

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 210

**SUSTAV ZA ODRŽAVANJE LABORATORIJSKIH VJEŽBI NA
DALJINU**

Antonio Bukovac

Zagreb, lipanj 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 210

**SUSTAV ZA ODRŽAVANJE LABORATORIJSKIH VJEŽBI NA
DALJINU**

Antonio Bukovac

Zagreb, lipanj 2023.

**SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

Zagreb, 10. ožujka 2023.

DIPLOMSKI ZADATAK br. 210

Pristupnik: **Antonio Bukovac (0036519160)**

Studij: Računarstvo

Profil: Programsко inženjerstvo i informacijski sustavi

Mentor: doc. dr. sc. Tomislav Jaguš

Zadatak: **Sustav za održavanje laboratorijskih vježbi na daljinu**

Opis zadatka:

Nedostatak opreme u školama, pogotovo u ruralnim krajevima, ali i nedavna pandemija koronavirusa, još su više aktualizirali potrebu za udaljenom nastavom, tj. održavanjem laboratorijskih vježbi na daljinu, za što je potrebna odgovarajuća računalna podrška. U sklopu diplomskog rada potrebno je istražiti dostupnu literaturu i postojeće sustave za izvođenje vježbi na daljinu i na temelju njih predložiti i ostvariti vlastiti sustav. Predloženi sustav treba biti generički, tj. mora omogućiti jednostavno programiranje različitih tipova uređaja (npr. Arduino, LEGO robot, MicroBit i sl.). Osim korisničkog sučelja, u kojem se piše program i vide rezultati prevođenja i izvođenja na stvarnom uređaju, sustav treba imati i administracijsko sučelje, kroz koje nastavnik može definirati zadatke, dodavati i uređivati podatke o korisnicima i sl. Za pohranu podataka o korisnicima, zadacima i povijesti korištenja sustava potrebno je ostvariti prikladnu bazu podataka i odgovarajuću podršku u administracijskom sučelju. Potrebno je istražiti i osmislit mogućnosti rada više istovremenih korisnika, kao i njihove suradnje u rješavanju zadatka. U sklopu diplomskog rada potrebno je sastaviti testni skup laboratorijskih zadataka i pripremiti potreban hardver kako bi se sustav mogao isprobati u praksi.

Rok za predaju rada: 23. lipnja 2023.

Želim izraziti svoju duboku zahvalnost svima koji su mi pružili podršku i omogućili da uspješno završim diplomski studij.

U prvom redu, neizmjerno sam zahvalan svojim roditeljima Jadranki i Tomi te bratu Dominiku, koji su mi uvijek bili tu i pružali mi neizmjernu moralnu i finansijsku podršku tijekom cijelog školovanja.

Posebno bih se želio zahvaliti svojoj djevojci Petri koja mi je pružala neprestanu podršku i vjerovala u moje sposobnosti čak i kada nisam vjerovao u sebe.

Također, želim se zahvaliti svim svojim prijateljima, posebice Nikolini, koji su uvijek bili tu uz mene tijekom svih izazova koje sam susreo. Njihova podrška, ohrabrenje i prisutnost su mi pružili iznimnu pomoć u prevladavanju svih izazova.

Na kraju se želim zahvaliti svom mentoru Tomislavu Jaguštu na vrijednim savjetima, strpljenju i vođenju tijekom diplomskog studija i pisanja ovog rada.

SADRŽAJ

1. Uvod	1
2. Usporedba tradicionalnog i udaljenog načina održavanja	2
2.1. Tradicionalni pristup	2
2.2. Udaljeni laboratorij	3
3. Postojeći udaljeni laboratorijski sistem	5
3.1. UNED laboratorijski sistem	5
3.1.1. Mobile robots control (multi-agents system)	5
3.1.2. Level control in a four tanks system	6
3.2. Udaljeni laboratorij za učenje programiranja i robotike	8
4. Arhitektura sustava	9
4.1. Korisnička aplikacija	10
4.2. Poslužitelj za upravljanje podacima i baza podataka	11
4.3. Medijski poslužitelj	13
4.4. Arduino mikrokontroler	13
5. Implementacija sustava	15
5.1. Poslužitelj za upravljanje podacima	15
5.1.1. Modularnost sustava	15
5.1.2. Višekorisnička podrška u sustavu	19
5.1.3. Dodavanje novih uređaja	21
5.2. Medijski poslužitelj	22
5.3. Arduino mikrokontroler	25
5.3.1. Arduino UNO	25
5.3.2. Arduino-cli	26
5.4. Korisnička aplikacija	27
5.4.1. STOMP klijent	28

5.4.2. React Player	29
6. Primjer uporabe aplikacije	30
6.1. Sučelje za unos koda	30
6.2. Administratorsko sučelje	33
7. Opis pripremljenih eksperimenata	34
7.1. RGB LED dioda	34
7.2. Senzor temperature	35
7.3. Servo motor	35
8. Moguća poboljšanja i nadogradnje	37
8.1. Suradnja u rješavanju zadataka	37
8.1.1. Videopozivi ili čavrljanje	37
8.1.2. Kolaboracijske datoteke	38
8.2. Automatsko ocjenjivanje	39
9. Zaključak	41
Literatura	42

1. Uvod

Laboratorijske vježbe su ključne za razvoj znanja i vještina učenika, posebno u području znanosti, tehnologije, inženjerstva i matematike (STEM) [1]. One omogućuju učenicima da primijene teorijsko znanje i steknu praktično iskustvo koje će im biti korisno u budućim profesionalnim aktivnostima. Stoga je važno da obrazovne institucije osiguraju učenicima pristup kvalitetnim i relevantnim praktičnim vježbama. Tradicionalna metoda provođenja laboratorijskih vježbi je da se u školama ili na fakultetu osigura prostorija s potrebnom opremom, gdje učenici mogu primijeniti teorijsko znanje stečeno na predavanjima. Međutim, porastom popularnosti *online* obrazovanja, fleksibilnost je postala ključna prednost za mnoge učenike. Kako bi se omogućilo praktično učenje i u *online* okruženju, prilagodbe su napravljene i za laboratorijske vježbe. Nedostatak opreme u školama, pogotovo u ruralnim područjima, kao i nedavna pandemija koronavirusa, dodatno su aktualizirali potrebu za udaljenom nastavom i održavanjem laboratorijskih vježbi na daljinu. Za to je potrebna odgovarajuća računalna podrška.

Udaljeni laboratoriji su jedno od rješenja koja omogućuju učenicima izvođenje eksperimenata i praktičnih zadataka putem Interneta, bez potrebe za fizičkim pristupom opremi [2]. Umjesto izravne interakcije s opremom kroz fizičke radnje, kao što su pokreti ruku ili pritiskanje tipki, udaljeni laboratoriji koriste korisničko sučelje i udaljenu infrastrukturu kako bi omogućili iste interakcije na daljinu.

Ovaj diplomski rad predstavlja udaljeni laboratorij za učenje programiranja mikrokontrolera. Sustav ima podršku za Arduino mikrokontrolere, ali je osmišljen tako da omogućuje laku integraciju drugih uređaja poput MicroBita ili Lego robota. Korisnici će putem web aplikacije imati mogućnost pisanja programskog koda koji će se izvoditi na mikrokontroleru, a rezultate izvršavanja mogu pratiti u stvarnom vremenu putem videoprijenosu uživo.

2. Usporedba tradicionalnog i udaljenog načina održavanja

Tradicionalni način provođenja laboratorijskih vježbi suočava se s nekolicinom nedostataka, posebice u kontekstu udaljenog učenja i zahtijeva nova rješenja koja će pružiti bolje iskustvo učenicima [3]. U ovom poglavlju istražuju se ti nedostatci i identificiraju se specifični problemi koje sustavi za održavanje laboratorijskih vježbi na daljinu rješavaju.

2.1. Tradicionalni pristup

Analizirat će se nedostatci tradicionalnog pristupa provođenju laboratorijskih vježbi. Identificiranje ovih nedostataka pomaže razumjeti zašto rješenja za održavanje laboratorijskih vježbi na daljinu postaju sve češća.

Neki od ključnih nedostataka su:

Ograničen pristup opremi: U tradicionalnim laboratorijima nerijetko postoji ograničeni broj specijaliziranih uređaja. To znači da učenici moraju dijeliti opremu ili čekati svoj red, što može otežati pristup određenim uređajima potrebnim za izvođenje vježbi.

Preklapanje rasporeda: Ovaj se problem nadovezuje na prethodni nedostatak. Naime, u slučaju da učenici moraju dugo čekati pristup uređaju, može doći do preklapanja s drugim obavezama, što dovodi do poteškoća u pronalaženju prikladnog vremena za sudjelovanje. To može rezultirati komplikacijama kod stvaranja rasporeda ili smanjenim brojem učenika koji mogu iskusiti praktični aspekt predmeta.

Nemogućnost pristupa nastavi: Postoje slučajevi kada neki učenici nisu u mogućnosti fizički prisustvovati laboratorijskim vježbama u zakazanom

terminu. To može rezultirati propuštanjem vježbi ili potrebom za promjenom termina, što stvara izazove u usklađivanju rasporeda, posebno kada se takve situacije neplanirano pojave, kao što su neočekivane bolesti ili drugi nepredviđeni razlozi. Kao primjer može se uzeti pandemiju koronavirusa. U trenutku kada je nastava prešla na *online* oblik, određene fizičke laboratorijske vježbe nisu mogle biti održane.

Ograničen broj osoblja: Ograničenje u pogledu osoblja, poput nastavnika ili asistenata, također može biti jedan od izazova tradicionalnog pristupa laboratorijskim vježbama. Svaki termin laboratorijske vježbe zahtijeva prisutnost osoblja koje nadzire učenike i osigurava sigurnost opreme. Kako se povećava broj termina ili grupa učenika, tako se povećava i potreba za dodatnim osobljem, što može rezultirati dodatnim troškovima za obrazovnu instituciju.

Analizom ovih nedostataka može se bolje razumjeti potreba za sustavima koji će prevladati ove nedostatke i pružiti bolje iskustvo učenicima i nastavnicima.

2.2. Udaljeni laboratorij

U ovom su poglavlju navedene prednosti sustava za održavanje laboratorijskih vježbi na daljinu. Zahvaljujući primjeni modernih tehnologija i alata ovakvi sustavi uvode kreativnost i donose nekoliko prednosti u usporedbi s tradicionalnim pristupom [4] [5].

U nastavku su istaknute neke od ključnih prednosti:

Fleksibilnost pristupa: Sustavi za održavanje laboratorijskih vježbi na daljinu omogućuju učenicima da, bez obzira na svoju fizičku lokaciju, izravno izvršavaju kod na mikrokontrolerima putem Interneta. Ova fleksibilnost pruža učenicima veću autonomiju u učenju i omogućuje im da prilagode svoje iskustvo vježbanja vlastitim potrebama i interesima, bez potrebe za skupom opremom ili strogo određenim rasporedom.

Manja potreba za nadzorom: Sustavi za udaljene laboratorije pružaju prednost u tome što su samoodrživi i ne zahtijevaju stalnu prisutnost osoblja za njihovo funkcioniranje. Ovi sustavi su oblikovani da omoguće automatiziranu interakciju i upravljanje, eliminirajući potrebu za neprekidnom nadzorom. To rezultira smanjenjem troškova vezanih uz angažman asistenata i drugog osoblja, što je korisno za institucije s ograničenim resursima.

Automatsko ocjenjivanje: Automatsko ocjenjivanje predanih rješenja predstavlja ključnu prednost sustava za udaljene laboratorije. Umjesto da nastavnici ručno ispravljaju i ocjenjuju svako pojedinačno rješenje, sustav može koristiti unaprijed definirane kriterije kako bi automatski provjerio ispravnost predanog rješenja. Ova automatizacija smanjuje opterećenje nastavnika, oslobađajući vrijeme i resurse koji bi inače bili potrebni za ručno ocjenjivanje [6]. To omogućuje nastavnicima da se usredotoče na druge važne aspekte nastave, kao što su pružanje podrške i mentorstva učenicima. Također, brže i objektivnije ocjenjivanje pruža učenicima brže povratne informacije o njihovom rješenju.

Udaljeni laboratorijski sustavi pružaju nekoliko prednosti u odnosu na tradicionalni pristup. Udaljeni laboratorijski omogućavaju pristup i iskustvo laboratorijskih vježbi bez fizičke prisutnosti, potiču samostalnost učenika, smanjuju potrebu za osobljem i omogućavaju automatizaciju provjere i ocjenjivanja, što rezultira efikasnijim i fleksibilnijim obrazovnim iskustvom.

3. Postojeći udaljeni laboratorijski

Udaljeni laboratorijski su popularni već neko vrijeme i kratkim pretraživanjem se mogu pronaći razni radovi na tu temu. U nastavku su opisane neke postojeće implementacije.

3.1. UNED laboratorijski

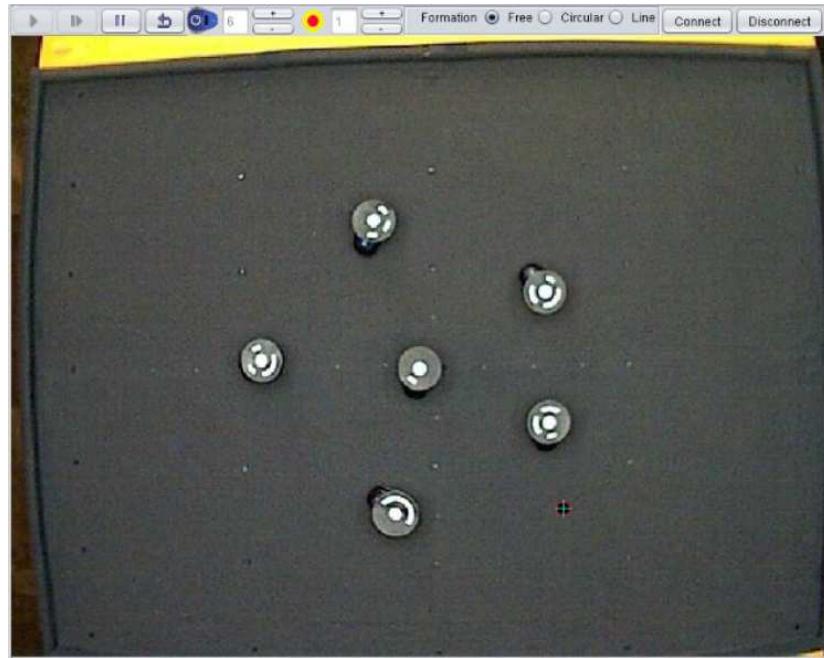
Universidad Nacional de Educación a Distancia (UNED) je Nacionalno Sveučilište za obrazovanje na daljinu. Nalazi se u Madridu, Španjolska. Kao što i samo ime kaže, usredotočeno je na obrazovanje na daljinu te stoga ima veliki broj udaljenih laboratorijskih. Na stranici <https://unilabs.dia.uned.es> se može vidjeti ponuda i opis njihovih laboratorijskih. U nastavku su ukratko opisani neki od laboratorijskih.

3.1.1. Mobile robots control (multi-agents system)

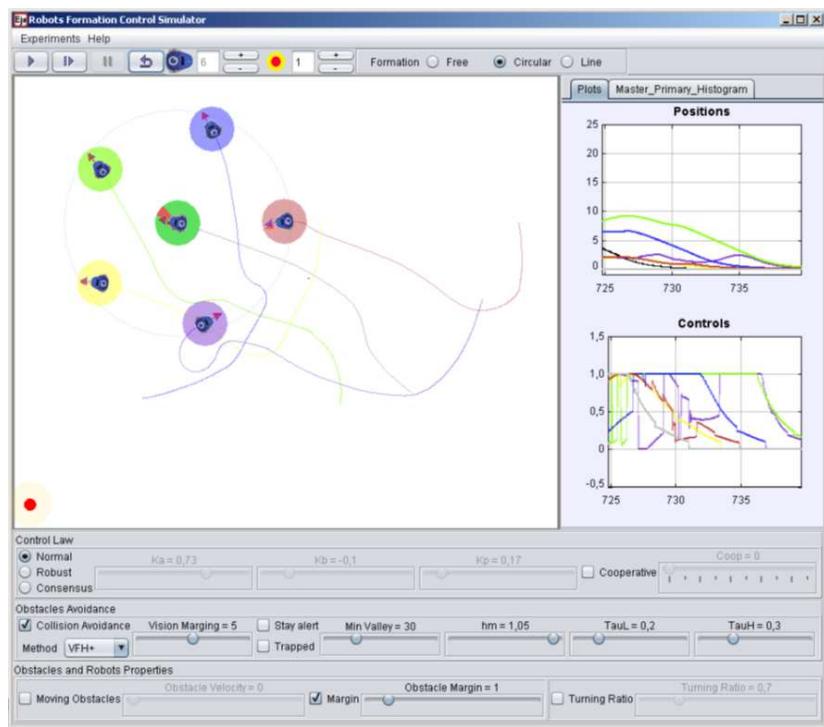
Ovaj se laboratorijski temelji na sustavu s više agenata koji se sastoje od skupine malih robota na kotačima (*Moway*) (Slika 3.1). Ovi roboti sastoje se od pet modula: sustava za obradu, pogonskog sustava, grupe senzora i indikatora, sustava napajanja i priključka za proširenje za radiofrekventnu bežičnu komunikaciju [7]. Roboti mogu međusobno komunicirati, izbjegavati prepreke i surađivati u arhitekturi nadzornik-podređeni. Korisnici upravljaju sustavom koristeći korisničko sučelje prikazano na slici 3.2

Ciljevi ovog laboratorijskog su:

- proučavanje statičkih i dinamičkih svojstava sustava
- proučavanje i projektiranje sustava upravljanja u brzini
- proučavanje i projektiranje upravljačkog sustava u položaju



Slika 3.1: Prikaz Moway robota



Slika 3.2: Korisničko sučelje za *Mobile robots control* laboratorij

3.1.2. Level control in a four tanks system

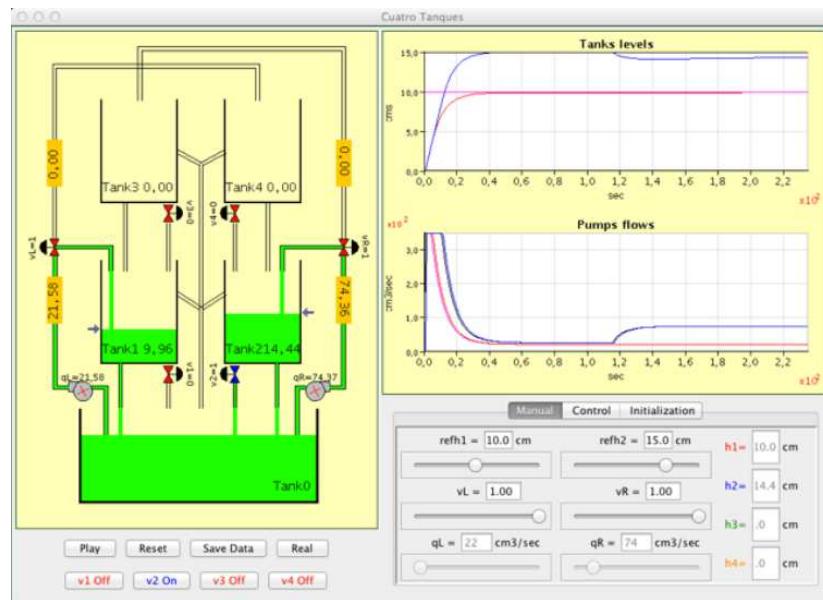
Ovaj laboratorij jedan je od najčešće korištenih u učenju više-varijabilnog upravljanja procesima. Sustav se sastoji od četiri spremnika. (Slika 3.3) Na dnu

svakog od njih nalazi se izlaz poznatog presjeka i još jedan izlaz nepoznatog presjeka, reguliran ventilom koji omogućuje ili onemogućuje odgovarajuću perturbaciju. Sustav također ima dva trosmjerna ventila koji omogućuju regulaciju toka. Tok dolazi iz dvije pumpe i ulazi u svaki od spremnika [8]. Na slici 3.4 prikazano je korisničko sučelje. Ciljevi ovog laboratorija su:

- proučavanje dinamičkih karakteristika sustava
- dizajn više-varijabilnog regulatora za regulaciju razine jednog spremnika



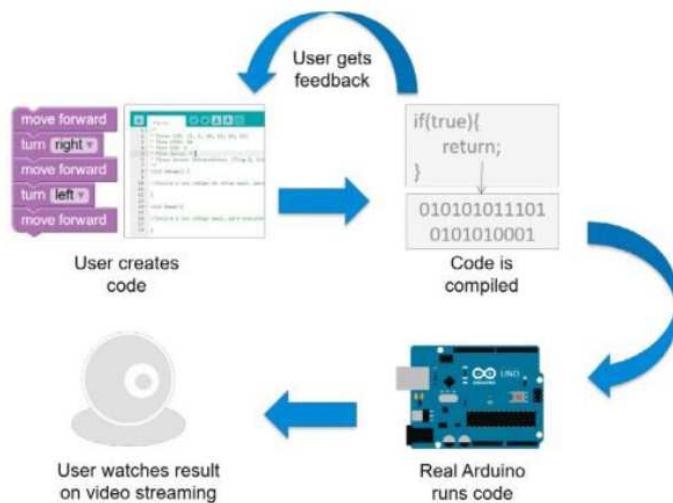
Slika 3.3: Sustav laboratorija *Level control in a four tanks system*



Slika 3.4: Korisničko sučelje za laboratorij *Level control in a four tanks system*

3.2. Udaljeni laboratorij za učenje programiranja i robotike

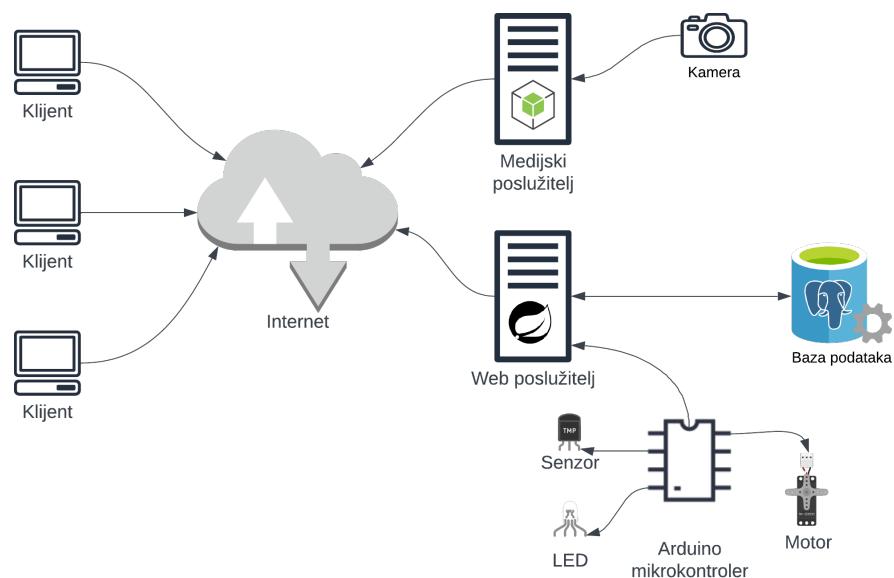
Ovaj laboratorij koji služi za učenje programiranja Arduino pločice i upoznavanje korisnika s osnovnim pojmovima o robotici. Korisnik se može upoznati s radom raznih komponenti, poput: servo motora, koračnog motora, potenciometra, LED sijalica, LCD-a, senzora (temperature, vlage, ultrazvučni senzor). Korisnik putem sučelja slaže programski kod, koji se prevodi (eng. *compile*) i šalje na Arduino. Korisnik cijelo vrijeme prati sustav video-prijenosom uživo (eng. *livestream*) [9] (Slika 3.5).



Slika 3.5: Grafički prikaz sustava Arduino laboratorija

4. Arhitektura sustava

U ovom poglavlju opisana je arhitektura ostvarenog sustava, s tehnologijama i komponentama od kojih se sustav sastoji. Naglasak je na prikazu arhitekte više razine. Konkretni implementacijski detalji objašnjeni su u poglavlju 5. Okvirna arhitektura sustava prikazana je na slici 4.1. S lijeve strane nalazi se korisnička aplikacija koja omogućuje korisnicima pristup našem sustavu. S desne strane nalaze se poslužitelji i baza podataka. Jedan od poslužitelja služi kao poslužitelj medija i na njega je spojena kamera koja snima laboratorijsku opremu. Drugi poslužitelj zadužen je za primanje i slanje podataka, spremanje u bazu podataka te prevođenje i slanje koda na povezani mikrokontroler. Na ovom primjeru koristi se Arduino mikrokontroler. Na mikrokontroler su povezani ostali uređaji poput senzora i motora.



Slika 4.1: Arhitektura sustava

4.1. Korisnička aplikacija

Korisnička aplikacija razvijena je korištenjem *JavaScript* biblioteke (eng. *library*) *React* i programskog jezika *TypeScript*. Uz njih se koriste i pomoćne biblioteke: *axios* i *Mantine*. Aplikacija je sučelje kojim se upravlja udaljenim laboratorijem.

React je popularna *JavaScript* biblioteka za izgradnju korisničkih sučelja [10]. Ona omogućava razvoj web aplikacija putem komponentne arhitekture. Komponente su modularne i ponovno iskoristive jedinice koda koje olakšavaju organizaciju i održavanje projekta. *React* koristi virtualni DOM (eng. *Document Object Model*) da bi učinkovito ažurirao samo promijenjene dijelove sučelja, čime postiže bržu i efikasniju izvedbu.

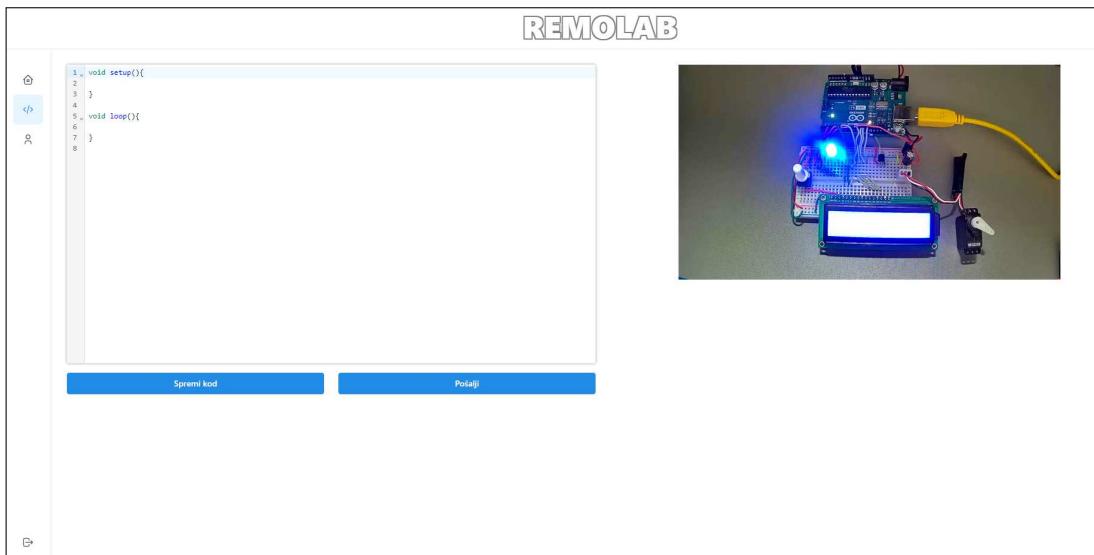
TypeScript je nadogradnja *JavaScripta* koja donosi statičku tipizaciju i poboljšava sigurnost, održavanje i razumijevanje koda. To znači da se definiraju tipovi podataka koje varijable mogu imati, što pomaže u otkrivanju grešaka pri pisanju koda. *TypeScript* se prevodi u *JavaScript* prilikom izvođenja, što znači da se može koristiti u svim modernim web-preglednicima i okolinama izvršavanja *JavaScripta*.

Axios je *JavaScript* biblioteka koja se često koristi za izvođenje HTTP zahtjeva prema poslužiteljima [11]. Pruža jednostavno sučelje za slanje zahtjeva i primanje odgovora, podržavajući različite HTTP metode poput: GET, POST, PUT, DELETE i druge. *Axios* također podržava mogućnost postavljanja zaglavlja zahtjeva, rukovanje pogreškama te automatsko pretvaranje podataka u različite formate kao što su JSON (eng. *JavaScript Object Notation*).

Mantine je *React* komponentna biblioteka koja pruža mnoge gotove komponente i alate za izgradnju modernih korisničkih sučelja [12]. Cilj je na jednostavnosti korištenja, prilagodljivosti i pružanja odgovarajućih stilova koji se mogu lako prilagoditi. *Mantine* nudi širok spektar komponenti poput tipografije, formi, gumbova, kartica, modla i još mnogo toga. Također pruža tematsku podršku za jednostavno prilagođavanje izgleda i stila komponenti. *Mantine* je popularan izbor za razvoj *React* aplikacija zbog svoje jednostavnosti korištenja, dobre dokumentacije i aktivne zajednice koja podržava daljnji razvoj i održavanje biblioteke.

Kako bi pristupili funkcionalnostima sustava, korisnici se moraju prijaviti. Sustav podržava dvije različite uloge: STUDENT i ADMIN, gdje ADMIN predstavlja nastavnika. Unutar korisničke aplikacije dostupni su tutorijali koji služe kao uvod za korisnike u osnovne komponente i funkcionalnosti sustava. Tutorijali sadrže informacije o tome kako se određeni mikrokontroler programira i kako se upravlja uređajima spojenim na mikrokontroler. Nastavnici imaju mogućnost kreiranja novih

tutorijala, uređivanja postojećih i uklanjanje neželjenih tutorijala. Primjeri tutorijala i eksperimentalnih zadatka detaljnije su opisani u poglavlju 7. Glavna funkcionalnost korisničke aplikacije je omogućiti učenicima da unesu vlastiti kod i pošalju ga na izvršavanje. Rezultati izvršavanja koda prikazuju se pomoću videoprijenosu uživo (eng. *livestream*). To omogućava korisnicima da prate izvođenje svog koda u stvarnom vremenu. Slika 4.2 prikazuje sučelje za unos koda.



Slika 4.2: Sučelje za unos koda

4.2. Poslužitelj za upravljanje podacima i baza podataka

Poslužitelj (eng. *server*) za upravljanje podacima razvijen je korištenjem Java programskog jezika i *Spring* radnog okvira (eng. *framework*). Za bazu podataka koristi se *PostgresSQL*, a za komunikaciju s korisničkom aplikacijom koristi se *REST* arhitekturalni stil.

Spring je popularni radni okvir, temeljen na Java ili Kotlin programskom jeziku, koji pruža sveobuhvatan skup alata i biblioteka za izgradnju robusnih i skalabilnih aplikacija [13]. *Spring* olakšava razvoj aplikacija svojom podrškom za različite module, kao što su ubrizgavanje ovisnosti (eng. *dependency injection*), upravljanje transakcijama, sigurnost i mnoge druge.

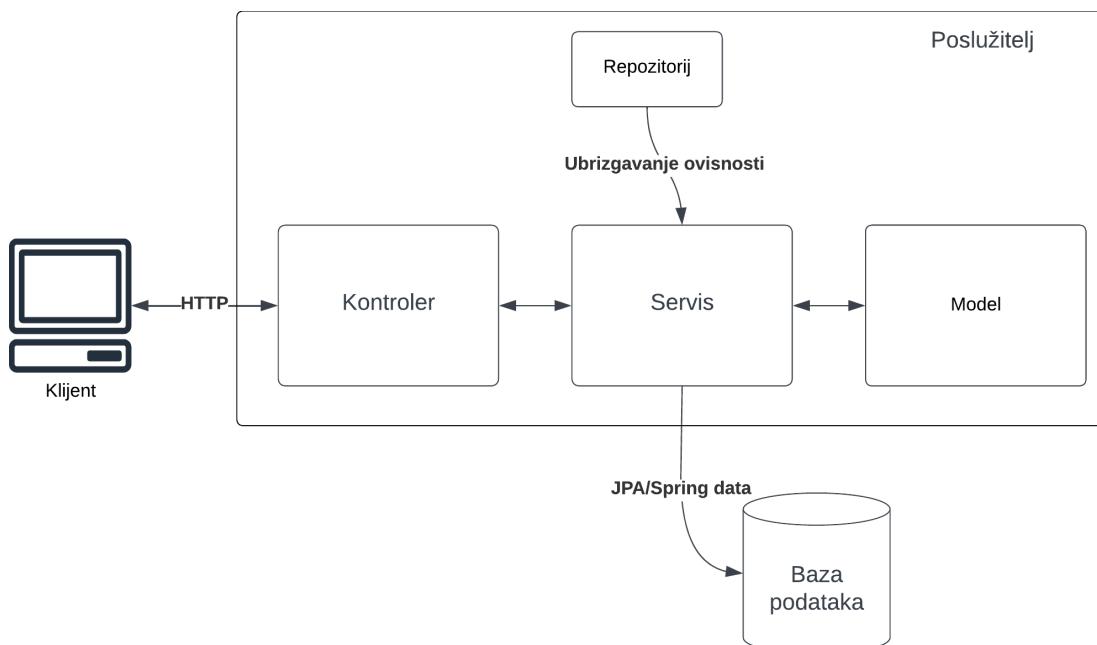
PostgresSQL je baza podataka otvorenog koda (eng. *open source*) temeljena na relacijskom modelu [14]. Ona pruža napredne značajke, poput podrške za

transakcije, upravljanja konzistentnošću podataka i podrške za različite vrste podataka. *PostgreSQL* je pouzdan i skalabilan sustav za pohranu i upravljanje podacima.

REST (eng. *Representational State Transfer*) je arhitekturalni stil koji se koristi za dizajniranje mrežnih servisa. On se oslanja na HTTP (eng. *Hypertext Transfer Protocol*) protokol i koristi jednostavne, standardne metode prijenosa podataka između klijenta i poslužitelja. REST arhitektura omogućava razmjenu resursa putem jedinstvenih *URI-ja* (eng. *Uniform Resource Identifier*) i podržava različite formate prijenosa podataka kao što su JSON i XML (*Extensible Markup Language*).

Glavna uloga ovog poslužitelja je spremanje podataka o korisnicima, tutorijalima i ostalim relevantnim informacijama. Poslužitelj prima zahtjeve od korisničke aplikacije i obrađuje ih da bi omogućio upis, dohvaćanje i manipulaciju tim podacima. Slika 4.3 prikazuje arhitekturu poslužitelja. Dodatno, poslužitelj je odgovoran za primanje zahtjeva koji sadrže kodove namijenjene izvršavanju na mikrokontroleru. Poslužitelj preuzima te zahtjeve, vrši prevođenje (eng. *compile*) koda u odgovarajući format za mikrokontroler te šalje rezultirajući kod na mikrokontroler. Mikrokontroler je povezan serijski na poslužitelj putem *USB* (eng. *Universal Serial Bus*).

Poslužitelj je dizajniran na način koji omogućuje jednostavno dodavanje podrške za nove vrste mikrokontrolera. Potrebno je samo implementirati apstraktni razred koji provodi potrebne operacije za prevođenje koda i slanje na određeni mikrokontroler.



Slika 4.3: Arhitektura poslužitelja

4.3. Medijski poslužitelj

Kako bi se učenicima omogućilo praćenje događanja na mikrokontroleru i određivanje ispravnosti koda, potrebno im je pružiti povratnu informaciju. Sustav koristi videoprijenos uživo (eng. *livestream*). Ovaj način pruža najbolji prikaz stvarnog stanja mikrokontrolera jer omogućuje korisnicima da uživo prate cijelokupan uređaj, uključujući i druge povezane komponente. Budući da korisnici nemaju izravan pristup uređaju, ovaj oblik povratne informacije im približava iskustvo korištenja. U tu svrhu koristi se *node-media-server* biblioteka za konfiguraciju i implementaciju medijskog poslužitelja. Ova biblioteka omogućuje strujanje (eng. *streaming*) video sadržaja prema korisnicima. Također koristi se *FFmpeg* (*Fast Forward Moving Pictures Expert Group*) kako bi video s kamere bio proslijeđen na medijski poslužitelj.

Node Media Server (NMS) je biblioteka otvorenog koda koja omogućuje prijenos videozapisa putem nekoliko protokola: **RTMP**, **HLS**, **FLV** i **DASH** [15]. *Node Media Server* omogućuje stvaranje vlastitog medijskog poslužitelja u *Node.js* okruženju. Sustav za pristup videoprijenosu koristi **FLV** protokol jer on ima najkraće kašnjenje od ovdje navedenih.

FFmpeg je popularan softverski alat za obradu i manipulaciju multimedijalnim sadržajem kao što su audio i video zapisi [16]. Podržava širok raspon formata i kodeka te nudi razne mogućnosti za pretvorbu, kodiranje, dekodiranje, snimanje, strujanje i još mnogo toga. *FFmpeg* koristi se za prikupljanje videa s web-kamere te strujanje tog videa u stvarnom vremenu na odabrani *streaming* poslužitelj (ranije spomenuti *Node Media Server*). Koristeći odgovarajuće naredbe i parametre moguće je postaviti kvalitetu videa, brzinu prijenosa, izbor kodeka i druge postavke.

4.4. Arduino mikrokontroler

Arduino je otvorena platforma za elektroniku temeljena na jednostavnoj hardverskoj i softverskoj infrastrukturi [17]. Arduino pločice mogu čitati ulazne signale, na primjer: svjetlost na senzoru ili pritisak na tipku te ih pretvoriti u izlazne signale poput pokretanja motora ili aktiviranja LED svjetla. Mikrokontroler na Arduino pločici izvršava upute koje mu se šalju putem Arduino programskog jezika (temeljen na programskom jeziku *Wiring* koji za temelj koristi *C++*) i Arduino softvera (IDE). Arduino je nastao na *Institutu za interakciju u Ivrei* (Italija) kao jednostavan alat za brzo prototipiranje, namijenjen učenicima. Slika 4.4 prikazuje Arduino mikrokontroler. Arduino se vremenom prilagođavao novim potrebama i

izazovima te je razvio svoju ponudu od jednostavnih 8-bitnih pločica do proizvoda za primjene u području *Interneta stvari* (eng. *Internet of Things*), nosivih tehnologija, 3D printanja i ugrađenih okruženja.

Sustav za održavanje laboratorija ne koristi Arduino IDE (*Integrated Development Environment*) za prevodenje i prenošenje programskog koda na mikrokontroler, već koristi *arduino-cli*.

Arduino-cli (*Arduino Command Line Interface*) je alat koji korisnicima pruža mogućnost upravljanja Arduino platformom putem naredbenog retka. Omogućuje jednostavnu interakciju s Arduino pločicama, uključujući učitavanje programa, upravljanje bibliotekama, provjeru sintakse koda i praćenje serijske komunikacije.



Slika 4.4: Arduino UNO mikrokontroler

5. Implementacija sustava

Ovo poglavlje opisuje implementacijske detalje pojedinih dijelova sustava. Navodi probleme koje je bilo potrebno riješiti i objašnjava način na koji su riješeni. Isjećima koda bit će pružen dublji uvid u razloge i razmišljanja koja su stajala iza implementacije.

5.1. Poslužitelj za upravljanje podacima

Sustav mora biti dizajniran tako da bude modularan i omogući lako dodavanje i korištenje novih vrsta uređaja. Ovo poglavlje detaljno opisuje kako je ostvarena ta modularnost i koje su tehnike korištene za podršku ovom zahtjevu. Također, sustav mora podržavati rad više korisnika istovremeno, što će također biti obuhvaćeno u ovom poglavlju.

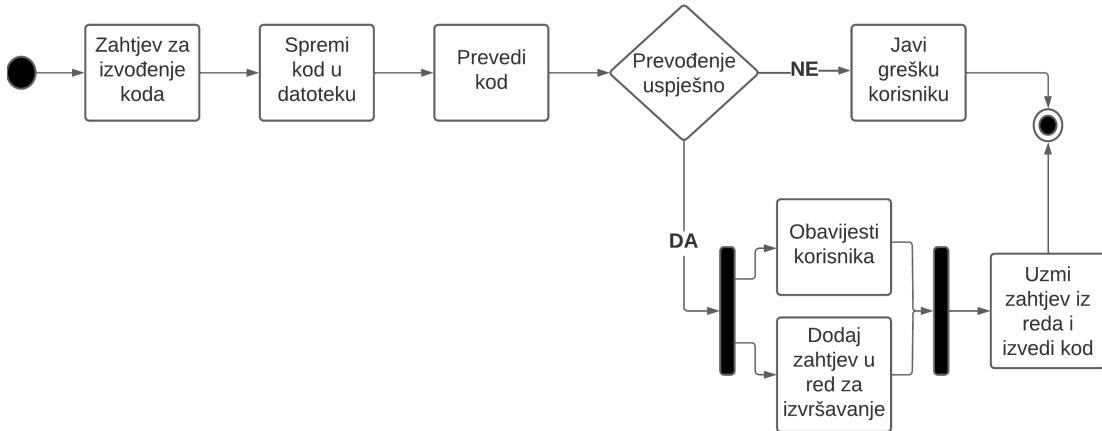
5.1.1. Modularnost sustava

Glavni cilj je omogućiti jednostavno i bezbolno dodavanje različitih vrsta mikrokontrolera. U primjeru koji se opisuje u ovom radu koristi se Arduino mikrokontroler, ali sustav je osmišljen tako da se lako mogu integrirati i druge platforme poput MicroBita ili Lego robota. Konfiguracija se svodi na implementiranje načina prevođenja (eng. *compile*) koda i prijenosa (eng. *upload*) koda na određeni mikrokontroler. Tako ostatak sustava ostaje nepromijenjen, omogućavajući prilagodljivost i fleksibilnost u izboru različitih mikrokontrolera za laboratorijske svrhe.

S obzirom na to da postoje različiti načini prijevoda i slanja koda na mikrokontrolere, koji ovise o pojedinom mikrokontroleru, važno je definirati postupak koji će biti primjenjiv na sve uređaje. Ključno je uspostaviti redoslijed operacija koje su neophodne za uspješno izvršavanje koda na mikrokontroleru.

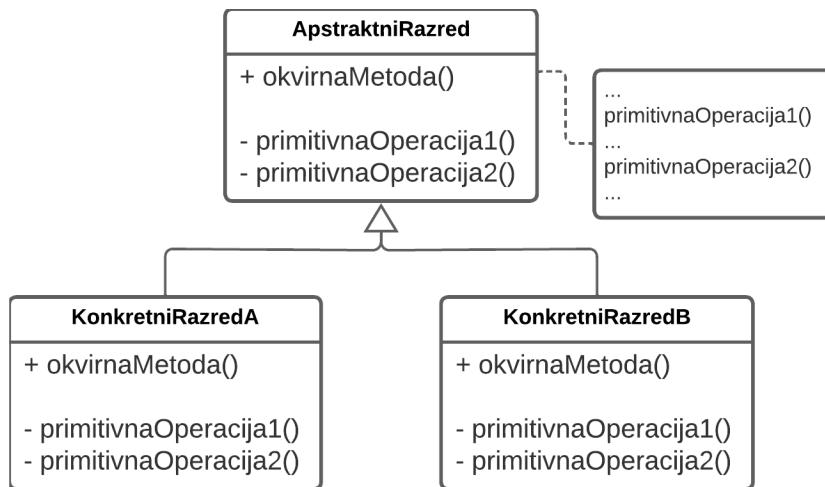
Prvo, korisnik šalje kod koji želi izvršiti, kod se zatim sprema u odgovarajuću

datoteku. Nakon toga, kod se prevodi i provjerava da bi se osigurala njegova ispravnost prije slanja i izvršavanja na uređaju. Konačno, prevedeni kod se prenosi na ciljni uređaj kako bi se mogao izvršiti. Na slici 5.1 prikazan je dijagram aktivnosti zahtjeva za izvršavanje koda.



Slika 5.1: Dijagram aktivnosti

Kako bi bilo omogućeno implementiranje pojedinih koraka algoritma koristi se oblikovni obrazac **okvirna metoda** (eng. *Template Method*). Okvirna metoda je oblikovni obrazac koji omogućuje definiranje osnovne strukture algoritma, dok detalje implementacije ostavlja potklasama [18]. Slika 5.2 prikazuje strukturni dijagram obrasca. Ovaj obrazac promiče ponovnu upotrebu koda, fleksibilnost i promjenjivost, omogućavajući potklasama da prilagode i prošire pojedine korake algoritma. To je korisno kada imamo isti redoslijed koraka u algoritmu, ali se implementacije tih koraka razlikuju.



Slika 5.2: Strukturni dijagram oblikovnog obrasca Okvirna metoda

Razred (eng. *class*) `CodeHandler` prikazan je u isječku koda 5.1. Razred ima tri članske varijable. `codeInputRequest` sadrži podatke o korisniku te kod kojeg je korisnik poslao. `requestProcessor` sadrži red u koji se dodaju zahtjevi i on se brine o redoslijedu izvršavanja kodova na mikrokontroleru (implementacija `requestProcessor`a detaljnije je objašnjena u potpoglavlju 5.1.2). Varijabla `filePath` služi za spremanje putanje do datoteke u kojoj je spremjen kod.

Metoda `compileAndAddToQueue()` je ovdje okvirna metoda i određuje redoslijed izvođenja pojedinih operacija. Postoje četiri apstraktne metode koje je potrebno implementirati kako bi okvirna metoda ispravno funkcionalala:

- **`saveToFile()`** - zadaća metode je spremiti korisnikov kod u datoteku u ispravnom formatu. Ovdje se određuje koji će biti tip datoteke (npr. za Arduino bi to bila `.ino` datoteka) i mjesto gdje će biti spremljena. Neki mikrokontroleri zahtijevaju posebni način spremanja datoteke. Za Arduino je potrebno datoteku spremiti u direktorij istog naziva kao datoteka. Metoda vraća putanju do datoteke.
- **`compile()`** - zadaća metode je obaviti prevođenje korisničkog koda, uglavnom pozivanjem nekog alata. Metoda vraća poruku koju prevoditelj (eng. *compiler*) vrati. Ta se poruka prosljeđuje metodi `isSuccessfullyCompiled()`.
- **`isSuccessfullyCompiled(String compileMessage)`** - metoda iz poruke prevoditelja zaključuje je li prevođenje bilo uspješno ili nije.
- **`upload()`** - metoda koja je zaslužena za prenošenje koda na mikrokontroler. Ova metoda se ne koristi u okvirnoj metodi `compileAndAddToQueue()` već nju poziva `requestProcessor`. Detaljnije o tome u potpoglavlju 5.1.2.

Okvirna metoda `compileAndAddToQueue()` određuje sljedeći redoslijed operacija. Prvo je potrebno spremiti kod u datoteku. Nakon toga se kod prevodi i spremi se poruka koju prevoditelj (eng. *compiler*) vrati. Na osnovu te poruke sustav određuje hoće li korisniku javiti grešku s priloženom porukom prevoditelja, kako bi korisnik mogao otkriti gdje je greška, ili će dodati zahtjev za izvođenje u red. Na liniji 12 u isječku koda 5.1 poziva se metoda `enqueueRequest(CodeHandler codeHandler)` nad objektom `requestProcessor` i predaje joj se referenca na sami objekt (`this`). Predaje se referenca kako bi kasnije mogla biti pozvana metoda `upload()`. Nakon dodavanja zahtjeva u red, šalje se odgovor korisniku da je prevođenje uspješno i da je zahtjev dodan u red.

```
1 public abstract class CodeHandler {
2     private final CodeInputRequest codeInputRequest;
3     private final RequestProcessor requestProcessor;
4     private String filePath;
5
6     // Compiles code and adds request to a queue
7     public CodeInputResponse compileAndAddToQueue() {
8         this.filePath = saveToFile();
9         String compilerMessage = compile();
10
11        if (isSuccessfullyCompiled(compilerMessage)) {
12            requestProcessor.enqueueRequest(this);
13            return CodeInputResponse.SuccessfulResponse();
14        } else {
15            return CodeInputResponse.ErrorResponse(compilerMessage);
16        }
17    }
18
19    // Saves code to file with right extension
20    protected abstract String saveToFile();
21
22    // Compiles code
23    protected abstract String compile();
24
25    // Checks from compile message if compilation was successful or not
26    protected abstract boolean isSuccessfullyCompiled(String compileMessage);
27
28    // Uploads code to the board
29    public abstract boolean upload();
30 }
```

Isječak koda 5.1: Apstraktni razred CodeHandler

Isječak koda 5.2. prikazuje implementaciju metode `upload()` za Arduino mikrokontroler. Najvažnija linija ovdje je `String command ="arduino-cli upload ..."` jer se tu postavlja naredba koja će se izvršiti i odraditi slanje koda na Arduino mikrokontroler. Alat *arduino-cli* i njegove naredbe detaljnije su objašnjeni u poglavljju 5.3.

```
1 @Override
2 public boolean upload() {
3     try {
4         log.info("Uploading code to Arduino board: ");
5         String path = "\\" + getFilePath() + "\\";
6         String command = String
7             .format("arduino-cli upload -p %s --fqbn %s %s",
8                 PortsConstants.COM5,
9                 ArduinoConstants.ARDUINO_UNO_FQBN,
10                path);
11
12     ProcessBuilder pb = new ProcessBuilder("cmd", "/c", command);
13     pb.redirectErrorStream(true);
14     pb.start();
15
16 } catch (IOException e) {
17     log.error("Error trying to upload to Arduino board.");
18     return false;
19 }
20 return true;
21 }
```

Isječak koda 5.2: Metoda upload() za ArduinoCodeHandler

5.1.2. Višekorisnička podrška u sustavu

Budući da je ovo sustav koji je namijenjen za više korisnika, primjerice u školskom okruženju, važno je podržavati istovremeni rad više korisnika. S obzirom na to da se koristi samo jedan mikrokontroler, potrebno je pronaći način za raspodjelu zahtjeva korisnika kako bi svaki korisnik dobio svoje vrijeme za izvršavanje koda i mogao pratiti rezultate svog rada.

Jedno rješenje je dodavanje korisničkih zahtjeva u red i periodično izvršavanje tih zahtjeva. Tako svaki korisnik može predati svoj kod, a sustav će ga, nakon uspješnog prevodenja, uvrstiti u red čekanja. Periodično će se izvršavati zahtjevi iz reda, osiguravajući pravednu raspodjelu vremena između korisnika. Tako će svi korisnici imati priliku izvršiti svoj kod i pratiti rezultate izvođenja. Prije izvođenja koda, sustav obavijesti korisnika da se njegov kod trenutno izvršava, kako bi korisnik znao pratiti videoprijenos. U sustavu je za to zadužen razred RequestProcessor. Razred je prikazan u isječku koda 5.3.

```
1 public class RequestProcessor {
2     private static final long SESSION_DURATION_MS = 30_000;
3     private final Queue<CodeHandler> requestQueue = new LinkedList<>();
4     private final WebSocketService webSocketService;
5     public static final Map<Integer, Boolean> userRequestStatus =
6         new HashMap<>();
7
8     public void enqueueRequest(CodeHandler request) {
9         requestQueue.offer(request);
10        Integer userId = request.getCodeInputRequest().getUserId();
11
12        changeUserRequestQueueStatus(userId);
13        notifyUserAboutEnqueue(userId, Notification.ADDED_TO_QUEUE);
14    }
15
16    @Scheduled(fixedDelay = SESSION_DURATION_MS)
17    public void processRequests() {
18        if (requestQueue.isEmpty()) return;
19
20        log.info("Started processRequests.");
21        CodeHandler request = requestQueue.poll();
22        Integer userId = request.getCodeInputRequest().getUserId();
23
24        notifyUserAboutUpload(userId, Notification.UPLOADING);
25        changeUserRequestQueueStatus(userId);
26
27        request.upload();
28    }
29
30    private void notifyUserAboutUpload(Integer userId, String message) {
31        webSocketService.notifyUserAboutUpload(userId, message);
32    }
33
34    private void notifyUserAboutEnqueue(Integer userId, String message) {
35        webSocketService.notifyUserAboutEnqueue(userId, message);
36    }
37
38    private void changeUserRequestQueueStatus(Integer userId) {
39        boolean hasRequest = userRequestStatus.getOrDefault(userId, false);
40        userRequestStatus.put(userId, !hasRequest);
41    }
42
```

```

43     public static boolean userAlreadyHasRequestInQueue(Integer userId) {
44         return userRequestStatus.getOrDefault(userId, false);
45     }
46 }
```

Isječak koda 5.3: Razred RequestProcessor

Glavni dio ovog razreda je red `requestQueue` i metoda `processRequests()`. Red je struktura podataka koja predstavlja kolekciju elemenata organiziranih prema principu *prvi unutra, prvi van* (eng. *first-in, first-out*, skraćeno *FIFO*). To znači da se elementi dodaju na kraj reda, a uklanjuju s početka reda. Važno je primijetiti da su objekti, koji se spremaju u red, tipa `CodeHandler`, koji je apstraktan razred. To omogućuje dodavanje različitih implementacija tog razreda u red, poput `ArduinoCodeHandler` ili `MicrobitCodeHandler`. Zahvaljujući tome, `RequestProcessor` može funkcionirati s različitim mikrokontrolerima, bez dodatnih izmjena.

Metoda `processRequests()` zadužena je za obrađivanje korisničkih zahtjeva. Periodički uzima zahtjev iz reda i pokreće metodu `upload()` te obavještava korisnika da se kod trenutno izvršava. Obavijesti se šalju koristeći *web-socket*. *Spring* anotacija `@Scheduled` s postavljenim parametrom `fixedDelay` omogućuje periodičko obavljanje metode u zasebnog dretvi (eng. *thread*). Vrijeme intervala između obrađivanja zahtjeva korisnika određeno je varijablom `SESSION_DURATION_MS`, koja predstavlja razdoblje izraženo u milisekundama.

Trenutno korisnik može imati samo jedan zahtjev u redu koji čeka na obradu. Stanje korisnika spremi se u strukturi `userRequestStatus`. To je mapa koja sadrži `ID` korisnika i `boolean` vrijednost ovisno o tome postoji li zahtjev tog korisnika u redu ili ne postoji.

5.1.3. Dodavanje novih uređaja

U ovom će poglavlju biti objašnjeno što je sve potrebno napraviti kako bi se dodala podrška za novi mikrokontroler. Za primjer je uzet Microbit.

- Potrebno je napraviti novi razred `MicrobitCodeHandler` i naslijediti apstraktni razred `CodeHandler`. To će zahtijevati implementaciju svih apstraktnih metoda: `isSuccessfullyCompiled()`, `saveToFile()`, `compile()`, i `upload()`

2. Nakon toga, potrebno je napraviti novu putanju u kontroleru te kreirati MicrobitCodeHandler objekt i pozvati metodu compileAndUploadCode(...) iz servisa codeInputService i predati joj novostvoreni objekt. Primjer metode kontrolera prikazan je u isječku 5.4

```

1 @PostMapping("/microbit")
2 ResponseEntity<CodeInputResponse> compileMicrobitCode(
3     @RequestBody CodeInputRequest codeInputRequest) {
4
5     log.info("Received microbit codeInput: {}", codeInputRequest);
6     MicrobitCodeHandler microbitCodeHanlder = new MicrobitCodeHandler(
7         codeInputRequest,
8         requestProcessor);
9
10    CodeInputResponse response = codeInputService
11        .compileAndUploadCode(microbitCodeHanlder);
12    return ResponseEntity.ok(response);
13 }
```

Isječak koda 5.4: Razred Metoda kontrolera

5.2. Medijski poslužitelj

Medijski poslužitelj (eng. *media server*) je implementiran koristeći *node-media-server* i *FFmpeg*. Snimanje videozapisa vrši se putem standardne web-kamere koja je povezana s računalom na kojem je pokrenut poslužitelj. Za prikupljanje videozapisa s kamere i slanje na medijski poslužitelj koristi se *FFmpeg*.

FFmpeg se koristi za prikupljanje videozapisa s uređaja "EasyCamera" koji je spojen na računalo, te ih kodira koristeći libx264 kodek s postavkama za brzo kodiranje i minimalno kašnjenje, te šalje u realnom vremenu putem RTMP protokola na adresu `rtmp://localhost/live/my-stream`. Na toj adresi "osluškuje" medijski poslužitelj. Naredba se pokreće sljedećim parametrima:

- `-f dshow`: Označava ulazni format, u ovom slučaju *dshow* se koristi za hvatanje videozapisa s DirectShow uređaja.
- `-i video="EasyCamera"`: Specifikacija ulaznog uređaja, u ovom slučaju se koristi web-kamera naziva "EasyCamera".

- `-c:v libx264`: Odabir video kodeka za kompresiju videozapisa. Ovdje se koristi "libx264", popularni H.264 kodek.
- `-preset ultrafast`: Odabir brzinskog profila za kompresiju. *Ultrafast* je postavka koja prioritet daje brzini, ali može rezultirati većom veličinom datoteke.
- `-tune zerolatency`: Postavljanje profila optimiziranog za minimalno kašnjenje. Ovo je korisno za prijenos uživo gdje je minimalna latencija važna.
- `-b:v 1000k`: Odabir ciljane brzine prijenosa videozapisa. Ovdje se koristi 1000 kilobita po sekundi.
- `-fflags nobuffer`: Isključuje međuspremnik, što znači da nema međuspremanja videozapisa prije slanja. Također može smanjiti kašnjenje, ali su mogući gubitci.
- `-probesize 500000`: Postavlja veličinu "sonde" za pretragu u video datoteci. Veća vrijednost može poboljšati prepoznavanje formata.
- `-g 30`: Postavlja broj slika ključa (*I-frame*) u sekundi. Ovdje je postavljeno na 30 ključnih slika u sekundi.
- `-an`: Isključuje audio snimanje, što znači da se neće snimati zvuk.
- `-f flv`: Odabir izlaznog formata, ovdje je postavljen na *flv* (*Flash Video*).
- `rtmp://localhost/live/my-stream`: Odredište za strujanje (eng. *streaming*) videa. Ovdje je postavljeno na lokalni RTMP poslužitelj s putanjom */live/my-stream*. Medijski poslužitelj osluškuje na ovoj putanji.

Da bi se video sadržaj proslijedio korisnicima potrebno je konfigurirati medijski poslužitelj. Isječak koda 5.5 prikazuje konfiguraciju medijskog poslužitelja.

```

1 const config = {
2   rtmp: {
3     port: 1935,
4     chunk_size: 60000,
5     gop_cache: true,
6     ping: 30,
7     ping_timeout: 60
8   },
9   http: {
10    port: 8000,
11    mediaroot: '/media/steamvideo',
12    webroot: './www',

```

```
13     allow_origin: '<izvor korisnicke aplikacije>'  
14   },  
15 };
```

Isječak koda 5.5: Konfiguracija medijskog poslužitelja

- **rtmp:** Ovaj dio konfiguracije odnosi se na postavke RTMP protokola (eng. *Real-Time Messaging Protocol*).

Definiraju se sljedeći parametri:

- **port:** 1935 - Vrata (eng. *port*) koja se koriste za RTMP protokol za *streaming* medija. Postavljen je na 1935.
- **chunk_size:** 60000 - Veličina paketa (eng. *chunk*) u bajtovima koji se koristi prilikom slanja videozapisa putem RTMP protokola. Postavljen je na 60000 bajtova.
- **gop_cache:** true - Parametar koji omogućava pohranjivanje ključnih slika (eng. *Group of Pictures*) u predmemoriju kako bi se poboljšala kvaliteta prijenosa. Postavljen je na `true`.
- **ping:** 30 - Vremenski interval (u sekundama) između slanja *ping* zahtjeva na RTMP vezi kako bi se provjerilo da je veza aktivna. Postavljen je na 30 sekundi.
- **ping_timeout:** 60 - Vremensko ograničenje (u sekundama) za odgovor na *ping* zahtjev na RTMP vezi. Ako ne primi odgovor unutar tog vremena, veza se smatra prekinutom. Postavljen je na 60 sekundi.

- **http:** Ovaj dio konfiguracije odnosi se na postavke HTTP protokola za web pristup.

Definiraju se sljedeći parametri:

- **port:** 8000 - Port koji se koristi za HTTP protokol za pristup medijskom poslužitelju putem preglednika. Postavljen je na 8000.
- **mediaroot:** '/media/steamvideo' - Putanja do mape koja sadrži medijske datoteke koje će biti dostupne putem poslužitelja. Postavljen je na '/media/steamvideo'.
- **webroot:** './www' - Putanja do mape koja sadrži web stranice i resurse povezane s medijskim poslužiteljem. Postavljen je na './www'.
- **allow_origin:** '<izvor korisnicke aplikacije>' - Opcija koja ograničava pristup poslužitelju. Postavljaju se izvori s

kojih se želi dozvoliti pristup. Moguće je koristiti zamjenski znak "*" (eng. *wildcard*) kako bise omogućio pristup poslužitelju sa svih izvora.

5.3. Arduino mikrokontroler

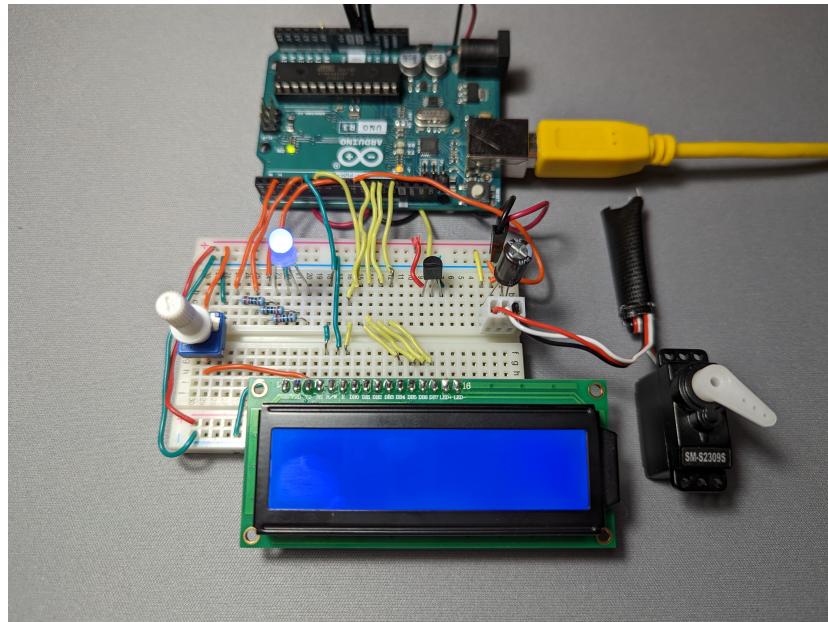
Sustav je integriran s Arduino mikrokontrolerom, pružajući već implementiranu podršku za njihovu upotrebu. Uz to, sustav sadrži niz tutorijala koji korisnicima objašnjavaju kako upravljati različitim uređajima koji su spojeni na Arduino. Za korištenje drugih mikrokontrolera potrebno je dodatno prilagoditi sustav (objašnjeno u prethodnim poglavljima). Kao primarni mikrokontroler odabran je Arduino UNO, ali sustav funkcioniра s bilo kojim Arduino mikrokontrolerom.

5.3.1. Arduino UNO

Arduino UNO je mikrokontroler temeljen na ATmega328P čipu koji radi na frekvenciji od 16 MHz. Sadrži 14 digitalnih ulazno/izlaznih pinova, od kojih se 6 koristi kao PWM (eng. *Pulse Width Modulation*) pinovi te 6 analognih ulaznih pinova. PWM pinovi omogućuju generiranje analognih signala pomoću digitalnih pinova. Također, posjeduje serijsko UART (eng. *Universal Asynchronous Receiver-Transmitter*) komunikacijsko sučelje. Zahvaljujući UART sučelju, Arduino UNO može komunicirati s drugim uređajima poput računala, drugih mikrokontrolera, senzora, modula ili vanjskih perifernih uređaja. Integrirani USB priključak omogućuje jednostavno povezivanje s računalom i programiranje mikrokontrolera.

Na Arduino mikrokontroler spojena su četiri uređaja koje učenici mogu programirati i kontrolirati (Slika 5.3). Prednost je imati sva četiri uređaja na istoj ploči jer učenici mogu jednostavno eksperimentirati s više uređaja istovremeno. Ovakva kombinacija uređaja omogućuje učenicima dublje razumijevanje o tome kako različiti uređaji surađuju i nudi mogućnosti za izvođenje složenijih projekata (na primjer, mogućnost promjene boje RGB LED diode na temelju izmjerene temperature).

Na Arduino su spojeni sljedeći uređaji: RGB LED dioda (eng. *Red-Green-Blue light-emitting diode*), senzor temperature, servo motor i LCD zaslon (eng. *Liquid Crystal Display*). Uređaji i njihovi tutorijali su detaljno objašnjeni u poglavljiju 7. Za prevodenje i slanje koda na Arduino koristi se *arduino-cli*.



Slika 5.3: Arduino sa spojenim uređajima

5.3.2. Arduino-cli

Arduino CLI (eng. *Command Line Interface*) je alat koji omogućuje upravljanje Arduino platformom putem naredbenog retka. To je moćan alat koji pruža različite funkcionalnosti za rad s Arduino mikrokontrolerima i projektima [19].

Arduino CLI se može koristiti za sljedeće svrhe

- **Izrada, upravljanje i uređivanje Arduino projekata:** Omogućuje stvaranje novih Arduino projekata, uređivanje postojećih projekata i upravljanje njihovom struktururom.
- **Upravljanje bibliotekama:** Omogućuje pretraživanje, instalaciju i ažuriranje Arduino biblioteka koje se koriste u projektima. Također podržava upravljanje verzijama biblioteka.
- **Prevodenje i izgradnja:** Omogućuje prevodenje Arduino koda i izgradnju *firmware* datoteke koja se može prenijeti na Arduino mikrokontroler.
- **Upravljanje mikrokontrolerima:** Pruža funkcionalnost za upravljanje Arduino mikrokontrolerom, uključujući prepoznavanje pločica, postavljanje njihovih parametara, ažuriranje *firmwarea* i prenošenje koda na uređaj.
- **Serijska komunikacija:** Omogućuje komunikaciju s Arduinom putem serijskog ulaza. Podaci se mogu slati i primati s uređaja te koristiti za razne svrhe kao što su praćenje senzora ili upravljanje vanjskim uređajima.

Sustav koristi *arduino-cli* za prevođenje korisničkih kodova i prenošenje koda na Arduino.

Dvije su glavne naredbe koje se koriste:

1. Naredba za prevođenje

```
arduino-cli compile --fqbn <board_fqbn> <putanja_do_skice>
```

Gdje su parametri:

- **-fqbn:** Označava potpuno kvalificirano ime pločice (eng. *Fully Qualified Board Name*). Ovaj parametar specificira koja Arduino pločica se koristi za kompiliranje koda. Na primjer, za Arduino UNO, FQBN je *arduino:avr:uno*. Za različite pločice, FQBN se razlikuje.
- **putanja_do_skice:** Putanja do Arduino skice (*.ino* datoteke) koju se želi prevesti.

2. Naredba za prenošenje koda

```
arduino-cli upload -p <port> --fqbn <board_fqbn> --input-dir  
<putanja_do_direktorija>
```

Gdje su parametri:

- **-p (port):** Označava serijski priključak (eng. *port*) na kojem je povezana Arduino pločica. Ovaj parametar specificira na koji priključak se šalje kod.
- **-fqbn:** Označava potpuno kvalificirano ime pločice. Ovaj parametar specificira koja Arduino pločica se koristi za prijenos koda.
- **--input-dir:** Označava izvorni direktorij u kojem se nalazi kod koji se želi prenijeti na pločicu. Ovdje treba navesti direktorij u kojem se nalazi *.ino* datoteka i sve druge pomoćne datoteke potrebne za izvršavanje koda.

5.4. Korisnička aplikacija

Kako bi korisničko sučelje bilo potpuno funkcionalno, osim Reacta, koriste se i dodatne tehnologije. Dva važna aspekta koja treba riješiti su obavještavanje korisnika o izvođenju njihovog koda i prikaz videoprijenosu uživo. Za obavještavanje korisnika o događajima koristi se **STOMP** protokol, koji omogućuje stvaranje komunikacijskog kanala između klijenta i poslužitelja putem **WebSocketa**. Za spajanje na medijski poslužitelj i prikazivanje videoprijenosu uživo koristi se **React Video Player**.

5.4.1. STOMP klijent

STOMP (eng. *Simple Text Oriented Messaging Protocol*) je protokol koji se često koristi u kontekstu *WebSocketa* za komunikaciju između klijenata i poslužitelja. *WebSocket* je tehnologija koja omogućuje dvosmjernu komunikaciju između web preglednika (klijenta) i web poslužitelja. Međutim, *WebSocket* sam po sebi ne definira specifičan format poruka ili protokol za komunikaciju. Upravo se za to koristi STOMP. On pruža standardizirani način za slanje i primanje poruka između web klijenta i poslužitelja putem *WebSocketa* [20]. STOMP definira naredbe i poruke koje se koriste za razmjenu podataka. Koristeći STOMP, klijent može slati poruke poslužitelju i primati odgovore od poslužitelja.

STOMP omogućuje pretplatu (eng. *subscribe*) na određene kanale (eng. *topics*) kako bi se primali samo odabrani podaci. Također podržava i slanje poruka na određene kanale kako bi se komuniciralo s drugim klijentima ili komponentama sustava. Kreiranje STOMP klijenta prikazuje isječak koda 5.6.

```
1 const socket = new SockJS('http://localhost:8080/websocket?userId=${user.id}');
2 const stomp = Stomp.over(socket);
3 stomp.connect({Authorization: 'Bearer 21'}, () => {
4     console.log('Connected to the server.');
5 });
```

Isječak koda 5.6: STOMP klijent

Ovaj isječak koda uspostavlja vezu s *WebSocket* poslužiteljem koristeći *SockJS* biblioteku. Prvo se stvara nova instanca *SockJS* objekta s odgovarajućom URL adresom poslužitelja. U ovom primjeru, adresa poslužitelja je *http://localhost:8080/websocket?userId=\${user.id}*, gdje *user.id* predstavlja ID korisnika. Pomoći ID-a poslužitelj odlučuje kojem korisniku treba poslati poruku. Nakon što je stvoren *SockJS* objekt, koristi se STOMP biblioteka da bi se omogućila komunikacija putem STOMP protokola. Pomoću *stomp.over(socket)* metode stvara se *Stomp* objekt koji koristi prethodno stvoreni *SockJS* objekt. Zatim se poziva *stomp.connect()* metoda kako bi se uspostavila veza s poslužiteljem.

Kako bi se korisnici obavijestili da se njihov kod trenutno izvršava, potrebno se pretplatiti na određene kanale. Isječak koda 5.7 prikazuje pretplaćivanje na kanal */user/topic/code-input-upload*.

```
1 stomp.subscribe('/user/topic/code-input-upload', function (message) {  
2     const msg: NotificationMessage = JSON.parse(message.body)  
3     setNotificationMessage(msg.message)  
4     showModalAndClose()  
5     setCodeStateMessage(msg.message)  
6 }) ;
```

Isječak koda 5.7: Preplata na kanal

Kada stigne poruka od poslužitelja da se korisnikov kod trenutno izvršava, pokreću se ostale operacije. Postavlja se poruka obavijesti i pokreće se ažuriranje sučelja da bi korisnik primijetio obavijest. Izgled sučelja kod primitka obavijesti prikazan je u poglavlju 6.

5.4.2. React Player

ReactPlayer je biblioteka za *React* koja omogućuje jednostavno uključivanje medijskih sadržaja, poput videozapisa i audiozapisa, u *React* aplikacije. Ova biblioteka pruža komponentu *ReactPlayer* koja podržava različite formate medijskih datoteka i omogućuje reprodukciju tih sadržaja na web stranici. *ReactPlayer* pruža podršku za uključivanje medijskih sadržaja s različitih poslužitelja u *React* aplikacije. Ova biblioteka podržava protokole kao što su FLV(eng. *Flash Video*) HLS (eng. *HTTP Live Streaming*), DASH (eng. *Dynamic Adaptive Streaming over HTTP*) i drugi. Isječak koda 5.8 prikazuje komponentu koja se spaja na medijski poslužitelj.

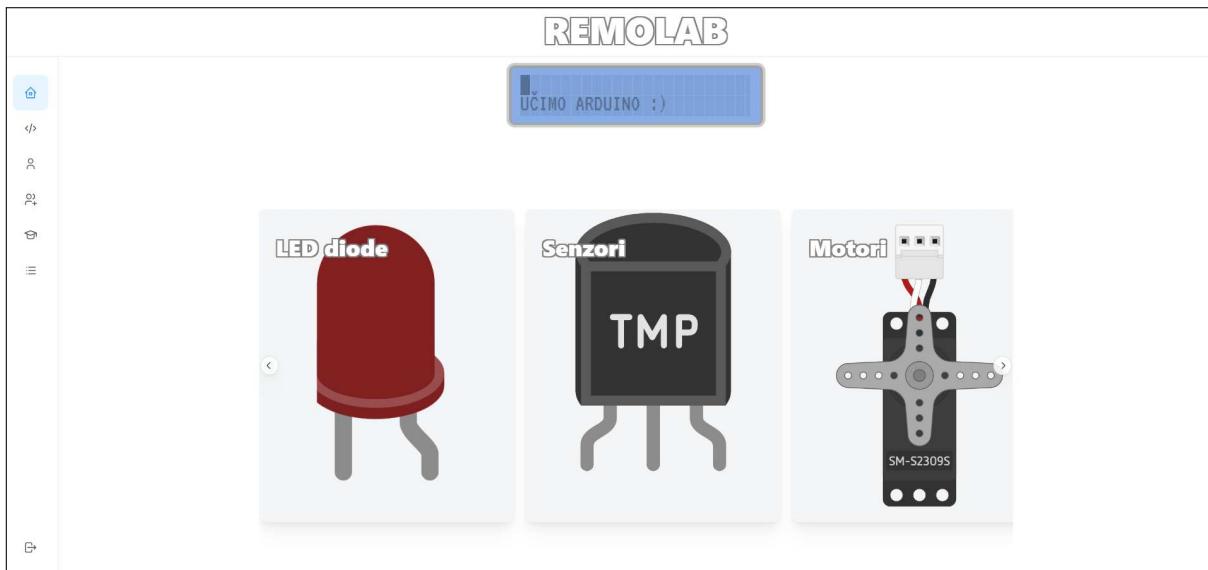
```
1 <ReactPlayer  
2     url="http://localhost:8000/live/my-stream.flv"  
3     playing={true}  
4     controls={false}  
5     muted={true}  
6     width="auto"  
7 />
```

Isječak koda 5.8: *ReactPlayer* komponenta

Potrebno je postaviti URL adresu videoprijenosu. Tu adresu određuje medijski poslužitelj. Ovo je jedna od putanja koje nudi *Node media server*. U primjeru se spaja na lokalni poslužitelj. Ostalim postavkama se osigurava da se video automatski pokreće bez upravljačkih gumba i bez zvuka.

6. Primjer uporabe aplikacije

U ovom poglavlju će biti opisane funkcionalnosti korisničkog sučelja. Korisničko sučelje pruža mogućnosti za dvije uloge: *administratora* i *studenata*. Ova uloga određuje razine ovlasti i pristupa određenim funkcionalnostima. Administrator, odnosno profesor, ima mogućnost upravljanja sustavom i korisnicima. Kroz korisničko sučelje, administrator može stvarati i upravljati korisničkim računima te kreirati i uređivati nastavne materijale. S druge strane, učenici imaju mogućnost pristupiti nastavnim materijalima i sučelju za upravljanje udaljenim laboratorijem. Na slici 6.1 prikazana je naslovna stranica.

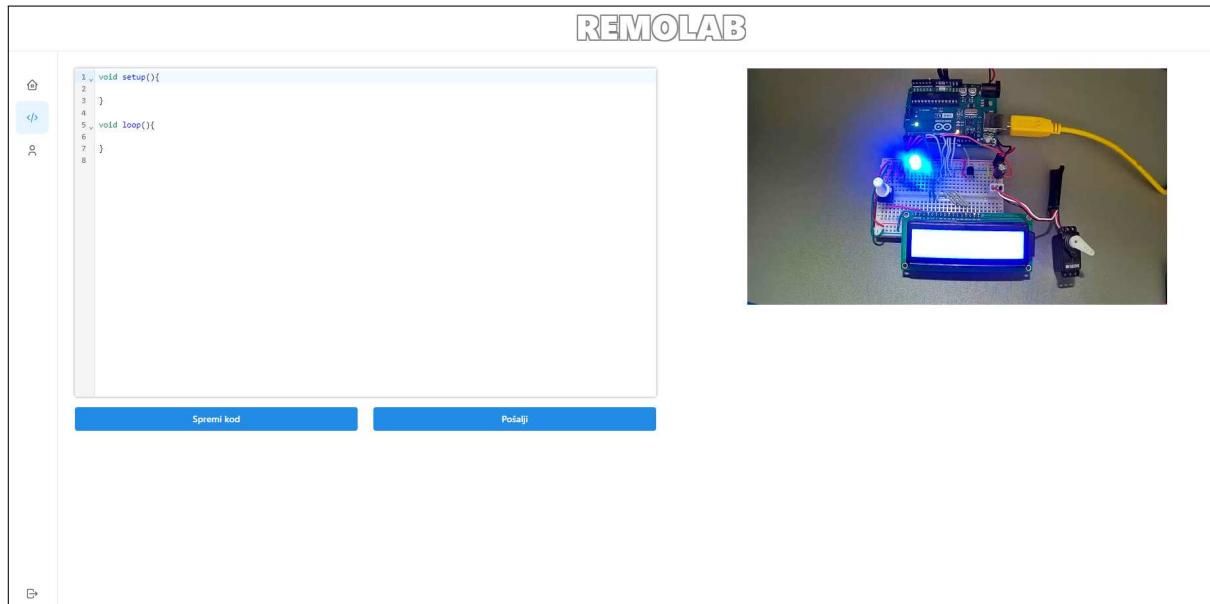


Slika 6.1: Početna stranica s navedenim kategorijama

6.1. Sučelje za unos koda

Kada korisnik pristupi stranici *codeInput*, prikaže se prostor za unos koda s postavljenim početnim kodom. Sustav ima podršku za Arduino uređaje i za potrebe sustava razvijen je udaljeni laboratorij za Arduino uređaje. Zbog toga se na polju

za unos nalazi osnova za Arduino kod. Svaki Arduino kod mora sadržavati dvije funkcije: `setup` i `loop`. Pored polja za unos koda, prikazan je videoprijenos uređaja. (Slika 6.2)



Slika 6.2: Sučelje za unos koda

Korisnik može upisati svoj kod i poslati ga na izvršavanje. Nakon što se kod prevede, korisnik će biti obaviješten o uspjehnosti prevodenja. Ako prevodenje nije uspješno, korisniku će biti prikazana poruka koju je vratio prevoditelj (eng. *compiler*) u okviru ispod prostora za unos koda (Slika 6.3). Cilj poruke je pomoći korisniku da razumije gdje je nastala greška i da je lakše ispravi.

```
1 const int PIN_RED = 2;
2 const int PIN_BLUE = 4;
3 const int PIN_GREEN = 8;
4
5 void setup() {
6   pinMode(PIN_RED, OUTPUT);
7   pinMode(PIN_GREEN, OUTPUT);
8   pinMode(PIN_BLUE, OUTPUT);
9 }
10
11 void loop() {
12   analogWrite(PIN_RED, 1);
13   analogWrite(PIN_BLUE, 1);
14   analogWrite(PIN_GREEN, 250);
15 }
16 }
```



A photograph of an Arduino Uno microcontroller connected to a breadboard. On the breadboard, there is a blue LED with a resistor in series, connected to digital pin 2 via a transistor. A 20x4 character LCD display is also connected to the breadboard. A yellow USB cable is connected to the Arduino Uno. The setup is placed on a light-colored surface.

Spremi kod

Poalji

204\204.ino:2:1: error: expected ';' or ')' before 'const' const int PIN_BLUE = 4; ^~~~~~ 204\204.ino: In function 'void setup()': 204\204.ino:8:2: error: 'PIN_BLUE' was not declared in this scope pinMode(PIN_BLUE, OUTPUT); ^~~~~~ 204\204.ino:14:15: note: suggested alternative: 'PIN_GREEN' pinMode(PIN_BLUE, OUTPUT); ^~~~~~ PIN_GREEN 204\204.ino: In function 'void loop()': 204\204.ino:14:15: error: 'PIN_BLUE' was not declared in this scope digitalWrite(PIN_BLUE, 1); ^~~~~~ PIN_BLUE 204\204.ino:14:15: note: suggested alternative: 'PIN_GREEN' digitalWrite(PIN_BLUE, 1); ^~~~~~ PIN_GREEN ||#2mUsed platform||0m ||92mVersion||0m ||90mPath||0m ||93marduino:avr||0m 1.8.6 ||90mArduino||15\packages\arduino\hardware\avr\avr\1.8.6||0m Error during build: exit status 1 Compilation completed with exit code: 1

Slika 6.3: Neuspješno prevođenje

U slučaju da je kod ispravan, korisniku se ispisuje poruka, ispod polja za unos, da je kod uspješno preveden i da je dodan u red čekanja (Slika 6.4).

The screenshot shows the REMOLAB interface with the following components:

- Code Editor:** On the left, there is a code editor window containing C++ code for an Arduino project. The code initializes pins for red, blue, and green LEDs and sets up analog outputs for each. It then enters a loop where it writes analog values to the pins.
- Output Window:** A central window displays the message "Vaš kod je dodan u red čekanja." (Your code has been added to the queue).
- Hardware Preview:** On the right, there is a preview image of the hardware setup, which includes an Arduino board connected to a breadboard with three LEDs and a 16x2 LCD display.
- Buttons:** At the bottom, there are two buttons: "Spremi kod" (Save code) and "Pošalji" (Send).
- Status Bar:** A green bar at the bottom indicates the status: "Prevođenje koda je uspješno, pričekajte izvršavanje..." (Code compilation was successful, please wait for execution...).

Slika 6.4: Uspješno prevođenje

Nakon toga korisnik čeka na izvršavanje svog koda. Kada kod dođe na red, korisnik ponovno dobiva obavijest. Ovaj puta se javlja poruka da se kod trenutno izvršava i da može pratiti videoprijenos. Obavijest je identična kao i obavijest za uspješno prevođenje samo s drugom porukom. Nakon toga korisnik može pratiti videoprijenos i vidjeti rezultat svog koda.

6.2. Administratorsko sučelje

Profesori imaju dodatne ovlasti u sustavu. Osim slanja i izvršavanja kodova na uređaju, profesori mogu obavljati i administrativne zadatke kao što je kreiranje i uklanjanje računa za učenike. Budući da je pristup laboratoriju ograničen samo na učenike, postoji potreba za registracijom koju obavljaju profesori.

Ova kontrola nad registracijom osigurava da samo relevantni korisnici, odnosno učenici, imaju pristup laboratoriju. Profesori mogu upravljati tim računima, omogućavajući pristup samo odabranim učenicima, osiguravajući tako sigurnost laboratorijskog okruženja.

Profesori mogu kreirati nove tutorijale. Tutorijali služe za objašnjenje komponenti spojenih na mikrokontroler, čemu služe i kako se koriste. Sustav ima tutorijale za spojene komponente: senzor temperature, RGB LED diodu i servo motor. Tutorijali su detaljno opisani u poglavlju 7. Tutorijali predstavljaju vrijedan izvor informacija za učenike i omogućavaju im da steknu dublje razumijevanje o radu s određenim komponentama. Slika 6.5 prikazuje formu za kreiranje novih tutorijala. Profesori također imaju mogućnost uređivanja postojećih tutorijala. Ovo im omogućuje da ažuriraju ili prilagode sadržaj tutorijala kako bi odgovarao najnovijim standardima, te da dodaju ili izbrišu zastarjele informacije.

The screenshot shows a web-based form titled 'Kreirajte novi tutorial' (Create new tutorial) under the 'REMO LAB' header. The form includes fields for 'Naslov' (Title), 'Kategorija' (Category), 'Opis tutorijala' (Tutorial description), and a rich text editor toolbar. A blue 'KREIRAJ' (Create) button is at the bottom.

Naslov *
[Empty input field]

Kategorija *
"kategorija" ▾

Opis tutorijala

Rich text editor toolbar:

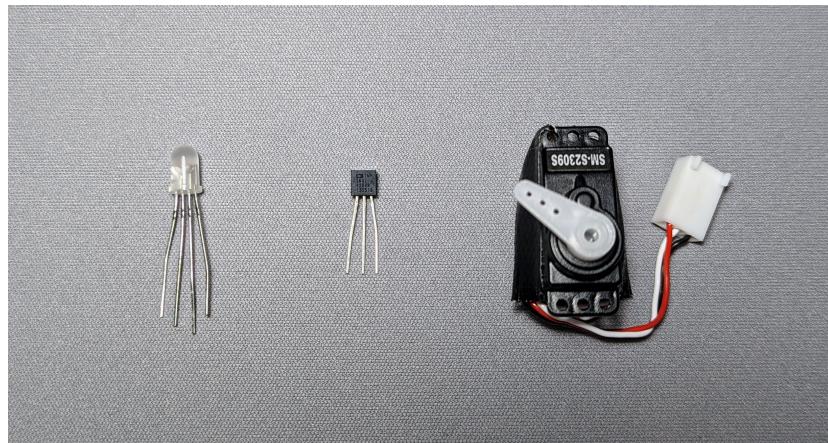
- B I U S T ∞ \times/\cancel{x}
- H₁ H₂ H₃ H₄
- $\frac{a}{b}$ \dots \equiv $\frac{1}{b}$ x_1 x^2
- \mathcal{O} \mathcal{Q}
- \equiv \equiv \equiv \equiv

KREIRAJ

Slika 6.5: Forma za registraciju učenika

7. Opis pripremljenih eksperimenata

U nastavu slijedi opis eksperimenata koji su dostupni u sustavu. Kao što je ranije spomenuto, sustav uključuje laboratorij s Arduino uređajem koji je opremljen s nekoliko komponenti. Među spojenim komponentama su: RGB LED dioda, servo motor, senzor temperature i LCD zaslon. (Slika 7.1) U nastavku su navedeni pojedinačni uređaji i tutorijali koji su napisani za svaki od njih.



Slika 7.1: RGB LED dioda, senzor temperature i servo motor

7.1. RGB LED dioda

RGB LED dioda (eng. *Red, Green, Blue Light-Emitting Diode*) je vrsta LED diode koja kombinira tri osnovne boje - crvenu, zelenu i plavu - kako bi stvorila široku paletu boja. Ove tri boje se mogu kontrolirati neovisno jedna o drugoj, omogućavajući stvaranje različitih boja svjetlosti. LED diodama se mogu napraviti zanimljive vježbe poput [21]. Za upravljanje RGB LED diodom koristi se Arduino mikrokontroler. Mikrokontroler omogućuje postavljanje svakog kanala (crvene, zelene i plave boje) na određenu vrijednost koja određuje intenzitet svjetlosti za taj kanal. Kada se ova tri kanala kombiniraju u različitim omjerima, dobiju se razne boje. Cilj ovog tutorijala je upoznati učenike s osnovnim funkcijama i naredbama koje se koriste za upravljanje

RGB LED diodom. Kroz primjer koda, korisnici će naučiti kako postaviti pinove na OUTPUT način, kako koristiti funkciju `analogWrite()` za kontrolu intenziteta svjetlosti te kako deklarirati varijable za pohranu vrijednosti boje. Primjer koda koji se nalazi u tutorijalu, mijenja boju svjetlosti na diodi između crvene, zelene i plave s određenim vremenskim razmakom između svake promjene. Postoje i naprednije tehnike poput stvaranja glatkih prijelaza boja ili animacija uz pomoć petlji i uvjeta. Kroz ovaj tutorijal, korisnici će također naučiti koncept RGB modela boja, koji se temelji na kombiniranju crvene, zelene i plave boje.

7.2. Senzor temperature

Senzor temperature je elektronički uređaj koji se koristi za mjerjenje temperature [22]. Omogućuje pretvaranje fizičke promjene temperature u električne signale koje Arduino mikrokontroler može interpretirati. Postoje različiti tipovi senzora temperature, a najčešće se koriste **termistori** ili **IC senzori**. Termistori su osjetnici temperature koji mijenjaju električni otpor ovisno o temperaturi. Kako se temperatura mijenja, vrijednost električnog otpora se mijenja proporcionalno. Arduino mikrokontroler može mjeriti taj električni otpor i koristiti ga za izračunavanje temperature. IC senzori temperature koriste digitalne ili analogue izlaze za prikaz temperature. Oni su obično već kalibrirani i pružaju precizne mjere temperature.

U tutorijalu za senzor temperature, učenici će naučiti osnove čitanja vrijednosti sa senzora i pretvaranje analognih ulaza u smislene vrijednosti. Također će se upoznati s konceptom pretvaranja vrijednosti u stupnjeve Celzijusa ili drugu željenu mjeru. Ovaj tutorijal će im pružiti temelje za daljnje istraživanje i eksperimentiranje s različitim senzorima, poput senzora vlage, svjetla i drugih.

7.3. Servo motor

Servo motor je elektromehanički uređaj koji se koristi za precizno upravljanje kutom rotacije. Sastoje se od: rotacijskog mehanizma, motora i elektroničke kontrolne jedinice. Servo motori su poznati po svom visokom momentu okretanja, preciznosti i sposobnosti zadržavanja stabilnog položaja. Često se koriste u robotici, automatskim sustavima za upravljanje, mehatronici i drugim aplikacijama gdje je potrebna precizna kontrola pokreta [23] [24]. U Arduino okruženju, servo motori se često koriste u kombinaciji s PWM (eng. *Pulse Width Modulation*) signalom. PWM signal

omogućuje upravljanje brzinom rotacije motora putem periodički generiranih impulsa visokog signala, čija širina odgovara željenom kutu rotacije. Ovo omogućuje precizno podešavanje položaja servo motora u skladu s potrebama.

U tutorijalu za servo motor, učenici će naučiti osnove upravljanja servo motorom pomoću Arduino mikrokontrolera. Počevši od korištenja *Servo* biblioteke, učenici će biti upućeni u korake za upravljanje servo motorom koristeći Arduino pločicu. Kroz primjer programskog koda, koji koristi petlje i funkcije za upravljanje kutom rotacije servo motora, učenici će shvatiti kako programirati precizne pokrete i promjene brzine. Ovaj tutorijal pruža praktično iskustvo za upravljanje servo motorom te razumijevanje koncepta i implementacije kontrolnih signala. Učenici će se također upoznati s uključivanjem vanjskih biblioteka koje nude već gotova rješenja za jednostavnije korištenje pojedinih uređaja.

8. Moguća poboljšanja i nadogradnje

Sustav za održavanje laboratorijskih vježbi je funkcionalan, ali postoje mogućnosti za poboljšanja koja bi unaprijedila korisničko iskustvo. U nastavku su navedene neke ideje i moguća rješenja za nadogradnju sustava:

8.1. Suradnja u rješavanju zadataka

Jedan od prijedloga je omogućiti učenicima da zajedno rješavaju zadatke ili na neki način surađuju unutar sustava. To može uključivati mogućnost stvaranja timova, dijeljenja koda, komentiranja i pregledavanja rješenja drugih članova tima. Suradnja je važna vještina koju učenici mogu razvijati kroz zajedničko rješavanje zadataka. Neki od načina za ostvarivanje suradnje su komunikacija i kolaboracijske datoteke.

8.1.1. Videopozivi ili čavrljanje

Videopozivi ili neki oblik čavrljanja (eng. *chat*) bili bi dobro rješenje za poboljšanje komunikacije između učenika. Videopozivi omogućuju učenicima da se međusobno vide i čuju u stvarnom vremenu. Učenici mogu postavljati pitanja, razgovarati o zadacima ili dijeliti ideje, na način koji je gotovo jednak razgovoru licem u lice. Učenici tako mogu odmah dobiti odgovore na svoja pitanja, umjesto da čekaju na odgovor putem e-pošte ili foruma. Neka rješenja su:

- **Twilio Video**

Twilio Video je programabilna platforma za komunikaciju u stvarnom vremenu koja omogućuje dodavanje funkcionalnosti video razgovora u web, iOS i Android aplikacije [25]. Ova platforma pruža API-je (eng. *Application Programming Interface*), SDK-ove (eng. *Software Development Kit*) i pomoćne alate za snimanje, distribuciju i prikazivanje visokokvalitetnog audio i video sadržaja. Korištenjem Twilio Video, mogu se integrirati video *chat*

mogućnosti u aplikacije, što omogućuje komunikaciju između korisnika u stvarnom vremenu.

- **Daily.co**

Daily.co je platforma za video komunikaciju koja omogućuje jednostavno integriranje video razgovora u web i mobilne aplikacije [26]. Pruža razne mogućnosti za stvaranje i upravljanje video pozivima, kao što su grupni pozivi, dijeljenje zaslona i slanje poruka. Daily.co pruža jednostavno sučelje za programiranje putem API-ja i SDK-ova, što olakšava integraciju video komunikacije u aplikacije. Platforma također nudi visokokvalitetan audio i video prijenos te prilagodljive postavke kako bi se zadovoljile potrebe korisnika.

- **WebRTC**

WebRTC (eng. *Web Real-Time Communication*) je projekt otvorenog koda (eng. *open source*) koji omogućuje komunikaciju u stvarnom vremenu između web preglednika bez potrebe za dodatnim proširenjima ili aplikacijama [27]. Ova tehnologija pruža mogućnost razvoja vlastitog rješenja za videopozive, što znači da nije potrebno plaćati gotova rješenja koja su dostupna na tržištu, omogućuje veću fleksibilnost i kontrolu nad videopozivima, a također omogućuje prilagođavanje posebnim potrebama projekta.

Međutim, važno je napomenuti da korištenje WebRTC-a za videopozive zahtijeva konfiguriranje poslužitelja i druge tehničke aspekte. Ovo može biti izazovno i zahtijeva određeno razumijevanje mrežnih protokola i infrastrukture.

8.1.2. Kolaboracijske datoteke

Kolaboracijske datoteke su rješenja poput Google dokumenata [28], gdje više korisnika može uređivati jedan dokument. Takvo rješenje bi omogućilo učenicima da imaju zajednički dokument gdje bi mogli pisati kod i zajedno rješavati zadatke. Neka od mogućih rješenja su:

- **TipTap**

Tiptap je biblioteka koja omogućuje implementaciju tekstualnih editora u *React* aplikacijama, uključujući podršku za suradnju u stvarnom vremenu [29]. Iako *Tiptap* ne pruža izravnu podršku za suradnju, može se integrirati s drugim alatima i bibliotekama kako bi se postigla funkcionalnost suradnje. Jedan od popularnih alata koji se često koristi u kombinaciji s *Tiptapom* za suradnju u

stvarnom vremenu je **Yjs** [30]. *Yjs* je *JavaScript* biblioteka koja omogućuje sinkronizaciju podataka u stvarnom vremenu između više korisnika. Može se integrirati s *Tiptapom* kako bi se omogućila istovremena suradnja više korisnika na uređivanju teksta. *Yjs* koristi tehnike poput CRDT (eng. *Conflict-Free Replicated Data Types*) za rješavanje sukoba i sinkronizaciju promjena između korisnika. Važno je napomenuti da implementacija suradnje u stvarnom vremenu, koristeći *Tiptap* i *Yjs*, može zahtijevati određenu razinu konfiguracije i prilagodbe prema potrebama projekta.

- **PubNub**

PubNub je globalna platforma za komunikaciju u stvarnom vremenu koja omogućuje razmjenu poruka i sinkronizaciju podataka između više korisnika [31]. *PubNubov* sustav poruka omogućuje istovremeno uređivanje tekstualnih datoteka više korisnika, pružajući mogućnosti za suradnju. *PubNubov* sustav poruka koristi model *objavi-preplati* (eng. *publish-subscribe*) Svi pretplaćeni korisnici koji su povezani na isti *PubNub* kanal dobivaju te poruke s ažuriranim sadržajem u stvarnom vremenu. Korisnici zatim mogu prikazati i ažurirati sadržaj svojih lokalnih kopija tekstualne datoteke temeljem tih poruka. *PubNub* pruža robusne mehanizme za upravljanje sinkronizacijom podataka, detekcijom kolizija (kada više korisnika pokušava istovremeno izmjeniti isti dio datoteke) i rješavanje sukoba kako bi osigurao dosljednost podataka i ispravno upravljanje promjenama. *PubNub* je dostupan kao usluga u oblaku (eng. *cloud*) i pruža skup API-ja i SDK-ova koji olakšavaju integraciju u različite platforme i tehnologije, uključujući web i mobilne aplikacije. Za razliku od *TipTapa* potrebno je manje konfiguracije, ali se plaća.

8.2. Automatsko ocjenjivanje

Automatsko ocjenjivanje može biti vrlo korisno jer omogućuje uštedu vremena budući da se provjera i ocjenjivanje koda odvija brzo i automatski, bez ručnog ispravljanja profesora. Učenici mogu odmah dobiti povratne informacije o svojem kodu, bez čekanja na ručnu provjeru, što povećava učinkovitost učenja. Automatsko ocjenjivanje je skalabilno i može se primijeniti na veliki broj kodova bez potrebe za dodatnim resursima ili vremenom profesora. To znači da se može efikasno vrednovati rad velikog broja učenika ili sudionika u sustavu za učenje, bez potrebe za povećanjem ljudskih resursa. Postoje alati koji služe za ocjenjivanje i provjeru kodova.

- **ArduinoUnit**

ArduinoUnit je biblioteka za jedinično testiranje Arduino projekata [32]. Omogućuje pisanje testova za provjeru ispravnosti Arduino koda. Mogu se definirati testni slučajevi koji provjeravaju očekivane izlaze koda i uspoređuju ih s dobivenim rezultatima.

- **Code Runner**

Code Runner je alat koji omogućuje izvršavanje koda u različitim jezicima programiranja, uključujući Arduino [33]. *Code Runner* se može koristiti za pokretanje Arduino koda i provjeru izlaza. To omogućuje brzo provjeravanje ispravnosti koda bez potrebe za prevodenjem ili prenošenjem koda na Arduino uređaj.

- **HackerRank**

HackerRank je popularna *online* platforma za provjeru vještina programiranja. Pruža mogućnost postavljanja zadataka, uključujući Arduino projekte, s referentnim rješenjima. *HackerRank* se može koristiti za ocjenjivanje i provjeru podudaranja predanih rješenja s referentnim rješenjima [34]. Platforma također omogućuje provjeru vremenskih ograničenja i pruža detaljne povratne informacije o performansama i točnosti rješenja.

- **Vlastiti alati**

Postoji mogućnost razvijanja vlastitih alata koji mogu služiti za provjeru i usporedbu kodova za Arduino uređaje. Ovisno o potrebama, može se napisati skripta ili program za provjeru i usporedbu koda. Prednost ovakvih alata je što mogu biti precizniji i bolje ispravljati i ocjenjivati predane kodove. Nedostatak je što je razvoj takvih rješenja zahtijevan.

9. Zaključak

Udaljeni laboratorijski predstavljanje inovativan pristup obrazovanju koji omogućuje učenicima da provode eksperimente i laboratorijske vježbe putem Interneta, bez fizičkog pristupa opremi. Udaljeni laboratorijski pružaju fleksibilnost i pristupačnost učenicima, omogućujući im da uče i eksperimentiraju sa stvarnim komponentama iz udobnosti vlastitih domova. Također, udaljeni laboratorijski omogućuju iterativno iskustvo učenja, gdje studenti mogu testirati, promatrati rezultate i tako prilagođavati svoja rješenja.

Ovaj rad predstavio je rješenje udaljenog laboratorijskog sistema koji dolazi s podrškom za Arduino mikrokontrolere. Sustav je fleksibilan i jednostavan za nadogradnju, omogućujući lako dodavanje novih uređaja. Također, sustav omogućuje istovremeni rad više korisnika i dolazi s pripremljenim materijalima koji olakšavaju razumijevanje i korištenje pojedinih uređaja. Upravljanje uređajima obavlja se kroz korisničko sučelje koje je u obliku web stranice. Za komunikaciju između aplikacije i uređaja koristi se HTTP poslužitelj koji prima programske kodove koje pošalju korisnici i izvršava ih na uređaju. Kako bi korisnici putem videoprijenosu uživo mogli pratiti rezultate izvršavanja svog koda na uređaju, bilo je potrebno implementirati medijski poslužitelj. Sustav dolazi s nekoliko pripremljenih tutorijala koji služe za upoznavanje osnovnih koncepata uređaja i nude nekoliko primjera korištenja.

LITERATURA

- [1] S. Martin, A. Fernandez-Pacheco, J. A. Ruipérez-Valiente, G. Carro, and M. Castro, “Remote experimentation through arduino-based remote laboratories,” *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, vol. 16, no. 2, pp. 180–186, 2021.
- [2] J. Corder, J. Nickerson, S. Esche, and C. Chassapis, “Remote versus hands-on labs: a comparative study,” in *34th Annual Frontiers in Education, 2004. FIE 2004.*, 2004, pp. F1G–17.
- [3] M. Abdulwahed and Z. Nagy, “Developing the trilab, a triple access mode (hands-on, virtual, remote) laboratory, of a process control rig using labview and joomla,” *Computer Applications in Engineering Education*, vol. 21, 12 2013.
- [4] Z. Nedic, J. Machotka, and A. Nafalski, “Remote laboratories versus virtual and real laboratories,” vol. 1, 12 2003, pp. T3E–1.
- [5] Z. Nurbekova, T. Tolganbaiuly, B. Nurbekov, A. Sagimbayeva, and Z. Kazhiakparova, “Project-based learning technology: An example in programming microcontrollers,” *International Journal of Emerging Technologies in Learning (iJET)*, vol. 15, p. 218, 06 2020.
- [6] D. Calderón, E. Petersen, and O. Rodas, “Salp: A scalable autograder system for learning programming - a work in progress,” in *2020 IEEE Integrated STEM Education Conference (ISEC)*, 2020, pp. 1–4.
- [7] S. Dormido, H. Vargas, J. Sánchez, R. Dormido, N. Duro, S. Dormido-Canto, and F. Morill, “Developing and implementing virtual and remote labs for control education: The uned pilot experience,” *17th IFAC World Congress*, pp. 8159–8164, 2008.
- [8] UNED, “University network of interactive laboratories,” <https://unilabs.dia.uned.es/blog/index.php?entryid=3>, pristupljeno: 20. veljače 2023.

- [9] L. M. Joao Paulo Cardoso de Lima, “Design and implementation of a remote lab for teaching programming and robotics,” *IFAC-PapersOnLine*, vol. 49, no. 30, pp. 86–91, 2016, 4th IFAC Symposium on Telematics Applications TA 2016. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2405896316325848>
- [10] Facebook. React. Poveznica: <https://react.dev>; pristupljeno: 15. travnja 2023.
- [11] Axios. Axios. Poveznica: <https://axios-http.com/>; pristupljeno: 1. ožujka 2023.
- [12] Mantine. Mantine. Poveznica: <https://mantine.dev/>; pristupljeno: 15. ožujka 2023.
- [13] Spring. Spring. Poveznica: <https://spring.io/>; pristupljeno: 1. veljače 2023.
- [14] PostgreSQL. Postgresql. Poveznica: <https://www.postgresql.org/>; pristupljeno: 1. veljače 2023.
- [15] Node-Media-Server. Node-media-server. Poveznica: <https://github.com/illuspas/Node-Media-Server>; pristupljeno: 8. svibnja 2023.
- [16] F. Project. Ffmpeg. Poveznica: <https://ffmpeg.org/>; pristupljeno: 15.4.2023.
- [17] A. Darmawan, R. Ratnadewi, E. Sartika, N. Pasaribu, and R. Arlando, “Basic arduino programming training for high school students,” *THE SPIRIT OF SOCIETY JOURNAL*, vol. 1, 09 2017.
- [18] E. Gamma, R. Helm, R. E. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*. Addison-Wesley Professional, 1995.
- [19] Arduino. Arduino cli. Poveznica: <https://arduino.github.io/arduino-cli/>; pristupljeno: 1. ožujka 2023.
- [20] i. netzpirat. Stomp.js. Poveznica: <https://www.npmjs.com/package/stomp-client>; pristupljeno: 4. svibnja 2023.
- [21] A. Fernández-Pacheco, S. Martin, and M. Castro, “Implementation of an arduino remote laboratory with raspberry pi,” in *2019 IEEE Global Engineering Education Conference (EDUCON)*, 2019, pp. 1415–1418.
- [22] M. Kusriyanto and A. A. Putra, “Weather station design using iot platform based on arduino mega,” in *2018 International Symposium on Electronics and Smart Devices (ISESD)*, 2018, pp. 1–4.

- [23] Y. Jing, L. Zhang, I. Arce, and A. Farajidavar, “Androrc: An android remote control car unit for search missions,” 05 2014, pp. 1–5.
- [24] V. Varshney, A. Kumar, T. Saxena, P. Jha, M. N. Tiwari, and Savita, “Design and development of intellectual robotic arm,” in *2019 6th International Conference on Computing for Sustainable Global Development (INDIACo)*, 2019, pp. 984–987.
- [25] T. Video. Twilio video. Poveznica: <https://www.twilio.com/docs/video>; pristupljeno: 6. lipnja 2023.
- [26] Daily.co. Daily.co. Poveznica: <https://www.daily.co/>; pristupljeno: 9. lipnja 2023.
- [27] WebRTC. Webrtc. Poveznica: <https://webrtc.org/>; pristupljeno: 7. lipnja 2023.
- [28] Google. Google docs. Poveznica: <https://www.google.com/docs/about/>; pristupljeno: 6. lipnja 2023.
- [29] Tiptap. Tiptap. Poveznica: <https://tiptap.dev/>; pristupljeno: 7. lipnja 2023.
- [30] Yjs. Yjs. Poveznica: <https://docs.yjs.dev/>; pristupljeno: 9. lipnja 2023.
- [31] PubNub. Pubnub. Poveznica: <https://www.pubnub.com/>; pristupljeno: 6. lipnja 2023.
- [32] W. MacEvoy. Arduino unit. Poveznica: <https://github.com/mmurdoch/arduinounit>; pristupljeno: 8. lipnja 2023.
- [33] C. Runner. Code runner. Poveznica: <https://coderunner.io/>; pristupljeno: 5. lipnja 2023.
- [34] HackerRank. Hackerrank. Poveznica: <https://www.hackerrank.com/>; pristupljeno: 10. lipnja 2023.

Sustav za održavanje laboratorijskih vježbi na daljinu

Sažetak

Ovaj rad predstavlja razrađenu ideju za edukacijski web sustav koji podučava programiranje i osnove mikrokontrolera. Sustav za održavanje laboratorijskih vježbi na daljinu omogućuje učenicima pisanje programskog koda koji se izvršava na stvarnoj hardverskoj opremi. Također, omogućeno je promatranje rezultata izvršavanja putem povezane web-kamere. Udaljeni laboratoriji predstavljaju inovativan pristup obrazovanju, pružaju fleksibilnost i pristupačnost učenicima te omogućuju iterativno iskustvo učenja. Sustav je jednostavan za nadogradnju i podržava istovremeni rad više korisnika, a dolazi s pripremljenim materijalima i tutorijalima za korištenje Arduino mikrokontrolera s nekoliko dodatnih uređaja.

Ključne riječi: udaljeni laboratorij, mikrokontroleri, programiranje, senzori, arduino, učenje

System for remote laboratory exercises

Abstract

This paper presents a developed idea for an educational web system that teaches programming and the basics of microcontrollers. The remote lab system allows students to write programming code that runs on real hardware. Also, it is possible to observe the execution results via the connected web camera. Remote laboratories represent an innovative approach to education, provide flexibility and accessibility to students, and enable an iterative learning experience. The system is easy to upgrade and supports the simultaneous work of several users. System also comes with prepared materials and tutorials for easy use.

Keywords: remote lab, microcontrollers, programming, sensors, arduino, learning