontrolera.
2. Izraditi program kojim se kontrolira više digitalnih izlaza.
3. Izmijeniti program tako da se tipkalom gasi cjelokupni sustav – tipkalo ima četiri stanja (1 – LED1, 2 – LED2, 3 – LED3 i 4 – gašenje sustava).
4. Izraditi program tako da se izabrani LED periodički pali i gasi.

## 2.6.3. Literatura i izvori


1. Blink Example, https://github.com/espressif/esp-idf/tree/32472536715d674c160a2565795895e0273f8cde/examples/get-started/blink
2. Example: GPIO, https://github.com/espressif/esp-idf/tree/32472536715d674c160a2565795895e0273f8cde/examples/peripherals/gpio/generic_gpio
3. GPIO & RTC GPIO, https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/gpio.html
4. Hello World Example, https://github.com/espressif/esp-idf/tree/32472536715d674c160a2565795895e0273f8cde/examples/get-started/hello_world

## 2.7. Izrada programa kojim se preko potenciometra naizmjenično aktiviraju tri digitalna izlaza mikrokontrolera ESP32

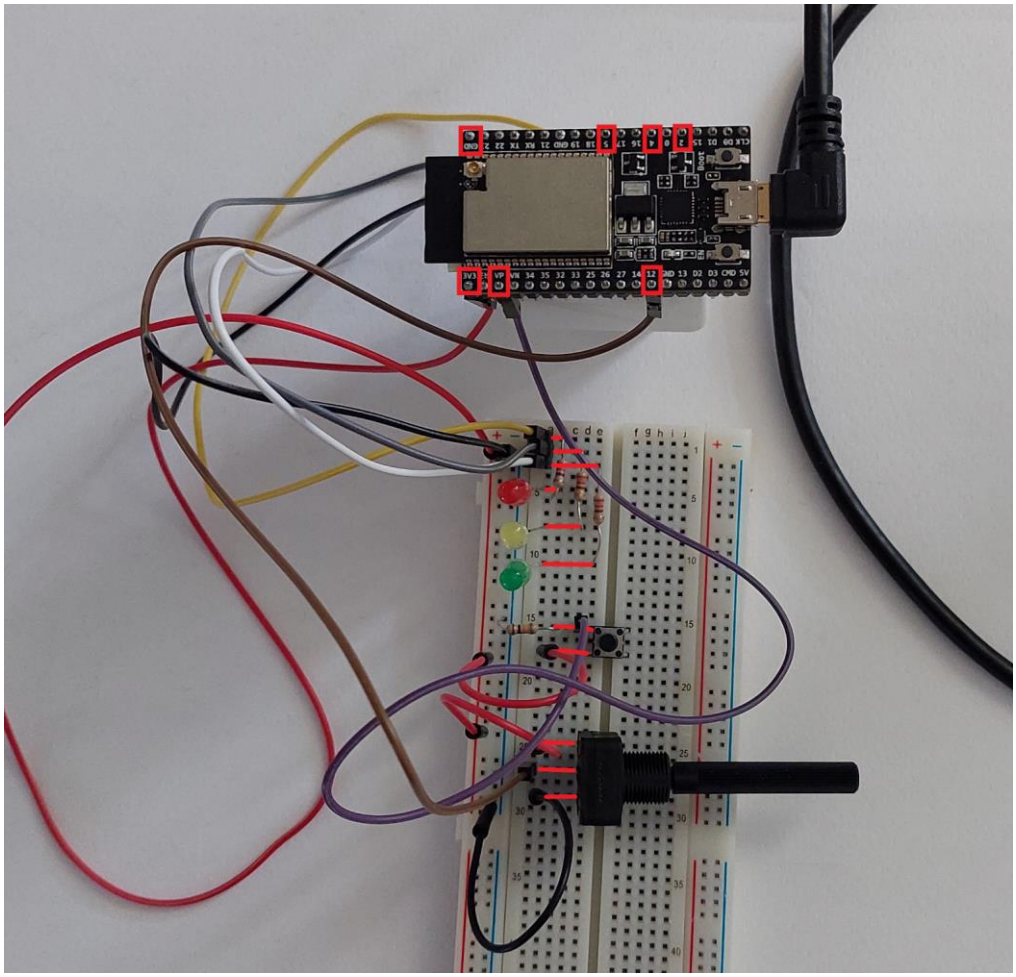
Preko potenciometra naizmjenično se aktiviraju digitalni izlazi GPIO2, GPIO4 i GPIO5. Na njih se trebaju spojiti tri LED indikatora. Potenciometar je potrebno spojiti na ulaz ADC GPIO12. Tipkalo je potrebno povezati s digitalnim ulazom GPIO36 i njega treba koristiti kako bi se cjelokupni sustav gasio/palio.

**Nakon usvajanja ove nastavne teme čitatelj će moći:**

1. Programirati mikroupravljač koristeći odabranu programsku potporu
2. Povezati senzor sa sklopovskom potporom
3. Programirati sklopovsku opremu za rad s priključenim senzorom
4. Povezati aktuator sa sklopovskom potporom
5. Programirati sklopovsku opremu za rad s priključenim aktuatorom
6. Izraditi programsku potporu za vlastiti sustav interneta stvari

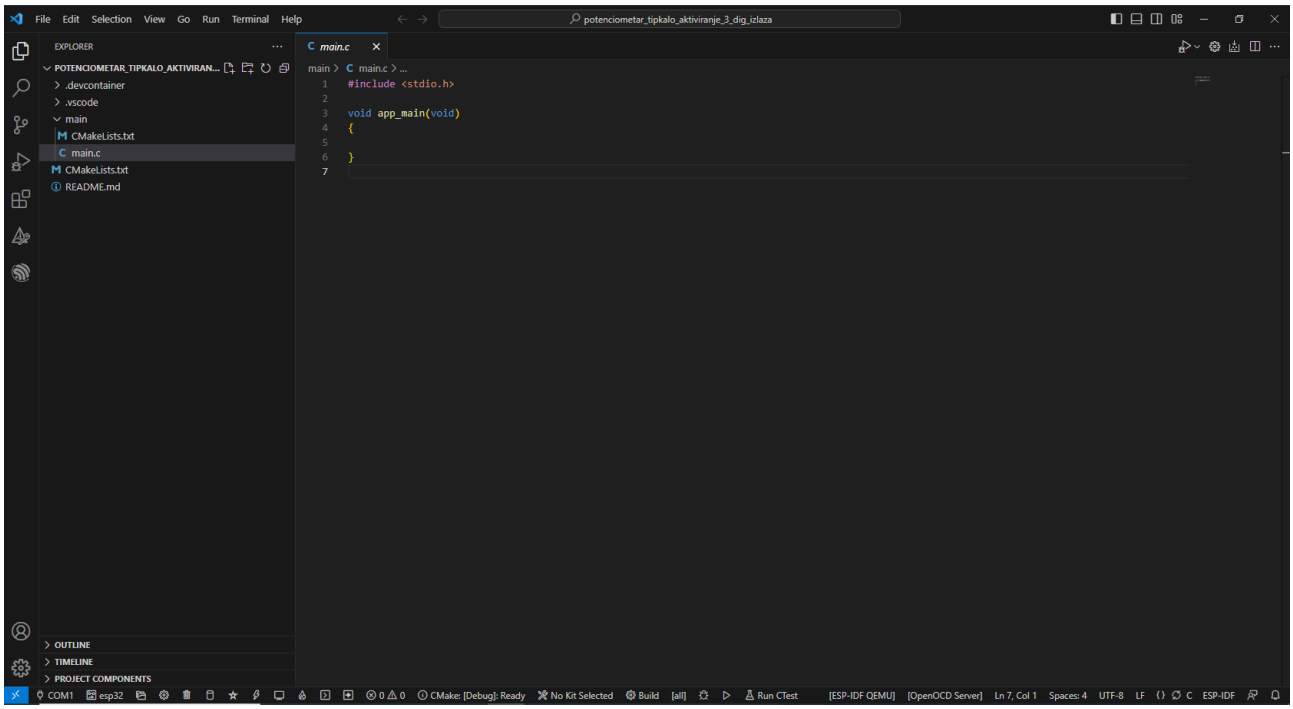
### 2.7.1. Rješenje radnog zadatka

Potenciometar je potrebno povezati na ulaz ADC GPIO12, dok je tipkalo potrebno povezati na digitalni ulaz GPIO36 – paziti na potrebni otpornik od 10 kOhm. Tri LED indikatora potrebno je povezati na digitalne ulaze GPIO2, GPIO4 i GPIO5 – paziti na potrebni otpornik od minimalno 200 ohm i na polaritet dioda.



*Slika 2.7.1 Povezivanje tipkala, potenciometra i LED indikatora s odgovarajućim priključcima mikrokontrolera*

U softveru Visual Studio Code potrebno je izraditi novi projekt naziva potenciometar_tipkalo_aktiviranje_3_dig_izlaza.




*Slika 2.7.2 Izrađen projekt potenciometar_tipkalo_aktiviranje_3_dig_izlaza*

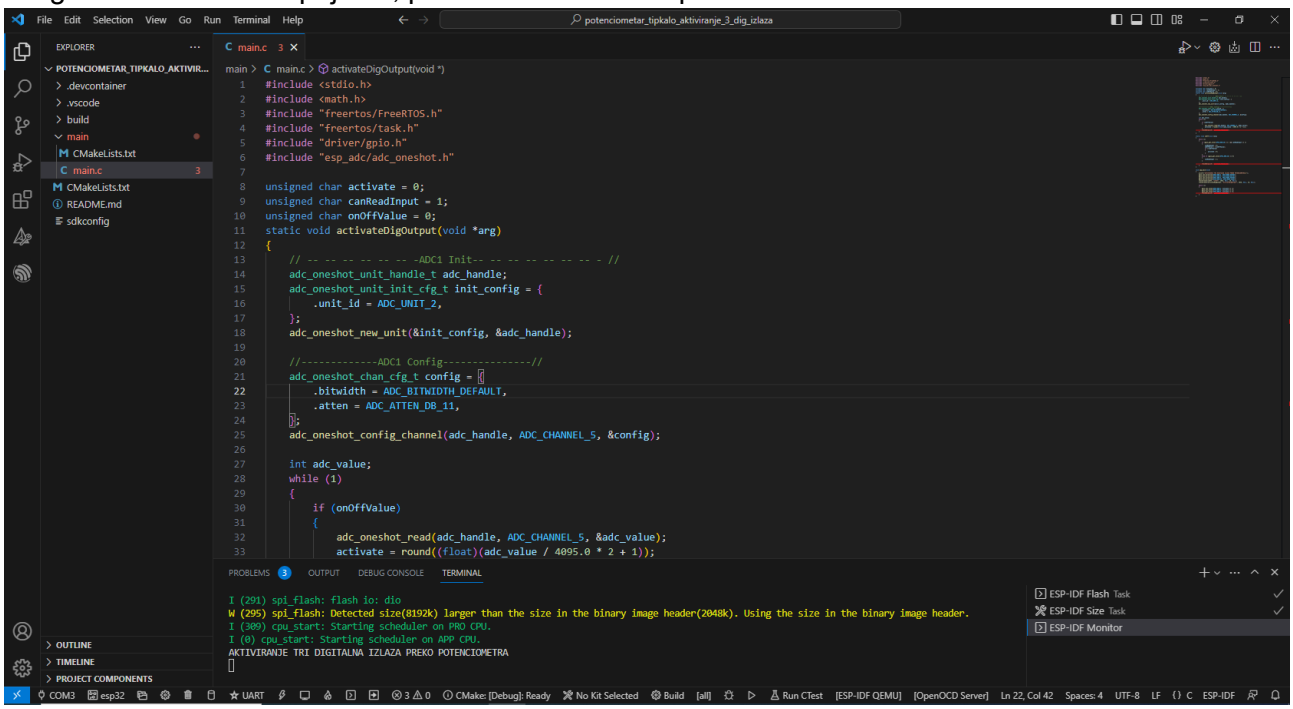
U datoteci main.c može se upisati program za naizmjenično aktiviranje tri digitalna izlaza preko potenciometra te paljenja/gašenja sustava preko tipkala.

```
#include <stdio.h>
#include <math.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/gpio.h"
#include "esp_adc/adc_oneshot.h"

unsigned char activate = 0;
unsigned char canReadInput = 1;
unsigned char onOffValue = 0;
static void activateDigOutput(void *arg)
{
    // -- -- -- -- -- -- -ADC1 Init-- -- -- -- -- -- -- - //
    adc_oneshot_unit_handle_t adc_handle;
    adc_oneshot_unit_init_cfg_t init_config = {
        .unit_id = ADC_UNIT_2,
    };
    adc_oneshot_new_unit(&init_config, &adc_handle);

    //-------------ADC1 Config---------------//
    adc_oneshot_chan_cfg_t config = {
        .bitwidth = ADC_BITWIDTH_DEFAULT,
        .atten = ADC_ATTEN_DB_11,
    };
```

```
    adc_oneshot_config_channel(adc_handle, ADC_CHANNEL_5, &config);

    int adc_value;
    while (1)
    {
        if (onOffValue)
        {
            adc_oneshot_read(adc_handle, ADC_CHANNEL_5, &adc_value);
            activate = round((float)(adc_value / 4095.0 * 2 + 1));
        }
        vTaskDelay(10 / portTICK_PERIOD_MS);
    }
}

static void onOff(void *arg)
{
    while (1)
    {
        if (gpio_get_level(GPIO_NUM_36) == 1 && canReadInput == 1)
        {
            canReadInput = 0;
            onOffValue = (!onOffValue);
            if (!onOffValue)
            {
                activate = 0;
            }
        }
        else if (gpio_get_level(GPIO_NUM_36) == 0)
        {
            canReadInput = 1;
        }

        vTaskDelay(10 / portTICK_PERIOD_MS);
    }
}

void app_main(void)
{
    printf("AKTIVIRANJE TRI DIGITALNA IZLAZA PREKO POTENCIOMETRA\n");
    gpio_set_direction(GPIO_NUM_2, GPIO_MODE_OUTPUT);
    gpio_set_direction(GPIO_NUM_4, GPIO_MODE_OUTPUT);
    gpio_set_direction(GPIO_NUM_5, GPIO_MODE_OUTPUT);
    gpio_set_direction(GPIO_NUM_36, GPIO_MODE_INPUT);
    xTaskCreate(onOff, "onOff", 2048, NULL, 10, NULL);
    xTaskCreate(activateDigOutput, "activateDigOutput", 2048, NULL, 10, NULL);

    while (1)
    {
        gpio_set_level(GPIO_NUM_2, activate == 1);
        gpio_set_level(GPIO_NUM_4, activate == 2);
```

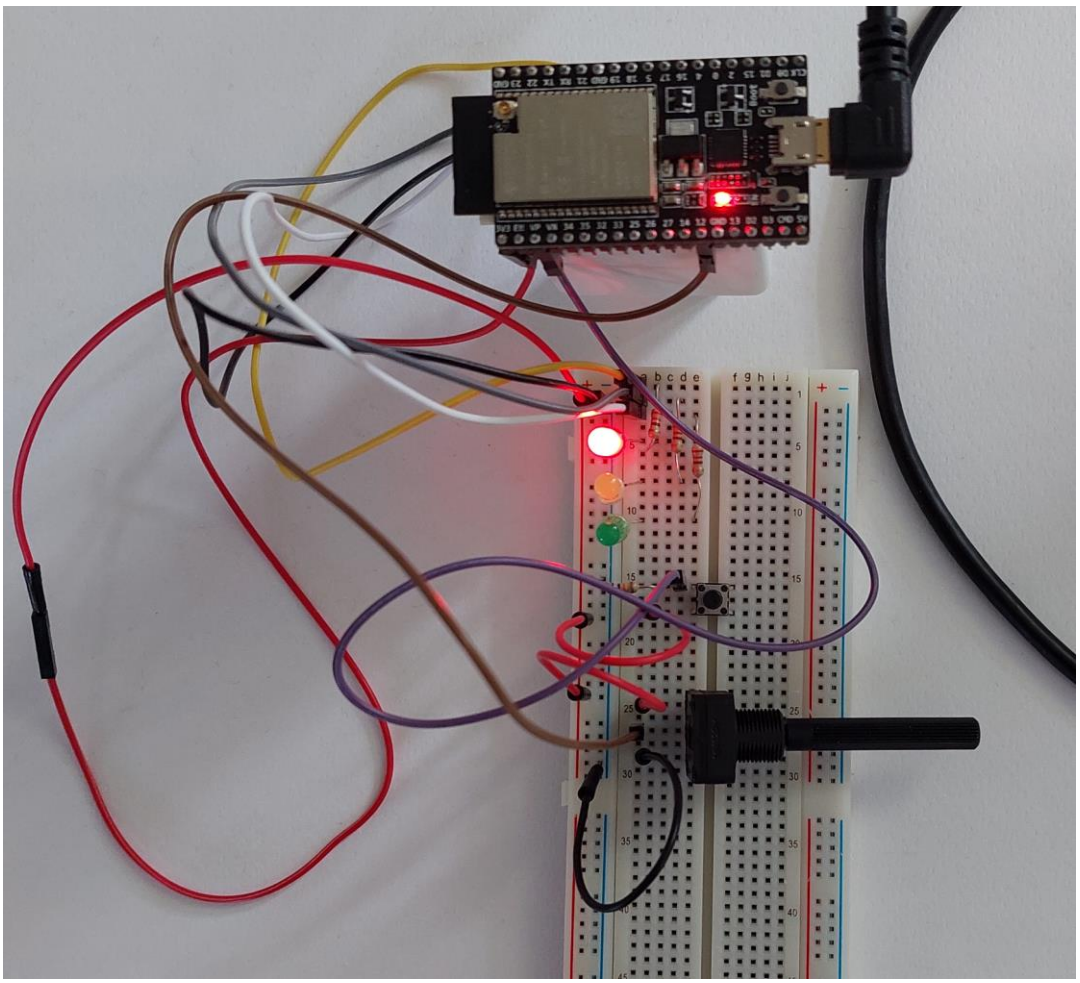
```
        gpio_set_level(GPIO_NUM_5, activate == 3);
        vTaskDelay(10 / portTICK_PERIOD_MS);
    }
}
```

U glavnom dijelu programa konfiguriraju se priključci mikrokontrolera kao digitalni izlaz (GPIO2, GPIO4 i GPIO5) i digitalni ulaz (GPIO36). Potom se stvaraju dvije zadaće za dvije funkcije: onOff i activateDigOutput. Na kraju se u beskonačnoj petlji aktivira neki od tri izlaza – ovisno o vrijednosti varijable activate.
U funkciji onOff postavlja se varijabla onOffValue (vrijednost 0 ili 1) pritiskom na tipkalo (odnosno dovođenjem 1 na digitalni ulaz GPIO36). Varijabla onOffValue postavlja se tako da poprima svoju suprotnu prethodnu vrijednost (operator negacije !). Također se vrijednost varijable activate postavlja na 0 ako je varijabla onOffValue jednaka 0 (odnosno ako se traži gašenje sustava). Time niti jedan uvjet za aktiviranje digitalnog izlaza nije zadovoljen čime se oni svi dovode u stanje 0.
U funkciji activateDigOutput očitava se digitalna vrijednost analognog signala koji se preko potenciometra dovodi na ADC GPIO12. Budući da vrijednost varijable activate može biti u intervalu [1, 3], provodi se postupak normalizacije očitane vrijednosti i njezino skaliranje na interval [1, 3]. Potom se tako dobivena vrijednost zaokružuje funkcijom round te se dodjeljuje varijabli activate.
Program se može kompajlirati, poslati na kontroler i pratiti u konzoli.




*Slika 2.7.3 Pokrenut program za naizmjenično aktiviranje tri digitalna izlaza mikrokontrolera ESP32 preko potenciometra*

Gotov sustav prikazuje naredna slika.



*Slika 2.7.4 Sustav za naizmjenično aktiviranje tri digitalna izlaza mikrokontrolera ESP32 preko potenciometra*

## 2.7.2. Pitanja i zadaci

1. Izraditi isti program, ali uz korištenje drugih priključaka mikrokontrolera.
2. Izraditi program kojim se kontrolira više digitalnih izlaza.
3. Izraditi program tako da se odabrani LED indikator periodički pali i gasi.
4. Izraditi program tako da se LED indikator odabire preko tipkala a da se perioda njezina paljenja i gašenja postavlja preko potenciometra.
5. Izraditi program tako da se LED indikator odabire preko potenciometra a da se perioda njezina paljenja i gašenja postavlja preko tipkala.

## 2.7.3. Literatura i izvori


1. Analog to Digital Converter (ADC) Oneshot Mode Driver, https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/adc_oneshot.html
2. Blink Example, https://github.com/espressif/esp-idf/tree/32472536715d674c160a2565795895e0273f8cde/examples/get-started/blink
3. ESP32 ADC – Read Analog Values with Arduino IDE, https://randomnerdtutorials.com/esp32-adc-analog-read-arduino-ide/
4. ESP32 ADC with ESP-IDF Measure Analog Inputs, https://esp32tutorials.com/esp32-adc-esp-idf/

5. Example: GPIO, https://github.com/espressif/esp-idf/tree/32472536715d674c160a2565795895e0273f8cde/examples/peripherals/gpio/generic_gpio
6. GPIO & RTC GPIO, https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/gpio.html
7. Hello World Example, https://github.com/espressif/esp-idf/tree/32472536715d674c160a2565795895e0273f8cde/examples/get-started/hello_world

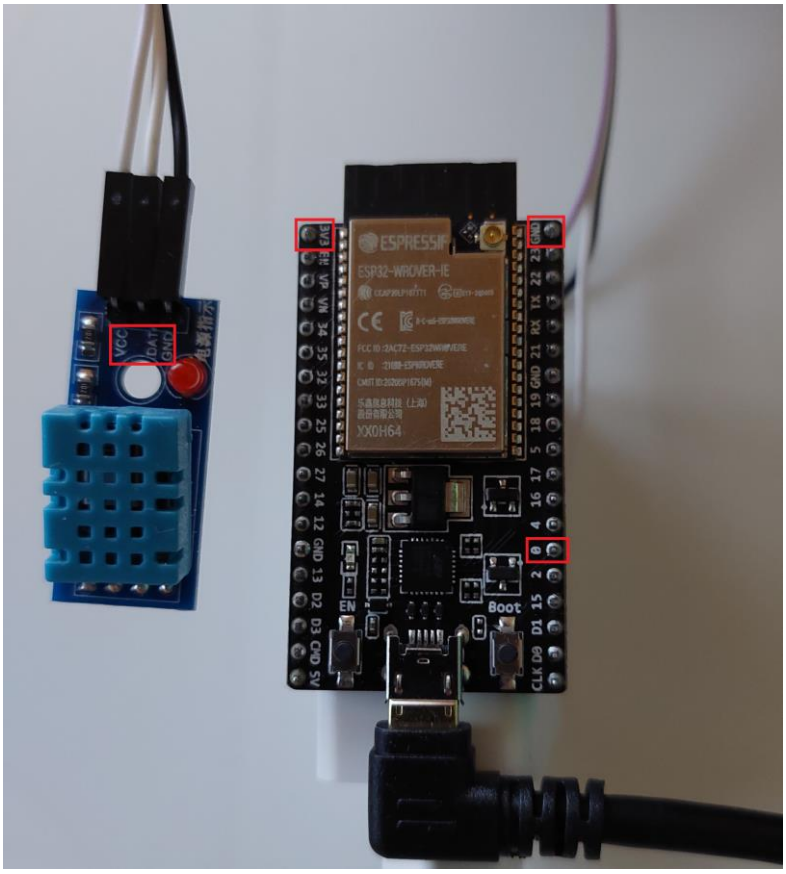
## 2.8. Izrada programa kojim se preko mikrokontrolera ESP32 očitavaju temperatura i vlažnost zraka

Potrebno je izraditi program koji periodički (npr. svakih pet sekundi) očitava temperaturu i vlažnost zraka sa senzora DHT 11 koji je povezan na mikrokontroler. Očitanje se treba ispisati u konzoli.

**Nakon usvajanja ove nastavne teme čitatelj će moći:**
1. Povezati senzor sa sklopovskom potporom
2. Programirati sklopovsku opremu za rad s priključenim senzorom
3. Izraditi programsku potporu za vlastiti sustav interneta stvari
4. Programirati mikroupravljač koristeći odabranu programsku potporu

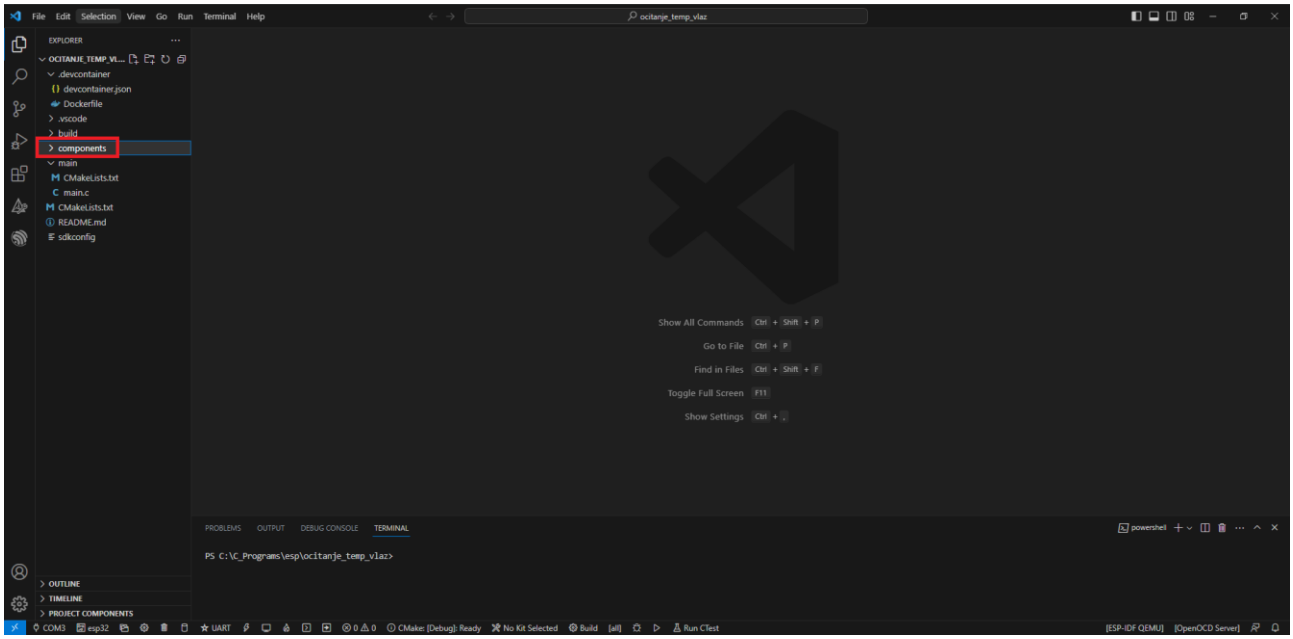
### 2.8.1. Rješenje radnog zadatka

Senzor DHT 11 preko priključaka Vcc i GND povezuje se na plus pol i GND napajanja. Priključak DATA povezuje se s priključkom mikrokontrolera koji ima funkciju input/output (npr. GPIO0).



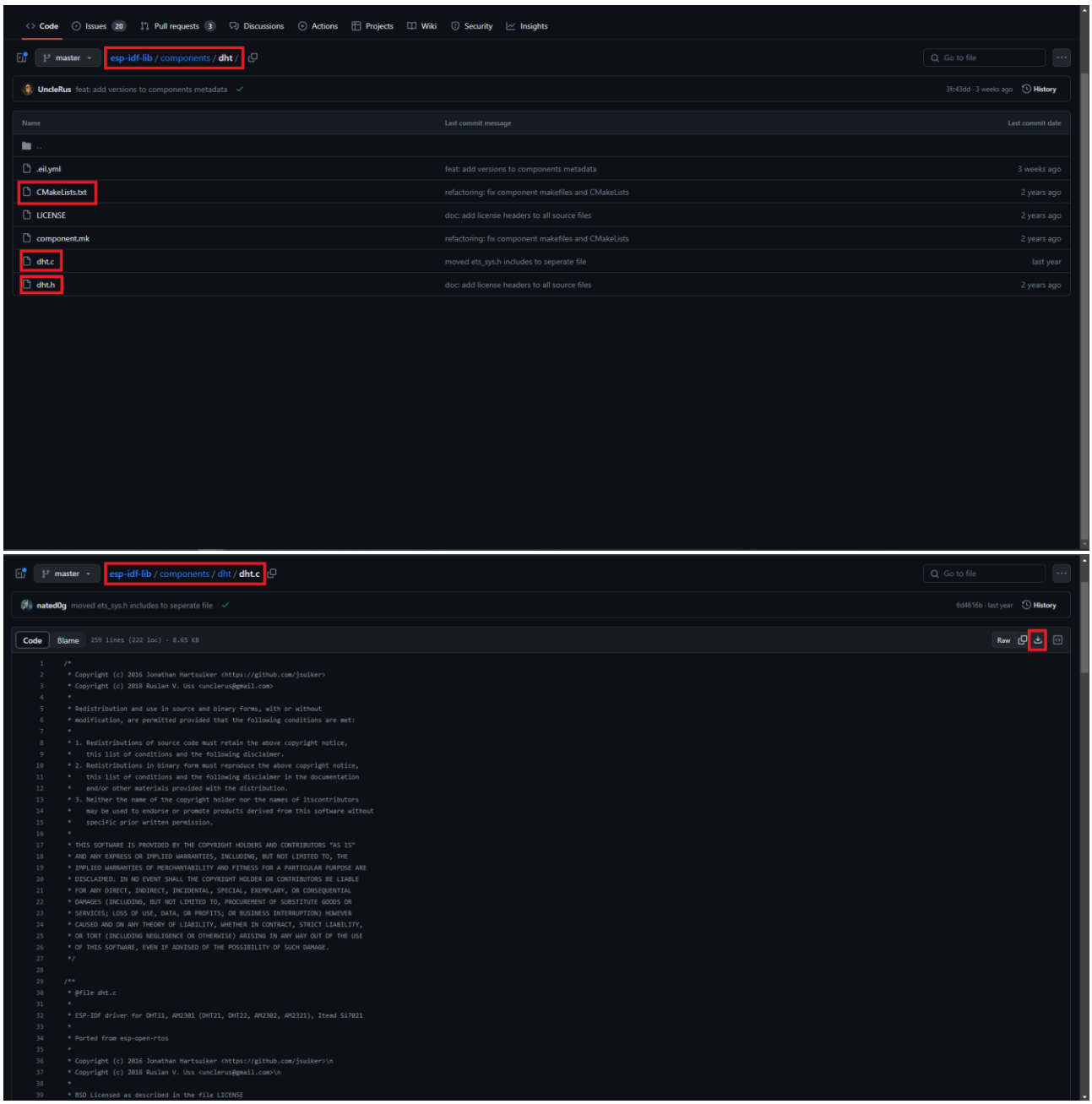

*Slika 2.8.1 Povezivanje senzora*

U softveru Visual Studio Code potrebno je izraditi novi projekt ocitanje_temp_vlaz. U projektnom stablu treba se izraditi nova mapa naziva components.



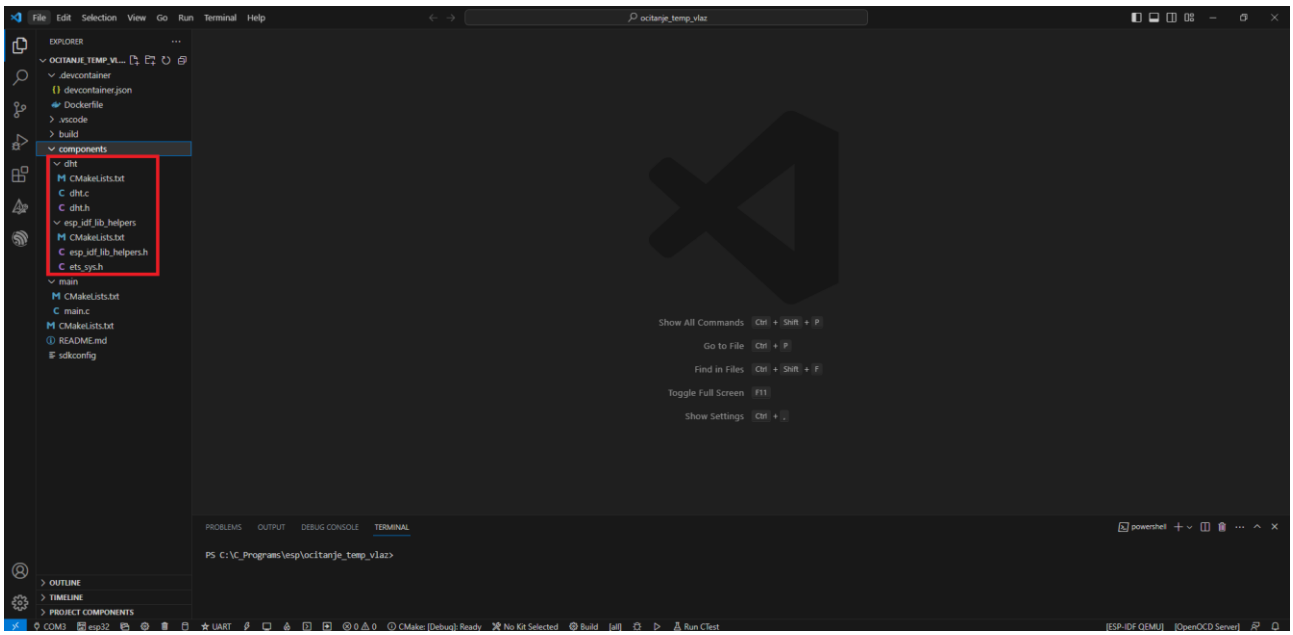

*Slika 2.8.2 Izrađeni projekt ocitanje_temp_vlaz i mapa components*

S poveznice https://github.com/UncleRus/esp-idf-lib/tree/master/components potrebno je preuzeti dvije komponente: dht i esp_idf_lib_helpers. Preuzimanje komponenti radi se tako da se otvori mapa u kojoj se nalaze datoteke komponente (.c, .h i CMakeLists.txt) te se klikom na pojedinu datoteku otvara mogućnost njezina preuzimanja.




*Slika 2.8.3 Preuzimanje komponenti i dht i esp_idf_lib_helpers*

U mapi components projekta treba izraditi podmape dht i esp_idf_lib_helpers i u njih ubaciti preuzete datoteke.



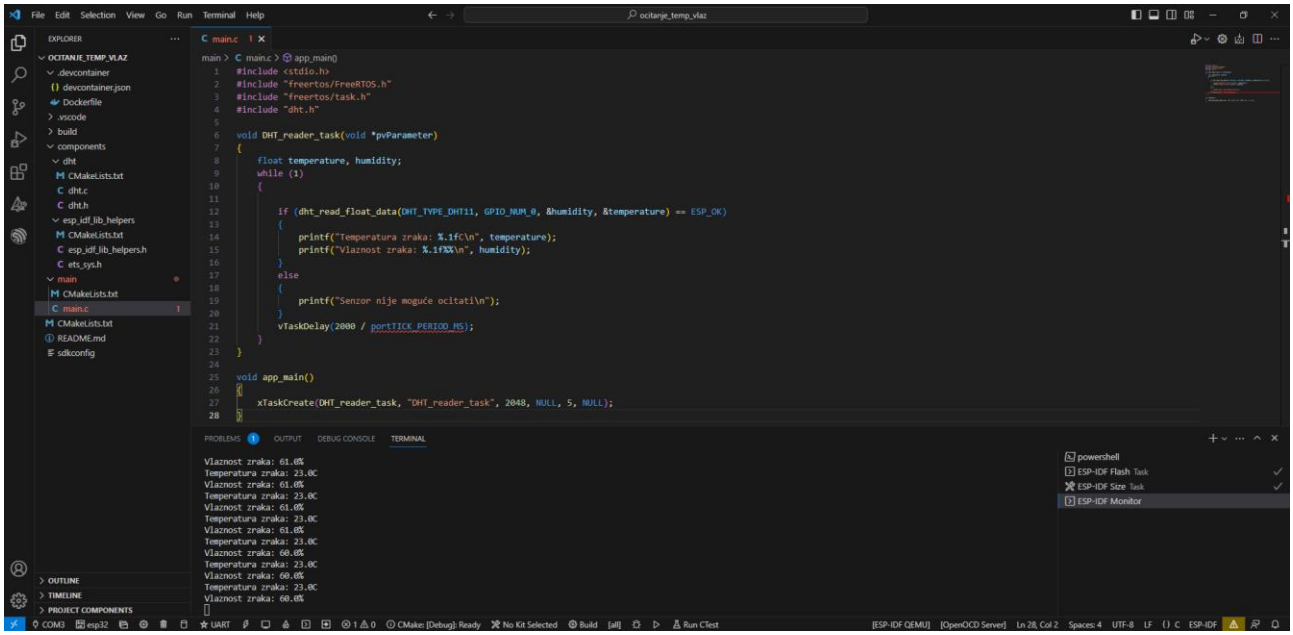

*Slika 2.8.4 Ubacivanje datoteka komponenti u odgovarajuće podmape*

U datoteku main.c može se ubaciti program za očitanje senzora DHT 11.

```
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "dht.h"

void DHT_reader_task(void *pvParameter)
{
    float temperature, humidity;
    while (1)
    {

        if (dht_read_float_data(DHT_TYPE_DHT11, GPIO_NUM_0, &humidity,
&temperature) == ESP_OK)
        {
            printf("Temperatura zraka: %.1fC\n", temperature);
            printf("Vlaznost zraka: %.1f%%\n", humidity);
        }
        else
        {
            printf("Senzor nije moguce ocitati\n");
        }
        vTaskDelay(2000 / portTICK_PERIOD_MS);
    }
}

void app_main()
{
    xTaskCreate(DHT_reader_task, "DHT_reader_task", 2048, NULL, 5, NULL);
}
```

U glavnom dijelu programa izradi se nova zadaća u kojoj se pokreće funkcija DHT_reader_task. U njoj se deklariraju dvije varijable (temperature i humidity) u koje će se spremiti očitanja sa senzora. Samo očitanje provodi se u beskonačnoj petlji svake dvije sekunde. Očitanje se provodi preko funkcije dht_read_float_data kojoj se kao argumenti šalju tip senzora (DHT_TYPE_DHT11), priključak na koji se šalju podaci (GPIO_NUM_0) te adrese varijabli u koje će se spremiti očitanje. Provjerava se je li očitanje bilo uspješno (ESP_OK) i ako je, u konzoli se ispisuje vrijednost temperature i vlažnosti. Ako očitanje nije bilo uspješno, ispisuje se prigodna poruka.

Nakon kompajliranja, slanja programa na mikrokontroler i pokretanja monitoringa mogu se vidjeti rezultati očitanja.




*Slika 2.8.5 Očitanje temperature i vlažnosti zraka te ispis u konzolu*

## 2.8.2. Pitanja i zadaci

1. Program izmijeniti tako da se preko tipkala odabere perioda očitanja senzora (npr. pet sekundi, osam sekundi i 10 sekundi).
2. Program izmijeniti tako da se preko potenciometra definira perioda očitanja senzora (npr. u rasponu 5 – 15 sekundi) a da se preko tipkala pali/gasi cjelokupni sustav.
3. U program dodati LED indikator koji se upali kada se dogodi očitanje senzora.

## 2.8.3. Literatura i izvori


1. DHT, https://github.com/UncleRus/esp-idf-lib/tree/master/components/dht
2. DHT22 with ESP32 ESP-IDF and Display Readings on OLED, https://esp32tutorials.com/dht22-esp32-esp-idf/
3. ESP-IDF Components library, https://github.com/UncleRus/esp-idf-lib
4. ESP-IDF lib helpers, https://github.com/UncleRus/esp-idf-lib/tree/master/components/esp_idf_lib_helpers
5. IDF Component Manager, https://docs.espressif.com/projects/esp-idf/en/v4.4/esp32s2/api-guides/tools/idf-component-manager.html#

## 2.9. Izrada programa kojim se pali LED koji odgovara razini termalne udobnosti izračunanoj u mikrokontroleru ESP32

Potrebno je izraditi program kojim se očitavaju temperatura i vlažnost zraka koristeći senzor DHT 11 i mikrokontroler. Nakon očitanja potrebno je izračunati razinu termalne udobnosti prema formuli

$$THI = (1.8 \cdot T + 32) - \big((0.55 - 0.0055 \cdot RH) \cdot (1.8 \cdot T - 26)\big)$$

gdje je:
*THI* – indeks temperature i vlažnosti
*T* – temperatura zraka u celzijusima
*RH* – vlažnost zraka u postocima

Izračunani THI označava razinu termalne udobnosti prema sljedećoj tablici:

| THI | Razina termalne udobnosti |
|---|---|
| < 72 | UGODNO |
| 72 – 79 | PODNOŠLJIVO |
| 80 – 89 | NEUGODNO |
| 90 – 99 | VRLO NEUGODNO |
| > 99 | OPASNO PO ZDRAVLJE |

S obzirom na razinu termalne udobnosti trebaju se upaliti odgovarajući LED indikatori i to:

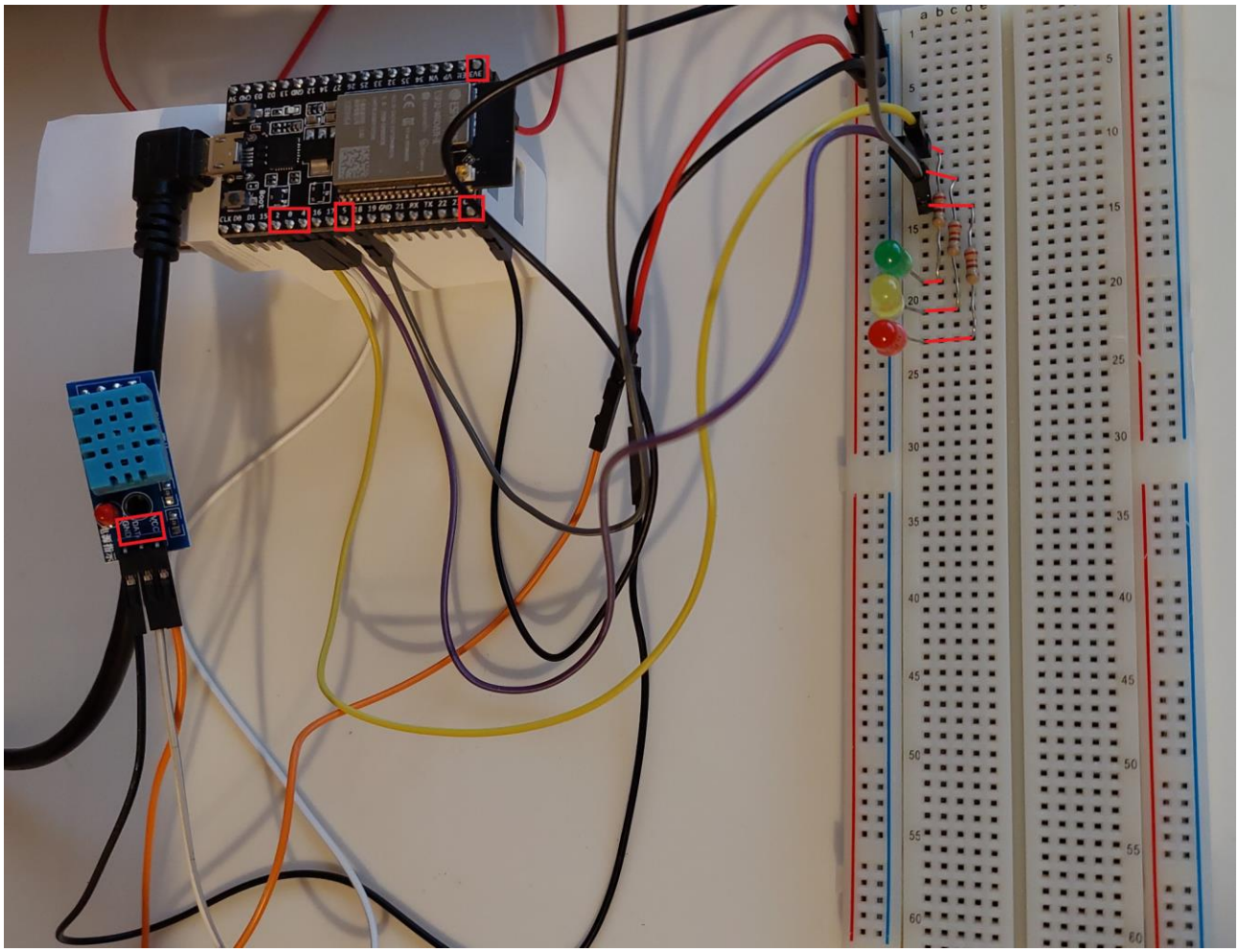
| Razina termalne udobnosti | LED |
|---|---|
| UGODNO | zelena |
| PODNOŠLJIVO | zelena i žuta |
| NEUGODNO | žuta |
| VRLO NEUGODNO | žuta i crvena |
| OPASNO PO ZDRAVLJE | crvena |

Razina termalne udobnosti računa se svih pet sekundi i ispisuje se u konzoli.

**Nakon usvajanja ove nastavne teme čitatelj će moći:**
1. Programirati mikroupravljač koristeći odabranu programsku potporu
2. Povezati senzor sa sklopovskom potporom
3. Programirati sklopovsku opremu za rad s priključenim senzorom
4. Povezati aktuator sa sklopovskom potporom
5. Programirati sklopovsku opremu za rad s priključenim aktuatorom
6. Izraditi programsku potporu za vlastiti sustav interneta stvari

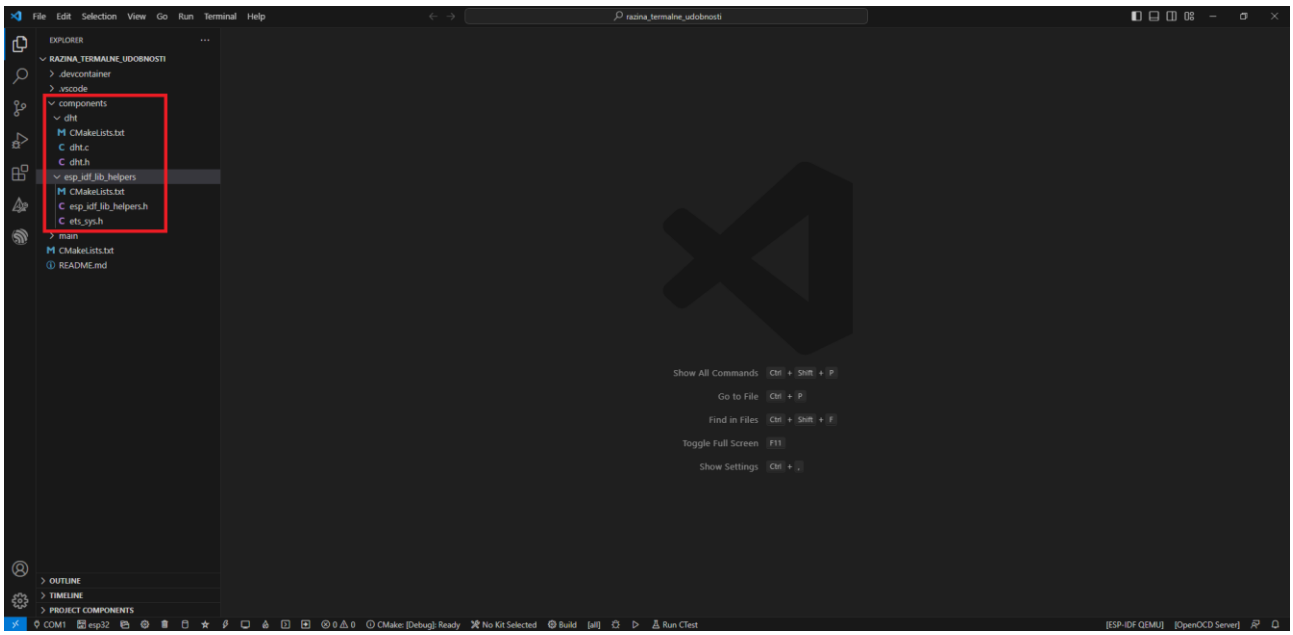
### 2.9.1. Rješenje radnog zadatka

Potrebno je povezati senzor DHT 11 s mikrokontrolerom tako da se podaci čitaju s ulaza GPIO0. Tri LED indikatora potrebno je povezati na izlaze GPIO2 (zelena), GPIO4 (žuta) i GPIO5 (crvena) – paziti na polaritet LED indikatora i postaviti odgovarajuće otpornike.



*Slika 2.9.1 Povezivanje senzora DHT 11, tri LED indikatora i mikrokontrolera*

U softveru Visual Studio Code potrebno je izraditi projekt razina_termalne_udobnosti. U njega je potrebno dodati dvije komponente, dht i esp_idf_lib_helpers.



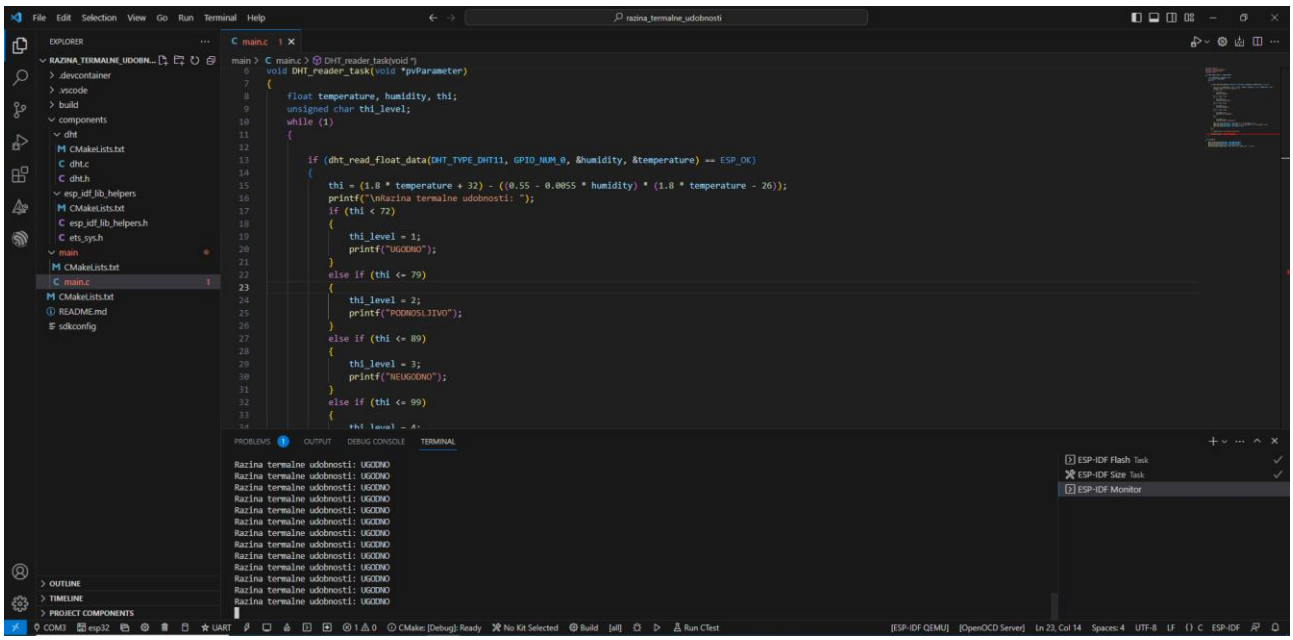

*Slika 2.9.2 Izrađen projekt razina_termalne_udobnosti*

U datoteci main.c može se upisati program kojim se računa razina termalne udobnosti i na osnovi nje pali se odgovarajući LED.

```
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "dht.h"

void DHT_reader_task(void *pvParameter)
{
    float temperature, humidity, thi;
    unsigned char thi_level;
    while (1)
    {

        if (dht_read_float_data(DHT_TYPE_DHT11, GPIO_NUM_0, &humidity,
&temperature) == ESP_OK)
        {
            thi = (1.8 * temperature + 32) - ((0.55 - 0.0055 * humidity) * (1.8 *
temperature - 26));
            printf("\nRazina termalne udobnosti: ");
            if (thi < 72)
            {
                thi_level = 1;
                printf("UGODNO");
            }
            else if (thi <= 79)
            {
                thi_level = 2;
                printf("PODNOSLJIVO");
            }
            else if (thi <= 89)
```

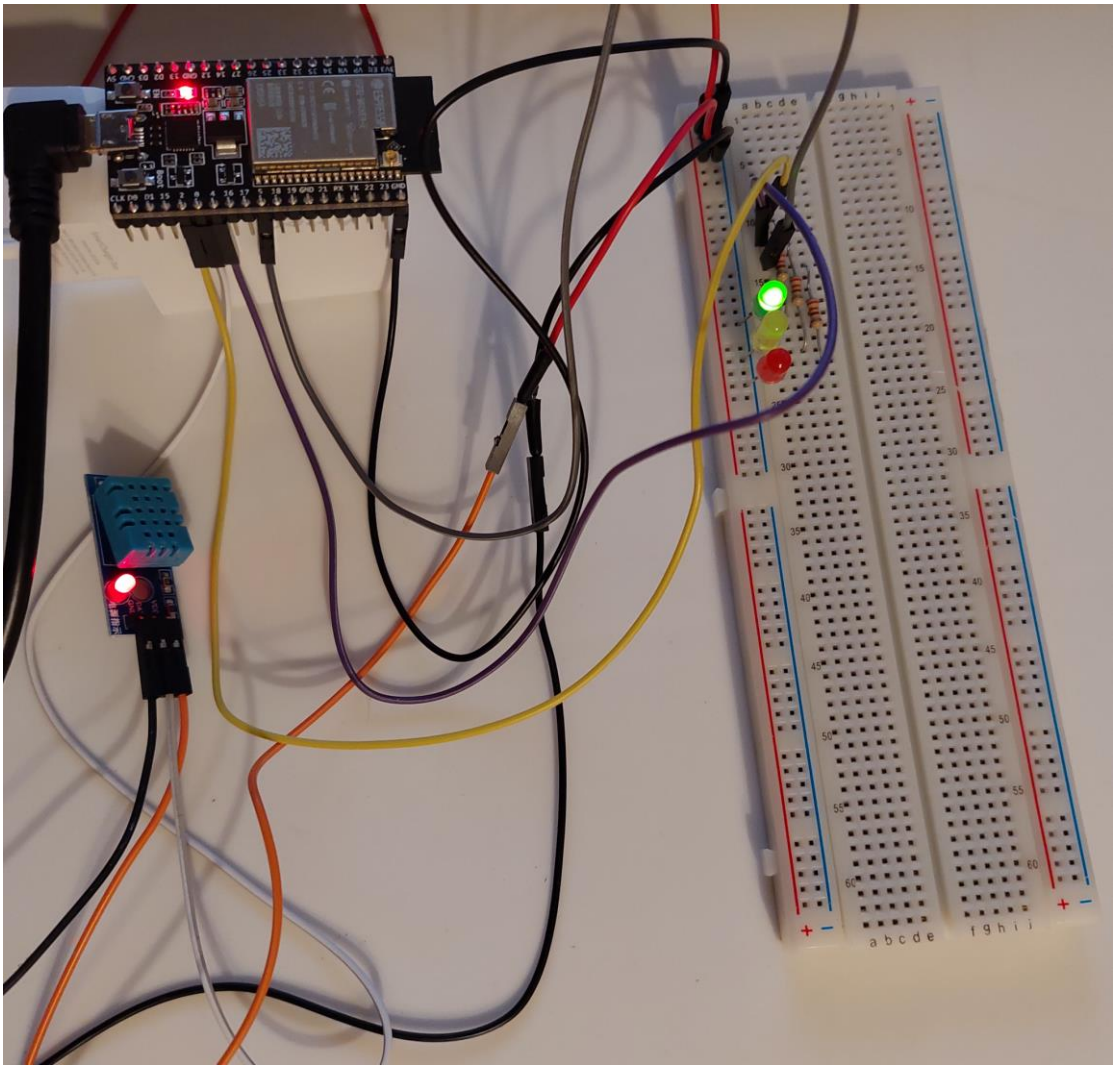
```
            {
                thi_level = 3;
                printf("NEUGODNO");
            }
            else if (thi <= 99)
            {
                thi_level = 4;
                printf("VRLO NEUGODNO");
            }
            else
            {
                thi_level = 5;
                printf("OPASNO PO ZDRAVLJE");
            }
            gpio_set_level(GPIO_NUM_2, (thi_level == 1 || thi_level == 2));
            gpio_set_level(GPIO_NUM_4, (thi_level == 2 || thi_level == 3 ||
thi_level == 4));
            gpio_set_level(GPIO_NUM_5, (thi_level == 5));
        }
        else
        {
            printf("Senzor nije moguce ocitati\n");
        }
        vTaskDelay(5000 / portTICK_PERIOD_MS);
    }
}

void app_main()
{
    gpio_set_direction(GPIO_NUM_2, GPIO_MODE_OUTPUT);
    gpio_set_direction(GPIO_NUM_4, GPIO_MODE_OUTPUT);
    gpio_set_direction(GPIO_NUM_5, GPIO_MODE_OUTPUT);
    xTaskCreate(DHT_reader_task, "DHT_reader_task", 2048, NULL, 5, NULL);
}
```

U glavnom dijelu programa postavljaju se priključci GPIO2, GPIO4 i GPIO5 kao digitalni izlazi (preko njih aktivirat će se odgovarajući LED indikator). Potom se izrađuje nova zadaća u kojoj se pokreće funkcija DHT_reader_task.
U toj se funkciji deklariraju četiri varijable: temperature (za spremanje očitane temperature), humidity (za spremanje očitane vlažnosti zraka), thi (za spremanje izračunane vrijednosti razine termalne udobnosti) i thi_level (za identifikaciju jedne od pet mogućih vrijednosti razine termalne udobnosti koja se kasnije koristi kod aktiviranja digitalnog izlaza, odnosno LED indikatora). U petlji se svakih pet sekundi očitava temperatura i vlažnost zraka te se izračunava thi. Potom se na temelju vrijednosti thi definira vrijednost varijable thi_level koja se kasnije u logičkom izrazu koristi za izračun aktivacije digitalnog izlaza (npr. ako je thi_level = 1, tada će se na GPIO2 poslati 1 a na ostale izlaz 0).
Naredna slika prikazuje pokrenut program.




*Slika 2.9.3 Pokrenut program aktiviranja LED temeljem razine termalne udobnosti*

Gotov sustav u radu prikazuje naredna slika.



*Slika 2.9.4 Sustav za aktiviranja LED temeljem razine termalne udobnosti u radu*

### 2.9.2. Pitanja i zadaci

1. Program izmijeniti tako da se preko tipkala odabere perioda očitanja senzora (npr. pet sekundi, osam sekundi i 10 sekundi).
2. Program izmijeniti tako da se preko potenciometra definira perioda očitanja senzora (npr. u rasponu 5 – 15 sekundi) a da se preko tipkala pali/gasi cjelokupni sustav.
3. U program dodati LED indikator koji se upali kada se dogodi očitanje senzora.

### 2.9.3. Literatura i izvori


1. Blink Example, https://github.com/espressif/esp-idf/tree/32472536715d674c160a2565795895e0273f8cde/examples/get-started/blink
2. DHT, https://github.com/UncleRus/esp-idf-lib/tree/master/components/dht
3. DHT22 with ESP32 ESP-IDF and Display Readings on OLED, https://esp32tutorials.com/dht22-esp32-esp-idf/
4. ESP-IDF Components library, https://github.com/UncleRus/esp-idf-lib
5. ESP-IDF lib helpers, https://github.com/UncleRus/esp-idf-lib/tree/master/components/esp_idf_lib_helpers
6. ESP32 ADC – Read Analog Values with Arduino IDE, https://randomnerdtutorials.com/esp32-adc-analog-read-arduino-ide/
7. ESP32 ADC with ESP-IDF Measure Analog Inputs, https://esp32tutorials.com/esp32-adc-esp-idf/
8. Example: GPIO, https://github.com/espressif/esp-idf/tree/32472536715d674c160a2565795895e0273f8cde/examples/peripherals/gpio/generic_gpio
9. GPIO & RTC GPIO, https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/gpio.html
10. Hello World Example, https://github.com/espressif/esp-idf/tree/32472536715d674c160a2565795895e0273f8cde/examples/get-started/hello_world
11. IDF Component Manager, https://docs.espressif.com/projects/esp-idf/en/v4.4/esp32s2/api-guides/tools/idf-component-manager.html#


## 2.10. Izrada programa kojim se mikrokontroler ESP32 povezuje na bežičnu računalnu mrežu

Mikrokontroler je potrebno povezati na pristupnu točku (*access point*) te mu IP adresu treba dodijeliti poslužitelj DHCP. S razvojnog računala koje je povezano na istu pristupnu točku potrebno je provjeriti povezanost mikrokontrolera preko naredbe ping. U konzoli je potrebno ispisati konfiguracijske podatke mrežne kartice mikrokontrolera koje je dobila od poslužitelja DHCP (IP adresa, adresa zadanog pristupnika – *default gateway* i mrežna maska).

**Nakon usvajanja ove nastavne teme čitatelj će moći:**

1. Programirati mikroupravljač koristeći odabranu programsku potporu
2. Objasniti opći model komunikacijskog sustava i funkcioniranje računalne mreže prema slojnom modelu
3. Objasniti proces enkapsulacije i dekapsulacije po slojevima
4. Objasniti ulogu mrežnog sloja
5. Izračunati i primijeniti mrežno adresiranje
6. Primijeniti IP adresiranje
7. Klasificirati osnovne komunikacijske kriterije

8. Objasniti osnovne karakteristike komunikacijskih mreža WAN
9. Objasniti osnovne karakteristike komunikacijskih mreža PAN i WLAN
10. Objasniti osnovne karakteristike LPWAN komunikacijskih mreža LPWAN
11. Izraditi sklopovsko i programsko rješenje koje koristi komunikacijske mreže WAN, PAN i WLAN ili LPWAN

## 2.10.1. Osnovni koncepti

Računalna mreža sastoji se od skupa međusobno povezanih računalnih uređaja koji komuniciraju razmjenjujući poruke preko nekog medija za prijenos podataka.
Računalni uređaji na mrežu se povezuju posebnim hardverskim sklopom koji se zove mrežna kartica (engl. *network interface card*), a koja omogućuje njihovo žičano ili bežično umrežavanje. Primjer računalnih uređaja koji se mogu povezati u mrežu jesu: prijenosno i stolno računalo, mobilni telefon, TV, kamera, igraća konzola, meteorološka stanica, razni senzori, klima-uređaj, pećnica, mikrokontroler itd.).
Kako bi se uspostavila računalna mreža među raznim računalnim uređajima, koriste se i mrežni uređaji kao što su: mrežni preklopnik/prospojnik (engl. *switch*), pristupna točka (engl. *access point*) i mrežni usmjerivač/usmjernik (engl. *router*).
Mrežni preklopnik/prospojnik mrežni je uređaj s više priključaka koji žičano povezuje računalne uređaje u mrežu. Mrežna se kartica računalnog uređaja ethernet kabelom povezuje na neki od priključaka na mrežnom preklopniku. Količina dostupnih priključaka može se povećati ako se mrežni preklopnici međusobno žičano povežu.

*Slika 2.10.1 Mrežni preklopnik s uključenim ethernet kabelima*

Pristupna točka mrežni je uređaj koji omogućuje bežično umrežavanje računalnih uređaja te njihovo umrežavanje s ostatkom žičane mreže. Mrežna kartica računalnog uređaja bežično se povezuje s pristupnom točkom koristeći odgovarajuću konfiguraciju (naziv pristupne točke i njezine sigurnosne postavke).


*Slika 2.10.2 Pristupna točka*

Mrežni usmjernik mrežni je uređaj koji povezuje računalne uređaje koji se nalaze u različitim mrežama (različitim prefiksima) – npr. računalni uređaji koji se nalaze u lokalnoj mreži neke ustanove povezani su s računalnim uređajima globalne mreže internet.
Međusobno umreženi računalni uređaji komuniciraju tako da razmjenjuju poruke. Strukturu i značenje poruke definira protokol. Protokol je „jezik" kojim računala međusobno komuniciraju. U komunikaciji među računalnim uređajima postoji niz problema koji se rješavaju protokolima (npr. pouzdana komunikacija, identifikacija računalnih uređaja, prijenos podataka preko medija i sl.). Vrste problema i njihovih rješenja zorno prikazuje tzv. OSI mrežni model koji opisuje funkcije u komunikacijskim i računalnim mrežama razdijeljene u obliku sedam slojeva: fizički sloj (engl. *physical layer*), podatkovni sloj (engl. *data link layer*), mrežni sloj (engl. *network layer*), prijenosni sloj (engl. *transport layer*), sesijski sloj (konferencijski, sjednički ili razgovorni sloj – engl. *session layer*), prezentacijski sloj (engl. *presentation layer*) i aplikacijski sloj (engl. *application layer*).
U narednoj tablici sažeto su prikazane funkcije pojedinog sloja OSI modela.

| **OSI model** | **Funkcija** |
|---|---|
| Aplikacijski sloj | Pruža mrežne usluge korisničkim aplikacijama (npr. Google Chrome, Skype, Viber, Outlook) preko API-a. Primjer protokola: HTTP, FTP, SMTP, DNS, DHCP. |
| Prezentacijski sloj | Prijevod podataka između formata podataka za aplikaciju i formata podataka za mrežu (npr. kodiranje znakova, kompresija podataka i enkripcija/dešifriranje). |
| Sesijski sloj | Uspostava, održavanje i prekidanje sesije. Pamti stanje sesije čime omogućuje sinkronizaciju nove sesije u odnosu na stanje prethodne. |
| Prijenosni sloj | Rastavljanje informacija na pakete i njihovo slaganje u ispravan redoslijed. Osiguravanje pouzdanog ili nepouzdanog prijenosa podataka. Upravljanje protokom podataka. Pouzdano otvaranje i zatvaranje veze. Primjer protokola: TCP, UDP. |
| Mrežni sloj | Osigurava prijenos paketa unutar iste mreže (mrežnog prefiksa) ili među različitim mrežama (mrežnih prefiksa). Primjer protokola: IP. |
| Podatkovni sloj | Formiranje okvira iz paketa koji se mogu slati kroz lokalnu mrežu. Primjer protokola: ethernet (MAC), ARP. |
| Fizički sloj | Prijenos i prijem bitova preko fizičkog medija (žičanog ili bežičnog). |

Kada računalni uređaj povezan u računalnu mrežu šalje podatke (npr. *e-mail*), tada ti podaci putuju „niz" slojeve OSI mrežnog modela sve do zadnjega fizičkog sloja. Kada računalni uređaj prima

podatke, tada oni putuju obrnutim putem, od najnižega fizičkog do najvišega aplikacijskog sloja. Proces slanja podataka niz slojeve OSI modela zove se enkapsulacija, dok se obratni proces naziva deenkapsulacija.
Osim OSI modela postoji i TCP/IP model koji također opisuje funkcije u komunikacijskim i računalnim mrežama, ali ih predstavlja u obliku četiri sloja: sloj pristupa mreži (engl. *network access layer*), mrežni sloj (engl. *network layer*), prijenosni sloj (engl. *transport layer*) i aplikacijski sloj (engl. *application layer*).
Odnos između modela OSI i TCP/IP prikazan je u narednoj tablici.

| **OSI model** | **TCP/IP model** |
|---|---|
| Aplikacijski sloj | Aplikacijski sloj |
| Prezentacijski sloj | |
| Sesijski sloj | |
| Prijenosni sloj | Prijenosni sloj |
| Mrežni sloj | Mrežni sloj |
| Podatkovni sloj | Sloj pristupa mreži |
| Fizički sloj | |

Kako bi komunikacija bila moguća, računalni uređaji u računalnoj mreži trebaju se moći jedinstveno identificirati. U biti postoje tri razine identifikacije računalnog uređaja.
Prva razina identifikacije jest fizička identifikacija – svaka mrežna kartica računalnog uređaja posjeduje fizičku (engl. *media access control* – MAC) adresu. Dodjeljuje ju proizvođač mrežne kartice. To je temeljna razina identifikacije računalnih uređaja i bez nje nije moguće uspostaviti komunikaciju u računalnoj mreži. Mrežne aplikacije pokrenute na računalnim uređajima uglavnom ne koriste taj način identifikacije računala.
Druga razina identifikacije jest logička identifikacija koju implementira mrežni sloj OSI modela preko IP protokola (IP adresa). Mrežne aplikacije koriste taj način identifikacije računalnih uređaja (npr. za povezivanje na bazu podataka ili pozivanje *web* aplikacijskoga programskog sučelja). IP adrese dodjeljuju se mrežnim karticama. Postoje dvije vrste IP adresa: v4 i v6. IPv4 se sastoji od 32 bita, dok se IPv6 sastoji od 128 bitova. U lokalnoj mreži uglavnom se koriste IPv4.
IPv4 adresa koja je 32-bitna zapisuje se u ljudima čitljivijem formatu tako da se pojedini okteti (grupe od 8 bita – bajtovi) zapišu dekadski i odijele točkom. Nula se pojavljuje kada su svi bitovi u oktetu 0, dok je 255 najveća vrijednost i označava osam jedinica.
Naredna tablica prikazuje primjere IPv4 adresa.

| **32-bitna IP ADRESA** | **DECIMALNA NOTACIJA** |
|---|---|
| 10000001 00110100 00000100 00000000 | 129.52.6.0 |
| 11000000 00000101 00110000 00000011 | 192.5.48.3 |
| 00001010 00000010 00000000 00100101 | 10.2.0.37 |
| 10000000 00001010 00000010 00000011 | 128.10.2.3 |
| 10000000 10000000 11111111 00000000 | 128.128.255.0 |

IPv4 adresa koja je 32-bitna sastoji se od dva dijela koji se nazivaju prefiks i sufiks. Prefiks ili adresa mreže identificira fizičku mrežu u kojoj se nalazi računalni uređaj, dok sufiks označava pojedinačni računalni uređaj u mreži. Dvije međusobno povezane računalne mreže ne mogu imati istu mrežnu adresu, a dva računalna uređaja u istoj računalnoj mreži ne mogu imati isti sufiks. Računalni uređaji u računalnoj mreži trebaju imati isti prefiks, ali različit sufiks.
Proizvoljno dijeljenje adrese IPv4 na prefiks i sufiks moguć je s pomoću adresne maske. Adresna maska jest 32-bitni broj koji se također zapisuje u dekadskoj notaciji.

Naredna tablica prikazuje primjere mrežnih maski.

| 32-bitna MREŽNA MASKA | DECIMALNA NOTACIJA |
|---|---|
| 11111111.00000000.00000000.00000000 | 255.0.0.0 |
| 11111111.11111111.00000000.00000000 | 255.255.0.0 |
| 11111111.11111111.11111111.00000000 | 255.255.255.0 |

Adresna maska primjenjuje se na IP adresu te se kao rezultat dobiva adresa mreže. Slijede dva primjera primjene različitih mrežnih maski na istu IP adresu:
IP adresa 168.15.1.100 i mrežna maska 255.255.0.0:

- binarni zapis IP adrese: 10101000.00001111.00000001.01100100
- mrežna maska za klasu B (binarno): 11111111.11111111.00000000.00000000
- adresa mreže (binarno): 10101000.00001111.00000000.00000000
- mrežna maska za klasu B (decimalno): 255.255.0.0
- adresa mreže (decimalno): 168.15.0.0
- adresa čvora (decimalno): 1.100

IP adresa 168.15.1.100 i mrežna maska 255.255.255.0:

- binarni zapis IP adrese: 10101000.00001111.00000001.01100100
- mrežna maska za klasu C (binarno): 11111111.11111111.11111111.00000000
- adresa mreže (binarno): 10101000.00001111.00000001.00000000
- mrežna maska za klasu C (decimalno): 255.255.255.0
- adresa mreže (decimalno): 168.15.1.0
- adresa čvora (decimalno): 100

Kako bi se skratilo zapisivanje IP adrese i mrežne maske, koristi se tzv. CIDR notacija. U prethodnim primjerima ta bi notacija glasila 168.15.1.100/16, odnosno 168.15.1.100/24. Broj iza znaka / govori koliko prvih bitova iz IP adrese predstavlja prefiks, odnosno adresu mreže. Vrijednost 255.0.0.0 u CIDR notaciji je /8, 255.255.0.0 je /16 i 255.255.255.0 je /24.
Dva su načina dodjeljivanja IP adresa mrežnim karticama računalnih uređaja – statičko i dinamičko. Statičko dodjeljivanje IP adresa predstavlja postavljanje IP adrese računalnog uređaja od strane mrežnog administratora. Tako dodijeljena IP adresa ne mijenja se ponovnim pokretanjem računalnog uređaja, a najčešće se koriste kod stacionarnih računalnih uređaja (npr. stolnih računala).
Dinamičko dodjeljivanje IP adresa predstavlja postavljanje IP adrese računalnog uređaja od strane poslužitelja DHCP (engl. *dynamic host configuration protocol*). Tako dodijeljena IP adresa mijenja se ponovnim pokretanjem računalnog uređaja – računalni uređaj zahtijeva dodjelu IP adrese.
Najčešće se koriste kod mobilnih računalnih uređaja (npr. prijenosna računala, pametni telefoni, tableti i sl.).
Treći je način identifikacije računalnih uređaja preko simboličkog naziva čime se ljudima olakšava njihova identifikacija. Određenoj IP adresi dodjeljuje se simbolički naziv. To zahtijeva postojanje mrežne usluge koja će ta imena pretvarati u IP adrese (poslužitelj DNS – engl. *domain name system*).
Mrežne aplikacije za komunikaciju mogu koristiti simboličke nazive računalnih uređaja i/ili IP adrese. Na primjer, u internetskom pregledniku (klijentska mrežna aplikacija) može se unijeti simbolički naziv računala (npr. www.facebook.com), ali i njegova IP adresa (npr. 31.13.84.36). Na oba načina zahtjev za internetskim stranicama doći će do poslužitelja WEB s IP adresom 31.13.84.36.
Ako se koriste simbolički nazivi, tada mora postojati mrežna usluga koja će ih pretvoriti u IP adresu (DNS poslužitelj).

Konačni prijenos paketa podataka moguć je tek s fizičkim adresama (MAC). Stoga je potrebno odredišnu IP adresu (npr. 31.13.84.36) pretvoriti u odredišnu fizičku adresu. To radi protokol (engl. *address resolution protocol*). Računalni uređaji mogu preko protokola ARP pretvoriti odredišnu IP adresu u fizičku samo onih računalnih uređaja koji su s njima u istoj mreži (mreži istog prefiksa).
Postoji više načina klasifikacije računalnih mreža. Jedna od njih je s obzirom na prostornu udaljenost računalnih uređaja koji su međusobno umreženi. Prema njoj računalne mreže možemo podijeliti na sljedeće:

- lokalne mreže (engl. *local-area networks* – LANs) – računalni uređaji koji su prostorno blizu jedan drugome (u istoj zgradi)
- globalne mreže (engl. *wide-area networks* – WANs) – računalni uređaji prostorno su udaljeni i povezani su telefonskim linijama ili radiovalovima
- mrežu kampusa (engl. *campus-area networks* – CANs) – računalni uređaji nalaze se na ograničenom prostoru kao što je kampus ili vojna baza
- gradsku mrežu (engl. *metropolitan-area networks* – MANs) – računalna mreža dizajnirana za područje grada
- kućnu mrežu (engl. *home-area networks* – HANs) – računalna mreža u kući korisnika koja povezuje osobne računalne uređaje
- osobnu mrežu (engl. *personal area networks* – PAN) – računalna mreža koja povezuje računalne uređaje u radnom prostoru osobe
- internet – globalna računalna mreža. Mreža svih mreža. Mreža umreženih mreža.

Ako se kao kriterij za klasifikaciju računalnih mreža uzima korišteni medij za prijenos podataka, tada se mreže mogu podijeliti na žičane i bežične mreže. Primjenom bežičnog medija za uspostavu lokalne mreže dobivamo bežičnu lokalu mrežu (engl. *wireless LAN* – WLAN).
Posebna vrsta mreže jest niskoenergetska globalna mreža (engl. *low-power wide-area networks* – LPWAN) koja omogućuje komunikaciju na velike udaljenosti, ali uz nisku energetsku potrošnju. Takve se mreže koriste u sustavima IoT.
Mikrokontroler ESP32 posjeduje mrežnu karticu koja omogućuje njegovo povezivanje u bežičnu računalnu mrežu. Procedura povezivanja jest sljedeća:

1. inicijalizirati particiju NVS u *flash* memoriji mikrokontrolera – naredba nvs_flash_init()
2. stvoriti grupu događaja u koju će se dodati oni koji će se osluškivati – naredba xEventGroupCreate()
3. inicijalizirati TCP/IP stog – naredba esp_netif_init()
4. kreirati osluškivanje događaja – naredba esp_event_loop_create_default()
5. konfigurirati bežičnu mrežnu karticu kao klijenta (stanicu) – naredba esp_netif_create_default_wifi_sta()
6. inicijalizirati bežičnu mrežnu karticu početnom konfiguracijom – naredba esp_wifi_init
7. u grupu događaja dodati događaje koji se odnose na bežičnu mrežu i na IP adresiranje – naredba esp_event_handler_instance_register
8. bežičnoj mrežnoj kartici postaviti podatke za povezivanje s pristupnom točkom – naredba esp_wifi_set_config
9. pokrenuti bežičnu mrežnu karticu – naredba esp_wifi_start
10. kada je bežična mrežna kartica spremna, pokrenuti povezivanje na pristupnu točku – naredba esp_wifi_connect().

## 2.10.2. Rješenje radnog zadatka

Mikrokontroler je potrebno povezati s razvojnim računalom koje je povezano na pristupnu točku.
U softveru Visual Studio Code potrebno je izraditi novi projekt pod nazivom wifi_povezivanje. U datoteku main.c potrebno je upisati program.

```c
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp_wifi.h"
#include "nvs_flash.h"

#define WIFI_SSID "Naziv pristupne točke"
#define WIFI_PASS "Lozinka za povezivanje na pristupnu točku"

static void event_handler(void *arg, esp_event_base_t event_base,
                          int32_t event_id, void *event_data)
{
    if (event_base == WIFI_EVENT && event_id == WIFI_EVENT_STA_START)
    {
        esp_wifi_connect();
    }
    else if (event_base == IP_EVENT && event_id == IP_EVENT_STA_GOT_IP)
    {
        ip_event_got_ip_t *event = (ip_event_got_ip_t *)event_data;
        printf("IP adresa: " IPSTR "\n", IP2STR(&event->ip_info.ip));
        printf("Default Gateway: " IPSTR "\n", IP2STR(&event->ip_info.gw));
        printf("Mrezna maska: " IPSTR "\n", IP2STR(&event->ip_info.netmask));
    }
}

void wifi_init_sta(void)
{
    s_wifi_event_group = xEventGroupCreate();
    esp_netif_init();

    esp_event_loop_create_default();
    esp_netif_create_default_wifi_sta();

    wifi_init_config_t cfg = WIFI_INIT_CONFIG_DEFAULT();
    esp_wifi_init(&cfg);

    esp_event_handler_instance_t instance_any_id;
    esp_event_handler_instance_t instance_got_ip;
    esp_event_handler_instance_register(WIFI_EVENT,
                                        ESP_EVENT_ANY_ID,
                                        &event_handler,
                                        NULL,
                                        &instance_any_id);
    esp_event_handler_instance_register(IP_EVENT,
                                        IP_EVENT_STA_GOT_IP,
                                        &event_handler,
                                        NULL,
                                        &instance_got_ip);

    wifi_config_t wifi_config = {
```

```
        .sta = {
            .ssid = WIFI_SSID,
            .password = WIFI_PASS,
        },
    };
    esp_wifi_set_config(WIFI_IF_STA, &wifi_config);
    esp_wifi_start();
}

void app_main(void)
{
    // Initialize NVS
    nvs_flash_init();

    wifi_init_sta();
}
```

U glavnom dijelu programa inicijalizira se *flash* memorija unutar mikrokontrolera. Potom se poziva funkcija wifi_init_sta() u kojoj se inicijalizira mrežna kartica mikrokontrolera (esp_netif_init()), stvara se petlja za praćenje događaja (esp_event_loop_create_default()) te se inicijalizira mrežna kartica da funkcionira kao Wi-Fi stanica (esp_netif_create_default_wifi_sta).
Nakon provedenih inicijalizacija postavlja se inicijalna konfiguracija bežične mrežne kartice:

```
wifi_init_config_t cfg = WIFI_INIT_CONFIG_DEFAULT();
    esp_wifi_init(&cfg);
```

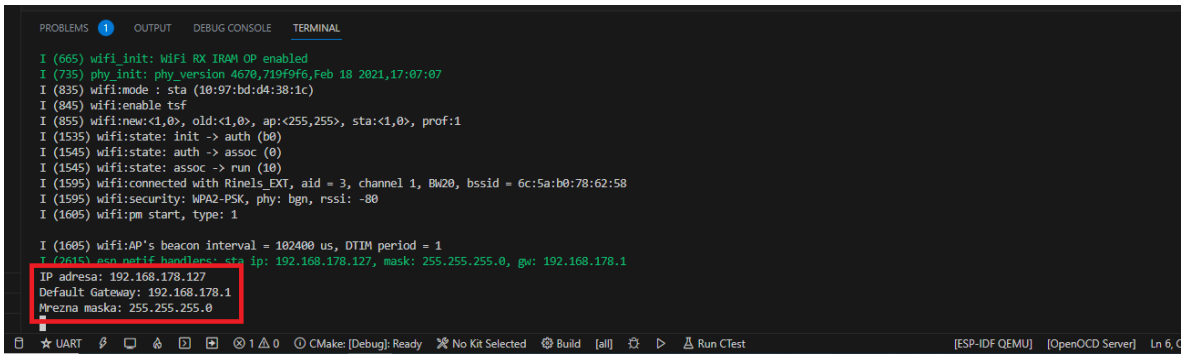
Potom se registriraju događaji (engl. *events*) koji će se pratiti unutar funkcije event_handler.

```
esp_event_handler_instance_t instance_any_id;
    esp_event_handler_instance_t instance_got_ip;
    esp_event_handler_instance_register(WIFI_EVENT,
                                        ESP_EVENT_ANY_ID,
                                        &event_handler,
                                        NULL,
                                        &instance_any_id);
    esp_event_handler_instance_register(IP_EVENT,
                                        IP_EVENT_STA_GOT_IP,
                                        &event_handler,
                                        NULL,
                                        &instance_got_ip);
```

Pratit će se svi događaji koji se odnose na Wi-Fi te samo do događaja dodjele IP adrese koji se odnosi na IP.
Funkcija završava postavljanjem konfiguracije za povezivanje na pristupnu točku i pokretanje bežične mrežne kartice.

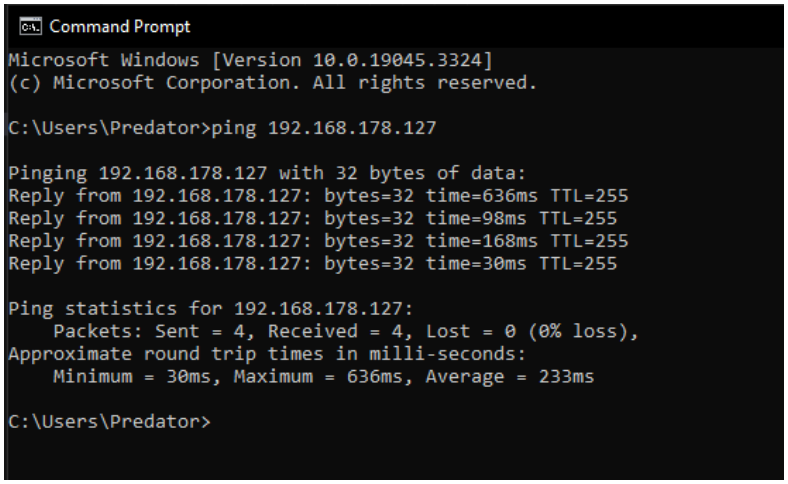
```
wifi_config_t wifi_config = {
        .sta = {
            .ssid = WIFI_SSID,
            .password = WIFI_PASS,
        },
    };
    esp_wifi_set_config(WIFI_IF_STA, &wifi_config);
    esp_wifi_start();
```

U funkciji event_handler (koja se automatski poziva kada se dogodi neki događaj – *event*) može se primijetiti da ako se dogodi događaj WIFI_EVENT_STA_START, pokrenut će se funkcija esp_wifi_connect() koja će pokušati povezati mikrokontroler s pristupnom točkom. Također se može primijetiti da ako se dogodi događaj IP_EVENT_STA_GOT_IP, ispisat će se dodijeljena konfiguracija mrežnoj kartici.
Naredna slika prikazuje rezultat u konzoli nakon kompajliranja programa, njegova slanja na mikrokontroler i pregleda u konzoli. Može se primijetiti koje je konfiguracijske podatke DHCP poslužitelj dodijelio mrežnoj kartici mikrokontrolera.



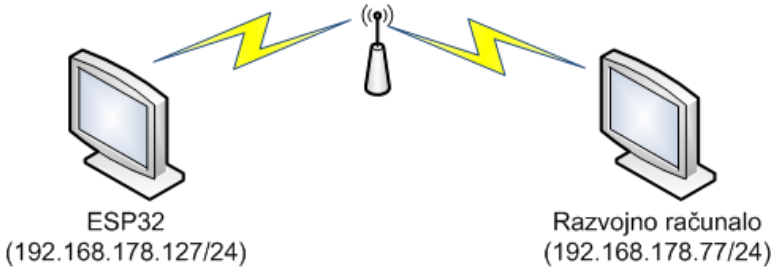
*Slika 2.10.3 Rezultat izvođenja programa prikazan u konzoli*

Koristeći dodijeljene konfiguracijske podatke, na razvojnom se računalu unutar naredbenog retka može pokrenuti ping naredba i time provjeriti povezanost mikrokontrolera s računalnom mrežom u kojoj se nalazi i razvojno računalo.



*Slika 2.10.4 Provjera povezanosti mikrokontrolera i razvojnog računala u računalnu mrežu*

Naredna slika prikazuje topologiju uspostavljene računalne mreže.



*Slika 2.10.5 Topologija uspostavljene računalne mreže*

### 2.10.3. Pitanja i zadaci

1. U program uključiti LED indikator (zeleno) kada se mikrokontroler poveže s mrežom (dobije ID adresu).

2. Pravilni raspored slojeva OSI mrežnog modela jest:
    a. aplikacijski, prezentacijski, sesijski, podatkovni, prijenosni, mrežni i fizički
    b. aplikacijski, prezentacijski, prijenosni, sesijski, podatkovni, mrežni i fizički
    c. aplikacijski, prezentacijski, sesijski, prijenosni, mrežni, podatkovni i fizički
    d. aplikacijski, prezentacijski, sesijski, mrežni, prijenosni, podatkovni i fizički
    e. aplikacijski, prezentacijski, sesijski, podatkovni, mrežni, prijenosni i fizički

3. Mrežni usmjernik uređaj je koji:
    a. povezuje računala unutar računalne mreže istog prefiksa
    b. povezuje računala iz računalnih mreža različitog prefiksa
    c. omogućuje uspostavu optičke računalne mreže
    d. omogućuje uspostavu bežične računalne mreže
    e. omogućuje uspostavu žične računalne mreže

4. Pristupna točka uređaj je koji:
    a. povezuje računala unutar računalne mreže istog prefiksa
    b. povezuje računala iz računalnih mreža različitog prefiksa
    c. omogućuje uspostavu optičke računalne mreže
    d. omogućuje uspostavu bežične računalne mreže
    e. omogućuje uspostavu žične računalne mreže

5. Mrežni preklopnik uređaj je koji:
    a. povezuje računala unutar računalne mreže istog prefiksa
    b. povezuje računala iz računalnih mreža različitog prefiksa
    c. omogućuje uspostavu optičke računalne mreže
    d. omogućuje uspostavu bežične računalne mreže
    e. omogućuje uspostavu žične računalne mreže

6. Postoje tri razine identifikacije računalnih uređaja povezanih u mrežu.
    a. Točno
    b. Netočno

7. Fizičko adresiranje računalnih uređaja provodi:
    a. administrator mreže
    b. poslužitelj DHCP
    c. poslužitelj DNS
    d. proizvođač mrežne kartice
    e. poslužitelj DSS

8. Statičko dodjeljivanje IP adresa provodi:
    a. administrator mreže
    b. poslužitelj DHCP
    c. poslužitelj DNS
    d. proizvođač mrežne kartice
    e. poslužitelj WEB

9. Dinamičko dodjeljivanje IP-a provodi:
    a. administrator mreže
    b. poslužitelj DHCP
    c. poslužitelj DNS
    d. proizvođač mrežne kartice
    e. poslužitelj WEB

10. Za prevođenje simboličkog imena računalnog uređaja u njegovu IP adresu koristi se sljedeći poslužitelj:
    a. poslužitelj DPP
    b. poslužitelj DHCP
    c. poslužitelj DNS
    d. poslužitelj DSN
    e. poslužitelj DSS

11. Pravilni raspored slojeva mrežnog modela TCP/IP jest:
    a. aplikacijski, mrežni, prijenosni, sloj pristupa mreži
    b. aplikacijski, prijenosni, mrežni, sloj pristupa mreži
    c. aplikacijski, sloj pristupa mreži, mrežni, prijenosni
    d. aplikacijski, sesijski, mrežni, prijenosni, sloj pristupa mreži
    e. aplikacijski, fizički, mrežni, prijenosni, sloj pristupa mreži

12. Kako se naziva postupak slanja podataka niz slojeve OSI modela:
    a. enkapsulacija
    b. deenkapsulacija

13. Od koliko se bitova sastoji IP adresa verzije 4:
    a. 30
    b. 31
    c. 32
    d. 33
    e. 34

14. Od koliko se bitova sastoji mrežna maska za IP adrese verzije 4:
    a. 30
    b. 31
    c. 32
    d. 33
    e. 34

15. Adresna maska je:
    a. pojam koji opisuje dodjelu IP adresa mrežnim usmjerivačima
    b. pojam koji omogućuje proizvoljno dijeljenje IP adrese na adresu mreže i adresu čvora
    c. pojam koji omogućuje dodjeljivanje adresa mreže u WAN mrežama
    d. pojam koji omogućuje dodjeljivanje adresa mreže u LAN mrežama
    e. pojam koji omogućuje dodjeljivanje adrese čvora u WAN mrežama

16. U CIDR notaciji /8 predstavlja sljedeću mrežnu masku:
    a. 255.255.255.255
    b. 255.255.255.0
    c. 255.255.0.0
    d. 255.0.0.0
    e. 0.255.0.0

17. U CIDR notaciji /16 predstavlja sljedeću mrežnu masku:
    a. 255.255.255.255
    b. 255.255.255.0
    c. 255.255.0.0
    d. 255.0.0.0
    e. 0.255.0.0

18. U CIDR notaciji /24 predstavlja sljedeću mrežnu masku:
    a. 255.255.255.255
    b. 255.255.255.0
    c. 255.255.0.0
    d. 255.0.0.0
    e. 0.255.0.0

19. Mrežna maska koja će od IP adrese 192.168.1.9 vratiti adresu mreže 192.168.0.0 glasi:
    a. 255.255.255.255
    b. 255.255.255.0
    c. 255.255.0.0
    d. 255.0.0.0
    e. 0.255.0.0

20. Što je adresa mreže IP adrese 10.10.10.10/24:
    a. 10.10.10.10
    b. 10.10.10.0
    c. 10.10.0.0
    d. 10.0.10.0
    e. 10.0.0.0

21. Za specifikaciju IP adrese v4 koriste se dekadski brojevi iz sljedećeg raspona:
    a. 1 – 256
    b. 0 – 256
    c. 1 – 255
    d. 0 – 255
    e. 0 – 254

22. Za specifikaciju mrežne maske IP adrese v4 koriste se dekadski brojevi iz sljedećeg raspona:
    a. 1 – 256
    b. 0 – 256
    c. 1 – 255
    d. 0 – 255
    e. 0 – 254

23. U koju vrstu spada mreža koja povezuje računalne uređaje u istoj zgradi?
    a. WAN
    b. PAN
    c. LAN
    d. CAN
    e. MAN

24. U koju vrstu spada mreža koja povezuje računalne uređaje unutar kampusa?
    a. WAN
    b. PAN
    c. LAN
    d. CAN
    e. MAN

25. U koju vrstu spada mreža koja povezuje računalne uređaje u kući korisnika?
    a. WAN
    b. PAN
    c. LAN
    d. AN
    e. MAN

26. U koju vrstu spada mreža koja povezuje računalne uređaje u radnom prostoru osobe?
    a. WAN
    b. PAN
    c. LAN
    d. HAN
    e. MAN

### 2.10.4. Literatura i izvori


1. ESP-NETIF, https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_netif.html#
2. Wi-Fi, https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_wifi.html
3. Wi-Fi Station Example, https://github.com/espressif/esp-idf/tree/52bca70b1a73248c9a66095d2cfac8c6ebb1f263/examples/wifi/getting_started/station


## 2.11. Izrada *web* aplikacijskoga programskog sučelja preko kojega se ubacuju podaci u bazu podataka

Potrebno je izraditi *web* aplikacijsko programsko sučelje preko kojega će se u tablicu u bazi podataka ubacivati podaci kao što su datum i vrijeme očitanja, temperatura i vlažnost zraka.

**Nakon usvajanja ove nastavne teme čitatelj će moći:**
1. Izraditi programsko rješenje koje će koristiti bazu podataka za vremenske sljedove za pohranu generiranih podataka

2. Objasniti razlike i primjenu čestih formata za razmjenu podataka putem aplikacijsko programskih sučelja
3. pomoću programskog okvira izraditi *web*-aplikacijsko programsko sučelje prikladno za upotrebu u sustavu interneta stvari
4. izraditi *web*-aplikacijsko programsko sučelje za vlastiti sustav interneta stvari

## 2.11.1. Osnovni koncepti

Baza podataka predstavlja skup podataka koji se čuvaju na strukturirani način kako bi se time olakšala njihova primjena. Postoje dvije vrste baza podataka: relacijske i nerelacijske. Relacijske baze podataka čuvaju podatke unutar međusobno povezanih tablica – svaka tablica ima niz stupaca. To znači da relacijska baza podataka treba imati unaprijed definiranu strukturu tablica kako bi se u njoj mogli pohraniti podaci. Nerelacijske baze podataka za pohranu podataka ne zahtijevaju unaprijed definiranu strukturu pohrane.

Primjera radi, baza podataka u kojoj se čuvaju očitanja senzora temperature i vlažnosti zraka unutar različitih prostorija neke zgrade može se sastojati od dvije međusobno povezane tablice, rooms i readings. Tablica rooms sastoji se od dva stupca: id – jedinstveni identifikator prostorije i name – naziv prostorije. Tablica reading sastoji se od pet stupaca: id – jedinstveni identifikator očitanja, id_room – identifikacija prostorije na koju se odnosi očitanje, reading_date_time – datum i vrijeme očitanja, temperature – vrijednost temperature i humidity – vrijednost vlažnosti zraka.

| rooms | |
|---|---|
| id | name |
| 1 | Sala za sastanke |
| 2 | Dvorana 1 |
| 3 | Laboratorij 1 |
| 4 | Ured 1 |

| readings | | | | |
|---|---|---|---|---|
| id | id_room | reading_date_time | temperature | humidity |
| 1 | 2 | 2023-09-02 08:20:33 | 22.0 | 68.0 |
| 2 | 4 | 2023-07-30 22:01:35 | 19.0 | 55.0 |
| 3 | 1 | 2023-08-30 13:06:23 | 18.0 | 63.0 |
| 4 | 1 | 2023-09-01 20:52:42 | 23.0 | 53.0 |

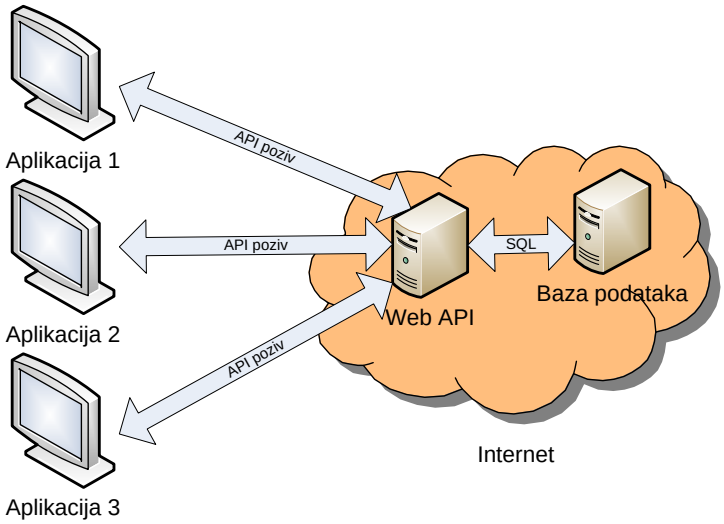
Iz ovako se strukturiranog zapisa podataka preko upitnog jezika SQL (engl. *structured query language*) mogu raditi razne analize, npr. koja je prosječna temperatura i/ili vlažnost zraka u nekom vremenskom periodu za neku prostoriju.

Treba primijetiti da je svaki stupac u tablici određenog tipa podatka (npr. stupac name je niz znakova, stupac id je integer itd.). Stupac reading_date_time jest tipa podatka date time (datumski i vremenski tip podatka). Ako je baza podataka posebno optimizirana za rad s takvim tipovima podataka, tada se govori o bazama podataka za vremenske sljedove (engl. *time-series database*).

Baza podataka implementira se na nekom poslužitelju baze podataka. Pitanje koje se postavlja jest kako neka korisnička aplikacija (ili aplikacija na mikrokontroleru) koristi bazu podataka. Čak je moguće i da više različitih aplikacija koristi istu bazu podataka. Jedan od načina jest da se izrade posebne funkcije za rad s bazom podataka koje će biti dostupne aplikacijama koje ih trebaju. Takve funkcije ne moraju biti predviđene samo za rad s bazom podataka, već se mogu koristiti i u druge svrhe (npr. za komunikaciju između dva dijela nekog sustava itd.). Budući da navedene funkcije trebaju biti dostupne različitim aplikacijama, one se također nalaze na posebnim poslužiteljima.

Jedan je od načina izvedbe tih funkcija preko *web* aplikacijskoga programskog sučelja (engl. *web application programming interface* – Web API).

Primjer arhitekture aplikacija koje koriste Web API kako bi pristupile bazi podataka prikazuje sljedeća slika.




*Slika 2.11.1 Arhitektura aplikacija koje pozivaju Web API koji dalje komunicira s bazom podataka*

Web API može, ali ne mora primiti argumente. Također, Web API može, ali ne mora vratiti podatke mjestu poziva (aplikaciji). Najčešći format dostavljanja podataka, odnosno primanja podataka od Web API-a jest JSON (engl. *JavaScript Object Notation* – JSON). On se u osnovi sastoji od naziva atributa i njegove vrijednosti – npr. {"name":"John", "age":30, "car":null}. Drugi format slanja i primanja podataka koji se koristi jest XML (engl. *eXtensible Markup Language*). On se sastoji od XML oznaka (engl. *tags*) kojima se definiraju naziv atributa i njegova vrijednost – npr.

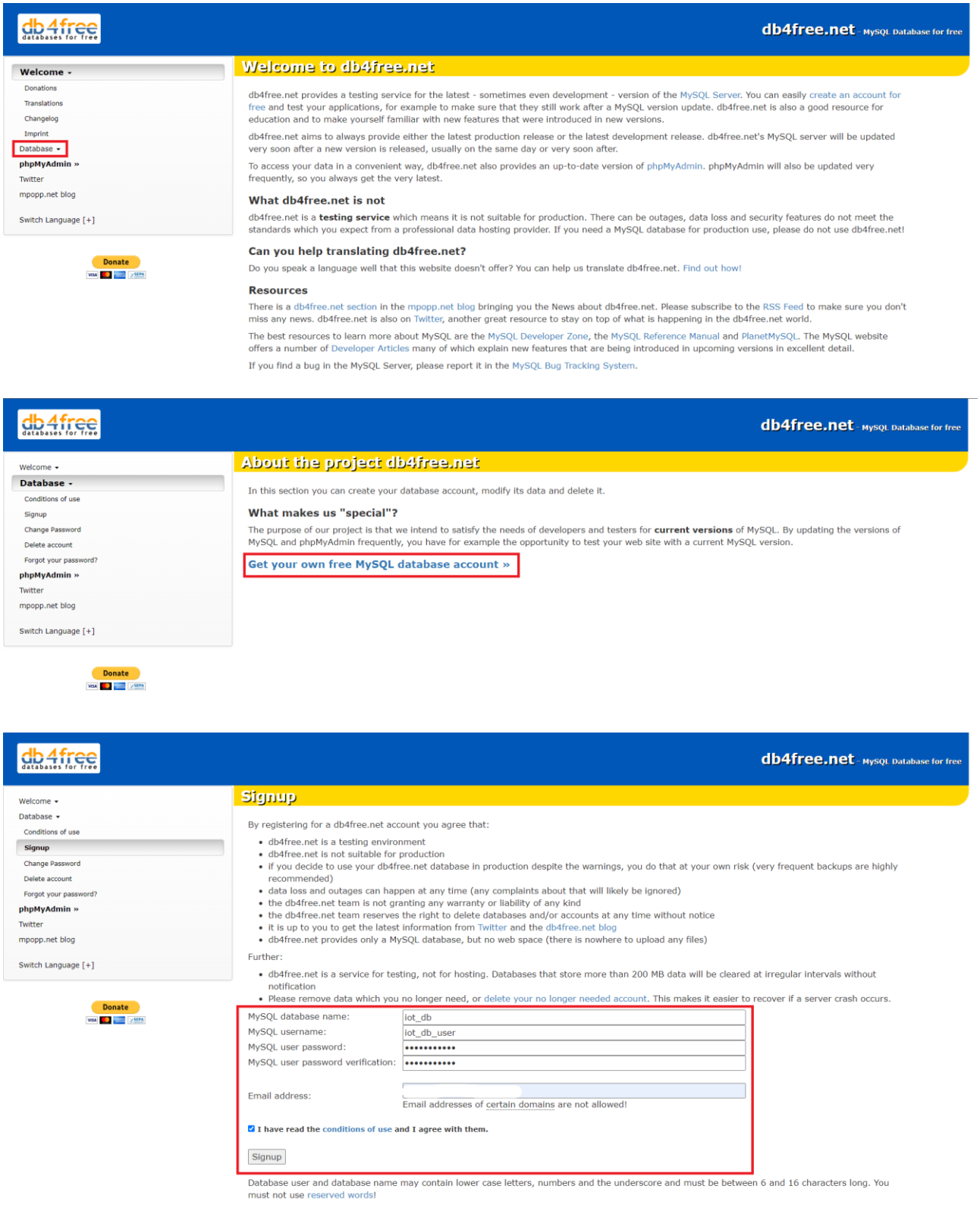
```
<root>
    <name>John</name>
    <age>30</age>
    <car />
</root>
```

Poziv Web API-ja aplikacija radi preko HTTP ili HTTPS protokola koristeći URL (engl. *uniform resource identifier*) – npr. http://universities.hipolabs.com/search?country=United+Kingdom mjestu poziva vraća listu sveučilišta u Velikoj Britaniji u formatu JSON. U pozivu Web API dio ?country=United+Kingdom treba primijetiti što je argument koji se dostavlja funkciji (može se isprobati poslati argument Croatia).
Aplikacija koja koristi Web API za rad s podacima ima na raspolaganju nekoliko HTTP naredbi:

- dohvat podataka (čitanje) - GET naredba
- kreiranje podataka - POST naredba
- izmjena podataka - PUT naredba
- brisanje podataka - DELETE naredba

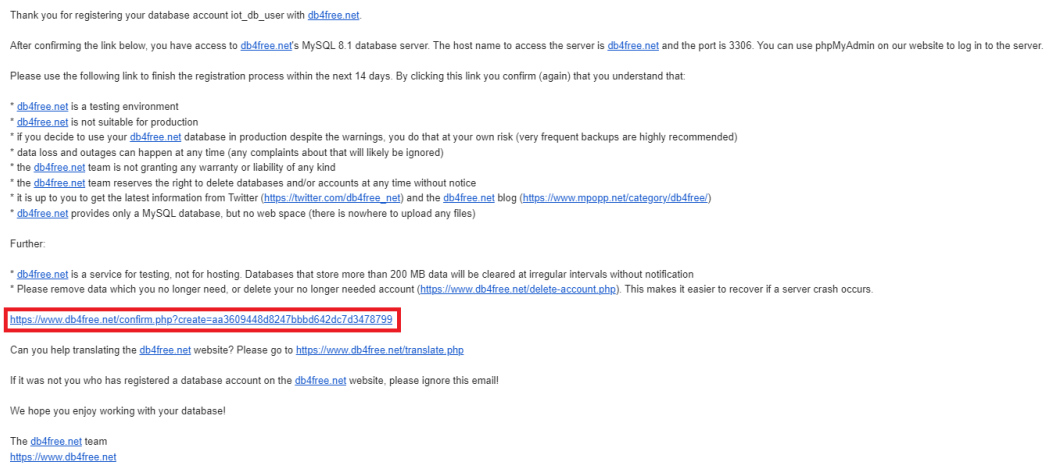
## 2.11.2. Rješenje radnog zadatka

Na poveznici https://db4free.net/ može se izraditi *online* dostupna baza podataka. Izradit će se baza podataka pod nazivom IoT_DB i u njoj će se izraditi jedna tablica pod nazivom readings koja ima četiri stupca: id, reading_dateTime, temperature i humidity.



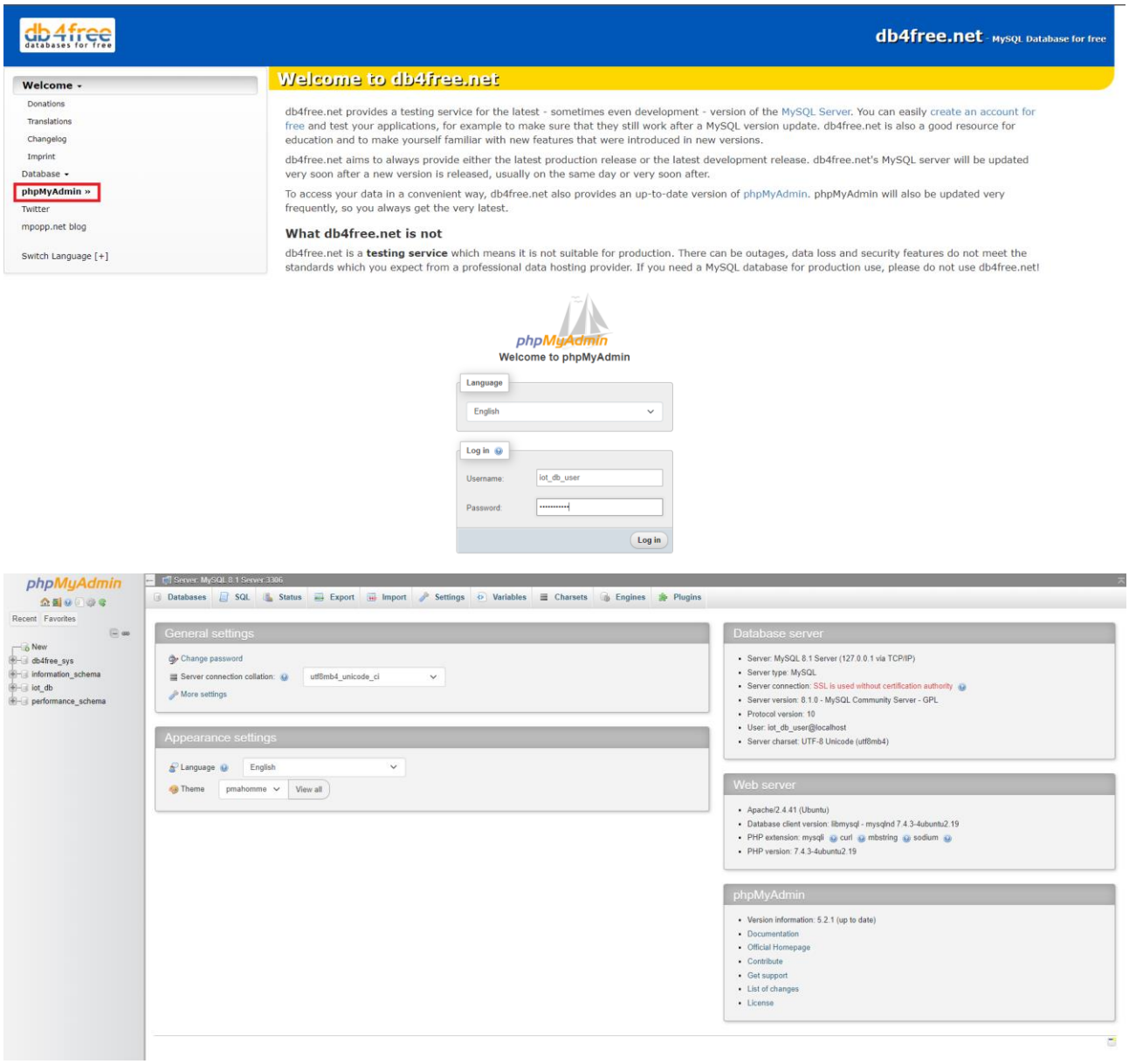

*Slika 2.11.2 Stvaranje baze podataka na poveznici [https://db4free.net/](https://db4free.net/)*

Nakon pritiska na gumb Signup dobit će se e-poruka u kojoj je preko dostavljene poveznice potrebno potvrditi ispravnost e-adrese.



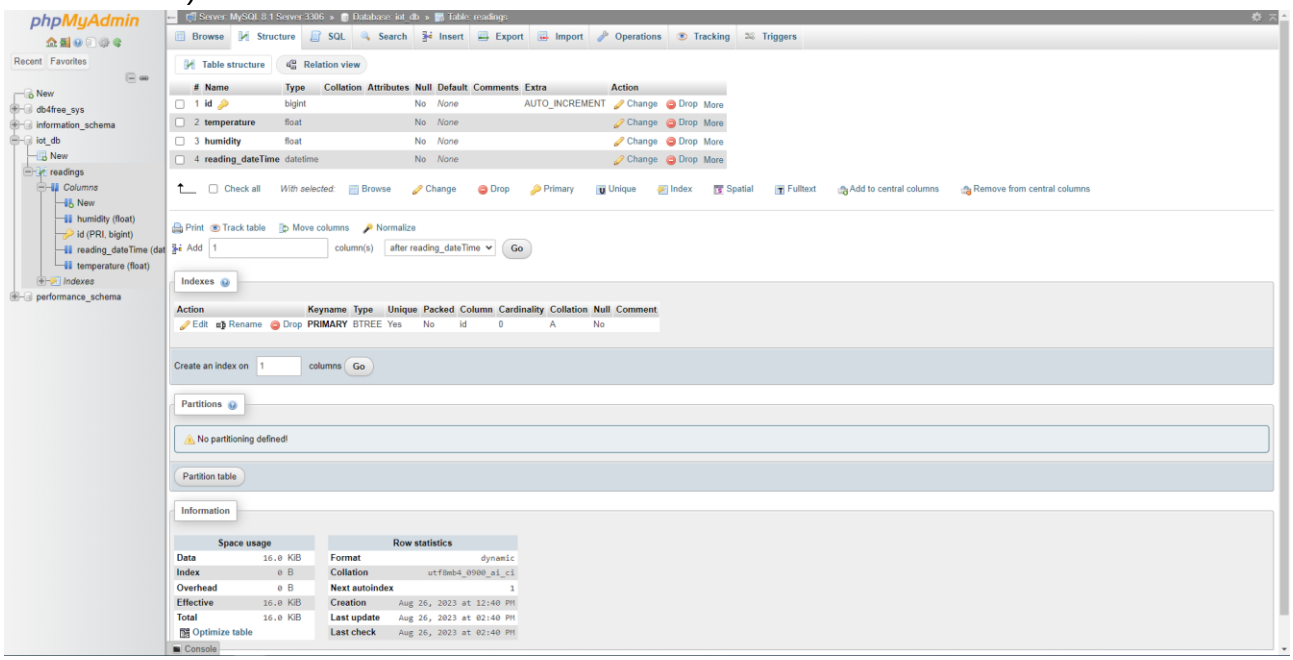

*Slika 2.11.3 Primljena poruka i poveznica za potvrdu e-adrese*

Nakon potvrde e-adrese baza podataka je izrađena. Sada je potrebno preko alata phpMyAdmin prijaviti se na izrađenu bazu podataka (korisnički račun jest onaj koji je izrađen tijekom procesa izrade baze podataka).




*Slika 2.11.4 Prijava na bazu podataka*

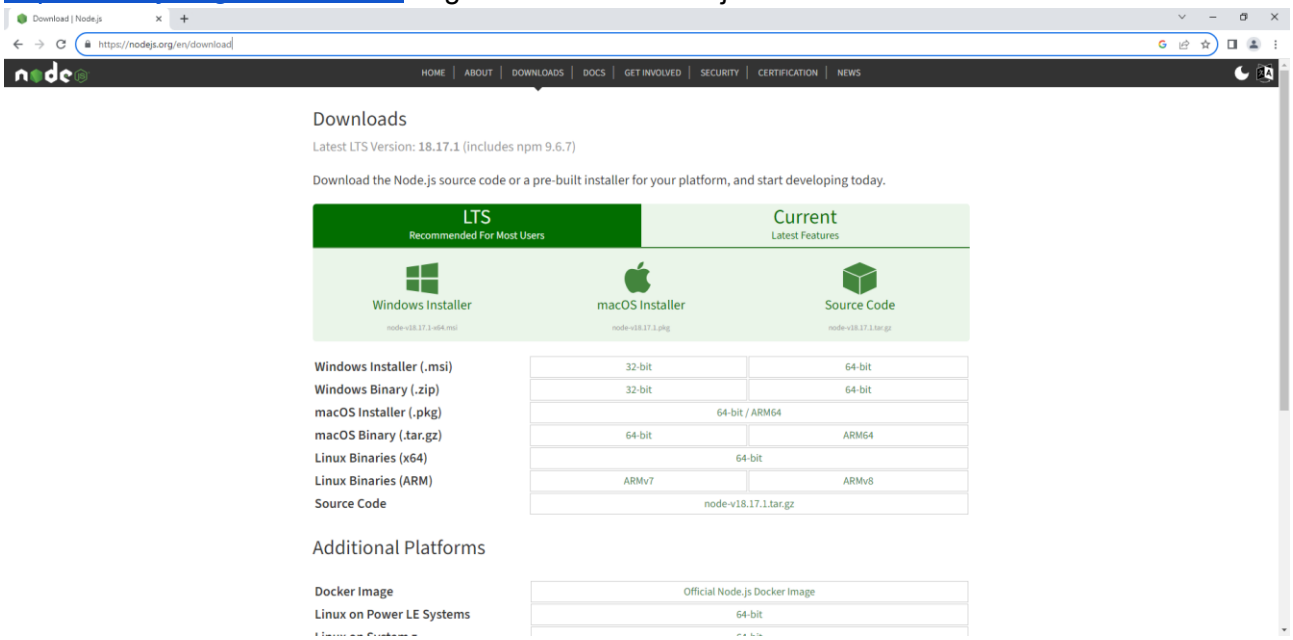
Sada se u bazi podataka iot_db treba izraditi tablica readings sa stupcima id, reading_dateTime, temperature i humidity. Naredna slika prikazuje strukturu tablice – obratiti pažnju da je id primarni ključ s postavljenim AUTO_INCREMENT (automatski će se postavljati vrijednost pri svakom unosu retka u tablicu).




*Slika 2.11.5 Izrađena tablica u bazi podataka*

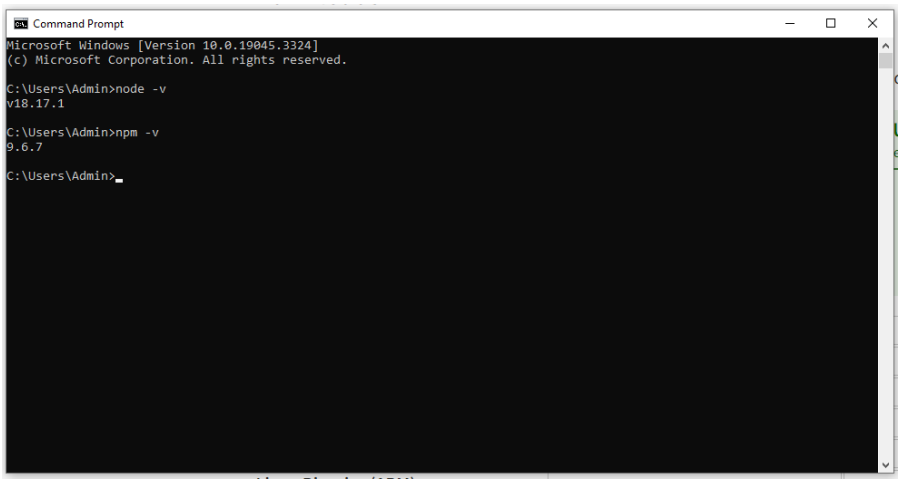
Nakon izrađene baze podataka slijedi izrada *web* aplikacijskoga programskog sučelja preko razvojnog okvira express.

Prvo je potrebno preuzeti Node.js (poslužiteljsko okruženje) preko poveznice https://nodejs.org/en/download te ga instalirati na razvojno računalo.



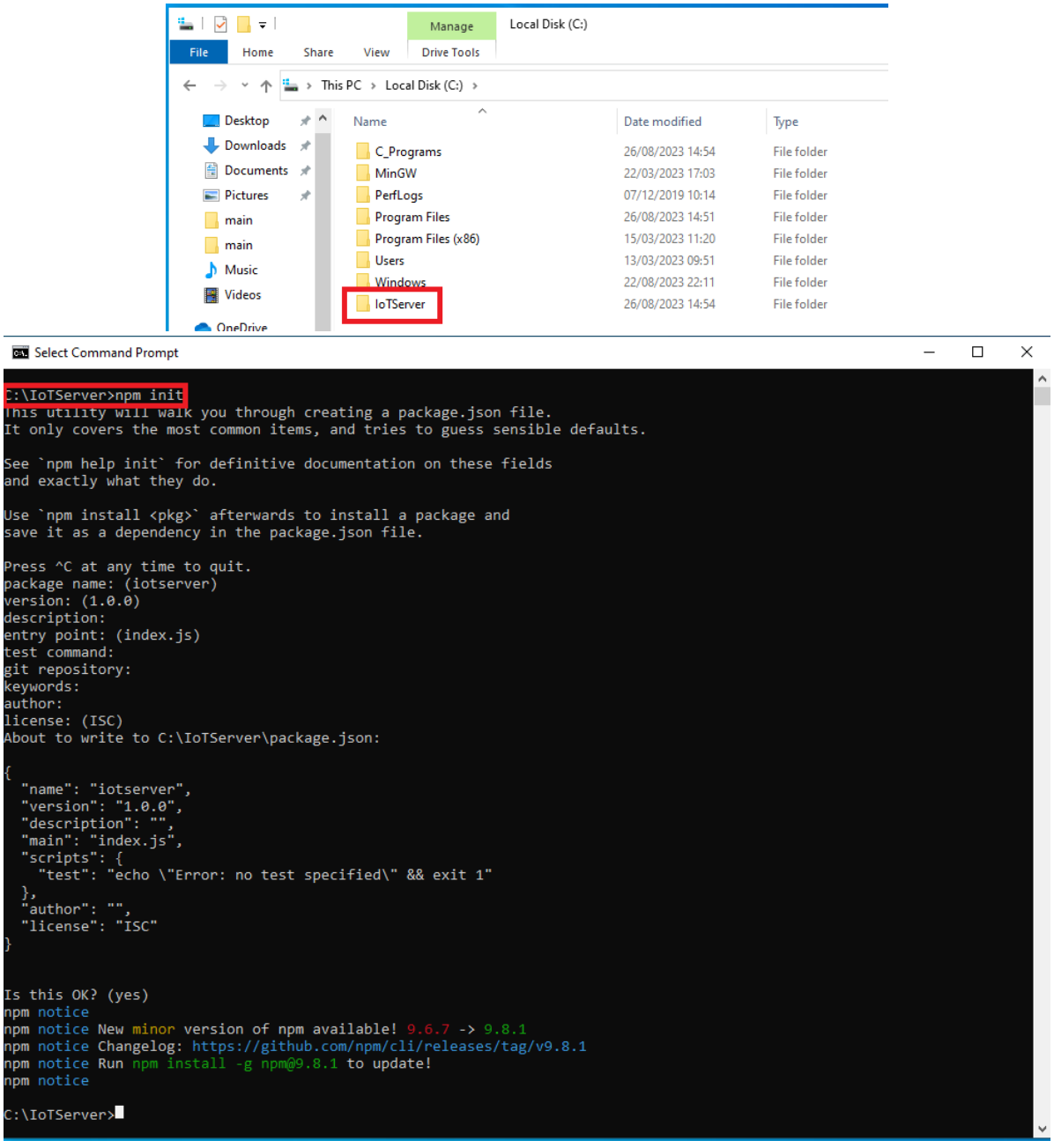

*Slika 2.11.6 Preuzimanje poslužiteljskog okruženja Node.js*

Nakon instalacije može se provjeriti verzija Node.js i npm programa.



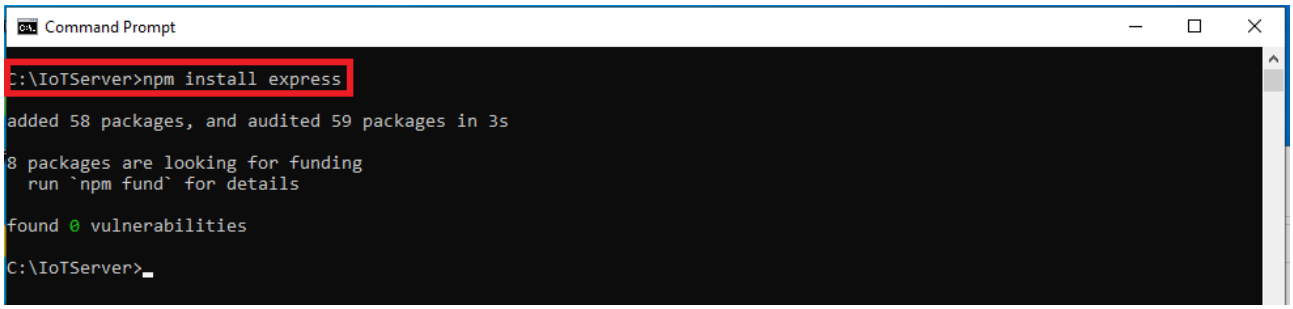

*Slika 2.11.7 Provjera instaliranosti Node.js i npm programa*

Stvorit će se mapa IoTServer u koju će se inicijalizirati Node.js projekt naredbom npm init.




*Slika 2.11.8 Inicijaliziran projekt u mapi IoTServer*

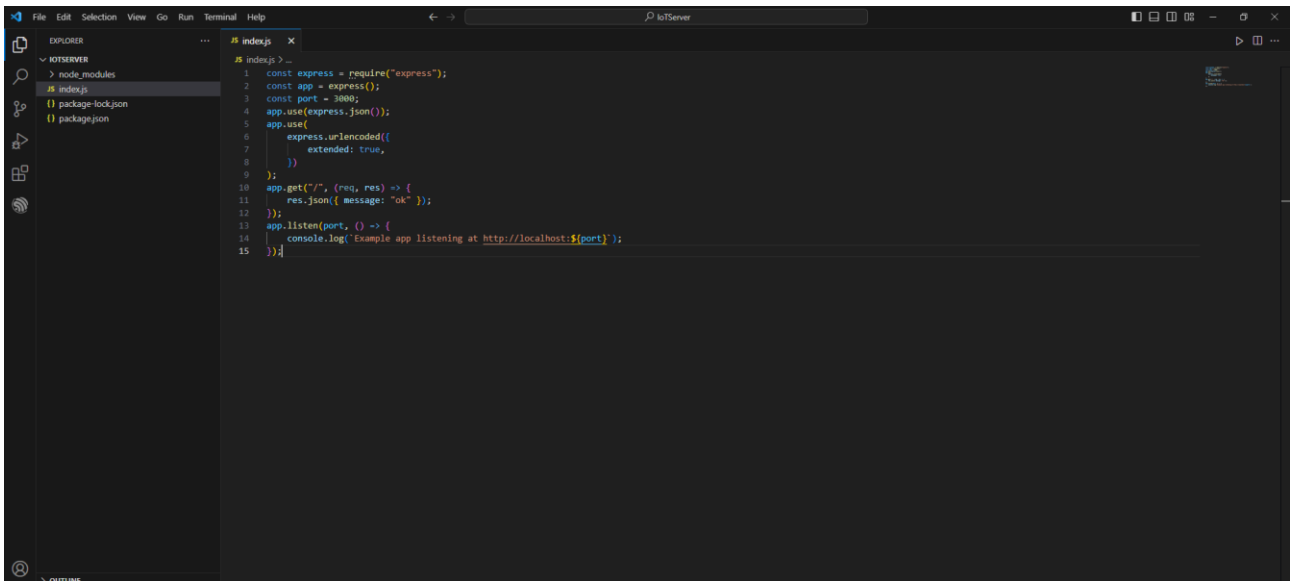
Sada će se instalirati razvojni okvir express tako da se unutar mape IoTServer pokrene npm install express.




*Slika 2.11.9 Instalacija razvojnog okvira express*

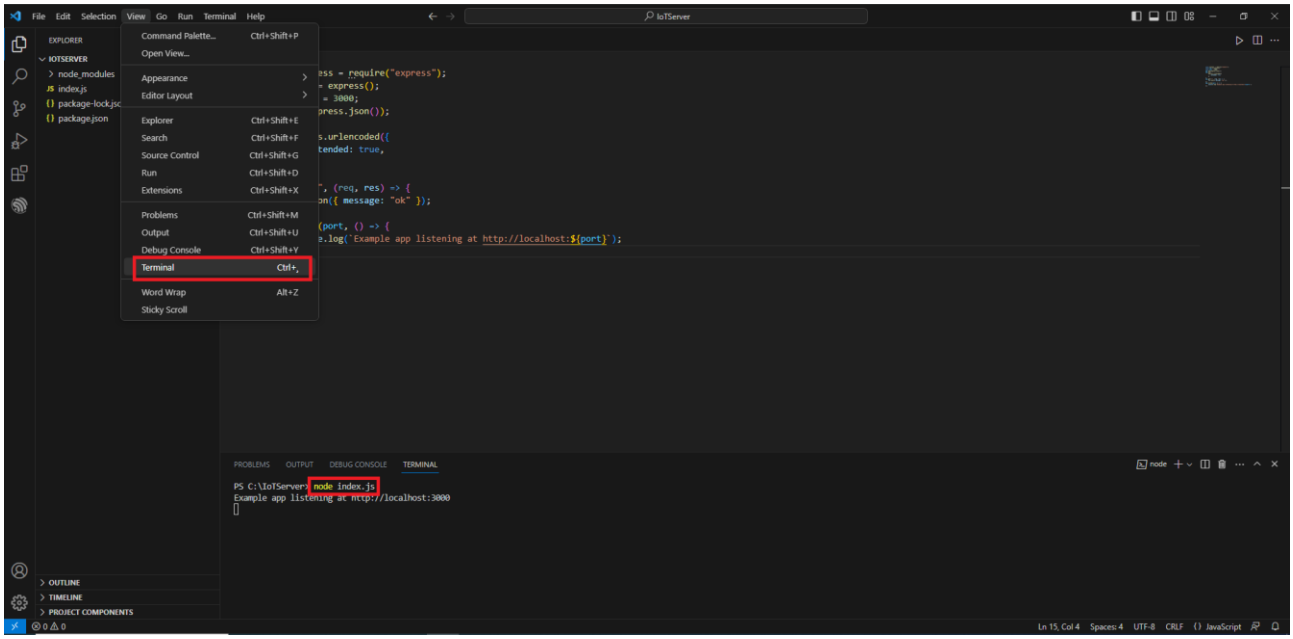
Nakon instalacije razvojnog okvira otvorit će se Visual Studio Code i u njemu će se otvoriti mapa IoTServer. Izradit će se nova datoteka index.js i u nju će se ubaciti programski kôd.

```
const express = require("express");
const app = express();
const port = 3000;
app.use(express.json());
app.use(
  express.urlencoded({
    extended: true,
  })
);
app.get("/", (req, res) => {
  res.json({ message: "ok" });
});
app.listen(port, () => {
  console.log(`Example app listening at http://localhost:${port}`);
});
```



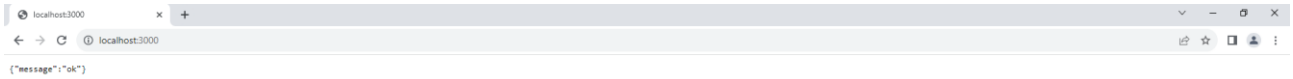
*Slika 2.11.10 Izrađena datoteka index.js*

Program se pokreće tako da se u terminalu pokrene naredna node index.js. Do terminala se može doći i preko softvera Visual Studio Code.



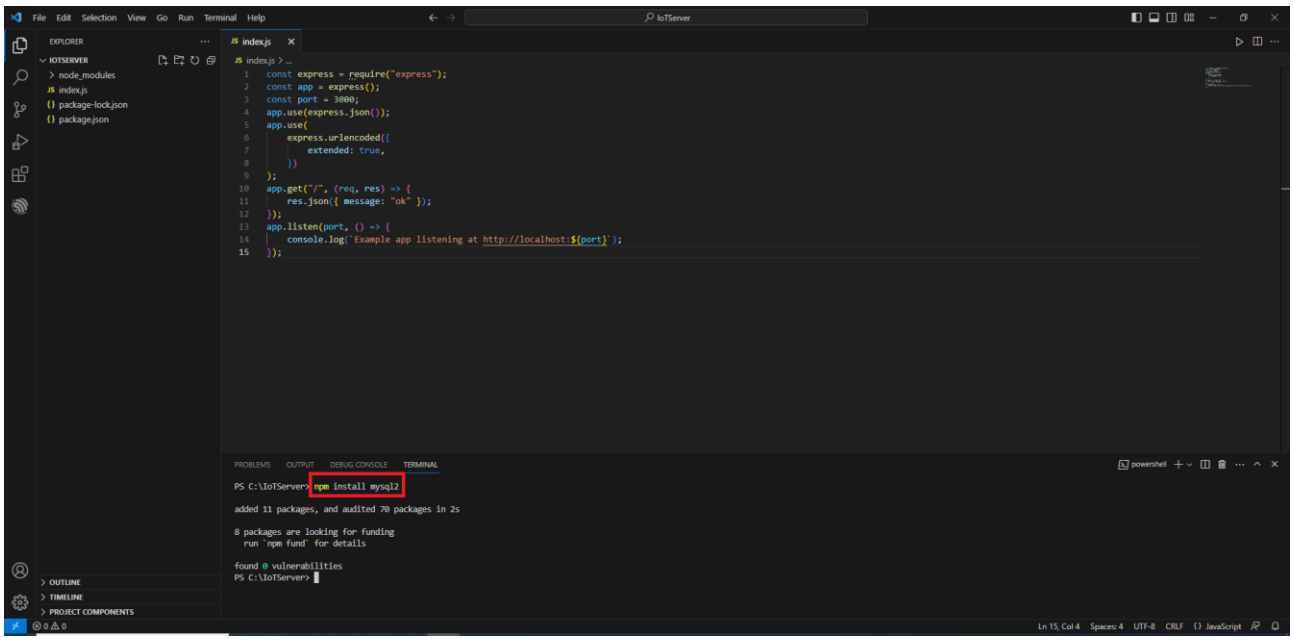

*Slika 2.11.11 Pokretanje poslužitelja preko terminala*

Ako se sada u internetskom pregledniku na razvojnom računalu upiše poveznica (ruta) http://localhost:3000, odgovorit će upravo pokrenuti poslužitelj.




*Slika 2.11.12 Odgovor poslužitelja na zahtjev*

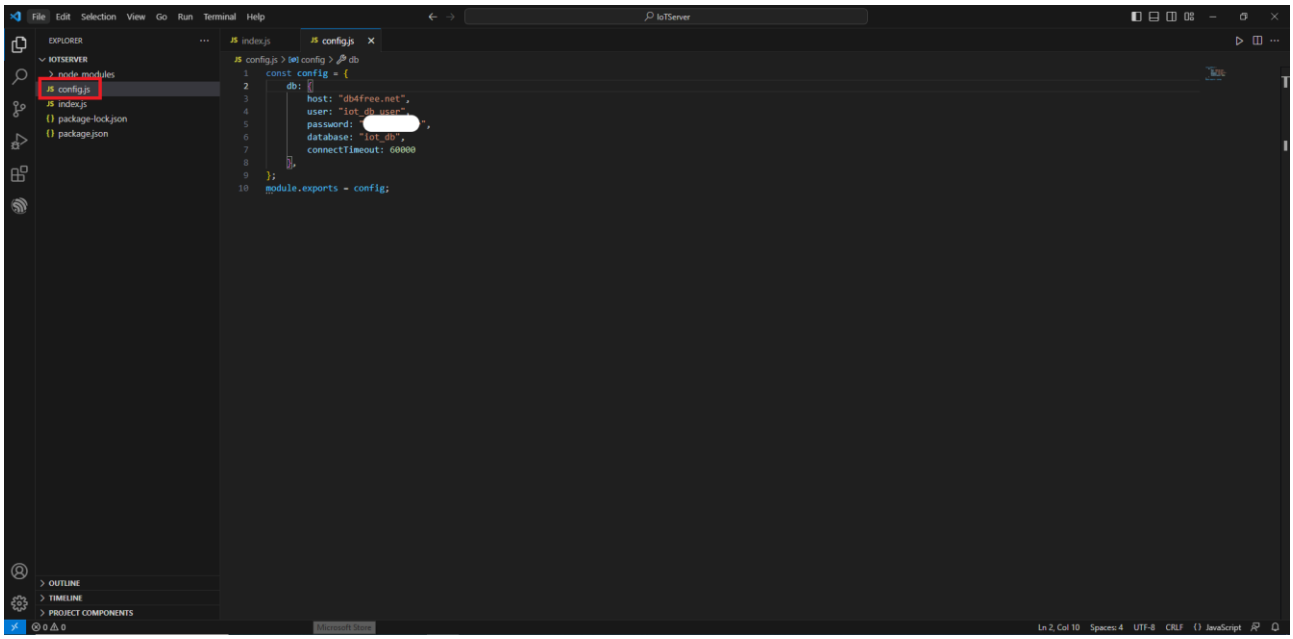
Ono što je sad potrebno napraviti jest ruta koja će ubacivati podatke u bazu podataka iot_db.
U tu je svrhu potrebno u program Node.js uključiti paket koji omogućuje rad s bazom podataka – mysql2. Taj se paket dodaje upisivanjem naredbe npm install mysql2 u terminal (može se koristiti i terminal unutar softvera Visual Studio Code).




*Slika 2.11.13 Instalacija paketa za rad s bazom podataka*

Sada će se stvoriti konfiguracijska datoteka (config.js) koja će se koristiti kako bi se poslužitelj mogao povezati s bazom podataka.
Sadržaj datoteke config.js:

```
const config = {
    db: {
        host: "db4free.net",
        user: "iot_db_user",
        password: "lozinka za iot_db_user",
        database: "iot_db",
        connectTimeout: 60000
    },
};
module.exports = config;
```




*Slika 2.11.14 Stvorena datoteka config.js*

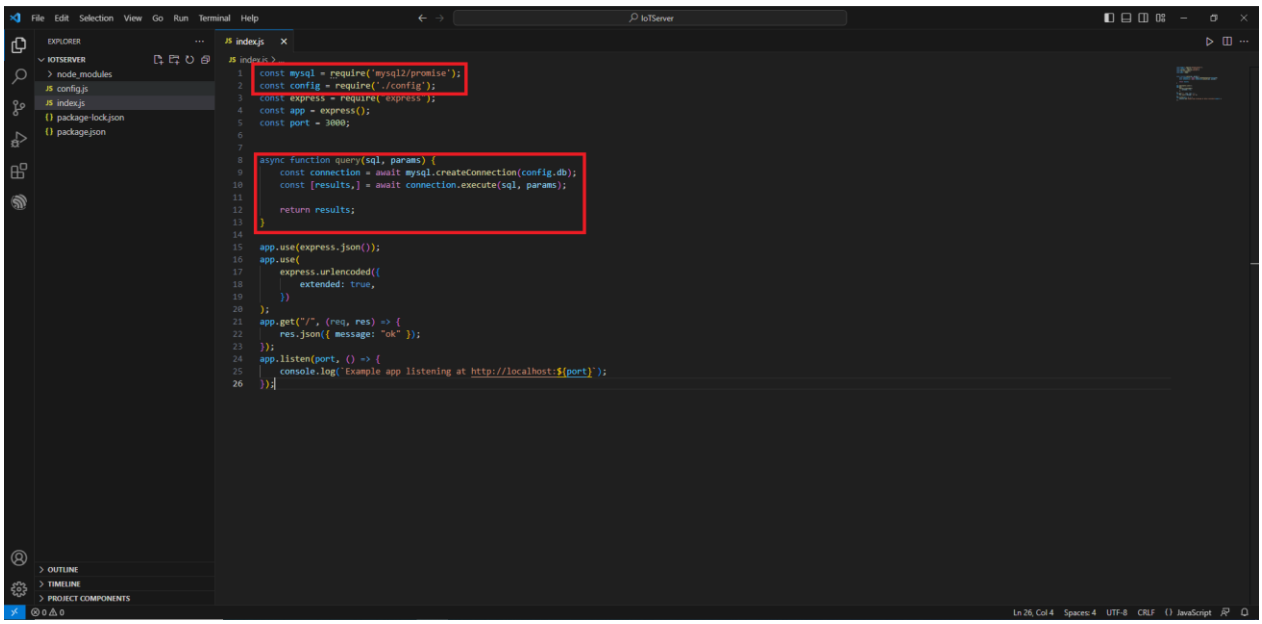
Slijedi povezivanje na bazu podataka iz datoteke index.js uz primjenu konfiguracijskih podataka iz datoteke config.js. Također će se izraditi i funkcija query kojoj će se slati SQL naredbe koje se trebaju izvesti na bazi podataka.
Ovo je programski kôd koji je potrebno dodati u datoteku index.js:

```
const mysql = require('mysql2/promise');
const config = require('./config');
async function query(sql, params) {
    const connection = await mysql.createConnection(config.db);
    const [results,] = await connection.execute(sql, params);

    return results;
}
```

Sada je potrebno izraditi novu rutu preko koje će se dodavati novi podaci u tablicu readings baze podataka. Nova ruta bit će http://localhost:3000/reading i preko nje će se podaci slati u bazu podataka.

Ovo je programski kod koji je potrebno dodati u index.js datoteku.

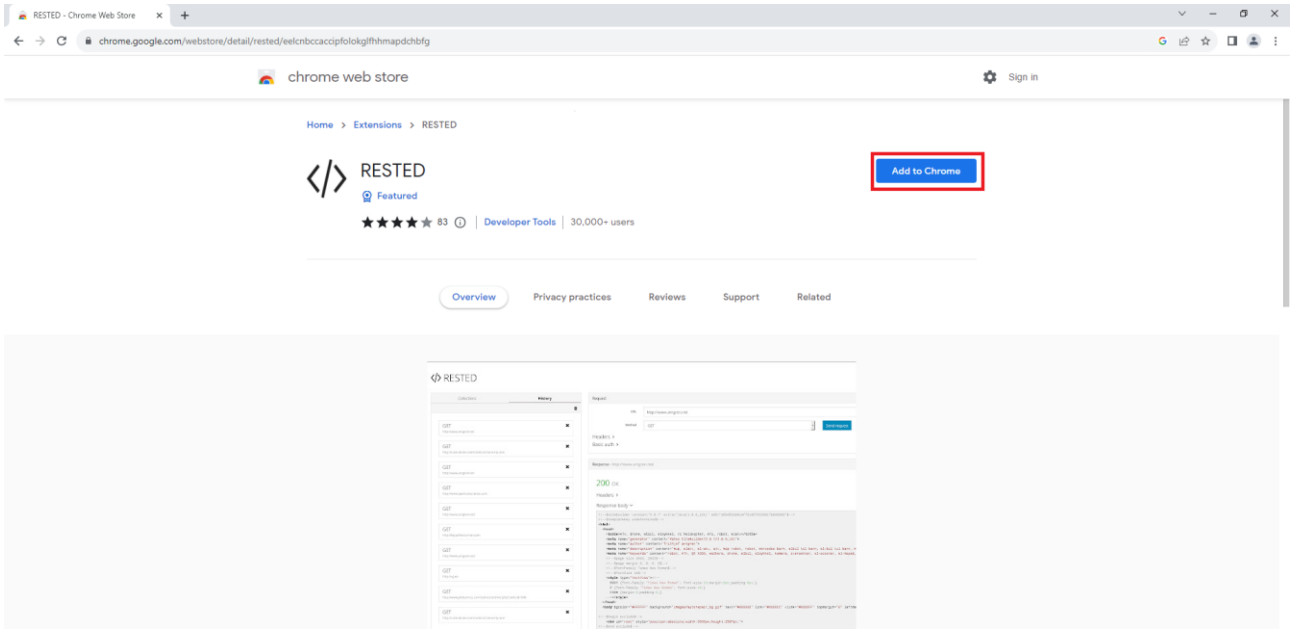
```
app.post('/reading', async function (req, res, next) {
    try {
        var currentDateTime = (new Date()).toISOString().slice(0,
19).replace('T', ' ');
        const result = await query(
            `INSERT INTO readings
            (temperature, humidity, reading_dateTime)
            VALUES
            (${req.body.temperature}, ${req.body.humidity},
'${currentDateTime}')`
        );


        if (result.affectedRows) {
            message = 'Očitanje uspješno spremljeno';
        }
        res.json({ message: "ok" });
    } catch (err) {
        console.error(`Greška u stvaranju očitanja `, err.message);
        next(err);
    }
});
```

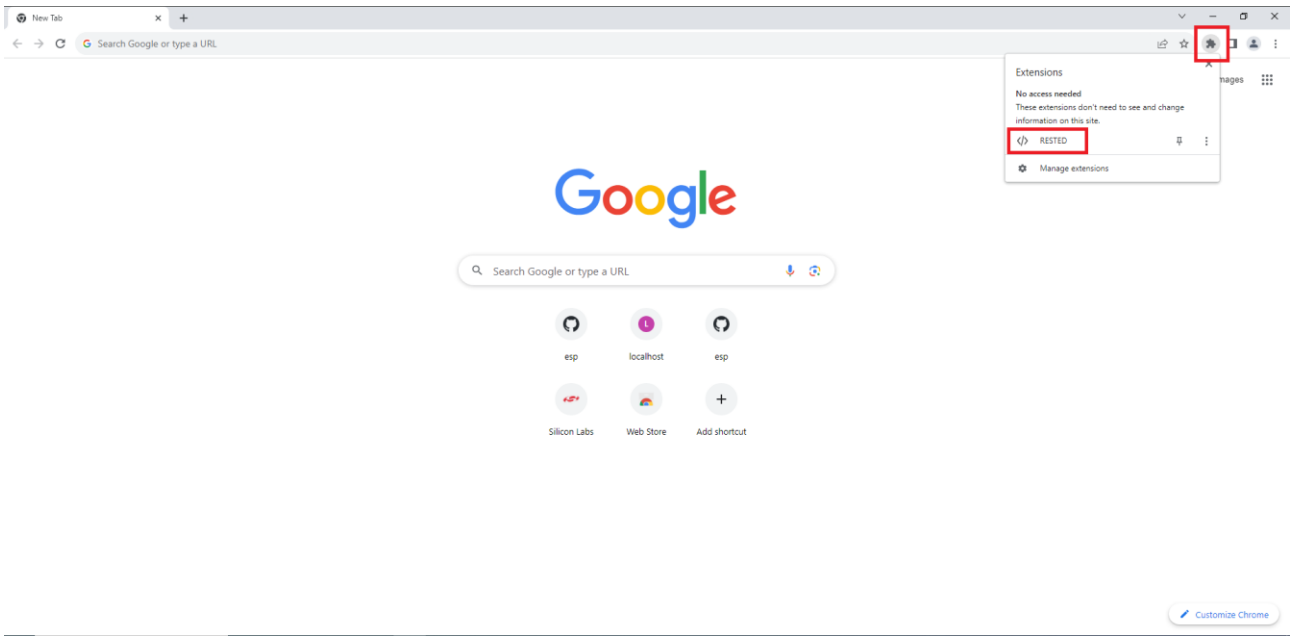



*Slika 2.11.15 Dodan programski kod za rad s bazom podataka*

U ovom programskom kodu, kada se pozove ruta ./reading (i to kao POST) kojoj se pošalje struktura podataka JSON {temperature: vrijednost, humidity: vrijednost}, tada se prvo uzme trenutačni datum i vrijeme u varijablu currentDateTime i to u formatu YYYY-MM-DD HH-mm-ss (što je format zapisa u bazu podataka). Potom se slaže naredba INSERT SQL u koju se dodaju podaci koji se trebaju ubaciti u tablicu readings. Ta se naredba šalje funkciji query koja je dalje šalje na bazu podataka.

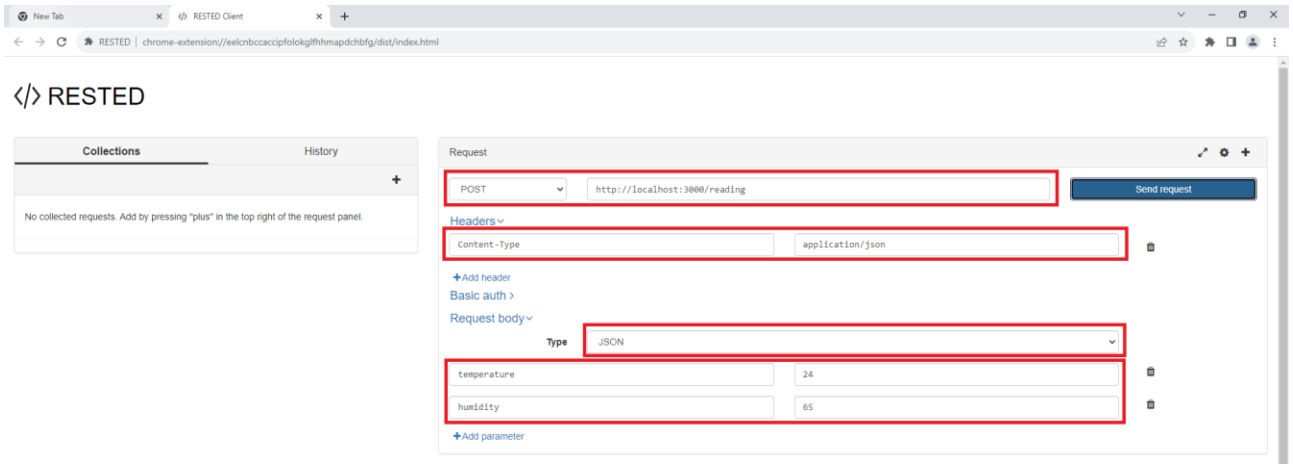
Kako bi se provjerila nova ruta, potrebno je u internetski preglednik Chrome instalirati dodatak RESTED preko kojega se može provjeriti funkcioniranje nove rute. Instalacija dodatka radi se preko poveznice https://chrome.google.com/webstore/detail/rested/eelcnbccaccipfolokglfhhmapdchbfg.





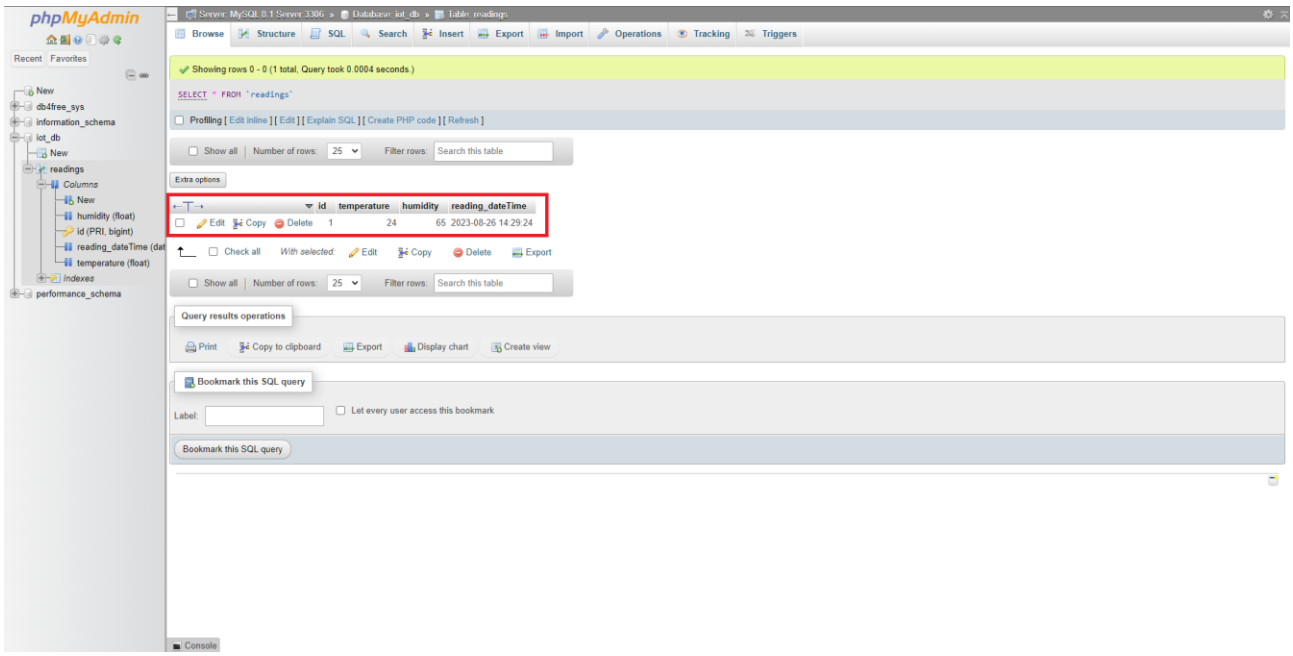

*Slika 2.11.16 Instalacija RESTED dodatka*

Kada se dodatak RESTED pokrene, u njega se može upisati ruta kojoj se mogu poslati podaci. Upisuju se vrsta poziva (POST), zaglavlje poziva (Content-Type: application/json), vrsta tijela poruke (JSON) i podaci u formatu JSON (polja i njihove vrijednosti).




*Slika 2.11.17 Pozivanje rute [http://localhost:3000/reading](http://localhost:3000/reading) i slanje podataka*

U bazi podataka može se vidjeti da je u tablicu readings ubačen redak s poslanim podacima.




*Slika 2.11.18 Ubačen redak u bazi podataka preko web aplikacijskoga programskog sučelja*

## 2.11.3. Pitanja i zadaci

1. Na sličan način kako je prikazano u radnom zadatku izraditi GET, PUT i DELETE *web* aplikacijska sučelja koja će manipulirati tablicom readings.
2. Neka je dana sljedeća baza podataka:

| rooms | |
|---|---|
| id | name |
| 1 | Sala za sastanke |
| 2 | Dvorana 1 |
| 3 | Laboratorij 1 |
| 4 | Ured 1 |

| readings | | | | |
|---|---|---|---|---|
| id | id_room | reading_date_time | temperature | humidity |
| 1 | 2 | 2023-09-02 08:20:33 | 22.0 | 68.0 |
| 2 | 4 | 2023-07-30 22:01:35 | 19.0 | 55.0 |
| 3 | 1 | 2023-08-30 13:06:23 | 18.0 | 63.0 |
| 4 | 1 | 2023-09-01 20:52:42 | 23.0 | 53.0 |

Koliko je očitanja temperature i vlažnosti zraka evidentirano za salu za sastanke?

a. 0
b. 1
c. 2
d. 3
e. 4

3. Neka je dana sljedeća baza podataka:

| rooms | |
|---|---|
| id | name |
| 1 | Sala za sastanke |
| 2 | Dvorana 1 |
| 3 | Laboratorij 1 |
| 4 | Ured 1 |
| 5 | Hodnik 1 |

| readings | | | | |
|---|---|---|---|---|
| id | id_room | reading_date_time | temperature | humidity |
| 1 | 2 | 2023-09-02 08:20:33 | 22.0 | 68.0 |
| 2 | 4 | 2023-07-30 22:01:35 | 19.0 | 55.0 |
| 3 | 1 | 2023-08-30 13:06:23 | 18.0 | 63.0 |
| 4 | 1 | 2023-09-01 20:52:42 | 23.0 | 53.0 |

Na koju se prostoriju odnosi očitanje temperature i vlažnosti zraka s identifikacijom 1?

a. Sala za sastanke
b. Dvorana 1
c. Laboratorij 1
d. Ured 1
e. Hodnik 1

4. Neka je zadan sljedeći podatak:

{"id":1, "name":"Dvorana 1"}

U kojem je formatu zapisan ovaj podatak?

a. HTML
b. WEB
c. CSV
d. XLS
e. JSON

5. Neka je zadan sljedeći podatak:

<root>

i. <id>1</ id >
ii. < temperature>21.0</ temperature >

</root>

U kojem je formatu zapisan ovaj podatak?

b. HTML
c. WEB
d. CSV
e. XLS
f. JSON

6. Web API može, ali i ne mora primiti argumente.
   a. Točno
   b. Netočno

7. Web API može, ali i ne mora vratiti podatke mjestu poziva.
    a. Točno
    b. Netočno

8. HTTP naredba za dohvat podataka jest:
    a. POST
    b. GET
    c. PUT
    d. DELETE
    e. PATCH

9. HTTP naredba za kreiranje podataka jest:
    a. POST
    b. GET
    c. PUT
    d. DELETE
    e. PATCH

10. HTTP naredba za izmjenu podataka jest:
    a. POST
    b. GET
    c. PUT
    d. DELETE
    e. PATCH

11. HTTP naredba za brisanje podataka jest:
    a. POST
    b. GET
    c. PUT
    d. DELETE
    e. PATCH

### 2.11.4. Literatura i izvori


1. Build a REST API with Node.js, Express, and MySQL, https://blog.logrocket.com/build-rest-api-node-express-mysql/
2. db4free, https://db4free.net/


## 2.12. Izrada programa za slanje temperature i vlažnosti zraka s mikrokontrolera ESP32 preko poziva *web* aplikacijskoga programskog sučelja

Potrebno je izraditi program kojim će se očitanja temperature i vlažnosti zraka sa senzora DHT11 poslati u bazu podataka preko unaprijed pripremljenog *web* aplikacijskog sučelja iz prethodnoga radnog zadatka. Pozivanje *web* aplikacijskoga programskog sučelja obavlja se samo ako je došlo do promjene očitane vrijednosti. Također se kod svakog pozivanja *web* aplikacijskoga programskog sučelja u konzoli ispisuju podaci koji se šalju.

**Nakon usvajanja ove nastavne teme čitatelj će moći:**

1. Programirati mikroupravljač koristeći odabranu programsku potporu
2. Objasniti ulogu aplikacijsko programskih sučelja u sustavima interneta stvari
3. Objasniti ulogu usluga u oblaku za internet stvari
4. razlikovati različite modele usluga u oblaku za internet stvari
5. Izraditi programsku potporu za vlastiti sustav interneta stvari

## 2.12.1. Osnovni koncepti

*Web* aplikacijsko programsko sučelje omogućuje implementaciju distribuiranoga hardversko-softverskog sustava – sustava koji se sastoji od niza hardversko-softverskih podsustava koji funkcioniraju neovisno jedan o drugome, ali međusobno komuniciraju preko računalne mreže i time ostvaruju neku aplikativnu zadaću. Primjera radi sustav IoT može se sastojati od niza mikrokontrolera koji očitanja sa svojih senzora dostavljaju u centraliziranu bazu podataka preko Web API-ja. Jednako tako, korisnik preko *web* aplikacije može određenom mikrokontroleru dostaviti konfiguraciju ili naredbe na temelju kojih on mijenja svoje ponašanje (npr. pali ili gasi alarm s obzirom na dostavljene granične vrijednosti očitanja).

Baza podataka i Web API nalaze se na nekom poslužitelju unutar računalne mreže. Ti se poslužitelji mogu nalaziti u lokalnoj mreži, ali i na internetu (npr. db4free je poslužitelj baze podataka koji se nalazi na internetu). Poslužitelji koji se nalaze na internetu čine oblak (engl. *cloud*), a usluge koje oni nude zovu se usluge u oblaku (engl. *cloud service*). Tako je db4free usluga u oblaku koja omogućuje upravljanje bazom podataka.

Usluge u oblaku mogu se podijeliti na sljedeće:

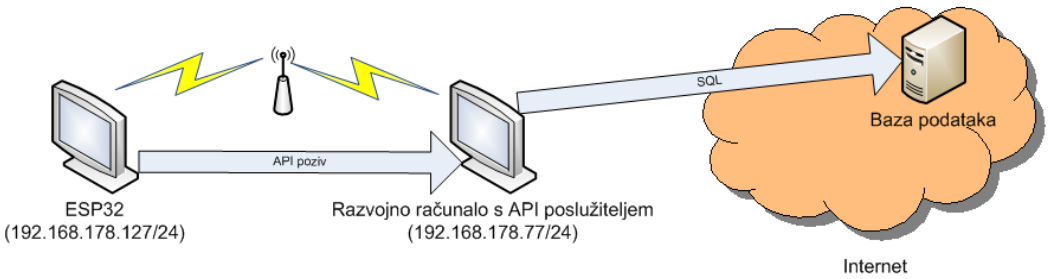
- IaaS (engl. Infrastructure as a Service) – usluga u oblaku koja korisniku omogućuje upravljanje virtualnom infrastrukturom (mrežom, računalima, spremištima podataka, sistemskim softverom itd.) – npr. AWS EC2, Google Compute Engine (GCE), Digital Ocean, Microsoft Azure;
- PaaS (engl. *Platform as a Service*) – usluga u oblaku koja korisniku omogućuje integralnu uporabu hardvera i softvera, najčešće u razvoju aplikacija (npr. Windows Azure, Heroku, AWS Elastic Beanstalk, Google App Engine);
- SaaS (engl. *Software as a Service*) – usluga u oblaku koja korisniku omogućuje uporabu nekog aplikativnog softvera (npr. Dropbox, Google Workspace, Slack itd.).

Procedura pozivanja Web API-ja iz mikrokontrolera ESP32 jest sljedeća:

1. konfigurirati http klijenta – naredba esp_http_client_init
2. postaviti mod, odnosno vrstu HTTP naredbe – naredba esp_http_client_set_method
3. postaviti zaglavlje poziva – naredba esp_http_client_set_header
4. postaviti podatke koji se šalju – npr. za post je naredba esp_http_client_set_post_field
5. izvesti HTTP poziv – naredba esp_http_client_perform
6. zatvoriti vezu – naredba esp_http_client_cleanup

## 2.12.2. Rješenje radnog zadatka

Topologija računalne mreže ovog radnog zadatka sastoji se od mikrokontrolera i razvojnog računala koji su umreženi preko pristupne točke. Razvojno računalo ima pristup internetu. U oblaku (na internetu) nalazi se baza podataka (usluga https://db4free.net/) u koju će se spremati očitanja.




*Slika 2.12.1 Topologija računalne mreže i komunikacija*

Potrebno je izraditi ožičenje senzora DHT11 i mikrokontrolera. U softveru Visual Studio Code treba izraditi novi projekt pod nazivom poziv_web_api. U njegovu datoteku main.c treba ubaciti programski kôd za spajanje mikrokontrolera na Wi-Fi mrežu te programski kôd za očitanje senzora DHT11 – perioda očitanja neka bude pet sekundi. U dio programskog koda za očitanje senzora DHT11 potrebno je dodati poziv *web* aplikacijskoga programskog sučelja kojemu se u formatu JSON šalju temperatura i vlažnost zraka.
Cjelokupan programski kôd koji treba upisati u main.c glasi:

```
#include <stdio.h>
#include <string.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp_wifi.h"
#include "nvs_flash.h"
#include "dht.h"
#include "esp_http_client.h"

#define WIFI_SSID "Naziv pristupne točke"
#define WIFI_PASS "Lozinka za povezivanje na pristupnu točku"

float old_temperature = 0;
float old_humidity = 0;

unsigned char is_IP_set = 0;

static void event_handler(void *arg, esp_event_base_t event_base,
                          int32_t event_id, void *event_data)
{
    if (event_base == WIFI_EVENT && event_id == WIFI_EVENT_STA_START)
    {
        esp_wifi_connect();
    }
    else if (event_base == IP_EVENT && event_id == IP_EVENT_STA_GOT_IP)
    {
        is_IP_set = 1;
    }
}

void wifi_init_sta(void)
{
    esp_netif_init();
```

```
    esp_event_loop_create_default();
    esp_netif_create_default_wifi_sta();

    wifi_init_config_t cfg = WIFI_INIT_CONFIG_DEFAULT();
    esp_wifi_init(&cfg);

    esp_event_handler_instance_t instance_any_id;
    esp_event_handler_instance_t instance_got_ip;
    esp_event_handler_instance_register(WIFI_EVENT,
                                        ESP_EVENT_ANY_ID,
                                        &event_handler,
                                        NULL,
                                        &instance_any_id);
    esp_event_handler_instance_register(IP_EVENT,
                                        IP_EVENT_STA_GOT_IP,
                                        &event_handler,
                                        NULL,
                                        &instance_got_ip);

    wifi_config_t wifi_config = {
        .sta = {
            .ssid = WIFI_SSID,
            .password = WIFI_PASS,
        },
    };
    esp_wifi_set_config(WIFI_IF_STA, &wifi_config);
    esp_wifi_start();
}

void DHT_reader_task(void *pvParameter)
{
    esp_http_client_config_t config = {
        .host = "192.168.178.128",
        .port = 3000,
        .path = "/reading",
    };

    esp_http_client_handle_t client;

    char post_data[100];
    float temperature, humidity;

    while (1)
    {
        if (is_IP_set)
        {
            if (dht_read_float_data(DHT_TYPE_DHT11, GPIO_NUM_0, &humidity,
&temperature) == ESP_OK)
            {
```

```
                if (old_humidity != humidity || old_temperature != temperature)
                {
                    printf("Temperatura zraka: %.1fC\n", temperature);
                    printf("Vlaznost zraka: %.1f%%\n", humidity);
                    old_humidity = humidity;
                    old_temperature = temperature;

                    sprintf(post_data, "{\"temperature\": %.1f, \"humidity\":
%.1f}", temperature, humidity);

                    client = esp_http_client_init(&config);
                    esp_http_client_set_method(client, HTTP_METHOD_POST);
                    esp_http_client_set_header(client, "Content-Type",
"application/json");
                    esp_http_client_set_post_field(client, post_data,
strlen(post_data));
                    esp_http_client_perform(client);
                    esp_http_client_cleanup(client);
                };
            }
            else
            {
                printf("Senzor nije moguce ocitati\n");
            }
        }

        vTaskDelay(5000 / portTICK_PERIOD_MS);
    }
}

void app_main(void)
{
    // Initialize NVS
    nvs_flash_init();

    wifi_init_sta();
    xTaskCreate(DHT_reader_task, "DHT_reader_task", 5000, NULL, 5, NULL);
}
```

U glavnom dijelu programa inicijalizira se *flash* memorija mikrokontrolera.
Potom se poziva funkcija wifi_init_sta unutar koje se obavlja povezivanje mikrokontrolera na pristupnu točku. Naposljetku se izrađuje zadaća u kojoj se poziva funkcija DHT_reader_task.
Način povezivanja mikrokontrolera na pristupnu točku opisan je u prethodnom zadatku. Jedina izmjena jest ta što je dodana statusna varijabla is_IP_set koja se postavlja na vrijednost 1 kada mikrokontroler dobije IP adresu. Ta se statusna varijabla kasnije koristi kao uvjet za pokretanje očitanja senzora u funkciji DHT_reader_task:

```
if (is_IP_set)
```

U funkciji DHT_reader_task prvo se definira konfiguracija za poziv *web* aplikacijskoga programskog sučelja i varijabla koja predstavlja klijenta koji je poziva.

```
esp_http_client_config_t config = {
        .host = "192.168.178.128",
        .port = 3000,
        .path = "/reading",
    };
esp_http_client_handle_t client;
```

Ono što se postavlja jest IP adresa računala na kojem je pokrenut poslužitelj za *web* aplikacijsko programsko sučelje (vidjeti prethodni zadatak), port na kojem poslužitelj osluškuje zahtjeve i ruta koja se poziva.
Budući da se *web* aplikacijskom programskom sučelju podaci šalju u formatu JSON ({“polje”: vrijednost}), to je kod svakog slanja podataka (očitane temperature i vlažnosti zraka) potrebno pretvoriti u JSON format – {“temperature”: value, “humidity”: value}. To se radi sljedećom linijom programskog koda:

```
sprintf(post_data, "{\"temperature\": %.1f, \"humidity\": %.1f}", temperature,
humidity);
```

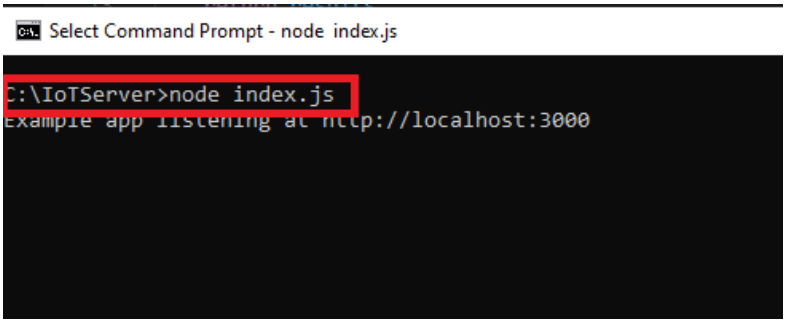
Sam poziv *web* aplikacijskoga programskog sučelja i slanje podataka obavljaju se sljedećim dijelom programa:

```
client = esp_http_client_init(&config);
                    esp_http_client_set_method(client, HTTP_METHOD_POST);
                    esp_http_client_set_header(client, "Content-Type",
"application/json");
                    esp_http_client_set_post_field(client, post_data,
strlen(post_data));
                    esp_http_client_perform(client);
                    esp_http_client_cleanup(client);
```

Procedura je da se prvo inicijalizira klijent za pozivanje *web* aplikacijskoga programskog sučelja, potom se specificiraju metoda poziva (POST) i zaglavlje poziva (Content-Type: application/json) te se postavljaju podaci koji se šalju (esp_http_client_set_post_field). Samo izvođenje poziva radi se preko funkcije esp_http_client_perform. Sve završava zatvaranjem konekcije prema poslužitelju funkcijom esp_http_client_cleanup.
Treba primijetiti da se *web* aplikacijsko programsko sučelje poziva samo onda ako se novo očitanje razlikuje od prethodnog.

```
if (old_humidity != humidity || old_temperature != temperature)
```

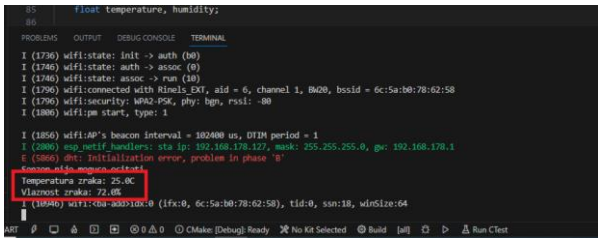
Prije kompajliranja programa i njegova slanja na mikrokontroler treba se pokrenuti poslužitelj s *web* aplikacijskim programskim sučeljem. Pokretanje se može odraditi preko naredbenog retka u kojem se unutar mape u kojoj se nalazi poslužitelj pokrene naredba node index.js.




*Slika 2.12.2 Pokretanje poslužitelja s web aplikacijskim programskim sučeljem*

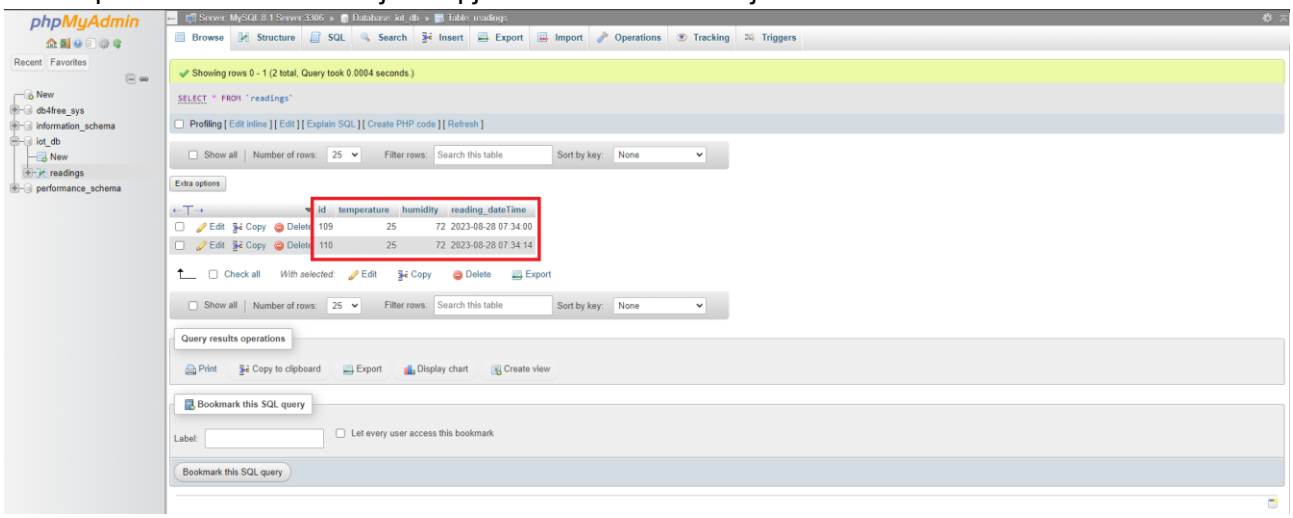
Nakon kompajliranja, slanja programa na mikrokontroler i pokretanja monitoringa vidi se rezultat

izvođenja programa.




*Slika 2.12.3 Pokrenuti program za slanje temperature i vlažnosti zraka na web aplikacijsko programsko sučelje*

U bazi podataka može se vidjeti uspješno ubačeno očitanje.




*Slika 2.12.4 Ubačena očitanja u bazu podataka*

## 2.12.3. Pitanja i zadaci

1. U program dodati LED indikatore – jedan koji se uključiti kada mikrokontroler dobije IP adresu i drugi koji se uključi kada mikrokontroler šalje podatke.
2. Program doraditi tako da se poziv *web* aplikacijskoga programskog sučelja dogodi tek ako temperatura i/ili vlažnost zraka budu u nekom unaprijed definiranom intervalu.
3. Program doraditi tako da se poziv *web* aplikacijskoga programskog sučelja dogodi tek kada korisnik pritisne tipkalo.

## 2.12.4. Literatura i izvori


1. HTTP Request Example, https://github.com/espressif/esp-idf/blob/52bca70b1a73248c9a66095d2cfac8c6ebb1f263/examples/protocols/http_request
2. IoT considerations — cloud services — IaaS, PaaS, SaaS, build your own, https://medium.com/lattice-research/iot-considerations-server-side-iaas-paas-saas-1f55afc03185
3. IaaS vs. PaaS vs. SaaS, https://www.redhat.com/en/topics/cloud-computing/iaas-vs-paas-vs-saas?sc_cid=7013a000002pgRcAAI&gclid=Cj0KCQjwusunBhCYARIsAFBsUP83DlQbxdVI1A-0PlmnT-t5Zd1O3HKWUfsb_LiMeep8WIuTwQavl8AaAnMnEALw_wcB&gclsrc=aw.ds

izv. prof. dr. sc. Alen Jakupović, prof. struč. stud.

**INTERNET STVARI**

ISBN 978-953-50797-0-5

PROSPEKT