Izrada hardverske podrške za sustavnu evidenciju nazočnosti studenata na nastavnim aktivnostima

Cindrić, Martin

VELEUČILIŠTE U BJELOVARU
STRUČNI PRIJEDIPLOMSKI STUDIJ RAČUNARSTVO

Izrada hardverske podrške za sustavnu evidenciju nazočnosti studenata na nastavnim aktivnostima

Završni rad br. 04/RAČ/2023

Martin Cindrić

Bjelovar, listopad 2023.

obrazac ZR - 001

Veleučilište u Bjelovaru

Trg E. Kvaternika 4, Bjelovar

1. DEFINIRANJE TEME ZAVRŠNOG RADA I POVJERENSTVA

Student: **Martin Cindrić** JMBAG: **0314022998**

Naslov rada (tema): **Izrada hardverske podrške za sustavnu evidenciju nazočnosti studenata na nastavnim aktivnostima**

Područje: **Tehničke znanosti** Polje: **Računarstvo**

Grana: **Informacijski sustavi**

Mentor: **Krunoslav Husak, dipl. ing. rač.** zvanje: **predavač**

Članovi Povjerenstva za ocjenjivanje i obranu završnog rada:

1. **dr. sc. Zoran Vrhovski, predsjednik**
2. **Krunoslav Husak, dipl. ing. rač., mentor**
3. **Ivan Sekovanić, mag. ing. inf. et comm. techn., član**

2. ZADATAK ZAVRŠNOG RADA BROJ: 04/RAČ/2023

U sklopu završnog rada potrebno je:

1. Analizirati i opisati Arduino razvojnu okolinu i usporediti ju s drugim razvojnim okolinama.
2. Analizirati i opisati ESP32 mikrokontroler i usporediti ga s drugim sličnim mikrokontrolerima.
3. Predložiti i opisati komponente korištene za izradu hardverske podrške za sustavnu evidenciju nazočnosti studenata na nastavnim aktivnostima.
4. Izraditi hardversko rješenje za evidenciju studenata na nastavnim aktivnostima.
5. Izraditi programsko rješenje za evidenciju studenata na nastavnim aktivnostima.
6. Predložiti i implementirati komunikaciju s ostalim dijelovima sustava za evidenciju nazočnosti studenata na nastavnim aktivnostima.

Datum: 09.06.2023. godine

Mentor: **Krunoslav Husak, dipl. ing. rač.**

Zahvala

Zahvaljujem se mentoru Krunoslavu Husaku, dipl. ing. rač. na vodstvu i potpori tijekom cijelog studija i pisanja završnog rada, uključujući i ostale djelatnike Veleučilišta u Bjelovaru. Naposljetku, želio bih se zahvaliti svojoj obitelji na nevjerojatnoj podršci koju su mi pružili tijekom cijelog mog školovanja.

Sadržaj

1. Uvod

Živimo u vremenu u kojemu tehnologija i inovacije kroje put prema budućnosti, pa tako i u obrazovanju. Tema ovog završnog rada „Izrada hardverske podrške za sustavnu evidenciju nazočnosti studenata na nastavnim aktivnostima“ pomoći će pri unapređenju standardne metode evidencije prisutnosti na predavanjima, laboratorijskim vježbama i ostalim oblicima nastave. Često su takve standardne metode bilježenja aktivnosti zahtjevne i podložne pogreškama koje utječu na samu evidenciju studenata., te ih je potrebno zamijeniti novim, modernijim, preciznijim i automatiziranim IoT (engl. *Internet of Things*) sustavom. IoT sustav odnosi se na mrežu međusobno povezanih uređaja koji prikupljaju i prenose informacije putem interneta. U ovom projektu IoT tehnologija omogućuje automatizaciju i digitalizaciju bilježenja evidencije studenata na nastavnim aktivnostima. Ključne komponente ovog sustava su RFID (engl. *Radio-Frequency Identification*) čitač koji omogućuje točnu identifikaciju studenta i razvojno okruženje za obradu i slanje tih podataka u bazu podataka. Studenti prilikom dolaska na neke od nastavnih aktivnosti, prislanjaju svoju studentsku ispravu na RFID čitač. Pomoću čitača se dohvaćaju sve potrebne informacije koje su sadržane u čipu studentske isprave, a to je UUID (engl. *Universally Unique Identifier*).

Integracija s .NET aplikacijom omogućuje profesorima i administrativnom osoblju lagan pristup, analizu i vizualizaciju podataka o prisutnosti svakog studenta u realnom vremenu. IoT pristupom, izbjegavamo ljudsku pogrešku koja se uvijek može dogoditi čime studenti ne mogu više izbjegavati svoje studentske obveze. Osim toga, automatizacijom se doprinosi održivosti, smanjujući se potreba za papirom čime se manje zagađuje okoliš.

Cilj ovog završnog rada je istražiti, analizirati i izraditi hardverskog rješenje za evidenciju prisutnosti na nastavnim aktivnostima koje će biti povezano s .NET aplikacijom za lakšu evidenciju i vizualizaciju podataka. Za izradu hardverskog rješenja korištene su tri komponente: ESP32 razvojno okruženje, RFID čitač i OLED (engl. *Organic Light Emitting Diodes*) zaslon za prikaz podataka. Upotrebom Arduino razvojnog programskog okruženja, isprogramiran je IoT uređaji koji ima mogućnost čitanja studentske isprave te slanje podataka preko interneta koji se spremaju u bazi podataka i prikazuju na .NET aplikaciji.

U nastavku rada biti će opisan detaljniji proces razvoja hardverskog rješenja, a zatim i programskog rješenja. Rad je podijeljen na pet poglavlja. Drugo poglavlje opisuje Arduino, EspressIf i MicroPython platforme, te njihove prednosti i nedostatke. Treće poglavlje opisuje mikrokontrolere. U četvrtom poglavlju se opisuje hardversko rješenje i uloga svake

komponente zasebno. Peto poglavlje opisuje programsko rješenje. Posljednja poglavlja sadržavaju zaključak, sažetak, literaturu i oznake.

2. Programska razvojna okruženja

U današnjem tehnološkom dobu, gdje je tehnologija postala sastavni i glavni dio svakodnevnog života, sve je više ljudi koji su zainteresirani za razumijevanje i stvaranje vlastitih uređaja i projekata.

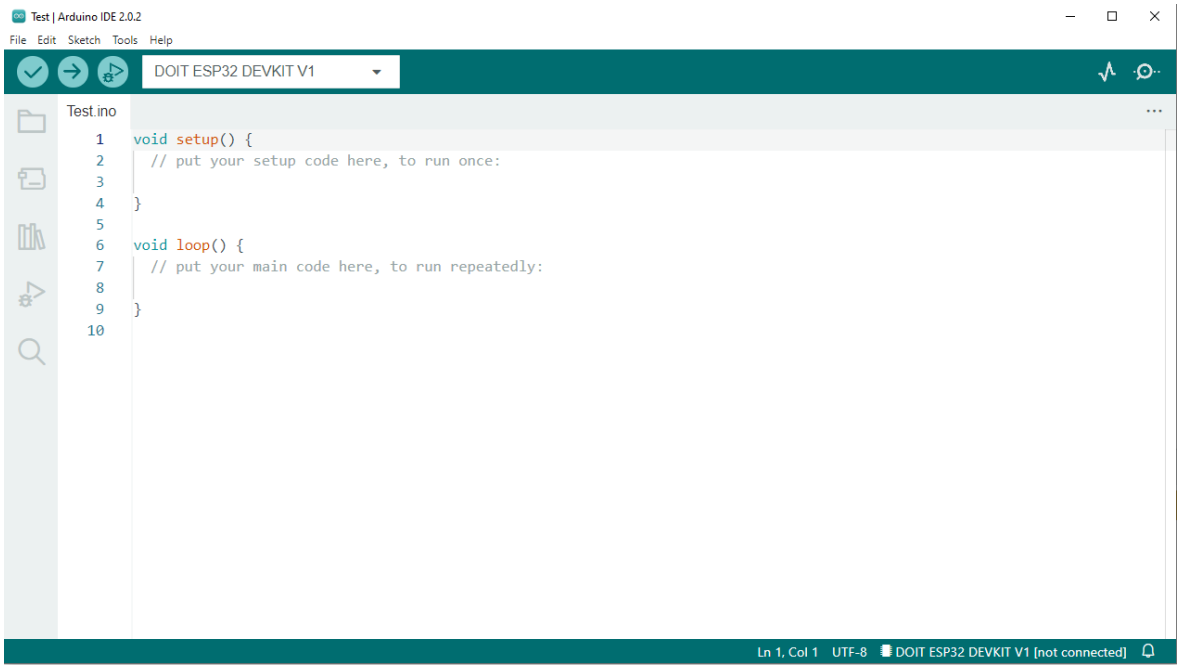
2.1 Arduino programsko razvojno okruženje

U svijetu elektronike i mikrokontrolera, Arduino programsko razvojno okruženje predstavlja jednu od najpopularnijih okruženja za stvaranje kreativnih inovacija i projekata. Ova moćna platforma omogućava entuzijastima i programerima da pretvore svoje ideje u stvarnost putem jednostavnog i svestranog razvojnog okruženja. Arduino programsko razvojno okruženje je besplatno razvojno okruženje tvrtke SmartProjects koje omogućuje korisnicima da pišu i prenose programski kod na Arduino platforme. Ovo razvojno okruženje uključuje podršku za jednostavnu sintaksu C/ C++ programskih jezika, programiranja, kompajliranja (engl. *Compile*) i prenošenja koda na Arduino platforme. Zbog jednostavnog i intuitivnog sučelja, čak i korisnici bez prethodnog znanja i iskustva u programiranju mogu vrlo brzo započeti s razvijanjem vlastitih programskih rješenja. Programi napisani pomoću Arduino programsko razvojno okruženje nazivaju se skice. Arduino programsko razvojno okruženje je dostupan na različitim operacijskim sustavima kao što su Windows, macOS i Linux. Na slici 2.1 prikazan je logo Arduino platforme [1].

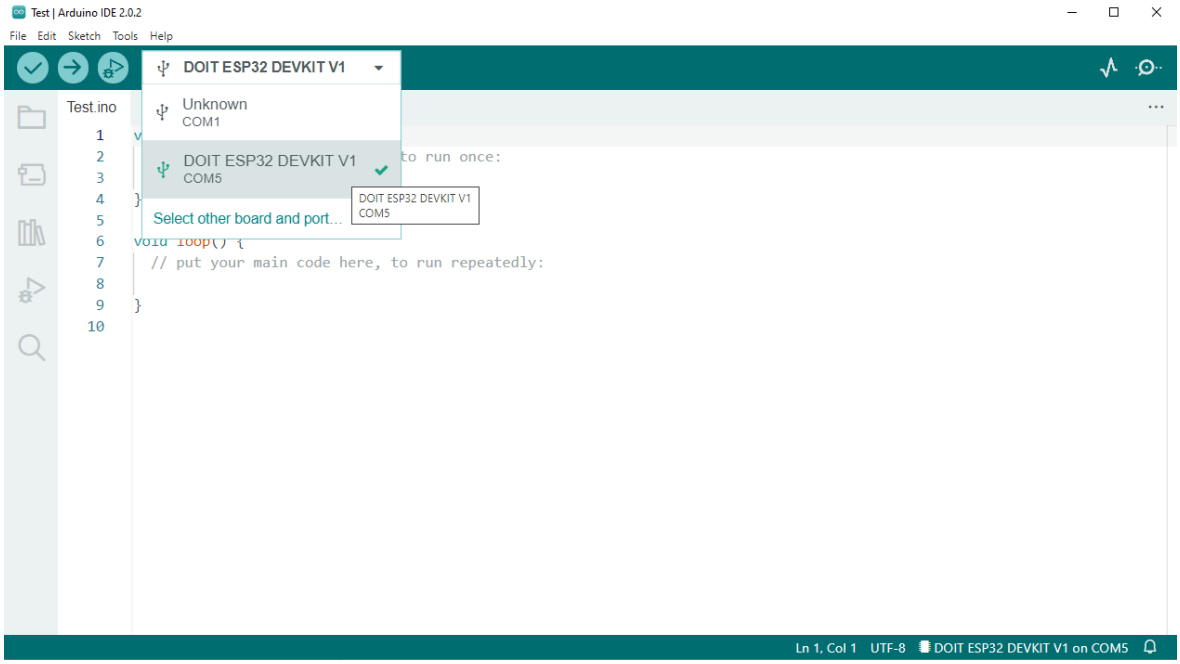
Slika 2.1: Logo Arduino platforme

2.1.1 **Pisanje skica**

Kao što je navedeno u poglavlju 2.1 svaki napisani programski kod pomoću Arduino programskog razvojnog okruženja naziva se skica. Svaka skica sprema se s ekstenzijom datoteke .ino. Za prikazivanje pogrešaka, izlaznih poruka i pogrešaka zaslužena je konzola. Također prilikom pisanja programa i odabira ulaznih i izlaznih pinova potrebno je odabrati i ispravni serijski priključak i razvojno okruženje tzv. pločica koji ima podršku Arduino platforme. Na slici 2.2 i 2.3 mogu se vidjeti prikaz sučelja Arduino programskog razvojnog okruženja i padajući izbornik za odabir razvojnog okruženja.



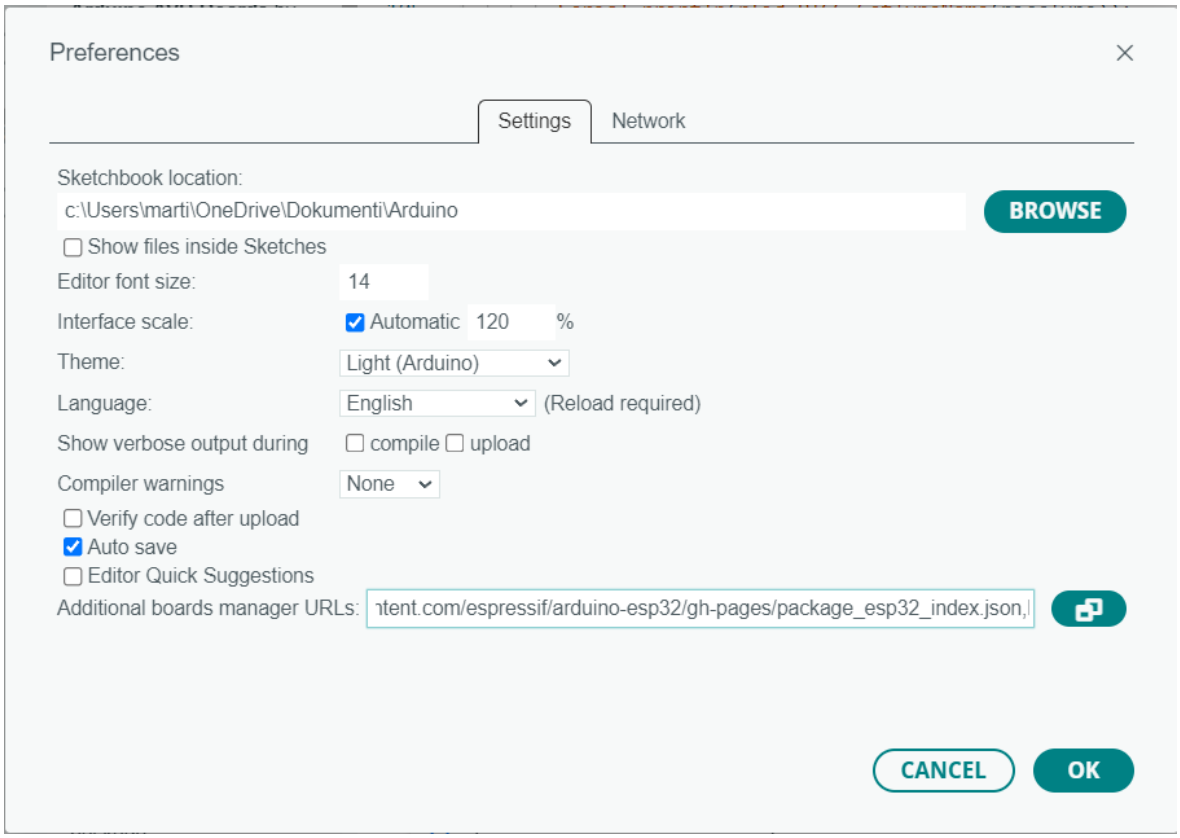
Slika 2.2: Prikaz sučelja Arduino programskog razvojnog okruženja



Slika 2.3: Prikaz padajućeg izbornika za odabir mikrokontrolera

2.1.2 **Dodatni upravitelj pločicama**

Dodatni upravitelj pločicama (engl. *Additional Boards Manager*) omogućuje korisnicima dodavanje podrške za platforme koji nisu na općenitom popisu Arduino programskog razvojnog okruženje. Ova proširena funkcionalnost omogućuje korisnicima rad s različitim razvojnim okruženjima s kojom se postigla fleksibilnost ovog razvojnog okruženja. Unatoč pločicama proizvedenih od strane tvrtke Arduino moguće je koristiti i razvojna okruženja drugih proizvođača poput Adafruit Feather, STM32 i ESP32 koji je korišten prilikom izrade ovog završnog rada. Osim ugrađenih primjera (engl. *Built-in Examples*), pristup ovim dodatnim resursima omogućuje i pristup dodatnim bibliotekama i primjerima koji su prilagođeni određenoj pločici. Sve što je potrebno je zalijepiti link u okvir naziva *Additional Boards Manager URLs* i time je omogućeno pretraživanje i instalacija podrške za nova razvojna okruženja. Na slici 2.4 može se vidjeti kako je to napravljeno za ovaj završni rad.



Slika 2.4: Dodatni upravitelj pločicama

2.1.3 **Prednosti i nedostatci**

Arduino programsko razvojno okruženje jednostavno je za početnike i entuzijaste koji se tek upoznavaju s elektronikom i programiranjem mikrokontrolera. Podržava objektno orijentirano programiranje - OOP (engl. *Object Oriented Programming*) što će poboljšati performanse samog izvođenja koda. Projekti se mogu jako brzo postavljati jer postoje unaprijed konfigurirane postavke i primjeri (engl. *Examples*) za svaku od pločica tj. razvojnih okruženja koji se koriste. Izrada prototipa je jako jednostavna. Uz nekoliko pretraga i klikova u pretraživaču *(*engl. *Browser)* moguće je pronaći električne sheme i primjere programskih kodova. Postoji veliki broj Arduino razvojnih okruženja koji su integrirani i dostupni instalacijom Arduino programskog razvojnog okruženja. Koristi intuitivno i jednostavno grafičko sučelje. Prisutnost raširene zajednice koja se susrela s većinom zadataka što doprinosi brzom razvoju i prototipiranju. Nedostatci se osjete u složenim zadacima koji traže dublje razumijevanje mikrokontrolera. Za takve projekte, Raspberry PI Pico i Python mogu biti korisni [2].

2.2 Espressif IDF

EspressIf IDF (engl. *IoT Development Framework*) omogućuje razvoj, programiranje i upravljanje aplikacijama na Espressif čipovima, posebno na popularnom ESP32 sustava na čipu. Ovaj razvojni okvir opremljen je bogatim skupim biblioteka i alata koji omogućuju razvoj IoT aplikacija brzim i lakšim. U ovom poglavlju bit će opisane funkcionalnosti Espressif IDF – a, te prednosti i mane.

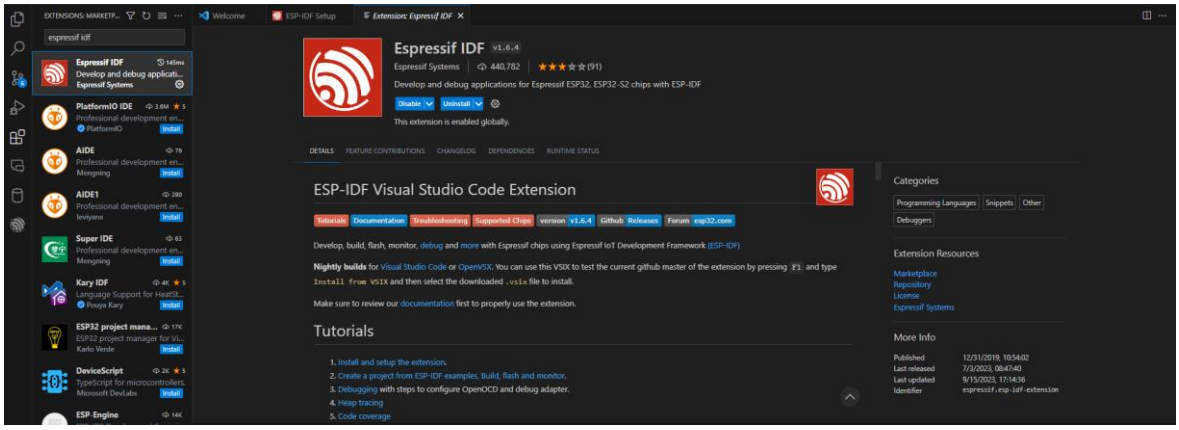
2.2.1 Osnovne karakteristike

Espressif IDF je službeno razvojni okvir za ESP32 i ESP32-S serije koje je razvila tvrtka Espressif. Espressif IDF ima integraciju s FreeRTOS (engl. *Free Real Time Operating System*) što omogućava izvođenje više zadaća koje se mogu paralelno izvršavati na mikrokontroleru. Naravno, zbog toga ESP32 serija sustava na čipu koristi više jezgreni čip kako bi *multitasking* bio moguć. Također, ovaj IDF sadrži alate za lakše praćenje izvođenja koda kako i alate za analizu potrošnje energije i otklanjanje pogrešaka (engl. *Debugging*). Kao programske jezike koristi C/C++ što omogućuje preciznu kontrolu nad hardverom. Fleksibilnost se upravo postiže tim jezicima, C se koristi za optimizaciju samog programskog koda dok se C++ koristi za OOP. Kako je u IoT svijetu vrlo važna sigurnost podataka ovaj IDF pruža hardverski ubrzanu kriptografiju i sigurnosne protokole koji osiguravaju da podaci ostanu zaštićeni. Vrlo bitno je napomenuti da sadrži sve upravljačke programe i biblioteke integrirane u sebi specifično za ESP32.
Iako Espressif IDF nije samo samostalni okvir može se instalirati kao ekstenzija za Visual Studio Code. Ova mogućnost pruža svim programerima da iskoriste prednosti ovog IDF-a unutar poznatog razvojnog okruženja što čini razvoj IoT aplikacija još učinkovitijima [3].

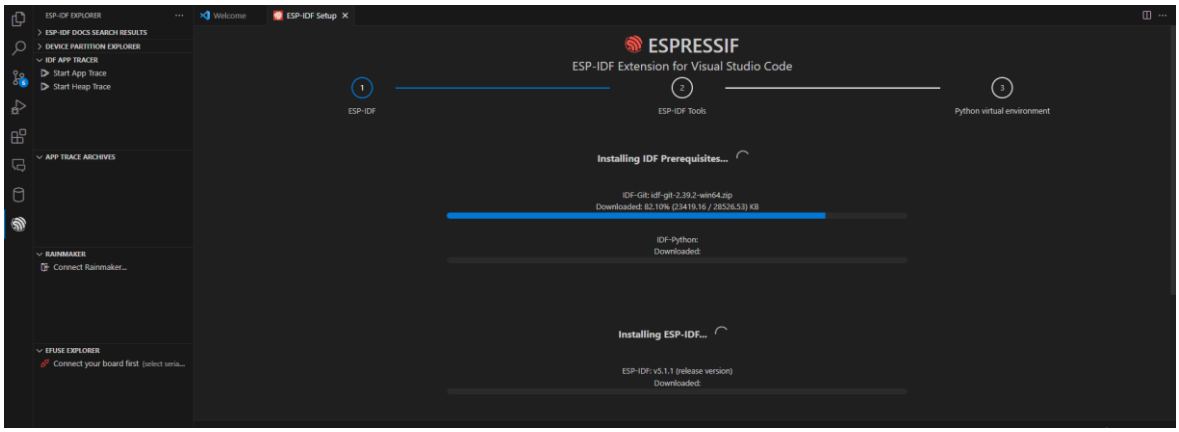
Slika 2.5: Logo tvrtke Espressif

2.2.2 **Prednosti i nedostatci**

Espressif IDF sadrži podršku za RTOS (engl. *Real Time Operating System*) što omogućuje *multitasking*. Namijenjen je za složenije projekte. Zbog veće kontrole nad hardverom programski kod rezultira optimiziranije i brže. Koristi *secure boot* i *flash* šifriranje što su vrlo bitne sigurnosne značajke. Kao napredne značajke za komunikaciju pruža podršku za WiFi i Bluetooth. Uz podršku za bežični komunikaciju nudi i podršku za bežično ažuriranje ugrađenog softvera (engl. *Firmware*). Zbog svoje kompleksnosti zahtjevniji je za početnike i entuzijaste. Za kompiliranje i razvoj zahtijeva više računalnih resursa, a za složenije projekte postavljanje je zahtjevnije. [4] Slike 2.6 i 2.7 prikazuju Espressif IDF dodatak unutar Visual Studio Code–a.



Slika 2.6: Prikaz Espressif IDF dodatka u Visual Studio Code-u



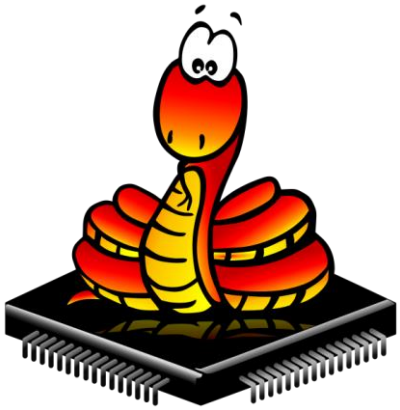
Slika 2.7: Prikaz instalacije Espressif IDF dodatka

2.3 MicroPython

MicroPython predstavlja jednostavnu i kompaktnu implementaciju Python programskog jezika za optimiziranu upotrebu na mikrokontrolerima. Ovo razvojno okruženje omogućuje brz i jednostavan razvoj aplikacija. U ovom poglavlju opisane su osnovne karakteristike te prednosti i mogući izazovi koji se pojavljuju prilikom korištenja MicroPythona u kontekstu razvoja IoT rješenja. Na slici 2.8 može se vidjeti logo MicroPython–a.

2.3.1 Osnovne karakteristike

MicroPython je verzija Pythona koja je optimizirana za rad s mikrokontrolerima. Za razliku od Pythona MicroPython je verzija koja radi sa sustavima ograničenih resursa. Idealan je za programere koji su već upoznati s Python razvojnim okruženjem ili su tek krenuli s programiranjem. Ovo razvojno okruženje čini programiranje bržim i jednostavniji zbog svoje jednostavnosti. MicroPython obogaćen je s REPL (engl. *Read-Eval-Print Loop*) okruženjem što omogućuje izvršavanje Python naredbi u stvarnom vremenu na mikrokontroleru. Posjeduje veliki broj ugrađenih biblioteka za podršku u mnogim zadacima. Uz MicroPython programski kod moguće je proširiti ga s funkcijama nižih programskih jezika kao što su C/C++ za brže izvršavanje ako je to potrebno. Kao i kod drugih razvojnih okruženja MicroPython može upravljati priključcima mikrokontrolera na koje mogu biti spojeni, LCD (engl. *Liquid Crystal Display*) zasloni, servo motori, I2C (engl. *Inter-Integrated Circuit*) komunikacijska sučelja i slično [5], [6].



Slika 2.8: Logo MicroPythona

2.3.2 **Prednosti i nedostatci**

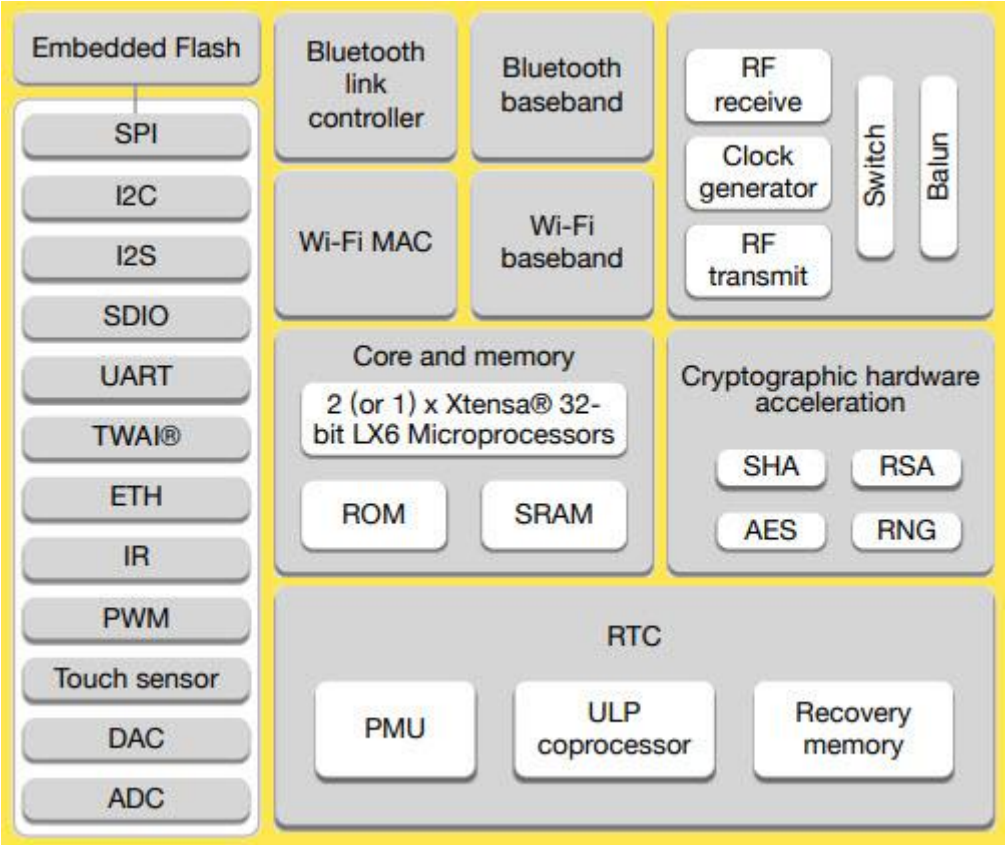
MicroPython nudi brojne prednosti, uključujući besplatno korištenje, što ga čini pristupačnim za širok spektar korisnika. Karakterizira ga jednostavnost i brzina razvoja, omogućujući brzo kreiranje aplikacija s poznavanjem osnovnog Python koda. Ovaj programski jezik dostupan je za veliku većinu mikrokontrolera, što proširuje njegovu primjenjivost. REPL dodatno olakšava proces, omogućujući interaktivno testiranje i otklanjanje pogrešaka (engl. *Debugging*) programskog koda. Međutim, postoje i mane koje prate MicroPython. U usporedbi s nižim jezicima, poput C/C++, MicroPython je sporiji i manje optimiziran, što rezultira većom potrošnjom energije. Također, moguće su nekompatibilnosti između Python i MicroPython biblioteka, što je posljedica ograničenih resursa MicroPythona.

3. Mikrokontroleri

Mikrokontroleri predstavljaju „srce“ mnogih modernih elektroničkih i IoT uređaja, omogućujući im kontrolu i automatizaciju širokog spektra. Ovi kompaktni uređaju su efikasna i prilagodljiva rješenja za različita područja primjene. U ovom poglavlju detaljno su opisani mikrokontroleri, opisane su njihove ključne karakteristike, arhitektura i funkcionalnosti, posebno u kontekstu IoT i ugrađenih sustava.

3.1 ESP32 razvojno okruženje

ESP32 potiče iz serije jeftinih i energetski učinkovitih sustava na čipu (engl. *System on Chip*) proizveden od strane tvrtke pod imenom Espressif Systems. Zbog svoje energetske učinkovitosti, niske cijene i performansi postao je jedan od najpoznatijih i najpristupačnijih sustava na čipu u svijetu IoT-a. Isto kao i njegov prethodnik ESP8266 dolazi je s WiFi sučeljem i obogaćen je za Bluetooth (može se koristiti i BLE (engl. *Bluetooth Low Energy*)) sučeljem koji omogućuju bežičnu komunikaciju s ostalim uređajima. Nebitno je radi li se o kućanskih uređajima ili industrijskim senzorima, ESP32 može se lako integrirati i komunicirati s drugim uređajima. Kao glavnu i kontrolnu jedinicu koristi dvojezgreni Xtensa 32 – bit LX6 mikroprocesor što mu omogućuje *multitasking* operacije i veću fleksibilnost. Zahvaljujući velikom broju GPIO (engl. *General Purpose Input/Output*) pinova koji se mogu konfigurirati za različite namjene uključujući ADC (engl. A*nalog to Digital Converter*) i DAC (engl. *Digital to Analog Converter*) pretvornike, senzor dodira, PWM (engl. *Pulse Width Modulation*), I2C komunikacija, I2S (engl. *Inter - Integrated Sound*) komunikacija, UART (engl. *Universal Asynchronous Receiver/Transmitter*) komunikacija, SPI (engl. *Serial Peripheral Interface*) komunikacije i sl. Ova fleksibilnost omogućuje korisnicima da povežu različite motore, senzore, LCD zaslone, čitače kartica itd. ESP32 ima vrlo nisku potrošnju električne energije uspoređujući ga s drugim razvojnim okruženjima i sustavima na čipu jer pruža različite mogućnosti upravljanja njome (BLE, *deep sleep* i sl.) i zbog toga je odličan u razvijanju raznih IoT projekata koji imaju zadatak dugoročnog rada bez punjenja. Velika prednost ovog sustava na čipu je mogućnost programiranja iz različitih razvojnih okruženja kao što su Espressif IDF, Arduino IDE i MicroPython. Sigurnost je od presudne važnosti zbog čega je ovaj sustav na čipu dizajniran nudeći hardversko ubrzanje za različite kriptografske algoritme, sigurno pokretanje (engl. *Secure boot*) i enkripciju *flash* memorije [7].



Slika 3.1: Prikaz ESP32 arhitekture [7]

3.2 Funkcionalnost i rad

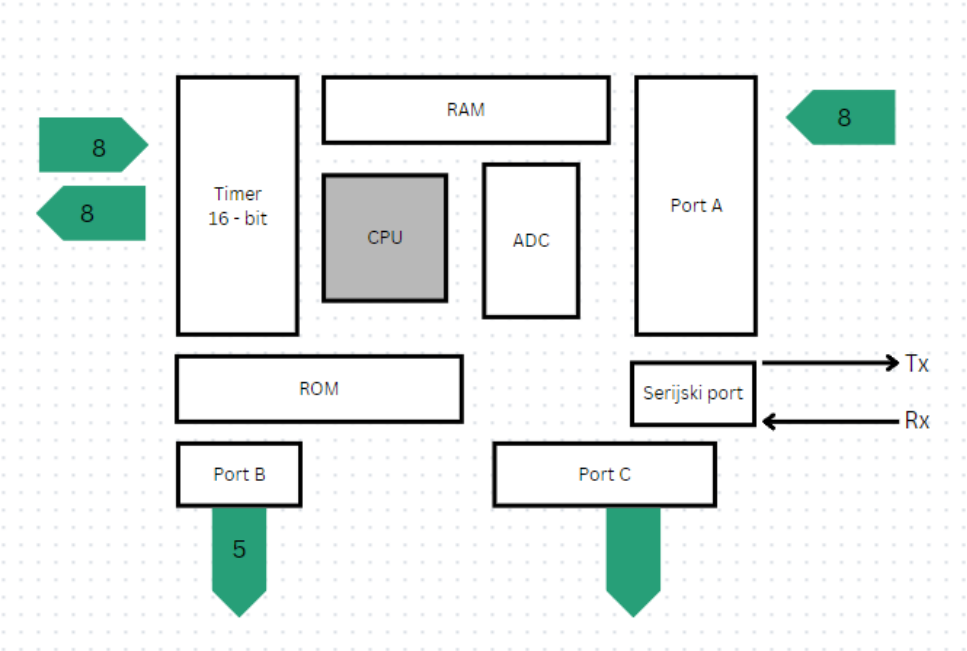
U svijetu elektronike, mikrokontroleri imaju ključnu ulogu kao mozak mnogih uređaja. Od pametnih telefona pa sve do kućanskih aparata i igračaka, mikrokontroleri omogućuju interakciju s korisnicima i „napredne“ funkcije. Poput komunikacije s ostalim uređajima i slično. Mikrokontroleri su elektronički uređaji koji obavljaju specifične zadaće u raznolikim slučajevima i namjenama ugrađenog sustava. Uobičajen mikrokontroler uključuje procesor, memoriju i ulazno/izlazne jedinice. Uglavnom su dizajnirani da upravljaju dijelovima elektroničkih uređaja kao što su mobiteli, pametni satovi, kućni uređaji, dijelovi proizvodnje, IoT uređaji i slično. Neke od ključnih prednosti mikrokontrolera su veličina, cijena i energetska učinkovitost. Mikrokontroler je ugrađen unutar sustava za kontrolu pojedinačne funkcije u uređaju. Upravo informacije koje prima putem svojih ulaznih i izlaznih jedinica obrađuje pomoću središnjeg procesora. Privremene informacije koje mikrokontroler prima pohranjuju se u podatkovnu memoriju kojima pristupa procesor koji koristi upute pohranjenim u njegovoj programskoj memoriji za

dešifriranje i primjenu podataka. Zatim koristi svoje ulazne i izlazne jedinice za daljnju komunikaciju i provođenje određene radnje [8], [9], [10], [11].

3.3 Arhitektura klasičnog mikrokontrolera

- Procesor ili CPU (engl. *Central Processing Unit*): Procesor je „mozak“ uređaja koji obrađuje i odgovara na različite instrukcije, uključujući izvođenje osnovnih aritmetičkih i ulazno/izlaznih operacija.
- Memorija: Mikrokontroleri koriste memoriju za pohranjivanje podataka i instrukcija koje procesor treba obaviti. Koriste dvije vrste memorija:
    - RAM (engl. *Random Access Memory*): Privremeno pohranjuje podatke tijekom izvršavanja instrukcija, a podaci brišu isključivanjem uređaja s napajanja.
    - ROM (engl. *Read Only Memory*): Trajno pohranjuje instrukcije koje mikrokontroler treba obaviti. Može se reprogramirati i držati pohranjene instrukcije kada je mikrokontroler isključen.
- Ulazno/izlazni priključci: Priključci povezuju sučelje procesora s vanjskim uređajima. Ulazne jedinice primaju informacije i šalju ih prema procesoru u binarnom obliku (0 i 1). Procesor iste podatke obrađuje i ako je potrebno šalje novi signal izlaznim uređajima pomoću izlaznih priključaka.
- Komunikacijska sučelja: Komunikacijska sučelja omogućuju komunikaciju s drugim uređajima putem serijske komunikacije. Tipovi komunikacije mogu biti:
    - UART (engl. *Universal Asynchronous Receiver/Transmitter*)
    - I2C (engl. *Inter – Integrated Circuit*)
    - SPI (engl. *Serial Peripheral Interface*)
    - Bluetooth
    - WiFi
- ADC (engl. *Analog-to-Digital Converter*): Pomoću ADC-a mikrokontroler analogne signale pretvara u digitalne vrijednosti.
- DAC (engl. *Digital-to-Analog Converter*): Pomoću DAC-a mikrokontroler digitalne vrijednosti pretvara u analogne vrijednosti.
- Oscilator (engl. *Oscillator*): Oscilator pruža stabilan takt mikrokontrolera, kako bi komunikacija između komponenata i u samom procesoru bila sinkronizirana.

- Tajmer (engl. *Timer*): Tajmer Omogućuje provođenje vremenski uvjetovanih zadataka [12].



Slika 3.2: Prikaz strukture 8 bit mikrokontrolera

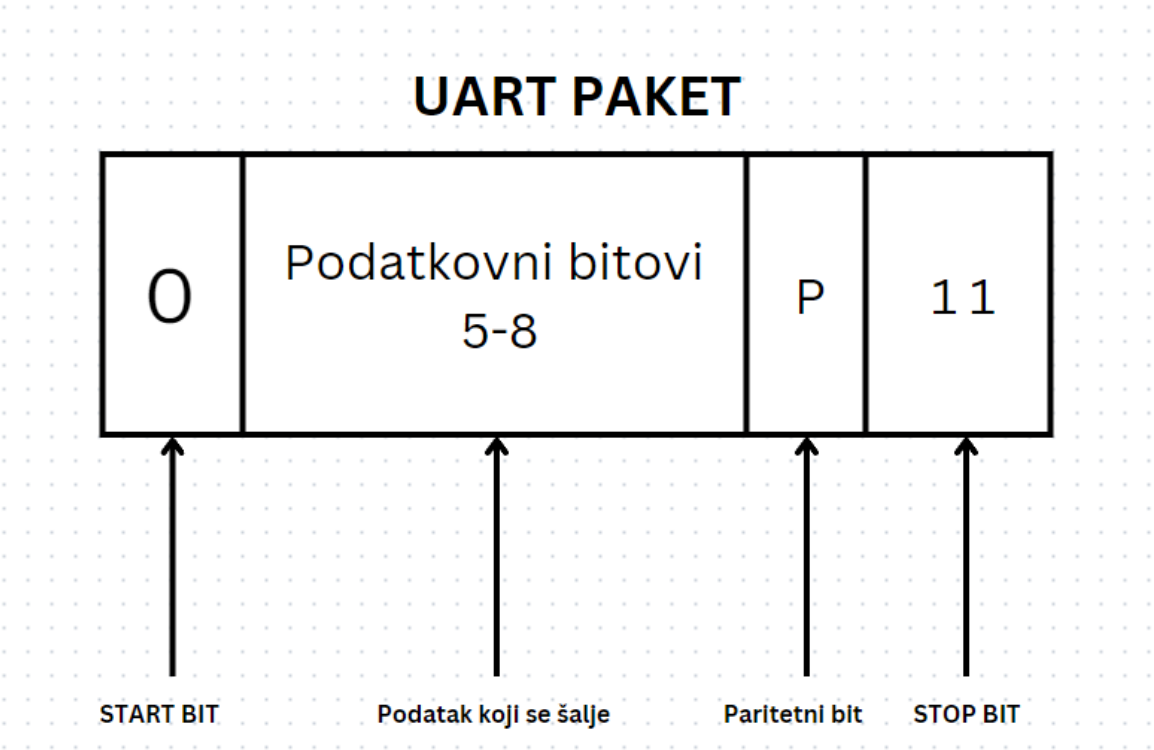
3.4 Komunikacija

U sljedećim potpoglavljima bit će opisane različite vrste komunikacije koje mikrokontroler može koristiti za komunikaciju s različitim vanjskim uređajima.

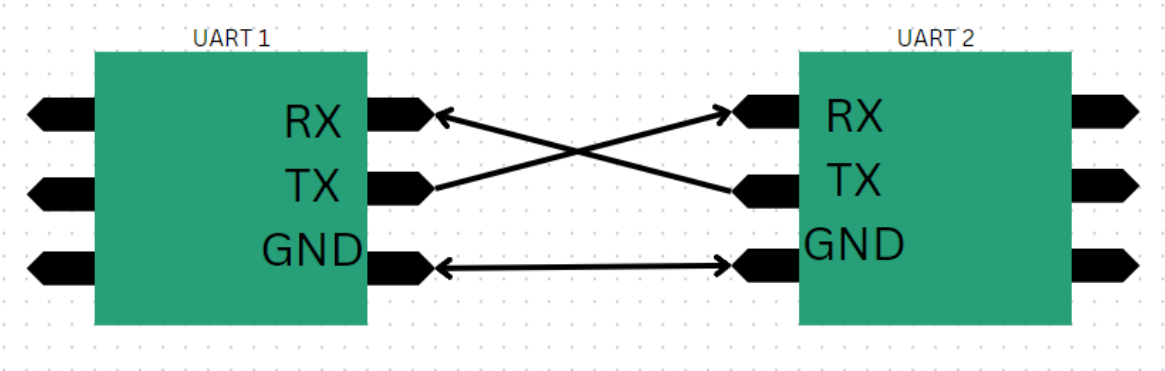
3.4.1 UART komunikacija

UART komunikacija je serijska i asinkrona komunikacija između dva uređaja. Asinkrona znači da ne postoji taktni signal koji će sinkronizirati slanje podataka s jednog na drugi uređaj. Ako je UART komunikacija ispravno konfigurirana tada UART može komunicirati s bilo kojim uređajem koji prima i odašilje podatke. Ovisno o potrebama UART komunikacija može biti jednosmjerna i dvosmjerna gdje se podaci šalju bit po bit. U UART komunikaciji između dva uređaja koriste se dva signala naziva odašiljač *(*engl. *Transmitter)* i prijamnik (engl. *Receiver).* Oba uređaja koja komuniciraju moraju imati unaprijed postavljenu brzinu prijenosa podataka koja na oba uređaja mora biti ista kako bi podaci uspješno bili poslani i primljeni. Svaki od podataka sadrži startni bit koji signalizira početak

prijenosa i bit za zaustavljanje koji signalizira kraj prijenosa podataka. Zbog same provjere pogreške u komunikaciji koristi se i paritetni bit koji pri tome može pomoći [12].



Slika 3.3: Prikaz podatka UART komunikacije

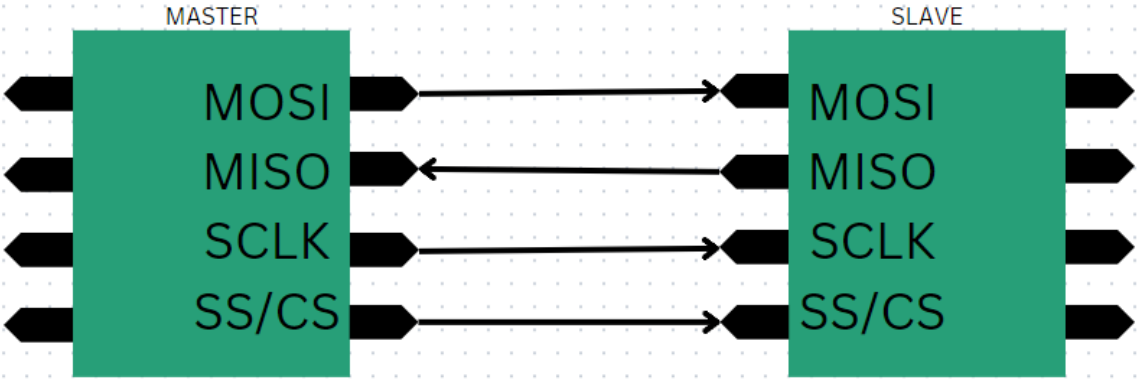


Slika 3.4: Prikaz spajanja žica UART komunikacije

3.4.2 **SPI komunikacija**

SPI komunikacija je serijska i sinkrona komunikacija koja se koristi za prijenos podataka između mikrokontroler i neke komponente ili drugog mikrokontroler. SPI koristi *master-slave* arhitekturu, gdje je jedan uređaj *master* i on kontrolira tok podataka, dok ostali uređaji *slave* primaju ili šalju podatke. Ova komunikacija je *Full-Duplex* komunikacija što

znači da omogućuje istodobno prijenos i prijem podataka. Master odabire slave uređaj postavljanjem SS/CS (engl. *Slave Select/Chip Select*) priključka na nisku razinu. Zatim generira taktni signal na SCK (engl. *Serial Clock*) priključku koji šalje podatke na MOSI (engl. *Master Out Slave In*) priključak i time slave uređaj čita podatke na svakom taktu signala SCK. Slanje podataka prema masteru vrši se preko MISO (engl. *Master In Slave Out*) priključka koji se isto tako očitava na svakom taktu signala. Kraj komunikacije se vrši postavljanjem SS/CS priključka na visoku razinu. Najčešći uređaji koji koriste ovakav tip komunikacije su LCD, OLED (engl. *Organic Light Emitting Diodes*) ekrani, memorijske kartice, različiti senzori i ostali uređaji koji zahtijevaju brz prijenos podataka [12].

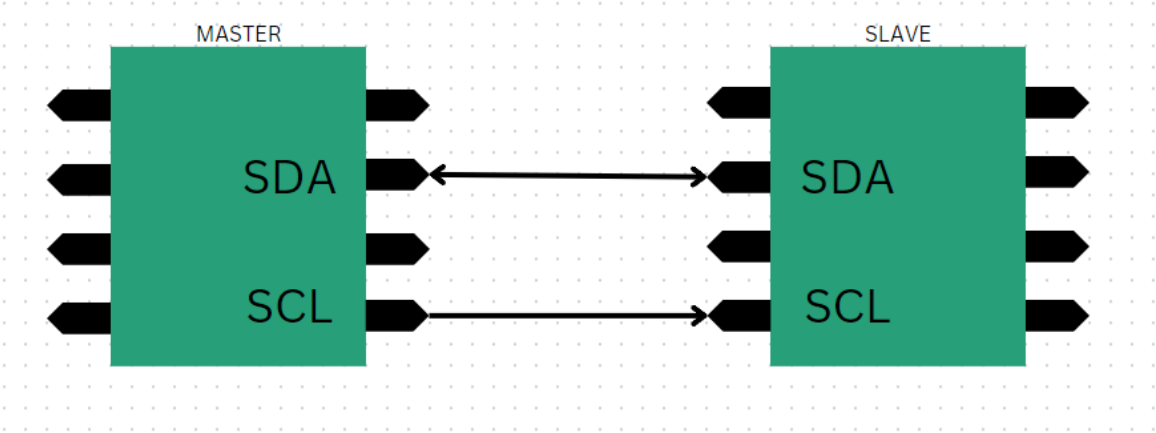


Slika 3.5: Prikaz SPI komunikacije

3.4.3 **I2C komunikacija**

I2C komunikacija popularna je sinkrona serijska komunikacija koja omogućuje dvama ili više uređajima da komuniciraju preko 2 žice. Postoje dva priključka koji su važni za komunikaciju, SDA (engl. *Serial Data Line*) i SCL (engl. *Serial Clock Line*). Prijenos podataka vrši se preko SDA priključka čijim prijenosom upravlja taktni signal koji se proizvodi na SCL priključku za sinkronizaciju podataka. Zasniva se isto kao i SPI komunikacija na *master–slave* komunikaciji. Svaku uređaj koji se nalazi u komunikacijskom kanalu sadrži jedinstvenu adresu na I2C sabirnici što omogućuje *master* uređaju da komunicira s određenim *slave* uređajem. Za razliku od SPI komunikacije koja je *Full-Duplex*, I2C je *Half-duplex* komunikacija što znači da se podaci ne mogu istovremeno prenositi u oba smjera. Način na koji radi jest da *master* uređaj postavljanjem SDA priključka na nisku razinu započinje komunikaciju, a zatim generiran taktni signal na SCL priključku koji omogućuje sinkronu komunikaciju. *Master* šalje adresu slave uređaja putem SDA priključka što rezultira slanjem potvrde *slave* uređaja s odgovarajućom adresom.

Postavljanjem SDA priključka na visoku razinu završava se komunikacija. Ovakva komunikacije većinom se koristi za komunikaciju s nekim senzorima poput senzora za vlažnost, temperaturu i svjetlost. Čak se koristi u internoj komunikaciji između modula na samom mikrokontroleru [12].



Slika 3.6: Prikaz I2C komunikacije

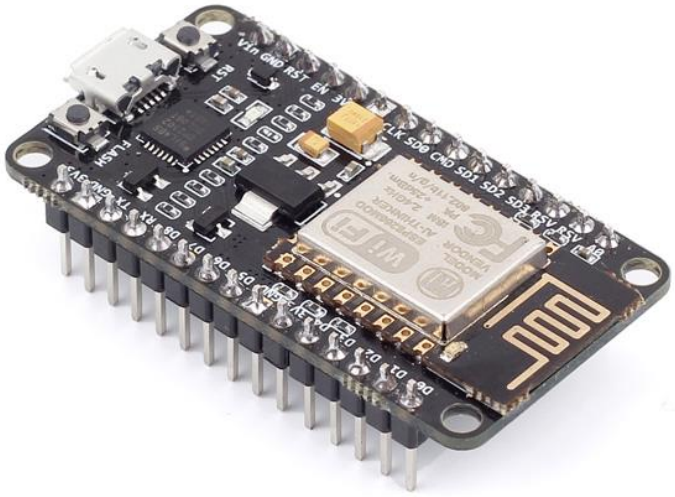
3.4.4 **Bluetooth komunikacija**

Bluetooth je bežična tehnologija koja pruža prijenos podataka bežično na kratkim udaljenostima. Bluetooth se često kod mikrokontrolera koristi za komunikaciju između više mikrokontrolera, mobilnih telefona, računala i ostalih uređaja. Bluetooth komunikacija omogućuje nisku potrošnju energije uvođenjem BLE u verziji Bluetooth 4.0, što je ključno u uređajima koji za napajanje koriste bateriju. Omogućuje povezivanje više uređaja istodobno, pa tako i njihovu međusobno razmjenu podataka. Postoji više sigurnosnih značajki kod ove komunikacije, a to su: uparivanje, autentifikacija i šifriranje podataka. Prije komunikacije uređaji moraju biti upareni. Tek nakon uparivanja uređaji mogu razmjenjivati podatke [13].

3.5 Usporedba s drugim razvojnim okruženjima

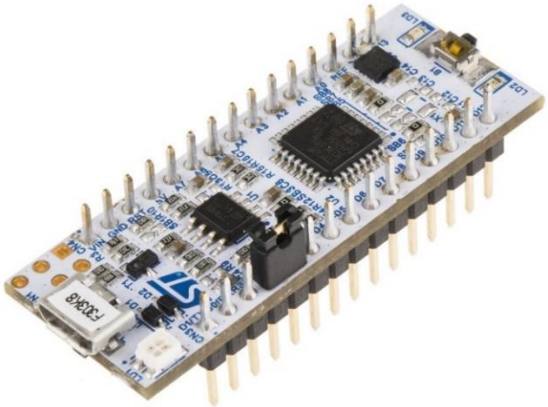
U svijetu ugrađenih sustava, izbor pravog mikrokontrolera može biti ključan za uspjeh projekta. Mikrokontroleri dolaze u različitim konfiguracijama, svaki sa svojim jedinstvenim setom značajki, prednosti i izazova. Ovo potpoglavlje će se fokusirati na detaljniju usporedbu između ESP32 i drugih popularnih platformi dostupnih na tržištu.

- ESP8266: Prethodnik ESP32 sustava na čipu. Razvijen je također od strane Espressif Systems tvrtke. Pruža WiFi povezivost, ali nema ugrađen Bluetooth modul. Koristi 32 – bit Tensilica Xtensa LX106 procesor s jednom jezgrom. Koristi se u mnogim slučajevima u kojima je potrebna niskobudžetna WiFi povezanost [14].

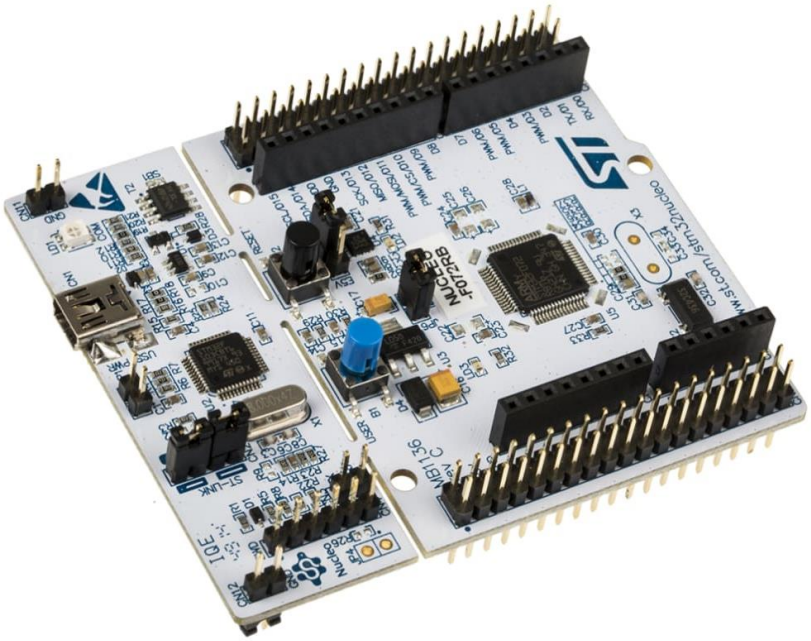


Slika 3.7: Prikaz ESP8266 razvojnog okruženja [15]

- STM32: STM32 je baziran na 32-bit ARM Cortex procesoru. Razvojno okruženje dolazi s Wi – Fi i Bluetooth sučeljima. STM 32 sadrži široku primjeru od IoT uređaja do industrije zbog svoje niske potrošnje energije. Razvila ga je tvrtka STMicroelectronics [16].

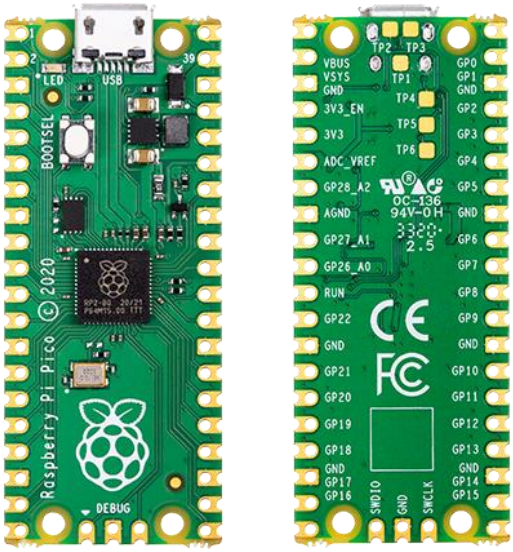


Slika 3.8: Prikaz STM32 razvojnog okruženja [17]



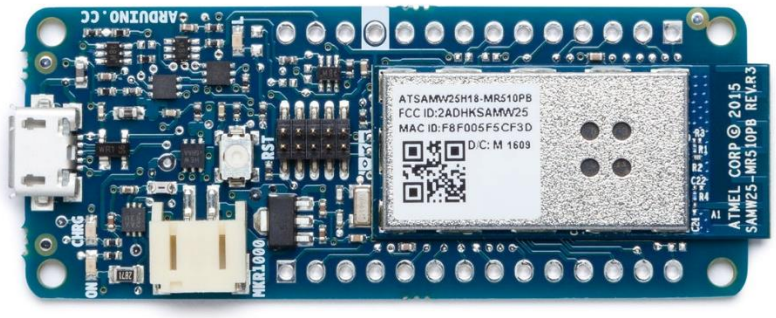
Slika 3.9: Prikaz STM32 Nucleo razvojnog okruženja [17]

- Raspberry Pi Pico: Raspberry Pi Pico je proizveden od strane Raspberry Pi Foundation-a. Koristi RP2040 mikrokontroler koji sadrži dvojezgreni ARM Cortex M0+ procesor. Uz 2 MB *flash* memorije ima 26 GPIO (engl. *General Purpose Input/Output*) priključaka i 3 analogna ulaza. Idealan je za početnike zbog svoje cijene i lakoće korištenja. Moguće ga je programirati pomoću C/C++ i MicroPython programskog razvojnog okruženja [18].



Slika 3.10: Prikaz Raspberry Pi Pico razvojnog okruženja [19]

- Arduino MKR1000: Koristi Arduino zero strukturu zajedno s WiFi modulom. Uz 256 KB *flash* memorije i 32 KB SRAM (engl. *Static Random Access Memory*) memorije koristi Atmel SAMD21 Cortex M0+ procesor . Također zbog same sigurnosti u komunikaciji koristi ECC508 za autentifikaciju [20].



Slika 3.11: Prikaz Arduino MKR1000 razvojnog okruženja [20]

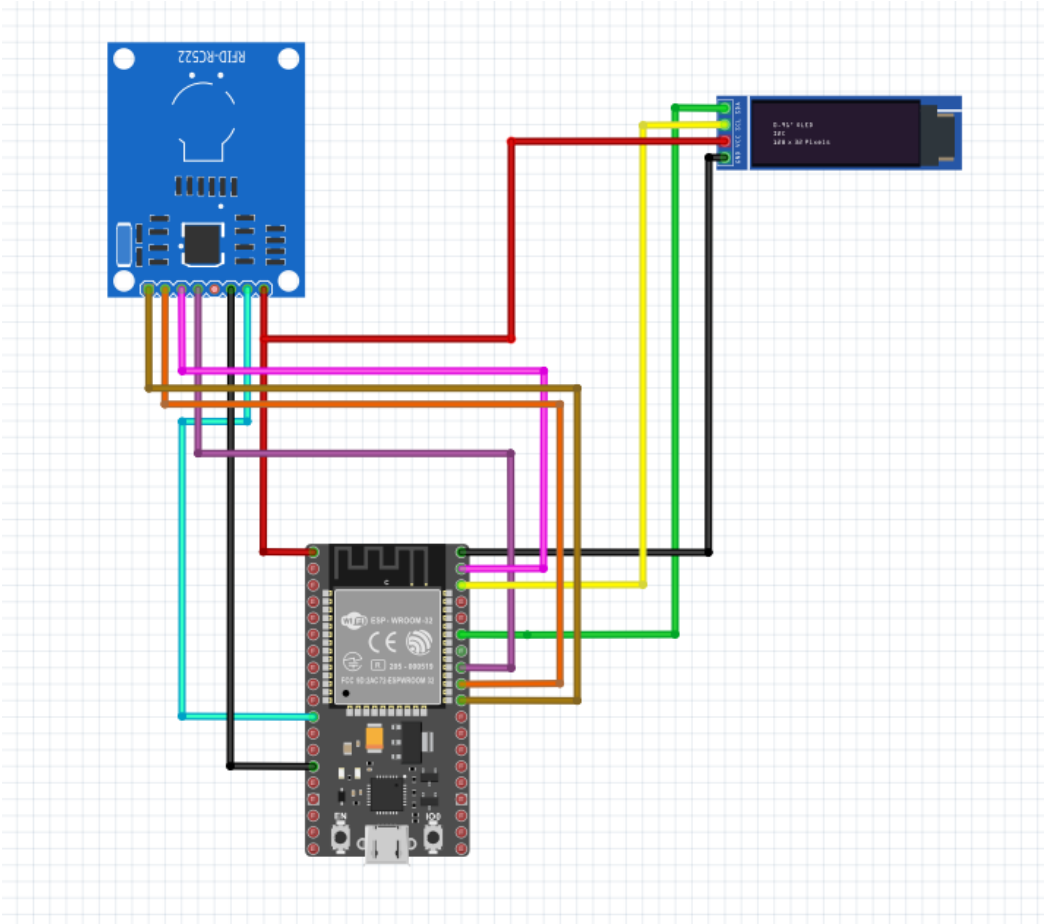
4. Izrada hardverskog rješenja

U ovom poglavlju biti će navedene i opisane komponente za izradu ovog završnog rada. Kako je već poznato da se radi o izradi hardverskog rješenja za praćenje aktivnosti studenata na nastavnim aktivnostima postoji glavni zahtjev kojega se moramo držati, a to je evidencija prisutnosti pomoću studentske isprave. Svaka studentska isprava u sebi sadrži zapisan jedinstveni identifikator (UUID) te kao i većina RFID kartica ima u sebi ugrađenu malu antenu i čip. Kada se takva isprava približi RFID čitaču, čitač emitira radiofrekvencijski signal koji napaja karticu pomoću radiofrekvencije od 13.56 MHz i omogućava prijenos podataka između kartice i čitača. Taj podatak koji čitač očitava je zapravo UUID, jedinstveni identifikator studenta čija je isprava prislonjena. Čitač koji je korišten u izradi hardverskog rješenja je MFRC522 RFID koji emitira radiofrekvenciju od 13.65 MHz. Za komunikaciju s mikrokontrolerom koristi SPI komunikaciju.

Nakon pažljivog razmatranja između raznih mikrokontrolera, odabran je ESP32 koji je opisan u poglavlju 4.5. Uz snažne performanse, integrirane Wi – Fi i Bluetooth povezivosti odličan je izbor za izradu ovog IoT hardverskog rješenja koje će uz sve to imati i .NET aplikaciju. Uz bogati skup biblioteka i alata omogućuje razvoj i implementaciju jednostavnijom.

Kako bi profesori i studenti mogli dobivati povratne informacije koristio sam OLED ekran za prikaz informacija dobivenih od strane .NET aplikacije.

Smatram da je ovakvo rješenje trenutno najefikasnije i skalabilno prema budućim potrebama. Mikrokontroler ima ulogu iščitavanja UUID oznake iz studentske isprave koju zatim prosljeđuje putem MQTT protokola prema bazi podataka tj. .NET aplikaciji koja vrši daljnju obradu podataka i vraća ako je potrebno informacije nazad prema mikrokontroleru. U nastavku ću detaljnije opisati mikrokontroler koji je korišten, čitač kartica i OLED ekran. Na slici 4.0 moguće je vidjeti električnu shemu hardverskog rješenja.



Slika 4.1: Prikaz električne sheme hardverskog rješenja

4.1 MFRC522 RFID čitač

MFRC522 je popularni RFID čitač koji se koristi za čitanje RFID kartica ili privjesaka čija je radna frekvencija 13.56 MHz. Ovakav čitač često se koristi u jednostavnim prototip projektima za kontrolu pristupa, identifikaciju i sl. Glavna karakteristika ovog modula je niska potrošnja energije što ga činim odličnim za baterijski napajane uređaje. MFRC522 čitač emitira radiofrekvencijski signal od 13.56 MHz koji time napaja RFID privjeske i kartice te im omogućuje da time pošalji svoj jedinstveni identifikator i druge podatke koji mogu biti pohranjeni natrag čitaču. Ovaj čitač koristan je u različitim IoT sustavima u kojima je potrebna niska potrošnja energije, jednostavnost i fleksibilnost. Uz IoT sustave lako ga je ugraditi i u ostale sustave slične namjene [21]. Slika 4.2 prikazuje MFRC522 RFID čitač i RFID karticu i privjesak.

Slika 4.2: Prikaz MFRC522 RFID čitača i kartice [22]

4.2 I2C OLED zaslon

U svrhu izrade projekta korišten je monokromatski zaslon koji koristi OLED tehnologiju za prikaz informacija. Njegova veličina iznosi 128 x 32 piksela što znači 128 piksela u horizontalnom smjeru i 32 piksela u vertikalnom smjeru. Koristi I2C komunikaciju kako bi primao informacije od mikrokontrolera. OLED tehnologija omogućava visoki kontrast i široke kutove gledanja uz nisko vrijeme odaziva. I2C komunikacija omogućuje komunikaciju putem dva priključka što pomaže pri uštedi na priključcima za eventualne nadogradnje u budućnosti. Pri odabiru zaslona bilo je važno nekoliko stvari, dimenzije i niska potrošnja energije što je s ovim zaslonom pun pogodak. OLED tehnologija pomaže pri niskoj potrošnji energije, a pritom je i rezolucija sasvim dovoljna za prikaz informacija [23]. Slika 4.3 prikazuje OLED zaslon.

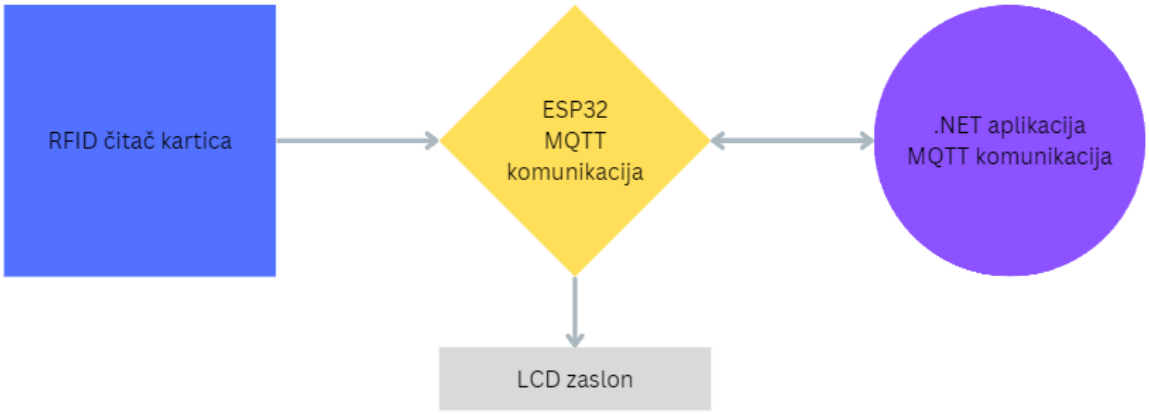
Slika 4.3: Prikaz OLED zaslona [24]

5. Izrada programskog rješenja

U ovom poglavlju bit će opisana implementacija programskog rješenja koje podržava hardverski sustav za evidenciju prisutnosti studenata. U sljedećim potpoglavljima, detaljno su opisani dijelovi programskog koda i korištene tehnologije za stvaranje robusnog, skalabilnog i pouzdanog programskog rješenja. Za izradu programskog koda korišteno je Arduino programsko razvojno okruženje koje uz bogatu količinu biblioteka i široke zajednice pruža jednostavnu i brzu izradu programskog rješenja

5.1 Dijagram tijeka

Kako je već objašnjeno, RFID čitač igra ključnu ulogu u procesu identifikacije i verifikacije studentskih isprava kroz očitavanja UUID podataka. U ovom integriranom sustavu, ESP32 preuzima centralnu ulogu u obradi i prijenosu tih podataka. Kada RFID čitač očita UUID, ESP32 ga ne samo da ispisuje na LCD zaslon, pružajući tako trenutnu vizualnu povratnu informaciju, već i inicira komunikacijski protokol za daljnji prijenos informacija. Koristeći MQTT protokol, ESP32 šalje UUID podatak na .NET aplikaciju. Ovaj protokol omogućuje brz i pouzdan prijenos podataka, optimiziran za situacije gdje su resursi i propusnost ograničeni. .NET aplikacija, koja djeluje kao pretplatnik u ovom scenariju, prima podatak, obrađuje ga i sprema u bazu podataka. Osim pohrane, aplikacija je zadužena i za pripremu podataka za vizualizaciju, omogućujući tako korisnicima da lako interpretiraju i analiziraju informacije. Interakcija između ESP32 i .NET aplikacije nije jednosmjerna. .NET aplikacija može objaviti poruku na određenu temu, koja će biti postavljena u *callback* funkciji na ESP32. Ovo omogućuje dinamičku dvosmjernu komunikaciju, gdje ESP32 može obraditi primljene informacije i prema potrebi ih ispisati na LCD zaslon. Ova funkcionalnost ne samo da poboljšava interaktivnost sustava, već i omogućuje ažuriranje u stvarnom vremenu i prilagodbu informacija, čime se povećava efikasnost i korisničko iskustvo u procesu evidencije prisutnosti studenata. Na slici 5.1 može se vidjeti dijagram tijeka IoT komunikacije.



Slika 5.1: Prikaz IoT komunikacije

5.2 Ostvarenje MQTT komunikacije

MQTT *(*engl. *Message Queuing Telemetry Transport)* je komunikacijski protokol koji se ističe svojom jednostavnošću i efikasnošću, posebno u kontekstu IoT ekosustava. Ovaj protokol omogućuje brz i pouzdan prijenos informacija između uređaja putem interneta, čineći ga idealnim za uređaje s ograničenim resursima koji zahtijevaju optimiziranu komunikaciju. U osnovi MQTT komunikacije leži model pošiljatelj-pretplatnik. Pošiljatelji su uređaji ili aplikacije koje generiraju i šalju poruke na određene teme. Pretplatnici, s druge strane, su entiteti koji su se pretplatili na te teme i koji primaju i obrađuju poruke koje pošiljatelji šalju. Ova hijerarhija omogućuje fleksibilnu i skalabilnu komunikaciju, podržavajući tako različite aplikacije i scenarije upotrebe. MQTT protokol nudi nekoliko razina QoS (engl. *Quality of Service*) koje određuju kako se poruke isporučuju. Na razini 0, poruka se šalje najviše jednom, što je najbrži način prijenosa, ali ne nudi potvrdu o isporuci. Na razini 1, poruka se šalje najmanje jednom, osiguravajući da će poruka biti primljena, ali može doći do dupliciranja. Na razini 2, poruka se šalje točno jednom, osiguravajući tako i isporuku i jedinstvenost poruke. Sigurnost je ključna komponenta svakog komunikacijskog protokola, a MQTT nije iznimka. Iako je sam po sebi jednostavan, MQTT nudi mogućnost integracije s različitim postojećim sigurnosnim protokolima. To omogućuje autentifikaciju, autorizaciju i enkripciju podataka, štiteći tako informacije od neautoriziranog pristupa i manipulacije. U kontekstu ovog rada, MQTT se koristi za omogućavanje komunikacije između ESP32 i .NET aplikacije. ESP32 djeluje kao pošiljatelj, šaljući UUID podatke očitane s RFID čitača na .NET aplikaciju koja djeluje kao pretplatnik. Ova dinamička *i real - time* komunikacija omogućuje brzu obradu i analizu podataka, čime

se postiže efikasnost i preciznost u evidenciji prisutnosti studenata [25]. Na slici 5.2 prikazan je MQTT logo.

Slika 5.2: Logo MQTT-a

5.3 Callback funkcija

```
1 void loop() {
2  //Ostatak programskog koda
3
4     //Povezivanje na MQTT broker
5     client.setServer(mqtt_broker, mqtt_port);
6     //Postavljanje callback funkcije naziva mqttCallback
7     client.setCallback(mqttCallback);
8
9  //Ostatak programskog koda
10 void mqttCallback(char* topic, byte* payload, unsigned int length) {
11    Serial.print("Primljena poruka s MQTT-a: ");
12    for (int i = 0; i < length; i++) {
13          response+=(char)payload[i];
14    }
15    printOLED(response);
16    Serial.println(response);
17    }
18}
```

Programski kod 5.1: Prikaz dijela programskog koda zaduženog za MQTT komunikaciju

Callback funkcija u programskom kodu 5.1 je dizajnirana da se pozove kada se primi poruka preko MQTT protokola i ispiše je na OLED zaslonu i *Serial Monitoru*. Funkcija prima tri parametra: *topic* koji predstavlja temu na kojoj je poruka primljena, *payload* koji je niz bajtova koji sadrži samu poruku, i *length* koji predstavlja dužinu poruke u bajtovima. 13. linija koda dodaje svaki bajt iz payload u *string response*, konvertirajući bajt u slovo prije dodavanja. Prislanjanjem studentske isprave na čitač UID studenta se šalje .NET

aplikaciji koja odgovara pozdravom studenta koji je prislonio studentsku ispravu. Ako je student već zabilježen na nastavnoj aktivnost šalje se upozorenje da je student već zabilježio prisutnost. 15. linija koda opisuje povratne informacije .NET aplikacije na OLED zaslon, vidljivo na slikama 5.3 i 5.4.

Slika 5.3: Prikaz na OLED zaslonu kada student prijavi prisutnost

Slika 5.4: Prikaz na OLED zaslonu nakon ponovnog pokušaja prijave prisutnosti

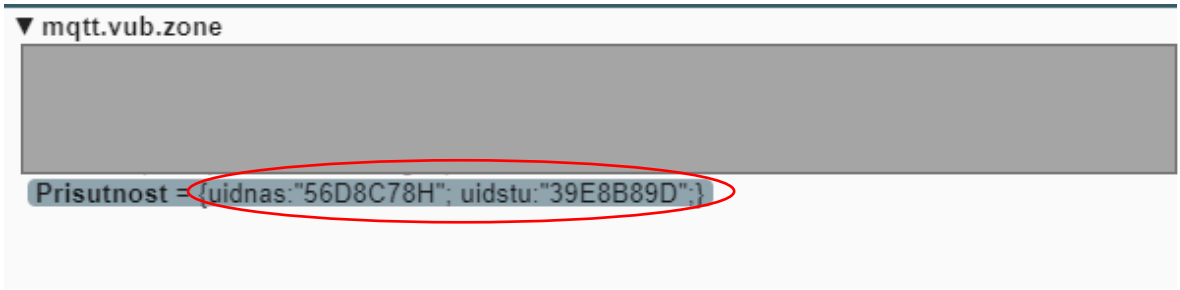
```
1void printOLED(String data) {
2     display.clearDisplay();
3     display.setTextSize(1);
4     display.setTextColor(SSD1306_WHITE);
5     display.setCursor(0, 0);
6     display.print(data);
7     display.display();
8     delay(1000);
9}
10
11// Ostatak programskog koda
```

Programski kod 5.2: Prikaz programskog koda funkcije za ispis na OLED zaslonu

Funkcija na programskoj liniji od 1 do 9 u programskom kodu 5.2 služi za ispisivanje teksta na OLED zaslon. Parametar *data* koji se prosljeđuje u funkciju je *string* koji sadrži tekst za ispis na zaslon. Postoji nekoliko parametara koje možemo mijenjati a to su, veličina slova, mjesto ispisa pomoću kursora i dvije funkcije koje potvrđuju unos 6 i 7 red.

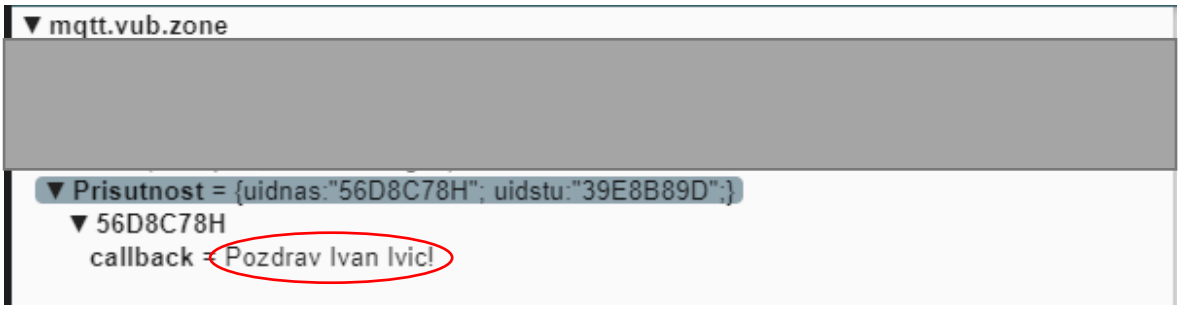
5.4 Pogled iz MQTT Explorera

MQTT Explorer pruža intuitivno korisničko sučelje za pregledavanje, analizu i manipulaciju porukama i temama u realnom vremenu, čime olakšava otklanjanje pogrešaka i optimizaciju MQTT baziranih aplikacija i sustava. MQTT Explorer je koristan za razvojne inženjere, testere i administratore koji rade s IoT uređajima i sustavima, omogućujući im da lako testiraju i dijagnosticiraju komunikaciju između uređaja i servera.



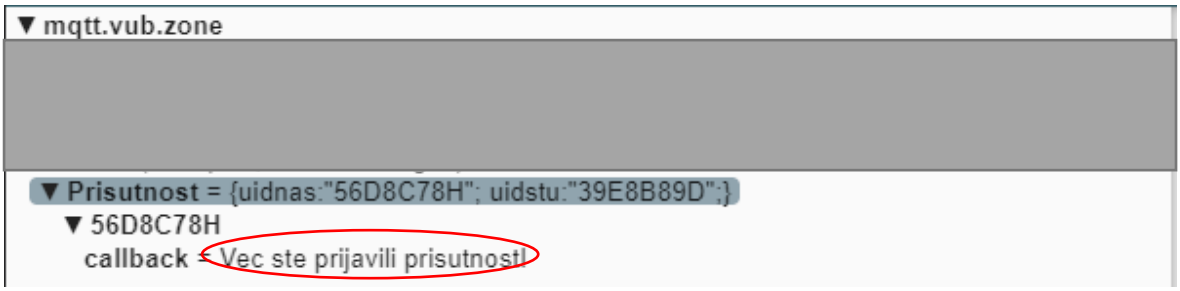
Slika 5.5: Prikaz poruke koju šalje ESP32 preko MQTT komunikacije

Slika 5.5 prikazuje poruku koja se šalje sa ESP32 u temu „Prisutnost“ u JSON *(*engl. *JavaScript Object Notation)* formatu na MQTT posrednik *(*engl. *MQTT Broker).* Poruka sadrži dva para ključa „uidnas“ i „uidstu“, a vrijednosti su odgovarajući UUID – ovi. Komunikacija je zamišljena tako da je jedan uređaj vezan za jednog nastavnika, te se dio JSON poruke s ključem „uidnas“ nikada s tog uređaja neće mijenjati već samo „uidstu“ svakog novog studenta koji se pokušava prijaviti na nastavnu aktivnost.



Slika 5.6: Prikaz poruke koja se šalje prema ESP32 kada student prijavi prisutnost

Slika 5.6 prikazuje sadržaj *callback* funkcije koji se šalje sa .NET aplikacije prema ESP32. Na slici možemo vidjeti kako se kreirala podtema s UUID – om nastavnika kako bi mogla ispisati ispravna poruka na točno registriranom uređaju.



Slika 5.7. Prikaz poruke koja se šalje prema ESP32 kada student pokušava ponovno prijaviti prisutnost

Na slici 5.7 možemo vidjeti sadržaj poruke ako student ponovno pokuša prijaviti prisutnost, a njegova prisutnost je već prijavljena.

6. ZAKLJUČAK

Tema ovog završnog rada bila je izrada hardverskog rješenja za sustavnu evidenciju nazočnosti na nastavnim aktivnostima. Poseban naglasak je na integraciji IoT tehnologija i MQTT komunikacijskog protokola. Hardversko rješenje koristi ESP32 sustav na čipu i RFID čitač za identifikaciju i registraciju prisutnosti studenata na aktivnostima, dok se podaci obrađuju i vizualiziraju kroz .NET aplikaciju. Kroz proces razvoja završnog rada, analizirane su prednosti i mane programskih razvojnih okruženja kao što su Arduino IDE, EspressIf IDF i MicroPython. Uz razvojna okruženja opisana je struktura i funkcionalnosti ESP32. MQTT komunikacijski protokol, s MQTT posrednikom kao ključnim elementom za potpunu funkcionalnost uređaja i aplikacije. Implementacija OLED ekrana dodatno je poboljšala korisničko iskustvo, omogućujući prikaz informacija u stvarnom vremenu. Korištenje MQTT Explorer-a je omogućilo praćenje i analizu komunikacije, dok je Arduino programsko razvojno okruženje pružio fleksibilnost i brzinu razvoja. Ovo rješenje ne samo da optimizira cijeli proces evidencije prisutnosti, već i otvara mogućnosti za daljnje inovacije i integracije, poput analitike podataka, automatiziranih sustava odgovora itd. Efikasnost, skalabilnost i sigurnost su ključne karakteristike razvijenog sustava što ga čini robusnim rješenjem za suvremene obrazovne institucije. U budućnosti je moguće dodati proširenja i unaprijediti sustav kroz integraciju dodatnih IoT uređaja, analitičkih alata i umjetne inteligencije, kako bi se dodatno optimizirala edukacijska iskustva za studente i profesore. Ovaj rad predstavlja temelj za daljnje istraživanje i inovacije u području automatizirane evidencije prisutnosti i IoT rješenja u obrazovnom sektoru.

7. LITERATURA

[1] Arduino. About Arduino [Online]. 15.10.2021.

Dostupno na:

https://www.arduino.cc/en/about

[2] Linuxhint. What Are Advantages and Disadvantages of Arduino [Online]. 2022.

Dostupno na:

https://linuxhint.com/advantages-and-disadvantages-arduino/

[3] EspressIF. Getting Started with ESP-IDF [Online]. 2023.

Dostupno na:

https://www.espressif.com/en/products/sdks/esp-idf

[4] ESPBorads. ESP-IDF vs Arduino Core [Online]. 18.3.2023.

Dostupno na:

https://www.espboards.dev/blog/esp-idf-vs-arduino-core/

[5] DigiKey. Micropython - basics. What is MicroPython? [Online]. 2022.

Dostupno na:

https://www.digikey.com/en/maker/projects/micropython-basics-what-is-micropython/1f60afd88e6b44c0beb0784063f664fc

[6] Random Nerd Tutorials. Getting started with MicroPython on ESP32 and ESP8266. [Online]. 2018

Dostupno na:

https://randomnerdtutorials.com/getting-started-micropython-esp32-esp8266/

[7] Electronics Hub. Getting started witg ESP32 | Introduction to ESP32. [Online]. 2016.

Dostupno na:

https://www.electronicshub.org/getting-started-with-esp32/

[8] The Astrology Page. Što je Mikrokontroler? [Online]. 2023.

Dostupno na:

https://hr.theastrologypage.com/microcontroller

[9] Automatiziraj se. Mikrokontroleri [Online]. 2023.

Dostupno na:

https://automatiziraj.se/mikrokontroleri/

[10] TechTarget. IotAgenda Microcontroller [Online]. Studeni 2019.
Dostupno na:
https://www.techtarget.com/iotagenda/definition/microcontroller
[11] All about circuits. What is Microcontroller? The defining characteristics and Architecture of a Common Component [Online]. 25.3.2019.
Dostupno na:
https://www.allaboutcircuits.com/technical-articles/what-is-a-microcontroller-introduction-component-characteristics-component/
[12] Fmuser. Uvod u SPI, I2C, UART, GPIO, SDIO, CAN [Online]. 2023.
Dostupno na:
https://hr.fmuser.org/news/IPTV-encoder/Introduction-to-SPI-I2C-UART-I2S-GPIO-SDIO-CAN/
[13] Intel. What Is Bluetooth Technology? [Online]. 2023
Dostupno na:
https://www.intel.com/content/www/us/en/products/docs/wireless/what-is-bluetooth.html
[14] EngineersGarage. Getting started with ESP8266 and ESP32 on Arduino IDE [Online] 2023.
Dostupno na:
https://www.engineersgarage.com/articles-gettting-started-with-esp8266-esp32-arduino-ide/
[15] Radiona Wiki. ESP8266 programiranje [Online]. 2023
Dostupno na:
https://radiona.org/wiki/project/esp8266_programming
[16] St life.augmented. STM32 Nucelo Boards [Online]. 2023
Dostupno na:
https://www.st.com/en/evaluation-tools/stm32-nucleo-boards.html
[17] RS-Online. STMicroelectronics STM32 Nucleo-32 MCU Development Board NUCLEO-F303K8 [Online]. 2023
Dostupno na:
https://twen.rs-online.com/web/p/microcontroller-development-tools/9092862
[18] Raspberry Pi. Raspberry Pi Pico [Online] 2023.
Dostupno na:
https://www.raspberrypi.com/products/raspberry-pi-pico/

[19] Mouser electronics. Raspberry Pi Pico [Online] 2023.
Dostupno na:
https://www.mouser.in/new/raspberry-pi/raspberry-pi-pico-boards/?_gl=1*1ev5qet*_ga*OTk5ODgxNDE2LjE2OTY0MzA3NTE.*_ga_15W4STQT4T*MTY5NjQzMDc1MC4xLjAuMTY5NjQzMDc1MS41OS4wLjA.
[20] Arduino. Arduino MKR1000 WiFi [Online] 2023.
Dostupno na:
https://store-usa.arduino.cc/products/arduino-mkr1000-wifi
[21] Last Minute Engineers. How RFID works RC522 [Online] 2023.
Dostupno na:
https://lastminuteengineers.com/how-rfid-works-rc522-arduino-tutorial/
[22] Amazon. SunFounder Reader Module Kit Mifare RC522 Reader Module with S50 White Card and Key Ring Compatible with Arduino Raspberry Pi [Online] 2023.
Dostupno na:
https://www.amazon.com/SunFounder-Mifare-Reader-Arduino-Raspberry/dp/B07KGBJ9VG
[23] Adafruit. Monochrome 0.91" 128x32 I2C OLED Display [Online] 2023.
Dostupno na:
https://www.adafruit.com/product/4440
[24] Smart Prototyping. 0.91" 128x32 I2C OLED Display [Online] 2023.
Dostupno na:
https://www.smart-prototyping.com/0.91-inch-OLED-display
[25] Hardver funti. MQTT: tvoreni mrežni protokol i njegova važnost u IoT-u [Online]. 2023.
Dostupno na:
https://www.hwlibre.com/hr/mqtt/

8. OZNAKE I KRATICE

BLE *(*engl. *Bluetooth Low Energy)*

CPU (engl. *Central Processing Unit)*

I2C *(*engl. *Inter-Integrated Circuit)*

IDE (engl. *Integrated Development Environment)*

IDF *(*engl. *IoT Development Framework)*

IoT *(*engl. *Internet of Things)*

JSON *(*engl. *JavaScript Object Notation)*

LCD *(*engl. *Liquid Crystal Display)*

MISO *(*engl. *Master In Slave Out)*

MOSI *(*engl. *Master Out Slave In)*

MQTT *(*engl. *Message Queuing Telemetry Transport)*

OLED *(*engl. *Organic Light Emitting Diodes)*

OOP *(*engl. *Object-Oriented Programming)*

REPL *(*engl. *Read Evaluate Print Loop)*

RFID *(*engl. *Radio-frequency identification)*

RTOS *(*engl. *Real Time Operating System)*

RTOS *(*engl. *Real Time Operating System)*

SCK *(*engl. *Serial Clock).*

SCL *(*engl. *Serial Clock Line).*

SDA *(*engl. *Serial Data Line)*

SPI *(*engl. *Serial Peripheral Interface)*

SS/CS *(*engl. *Slave select/Chip Select)*

UART *(*engl. *Universal Asynchronous Receiver/Transmitter)*

UUID *(*engl. *Universally Unique Identifier)*

9. SAŽETAK

Naslov: Izrada hardverske podrške za sustavnu evidenciju nazočnosti studenata na nastavnim aktivnostima

Ovaj rad fokusira se na razvoj i implementaciju hardverske podrške za sustavnu evidenciju nazočnosti studenata na nastavnim aktivnostima, koristeći inovativne IoT tehnologije i MQTT komunikacijski protokol. Centralni element sistema je ESP32, koji u kombinaciji s RFID čitačem, omogućuje automatsko i efikasno bilježenje prisutnosti studenata. Studenti svoju prisutnost potvrđuju prislanjanjem studentske isprave na RFID čitač. Podaci se prikupljaju i obrađuju u realnom vremenu, a rezultati se vizualiziraju kroz intuitivno korisničko sučelje .NET aplikacije. Implementacija OLED ekrana na ESP32 omogućuje direktan prikaz informacija studentima i nastavnom osoblju na licu mjesta. Sustav je testiran u stvarnom okruženju, pri čemu je pokazao visoku razinu preciznosti, efikasnosti i pouzdanosti. Razvoj ovog sistema demonstrira potencijal integracije IoT tehnologija u edukacijski sektor, nudeći automatizirana, precizna i skalabilna rješenja za evidenciju prisutnosti i druge aspekte upravljanja nastavnim aktivnostima. Ovaj rad pruža temelj za daljnje istraživanje i razvoj naprednih IoT rješenja u kontekstu optimizacije edukacijskih procesa i iskustava.

Ključne riječi: IoT, MQTT, RFID, .NET, evidencija prisutnosti.

10. ABSTRACT

Title: Development of Hardware Support for Systematic Attendance Recording of Students in Educational Activities

This study focuses on the development and implementation of hardware support for the systematic recording of student attendance during educational activities, utilizing innovative IoT technologies and the MQTT communication protocol. The central element of the system is the ESP32, which, in combination with an RFID reader, enables automatic and efficient recording of student attendance. Students confirm their presence by presenting their student ID to the RFID reader. Data is collected and processed in real-time, and the results are visualized through the intuitive user interface of a .NET application. The implementation of an OLED screen on the ESP32 microcontroller allows for the direct display of information to students and teaching staff on-site. The system was tested in a real environment, where it demonstrated a high level of accuracy, efficiency, and reliability. The development of this system showcases the potential of integrating IoT technologies into the educational sector, offering automated, precise, and scalable solutions for attendance recording and other aspects of managing educational activities. This work lays the foundation for further research and development of advanced IoT solutions in the context of optimizing educational processes and experiences.

Keywords: IoT, MQTT, RFID, .NET, attendance monitoring.

IZJAVA O AUTORSTVU ZAVRŠNOG RADA

Pod punom odgovornošću izjavljujem da sam ovaj rad izradio/la samostalno, poštujući načela akademske čestitosti, pravila struke te pravila i norme standardnog hrvatskog jezika. Rad je moje autorsko djelo i svi su preuzeti citati i parafraze u njemu primjereno označeni.

Mjesto i datum	Ime i prezime studenta/ice	Potpis studenta/ice
U Bjelovaru, 4.10.2023	Martin Cindrić	Martin Cindrić

U skladu s čl. 58, st. 5 Zakona o visokom obrazovanju i znanstvenoj djelatnosti, Veleučilište u Bjelovaru dužno je u roku od 30 dana od dana obrane završnog rada objaviti elektroničke inačice završnih radova studenata Veleučilišta u Bjelovaru u nacionalnom repozitoriju.

Suglasnost za pravo pristupa elektroničkoj inačici završnog rada u nacionalnom repozitoriju

Martin Cindrić
ime i prezime studenta/ice

Dajem suglasnost da tekst mojeg završnog rada u repozitorij Nacionalne i sveučilišne knjižnice u Zagrebu bude pohranjen s pravom pristupa (zaokružiti jedno od ponuđenog):

a) Rad javno dostupan
b) Rad javno dostupan nakon _____ (upisati datum)
c) Rad dostupan svim korisnicima iz sustava znanosti i visokog obrazovanja RH
d) Rad dostupan samo korisnicima matične ustanove (Veleučilište u Bjelovaru)
e) Rad nije dostupan.

Svojim potpisom potvrđujem istovjetnost tiskane i elektroničke inačice završnog rada.

U Bjelovaru, 4.10.2023.

Martin Cindrić
potpis studenta/ice