Mikroupravljački sustav za IoT autonomni akvarij

Štrasser, Fran

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni diplomski studij

MIKROUPRAVLJAČKI SUSTAV ZA IOT AUTONOMNI AKVARIJ

Diplomski rad

Fran Štrasser

Osijek, 2024

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac D1: Obrazac za ocjenu diplomskog rada na sveučilišnom diplomskom studiju

Ocjena diplomskog rada na sveučilišnom diplomskom studiju

Ime i prezime pristupnika:	Fran Štrasser
Studij, smjer:	Sveučilišni diplomski studij Računarstvo
Mat. br. pristupnika, god.	D-1253R, 08.10.2021.
JMBAG:	0165071994
Mentor:	izv. prof. dr. sc. Ivan Aleksi
Sumentor:	doc. dr. sc. Anita Katić
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	izv. prof. dr. sc. Tomislav Matić
Član Povjerenstva 1:	izv. prof. dr. sc. Ivan Aleksi
Član Povjerenstva 2:	doc. dr. sc. Ivan Vidović
Naslov diplomskog rada:	Mikroupravljački sustav za IoT autonomni akvarij
Znanstvena grana diplomskog rada:	**Procesno računarstvo (zn. polje računarstvo)**
Zadatak diplomskog rada:	U ovom diplomskom radu je potrebno napraviti sklopovsko i programsko rješenje za autonomni rad akvarija. cilj je izraditi elektroničku pločicu za upravljanje perifernim ulazno izlaznim uređajem te mikroupravljačem ESP32. Od ulazno-izlaznih uređaja potrebno je koristiti LED svjetlo, grijač, termometar, temperatura, tipkala, oled pokaznik...

Datum ocjene pismenog dijela diplomskog rada od strane mentora:	17.09.2024.
Ocjena pismenog dijela diplomskog rada od strane mentora:	Izvrstan (5)
Datum obrane diplomskog rada:	2.10.
Ocjena usmenog dijela diplomskog rada (obrane):	Vrlo dobar (4)
Ukupna ocjena diplomskog rada:	Izvrstan (5)
Datum potvrde mentora o predaji konačne verzije diplomskog rada čime je pristupnik završio sveučilišni diplomski studij:	08.10.2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O IZVORNOSTI RADA

Osijek, 08.10.2024.

Ime i prezime Pristupnika:	Fran Štrasser
Studij:	Sveučilišni diplomski studij Računarstvo
Mat. br. Pristupnika, godina upisa:	D-1253R, 08.10.2021.
Turnitin podudaranje [%]:	11

Ovom izjavom izjavljujem da je rad pod nazivom**: Mikroupravljački sustav za IoT autonomni akvarij**

izrađen pod vodstvom mentora izv. prof. dr. sc. Ivan Aleksi

i sumentora doc. dr. sc. Anita Katić

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis pristupnika:

SADRŽAJ

1. UVOD

Čovjek višednevnim izbivanjem iz doma, ukoliko ima akvarij, mora naći rješenje kako nahraniti ribe svaki dan. Najčešće to bude molba njegovim najbližima da oni preuzmu brigu o hranjenju riba. Cilj ovog diplomskog rada je realizirati mikroupravljački sustav za IoT (engl. *Internet of things*, Internet stvari) autonomni akvarij koji bi se predstavio kao rješenje tog problema. IoT podrazumijeva upravljanje nekog tijela ili objekta na daljinu.

U ovome radu predstavljena je primjena ESP32 komunikacijskog modula u udaljenom upravljanju akvarijem. Sustavom će upravljati mikroupravljač koji korisniku prikazuje trenutno stanje različitih komponenti. Za svrhe ostvarivanja sklopovskog i programskog rješenja rada će se koristiti Dasduino CONNECTPLUS mikroupravljač stoga će se za programiranje njega koristiti Arduino IDE (engl. *Arduino Integrated Development Environment*, Arduino integrirano razvojno okruženje).

Potrebno je napraviti programsko i sklopovsko rješenje na tu temu kao i maketu kućišta koje se može staviti na akvarij. Za izradu sustava korišten je mikroupravljački sustav, PCB pločica (engl. *Printed Circuit Bord,* tiskana pločica), kućište izrađeno od drveta i hranilicu za ribe. Kako bi se napravila hranilica za ribe potrebno je istu dizajnirati i 3D isprintati. Za svrhe dizajniranja hranilice koristi se Autodesk Fusion dok se za svrhu dizajniranja PCB i naručivanja izrađene pločice koristi EasyEDA.

1.2. Zadatak diplomskog rada

U ovom diplomskom radu potrebno je napraviti maketu za IoT autonomni akvarij. Potrebno je koristiti mikroupravljački sustav i odgovarajuće komponente izabrane prema namjeni.

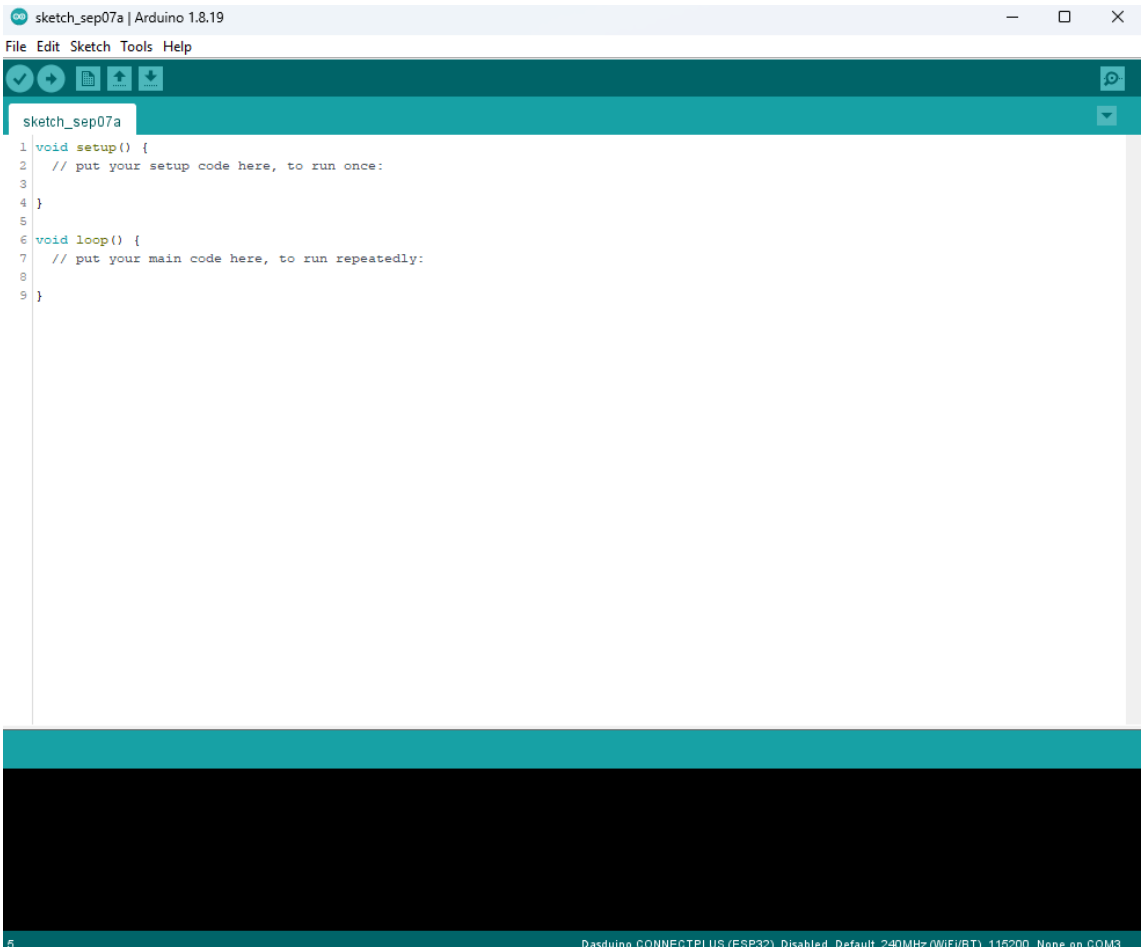
2. KORIŠTENE TEHNOLOGIJE

U procesu izrade mikroupravljačkog sustava korišteni su Arduino integrirano programsko okruženje za programiranje mikroupravljača, EasyEda za dizajniranje PCB pločice i Autodesk Fusion za modeliranje hranilice kojom upravlja stepper motor.

2.1. Arduino IDE

Arduino IDE je besplatni program koji omogućava korisnicima programiranje Arduino mikroupravljača. Arduino platforma se sastoji od hardvera (raznih Arduino ploča) i programa (Arduino IDE), što omogućava razvoj različitih elektroničkih projekata.

Prema [1] Arduino IDE je prvenstveno razvijen kako bi programiranje mikroupravljača bilo dostupno i jednostavno za sve, bez obzira na predznanje o programiranju. Program je kompatibilan s više operativnih sustava, uključujući Windows, macOS i Linux. Sam IDE koristi jednostavan jezik temeljen na C/C++ sintaksi, što olakšava početnicima da uče programiranje i elektroniku.



Slika 2.1. *Prikaz korisničkog sučelja Arduino IDE za programiranje mikroupravljača*

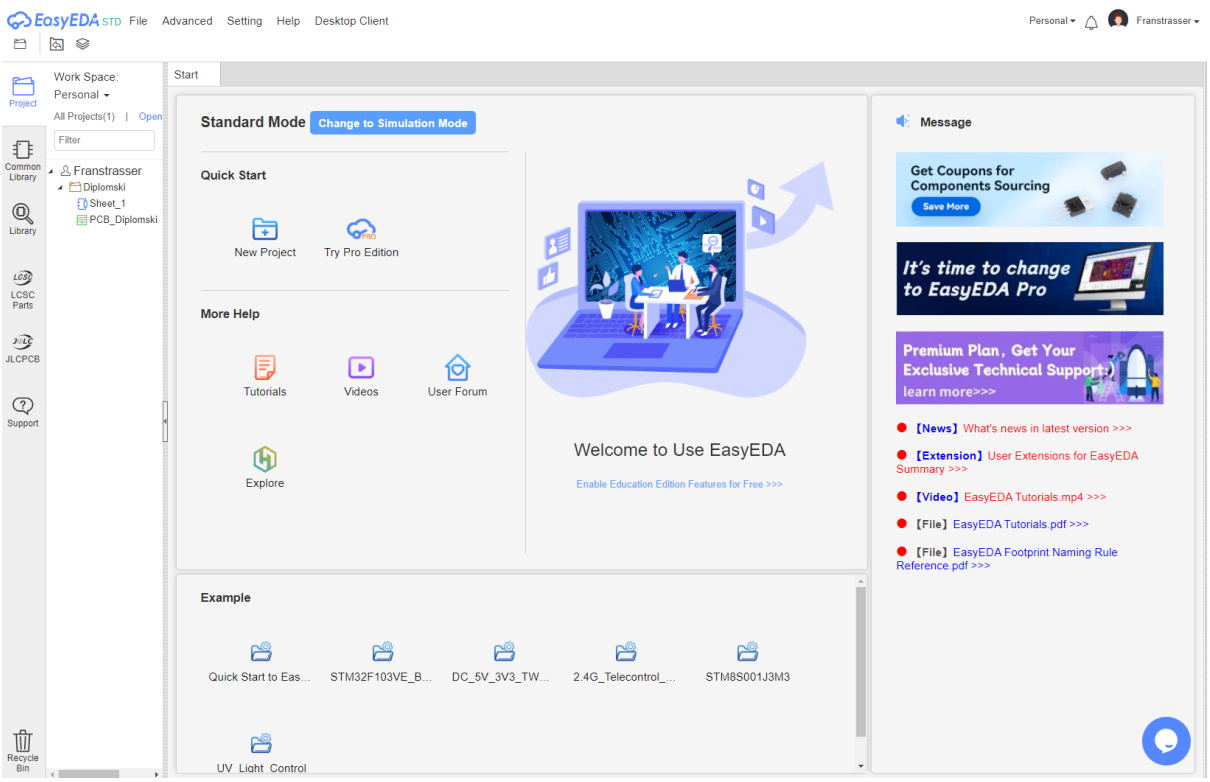
Jedna od ključnih prednosti Arduino IDE-a je njegova intuitivnost. Glavno sučelje se sastoji od nekoliko osnovnih dijelova: prostora za pisanje koda, konzole za prikazivanje poruka o pogreškama, alatne trake za učitavanje i provjeru koda te serijskog monitora, koji omogućava interakciju s pločom u stvarnom vremenu (Slika 2.1.).

Arduino IDE dolazi s ugrađenim bibliotekama, što omogućava lakše korištenje različitih senzora, aktuatora i modula. Korisnici također mogu preuzeti dodatne biblioteke i proširiti mogućnosti svojih projekata. IDE automatski uključuje kompilator, koji prevodi kod u oblik pogodan za mikroupravljač, te bootloader, koji omogućava jednostavno učitavanje koda na ploču.

2.2. EasyEDA

EasyEDA je besplatni, web-bazirani (engl. *web-based*) alat za dizajn elektroničkih sklopova i izradu tiskanih pločica (PCB). Razvijen je s ciljem da olakša inženjerima i studentima proces dizajniranja i simulacije elektroničkih krugova. EasyEDA kombinira korisničku jednostavnost s moćnim alatima, pružajući sve što je potrebno za izradu elektroničkih projekata, od shematskih dijagrama do gotovih PCB-ova.

Prema [2] jedna od ključnih značajki EasyEDA alata je njegova web-bazirana priroda. To znači da korisnici ne moraju instalirati program na svoje računalo; sve što im je potrebno je pristup internetu. Međutim, EasyEDA nudi i offline verziju za one koji preferiraju rad bez internet veze. Također, omogućava jednostavno spremanje projekata u oblak, što korisnicima olakšava pristup projektima s bilo kojeg uređaja i u svakom trenutku.



Slika 2.2. *Prikaz korisničkog sučelja EasyEDA alata*

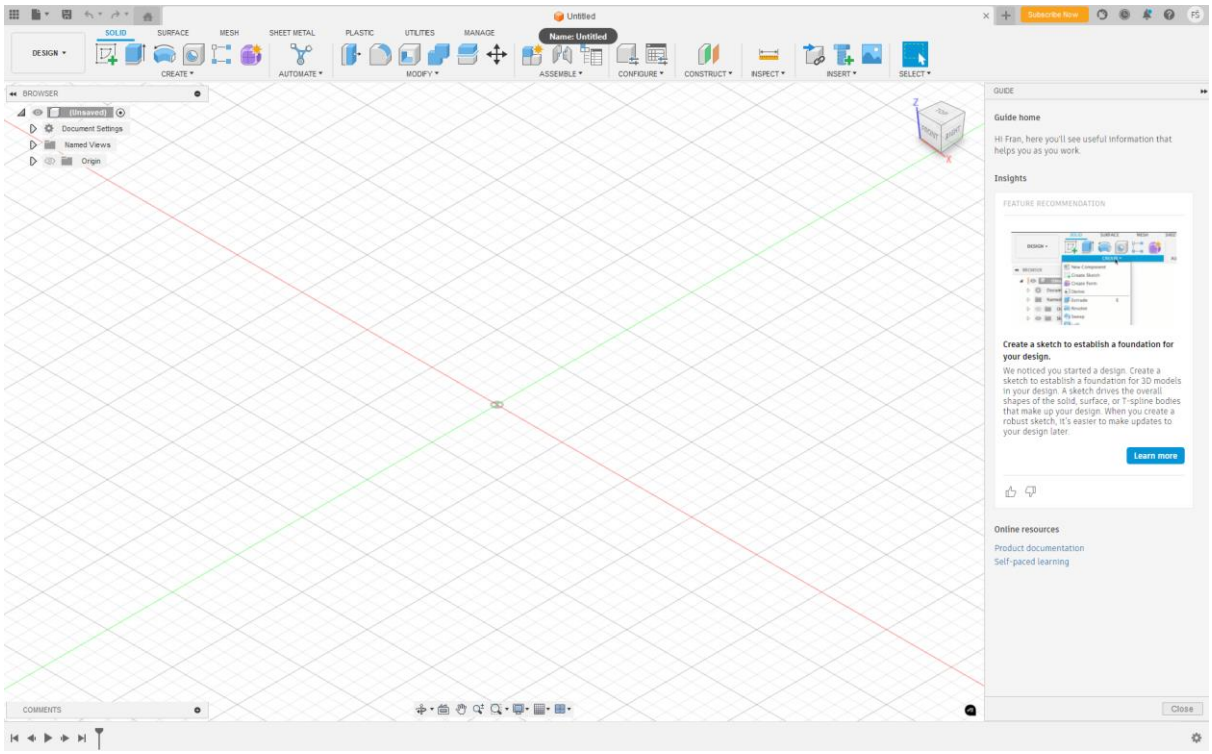
EasyEDA integrira tri glavne funkcionalnosti: dizajn sheme, simulaciju kruga i dizajn PCB-a. Korisnici mogu započeti s crtanjem shematskih dijagrama koristeći opsežnu biblioteku komponenata koja uključuje sve od osnovnih otpornika i kondenzatora do složenih IC-ova (engl. *Integrated circuit*). Nakon izrade sheme, alat omogućava simulaciju elektroničkog kruga, što je korisno za testiranje funkcionalnosti i provjeru grešaka prije izrade samog PCB-a. U svrhu izrade PCB-a, EasyEDA nudi sučelje za postavljanje komponenti i kreiranje i podešavanje veza među različitim komponentama. Alat nudi i automatsko podešavanje veza među komponentama.

EasyEDA također podržava uvoz i izvoz datoteka iz drugih alata poput Eagle-a, što olakšava prijenos projekata.

2.3. Autodesk Fusion

Prema [3] Autodesk Fusion je CAD (engl. *Computer-Aided Design*), CAM (engl. *Computer-Aided Manufacturing*) i CAE (*Computer-Aided Engineering*) alat koji je razvila tvrtka Autodesk. Namijenjen je za 3D modeliranje, dizajn, inženjering i proizvodnju, a zbog svoje svestranosti i moćnih funkcija koristi se u raznim industrijama, uključujući proizvodnju, arhitekturu i industrijski dizajn.

Jedna od ključnih prednosti Fusion 360 alata je njegova mogućnost odrađivanja različitih faza razvoja proizvoda unutar jedne platforme. Korisnici mogu započeti s konceptom, zatim stvoriti detaljne inženjerske modele, provesti simulacije kako bi testirali performanse te pripremiti modele za proizvodnju putem CNC strojeva ili 3D printera. Ova integracija omogućuje učinkovito upravljanje cijelim procesom dizajna, bez potrebe za korištenjem više različitih alata.



Slika 2.3. *Prikaz Autodesk Fusion korisničkog sučelja*

Autodesk Fusion podržava različite tehnike modeliranja, uključujući parametarsko modeliranje, slobodno modeliranje i površinsko modeliranje. Parametarsko modeliranje omogućava precizno definiranje geometrije uz upotrebu dimenzija i ograničenja, dok slobodno modeliranje pruža fleksibilnost u kreiranju kompleksnih organskih oblika. Površinsko modeliranje je korisno za stvaranje složenih oblika i detaljnih površina, što ga čini idealnim za industrijski dizajn.

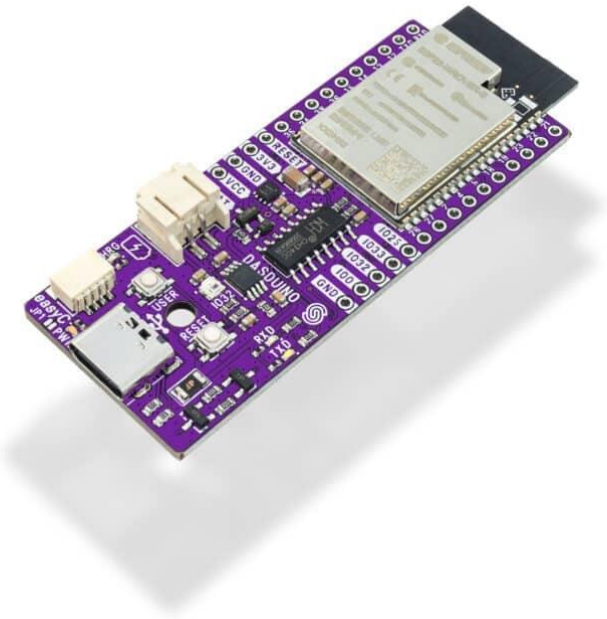
3. KORIŠTENE KOMPONENTE

U svrhu realizacije projektnog zadatka korištene su brojne komponente koje su odabrane istraživanjem uz savjet mentora.

3.1. Dasduino CONNECTPLUS

Dasduino CONNECTPLUS je razvojna ploča dizajnirana za kreiranje naprednih IoT rješenja, sadržavajući ESP32-WROVER-E mikročip koji pruža više mogućnosti bežične komunikacije putem Wi-Fi i Bluetooth modula. [4] Programiranje ove ploče se odvija u Aduino integriranom programskom okruženju koje pruža pregršt dostupnih biblioteka za različite komponente.

Wi-Fi modul omogućava Dasduino CONNECTPLUS ploči pristup bežičnim mrežama, što omogućuje prijenos podataka u stvarnom vremenu, daljinsko upravljanje uređajima i povezivanje s cloud servisima. Bluetooth modul omogućava lokalnu bežičnu komunikaciju s mobilnim uređajima, što je korisno za aplikacije koje zahtijevaju upravljanje na kratkoj udaljenosti.



Slika 3.1. *Prikaz Dasduino CONNECTPLUS razvojne ploče*

Tablica 3.1. *Dasduino CONNECTPLUS tehničke karakteristike*

Napon rada	3.3V (integrirani 5V regulator)
Broj pinova	30 pinova (digitalni, analogni, PWM)
Komunikacija	UART, SPI, I2C
Konektori	easyC, USB Type-C (ženski), JST baterijski

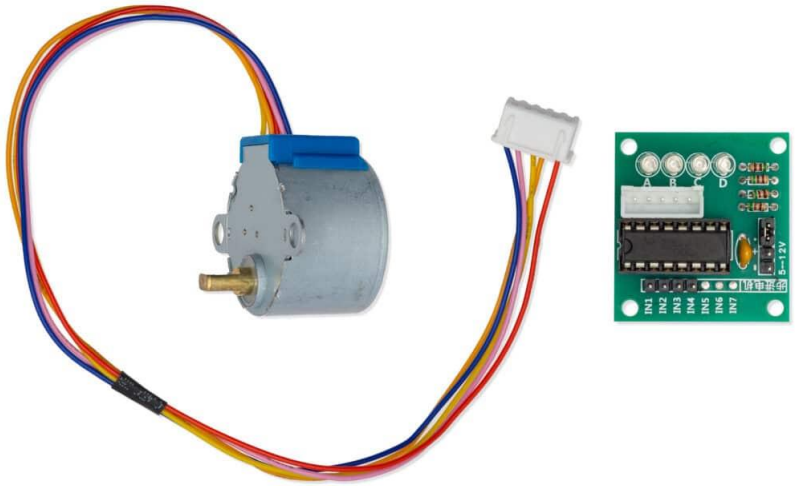
3.2. Stepper motor sa ULN2003 driver-om

Stepper motor je vrsta motor koja može vrlo precizno kontrolirati svoje okrete. Specifičan je po tome što se rotira u koracima, čime omogućuje točno kontroliranu kutnu poziciju.

ULN2003 je driver koji omogućuje jednostavno upravljanje stepper motorom putem mikroupravljača kao što je Dasduino CONNECTPLUS.

Kombinacija stepper motora i ULN2003 driver-a omogućava preciznu kontrolu brzine i smjera rotacije. [5] Stepper motor se obično sastoji od četiri namotaja, a ULN2003 omogućava upravljanje njima kroz četiri izlaza, omogućujući korak po korak rotaciju motora. Ovisno o potrebnoj preciznosti, može se birati između punih koraka, polu-koraka ili mikrokoraka, čime se dodatno povećava kontrola nad pozicioniranjem.

Na Dasduino se spaja sa četiri komunikacijske žice i po jednom žicom za napajanje i uzemljenje, a za programiranje ove komponente unutar Arduino integriranog razvojnog okruženja dostupno je više biblioteka.



Slika 3.2. *Prikaz stepper motora sa ULN2003 driver-om*

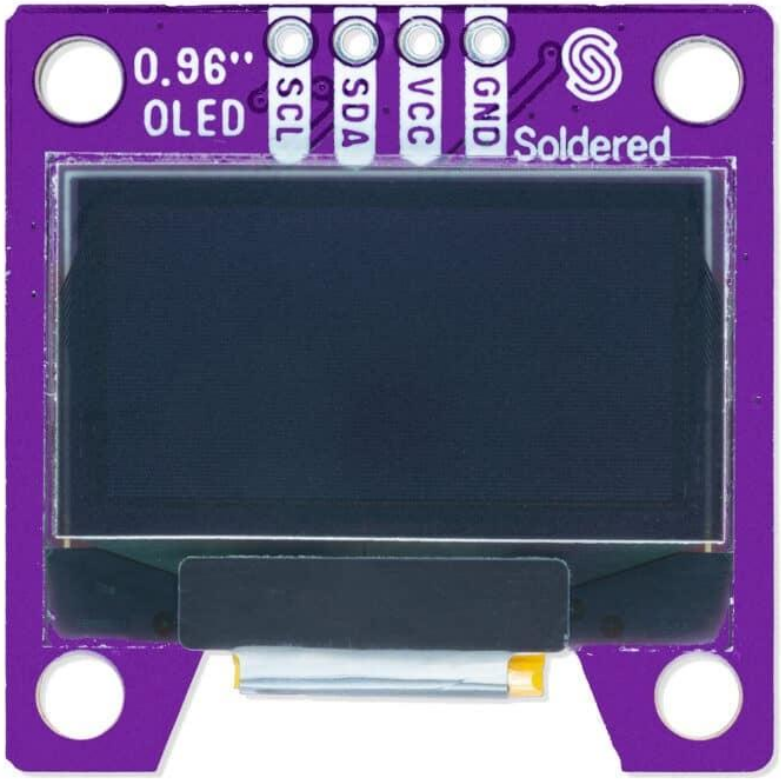
Tablica 3.2. *Tehničke karakteristike stepper motora sa ULN2003 driver-om*

Napon driver-a	5V - 12V
Napon motora	
Korak motora	5.625º
Maksimalna struja driver-a	500 mA
Dimenzije pločice driver-a	41mm x 21 mm

3.3. SSD1306 OLED ekran

OLED I2C 0.96" SSD1306 je zaslon koji se koristi u raznim elektroničkim projektima zbog svoje energetske učinkovitosti, jednostavne integracije i jasne grafike. [6] Zaslon koristi OLED tehnologiju, što bi značilo da svaki piksel sam emitira svjetlost pa zaslon ne zahtijeva pozadinsko osvjetljenje.

Zaslon koristi SSD1306 upravljački čip koji omogućava upravljanje pojedinačnim pikselima putem serijskog sučelja. Upravljački čip podržava I2C (engl. *Inter-Integrated Circit*) i SPI (engl. *Serial Peripheral Interface*) načine komunikacije.



Slika 3.3. *Prikaz SSD1306 OLED ekrana*

Tablica 3.3. *Tehničke karakteristike SSD1306 OLED ekrana*

Dijagonala zaslona	širina 0,96 inča
Radni napon	5V (ugrađeni regulator za 3,3V)
Komunikacija	I2C (adresa: 0x3C)
Priključci	2x easyC

3.4. XL6009 step-up modul

XL6009 je step-up (boost) pretvarač napona koji se koristi za povećanje ulaznog napona na zahtjevanu razinu izlaznog napona pomoću potenciometra smještenog na modulu. [7] Radi sa ulaznim naponima u rasponu od 5V do 32V dok na izlazu daje napon u rasponu od 5V do 35V (Tablica 3.4). Jedna od prednosti ovog pretvarača je njegova efikasnost, koja je veća od 94% što znači da dolazi do minimalnog rasipanja energije u obliku topline što znači manja potreba za hlađenjem i povećanje dugotrajnosti sustava.

Slika 3.4. *Prikaz XL6009 step-up modula*

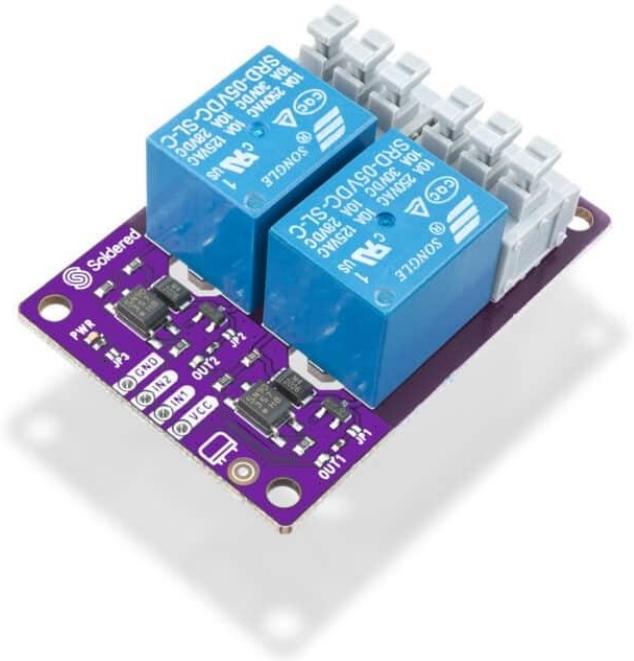
Tablica 3.4. *Tehničke karakteristike XL6009 step-up modula*

Ulazni napon	5-32V
Izlazni napon	podesiv 5-35V
Ulazna struja	maksimalno 10A
Efikasnost	veća od 94%

3.5. 2-kanalni relej

Releji su široko korištena komponenta u Arduino projektima. Pružaju mogućnost upravljanja uređajima visoke struje sa signalom mikroupravljača niske struje. Ovaj modul omogućuje Arduinu da pali/gasi električne uređaje čineći ga idealnim za automatizaciju i kontrolu.

2-kanalni relej se sastoji od dva releja, svaki od njih može upravljati zasebnim krugom. Svaki relej ima tri terminala: common (COM), normally open (NO), normally closed (NC). [8] Kad se relej aktivira digitalnim signalom common terminal je spojen na normally open terminal zatvarajući krug. U suprotnom, ako nije aktiviran, common terminal je spojen na normally closed terminal.



Slika 3.5. *Prikaz 2- kanalnog releja*

Tablica 3.5. *Tehničke karakteristike 2-kanalnog releja*

Maksimalni DC napon	30V
Maksimalna DC struja	10A
Maksimalni AC napon	250V
Maksimalna AC struja	10A

3.6. Pumpa za vodu

12V DC pumpa za vodu pruža efikasno rješenje za protok vode u Arduino projektima. [9] Pumpom se upravlja preko releja , a kao njeno napajanje se koristi XL6009 step-up modul.

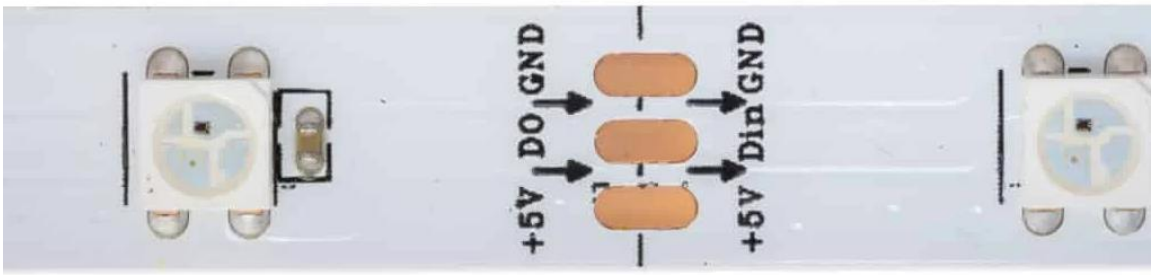
Slika 3.6. *Prikaz pumpe za vodu*

Tablica 3.6. *Tehničke karakteristike 12V DC pumpe za vodu*

Napon	12V
Brzina protoka	240L/h
Vanjski promjer cjevčice za odvod	8 mm

3.7. WS2812B RGB LED traka

Ova fleksibilna RGB LED traka je jednostavan način kako dodati kompleksno osvjetljenje u bilo koji projekt. Svaka pojedina LED dioda ima integrirani driver koji omogućuje kontroliranje boje i svjetline svake LED zasebno. [10] IC ove trake je WS2812B. Pomoću samo jedne žice može se kontrolirati tisuće ovakvih dioda. Za programiranje i upravljanje trakom u Arduino projektima postoje već definirane biblioteke za njeno jednostavno korištenje.



Slika 3.7. *Prikaz RGB LED trake*

Tablica 3.7. *Tehničke karakteristike WS2812B RGB LED trake*

Napon	5V
Snaga	8W/m
Struja	1.6A/m
Broj dioda	30 dioda po metru
Vrsta dioda	WS2812 SMD LED

3.8. Membranska tipkovnica

Za svrhu projekta koristi se membranska tipkovnica sa dvije tipke. Ona je često korištena u Arduino projektima zbog svoje jednostavnosti i niske cijene. Dva gumba smještena su na tankom sloju fleksibilne plastike s električnim kontaktima koji omogućuju komunikaciju s Dasduino mikroupravljačem kada se pritisnu.

U projektu gornji gumb služi za mijenjanje OLED zaslona, a donji gumb koristi za pokretanje akcije ovisno o onome na kojem se zaslonu trenutno korisnik nalazi. Na mikroupravljač se spaja sa jednim 3-pinskim konektorom koji služi za komunikaciju i predstavlja broj redaka i stupaca na tipkovnici (Tablica 3.8). [11] Upravljanje je omogućeno već dostupnim bibliotekama u Arduino integriranom razvojnom okruženju.

Slika 3.8. *Prikaz membranske tipkovnice sa dvije tipke*

Tablica 3.8. *Tehničke karakteristike membranske tipkovnice*

Broj tipki	2
Dužina tipkovnice sa konektorom	90 mm
Tip konektora	1x3 pin 2.54 mm ženski konektor

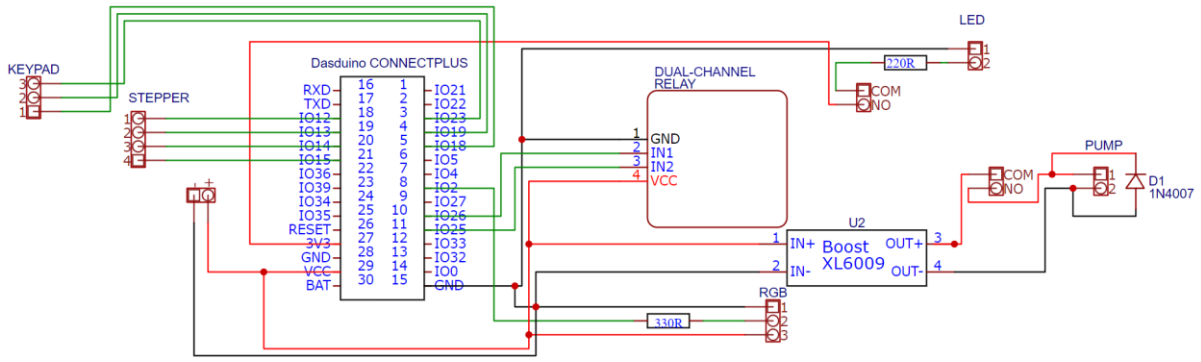
4. REALIZACIJA SUSTAVA

4.1. Realizacija sklopovskog rješenja

Realizacija sklopovlja ovisi o brojnim faktorima. Bilo je potrebno odabrati odgovarajuće komponente s obzirom na potrebe sustava što je učinjeno na temelju istraživanja i prijedlogu mentora. Također je bilo potrebno odabrati odgovarajuću Arduino pločicu. S obzirom da sustav mora biti spojen na internet, u obzir su dolazile pločice sa ESP32 i ESP8266 čipom. Na prijedlog mentora odabrana je pločica sa ESP32 čipom zbog njezinih performansi u odnosu na ESP8266. ESP32 čip ima dvojezgreni procesor sa taktnom frekvencijom od 160MHz do 240MHz dok ESP8266 ima jednojezgreni procesor sa taktnom frekvencijom od 80MHz.

4.1.1. Električna shema sustava

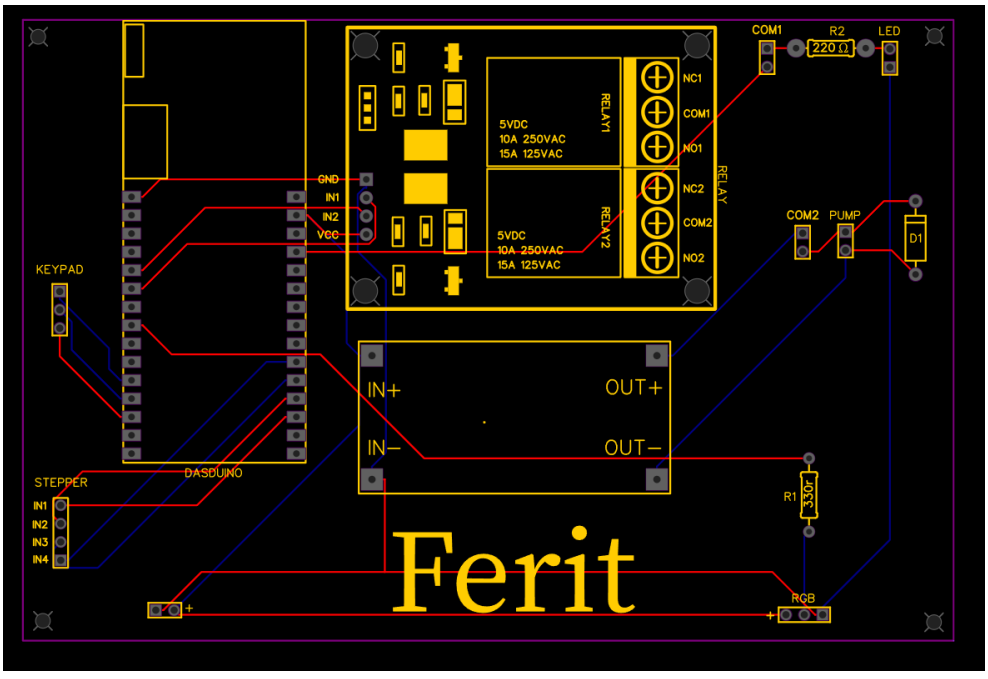
Analizom karakteristika komponenti zaključeno je utvrđeno je da većina komponenti zahtijeva napon od 5V za rad dok je izuzetak pumpa za vodu kojoj je potreban napon od 12V. Pumpa se napaja pomoću XL6009 step-up modula koji na ulazu prima 5V od Dasduina, a na izlazu daje napon od 12V podešen pomoću potenciometra na komponenti. Paralelno na pumpu za vodu je spojena 1N4007 flyback dioda koja rješava problem naglog porasta napona pri prekidu strujnog kruga kojeg radi relej (Slika 4.1.). MIkroupravljač radi pod naponom od 5V i ima USB-C priključak preko kojeg je spojen u AC-DC strujni adapter. Adapter daje izlazni napon od 5V i struju od 2A.



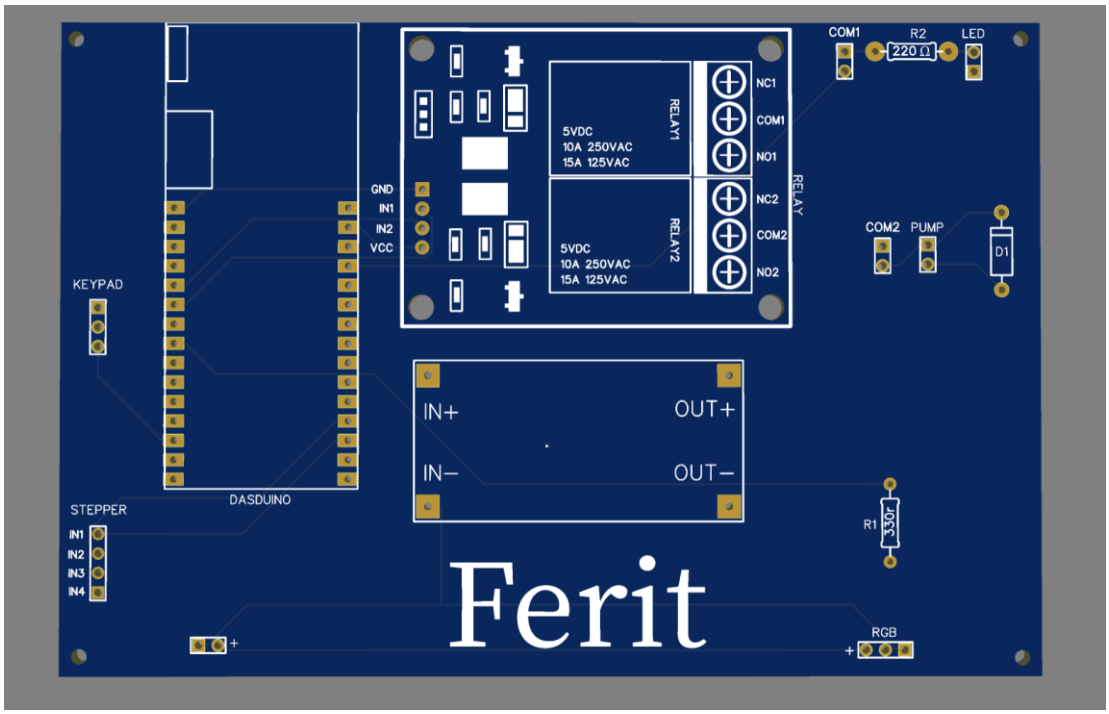
Slika 4.1. *Prikaz električne sheme sustava*

4.1.2. Dizajn PCB-a

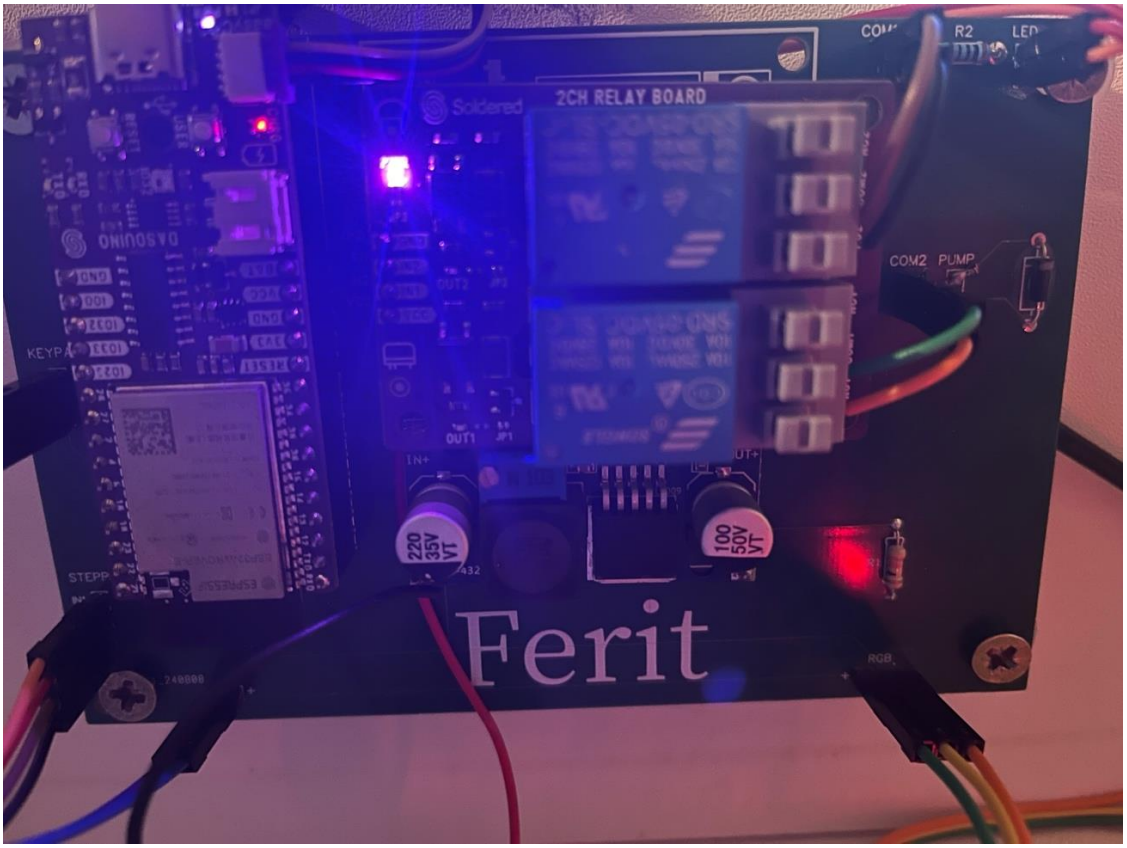
Dizajn PCB-a je napravljen u programu EasyEDA pomoću električne sheme sustava. Sve komponente su postavljene unutar mjesta označenog za PCB i odabran je auto route u izborniku kako bi se automatski podesile veze između komponenti (Slika 4.2.). Nakon automatskog podešavanja veza među komponentama EasyEDA omogućuje 3D prikaz PCB-a i njegovo naručivanje preko partnera JLCPCB-a (Slika 4.3.).



Slika 4.2. *Prikaz veza među komponentama u EasyEDA programu*

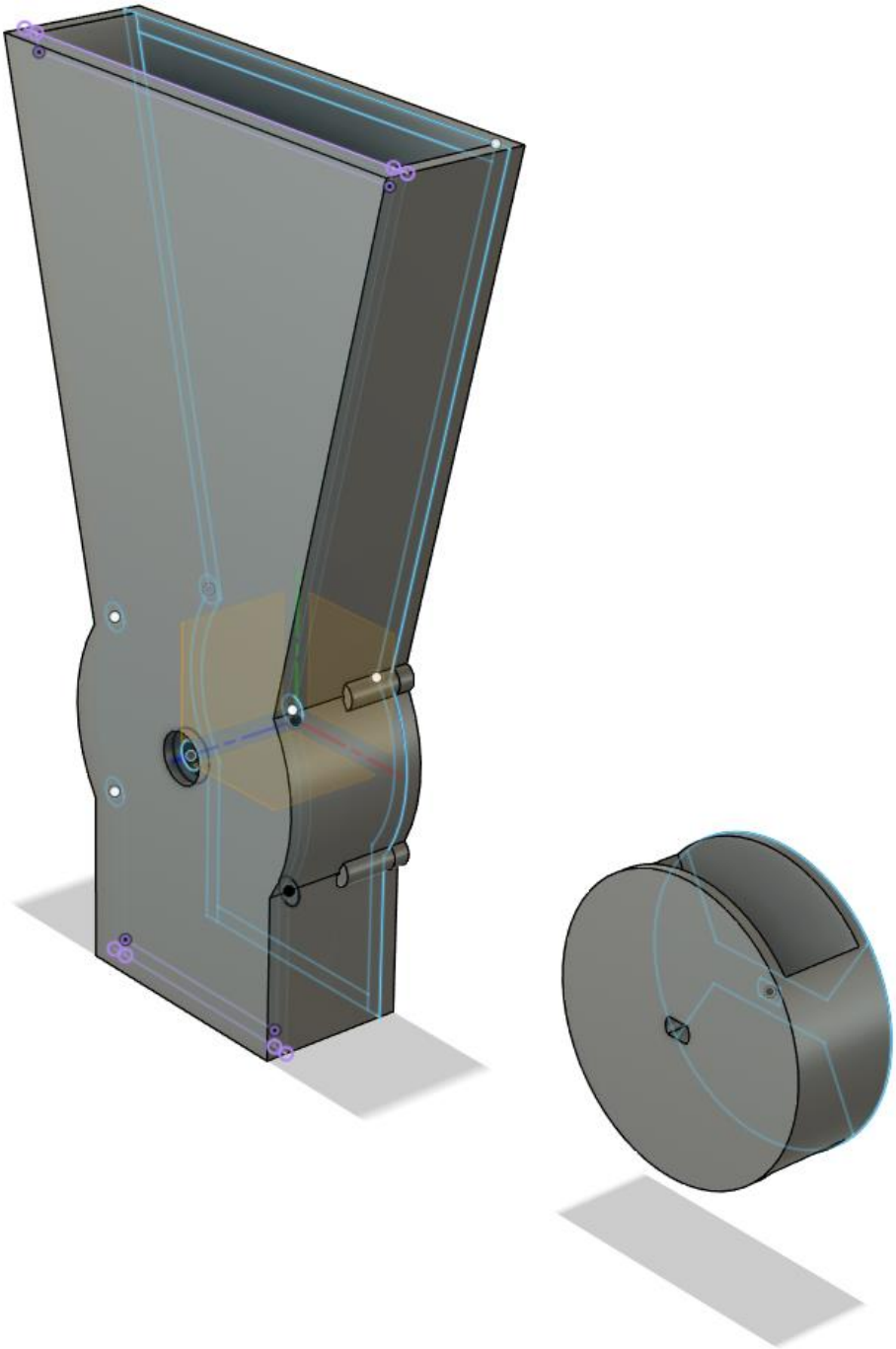


Slika 4.3. *Prikaz dizajniranog PCB-a unutar EasyEDA programa*



Slika 4.3. *Prikaz PCB-a sa zalemljenim komponentama*

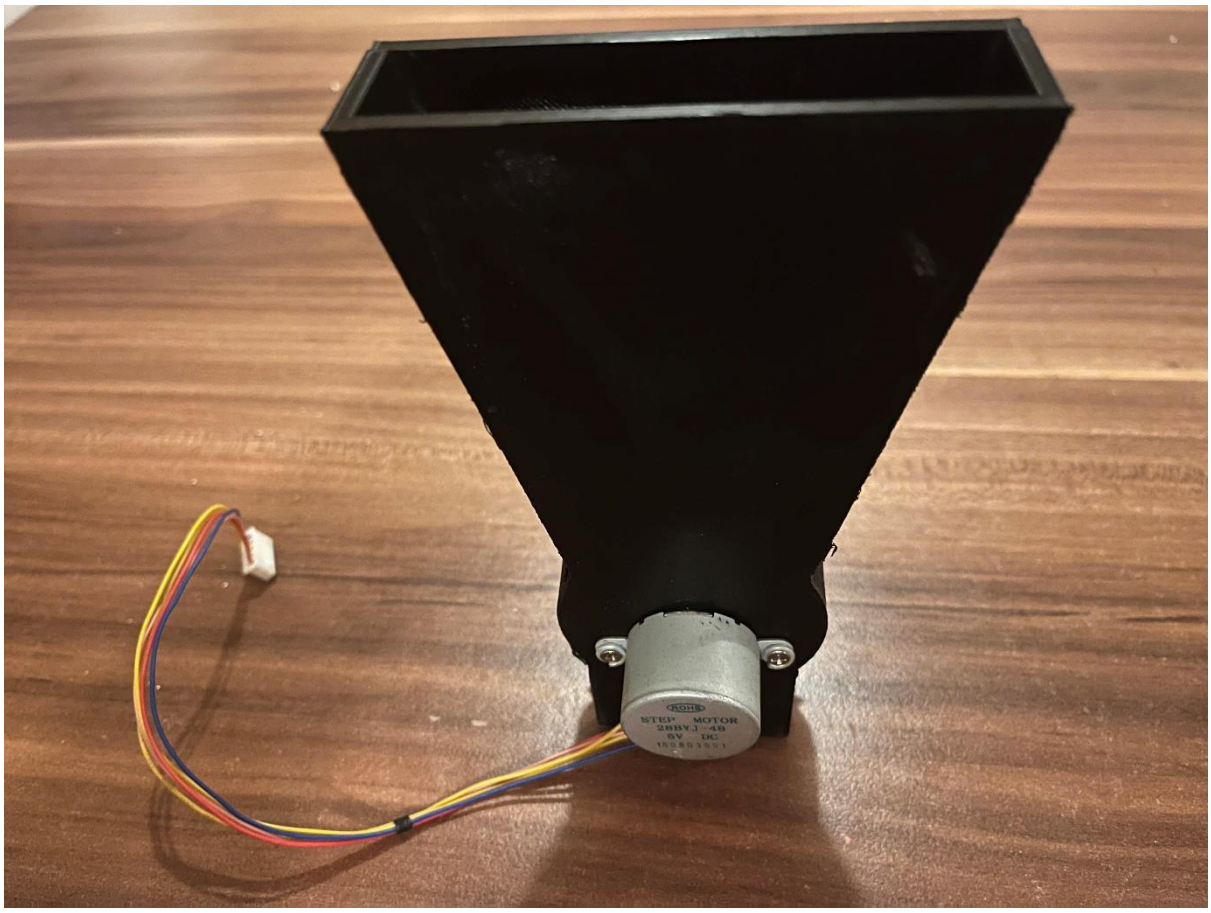
4.1.3. Dizajn hranilice



Slika 4.4. *Prikaz 3D modela hranilice unutar Autodesk Fusion programa*

3D model hranilice je izrađen u već spomenutom Autodesk Fusion-u. Napravljena je u tri dijela. Jedan dio predstavlja kućište, jedan poklopac tog kućišta i dio koji se okreće unutar tog kućišta. Na jednoj strani kućišta je napravljena okrugla rupa promjera većeg od širine priključka na stepper motoru kako bi se samo odgovarajući dio neometano okretao. Na dijelu koji se okreće je napravljena rupa oblika i dimenzija priključka na stepper motoru. Poklopac se učvrsti na kućište hranilice pomoću četiri šarafa dok se stepper motor učvrsti na hranilicu pomoću dva šarafa.

Kućište ima dva otvora: gornji za nadopunu hrane i donji gdje hrana upada u akvarij. Hranilica je zamišljena i napravljena na principu da dio koji se okreće ima dva prostora za hranu smještena nasuprot jedan drugome što znači da stepper motor u jednoj revoluciji nahrani ribe dvaput. Jedno hranjenje predstavlja okretanje stepper motora za kut od 180°.



Slika 4.5. *Prikaz 3D isprintanog modela hranilice sa stepper motorom*

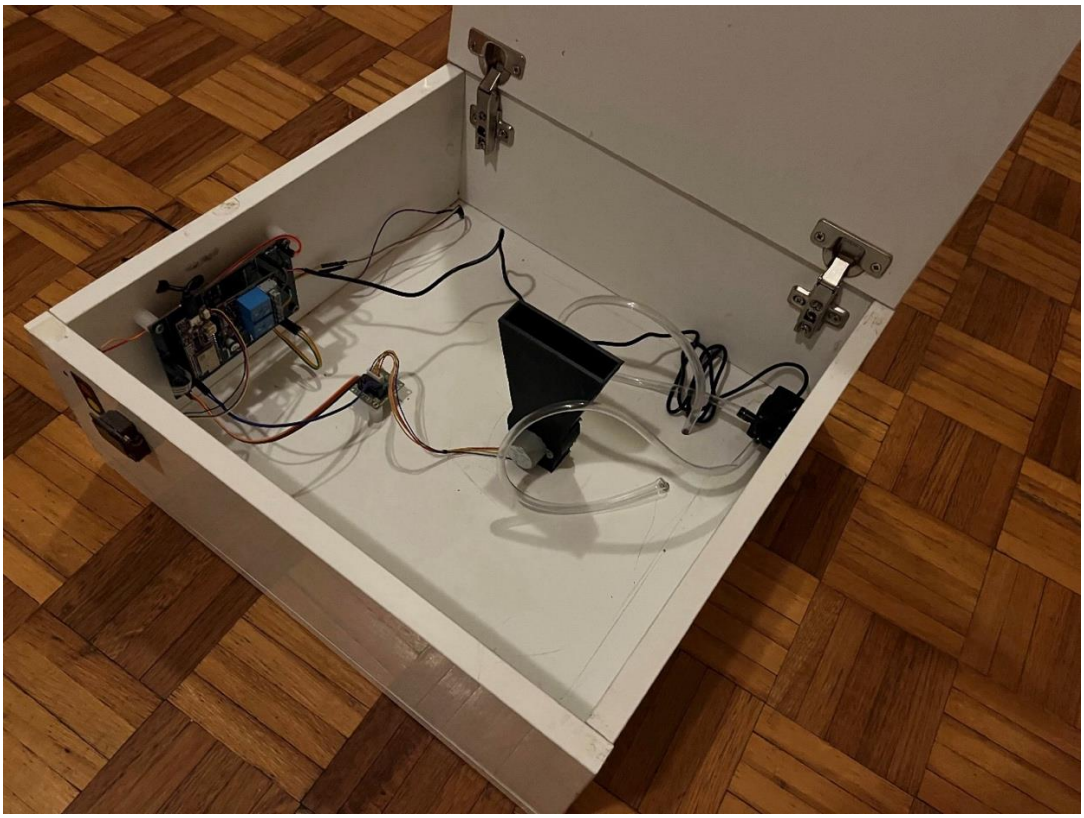
4.1.4. Dizajn kućišta

Kućište je napravljeno od drveta na način da su sve strane kućišta spojene drvenim tiplovima i drvofix-om (Slika 4.6.). Stavljena je šarka kako bi se gornji dio kućišta mogao otvarati da se hrana za ribice može napuniti. U podnožje kućišta urezano je mjesto za RGB LED traku u koje je ona stavljena i zalivena epoxy smolom kako se nebi moralo brinuti o tome da traka dolazi u kontakt s vodom. Izbušena je po jedna rupa za:

- Dovod vode do pumpe za vodu
- Odvod vode od pumpe za vodu
- Kabel za napajanje sustava
- LED diodu koja predstavlja grijanje
- Kabel od OLED ekrana

U podnožju kućišta je također napravljena pravokutna rupa dimenzija koje odgovaraju donjem otvoru hranilice kako bi se hranilica mogla uglaviti u nju i hrana mogla upadati u akvarij.

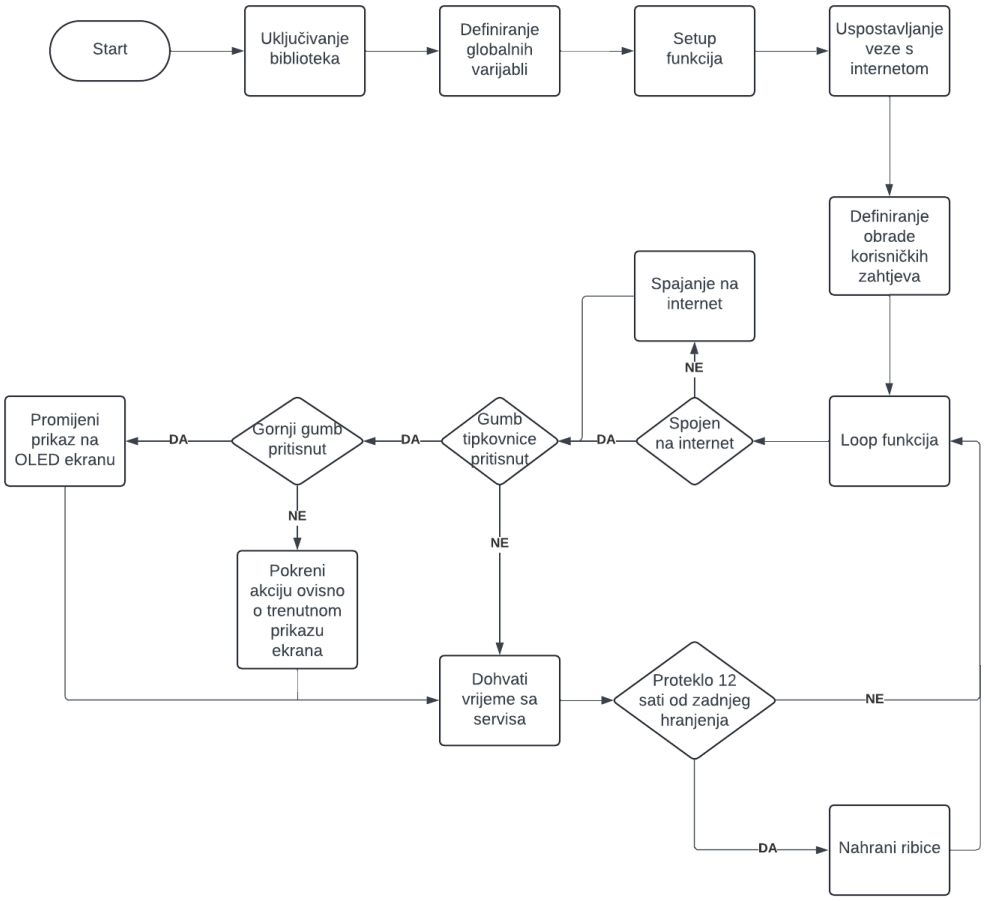
Slika 4.6. *Prikaz izrađenog kućišta*



Slika 4.7. *Prikaz unutrašnjosti kućišta*

4.2. Realizacija programskog rješenja

Nakon spajanja svih komponenti sa mikroupravljačem, izrade hranilice i kućišta, programiran je rad sustava.



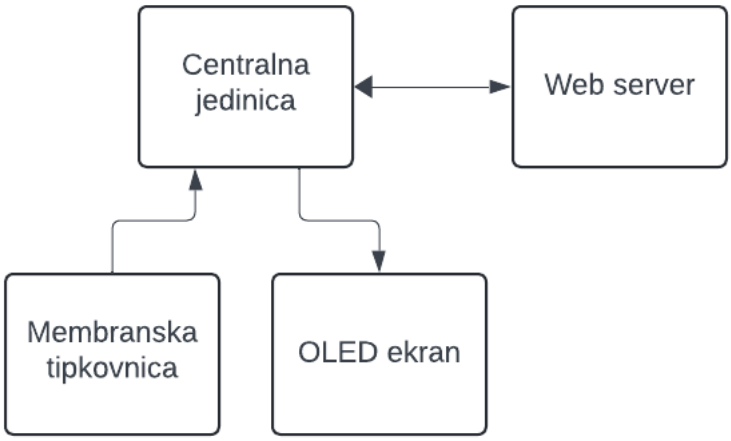
Slika 4.8. *Blok dijagram algoritamskog rješenja*

Za pravilno korištenje vanjskih biblioteka potrebno ih je prvo uključiti u projekt. Korištene biblioteke su:

- <WiFi.h> - [12] biblioteka koja omogućava WiFi support ESP32 čipovima
- <time.h> - [13] biblioteka za dohvaćanje vremena
- <Adafruit_NeoPixel.h> - [14] biblioteka za rad sa LED trakama
- <Keypad.h> - [15] biblioteka za rad s membranskim tipkovnicama
- <Wire.h> - [16] biblioteka koja omogućava komunikaciju sa I2C uređajima
- <Adafruit_SSD1306.h> - [17] biblioteka koja omogućava rad s OLED ekranima sa SSD1306 driver-ima
- <Adafruit_GFX.h> - [18] biblioteka za rad s LCD i OLED ekranima
- <Stepper.h> - [19] biblioteka za rad s unipolarnim i bipolarnim stepper motorima
- <ESPAsyncWebServer.h> - [20] biblioteka koja omogućava rad asinkronog web server-a

Nakon uključivanja biblioteka definirane su globalne varijable i inicijalizirani su objekti. Potom se pokreće setup funkcija u kojoj se pokreću objekti, čeka se dok se mikroupravljač ne spoji na internet, dohvati se trenutno vrijeme, nahrane se ribice i spremi se vrijeme hranjenja u globalnu varijablu koja nam služi kako bi pratili zadnje vrijeme hranjenja. Prvi ispis na OLED ekran nakon spajanja mikroupravljača na internet je ispis IP adrese za pristup web server-u uređajima koji su spojeni na istu mrežu. Algoritamski je podešeno da, ukoliko korisnik ručno ne pokrene hranjenje putem web server-a ili putem membranske tipkovnice, se hranjenje ribica odvija svakih 12 sati. U setup funkciji je također definirano koje funkcije će se pozivati ukoliko korisnik uputi zahtjev prema centralnoj jedinici putem asinkronog web server-a. Po završetku setup funkcije dolazimo do loop funkcije u kojoj se provjerava je li centralna jedinica spojena na internet te ukoliko nije odmah pokreće ponovno spajanje. Ukoliko je spojena ili spajanje bude uspješno dohvaća se vrijeme i uspoređuje je li prošlo 12 sata od prošlog hranjenja, ako je pokreće se postupak hranjenja ribica. Potom se unutar loop funkcije dohvaća unos sa tipkovnice ukoliko je do njega došlo te shodno tome pokreće akciju ili promjene ekrana ili, ovisno na kojem se klijent ekranu nalazi, pokreće određenu akciju. Za detaljan pregled programskog koda pogledati prilog **P.1.**

4.3 Realizacija upravljačkog i komunikacijskog sučelja

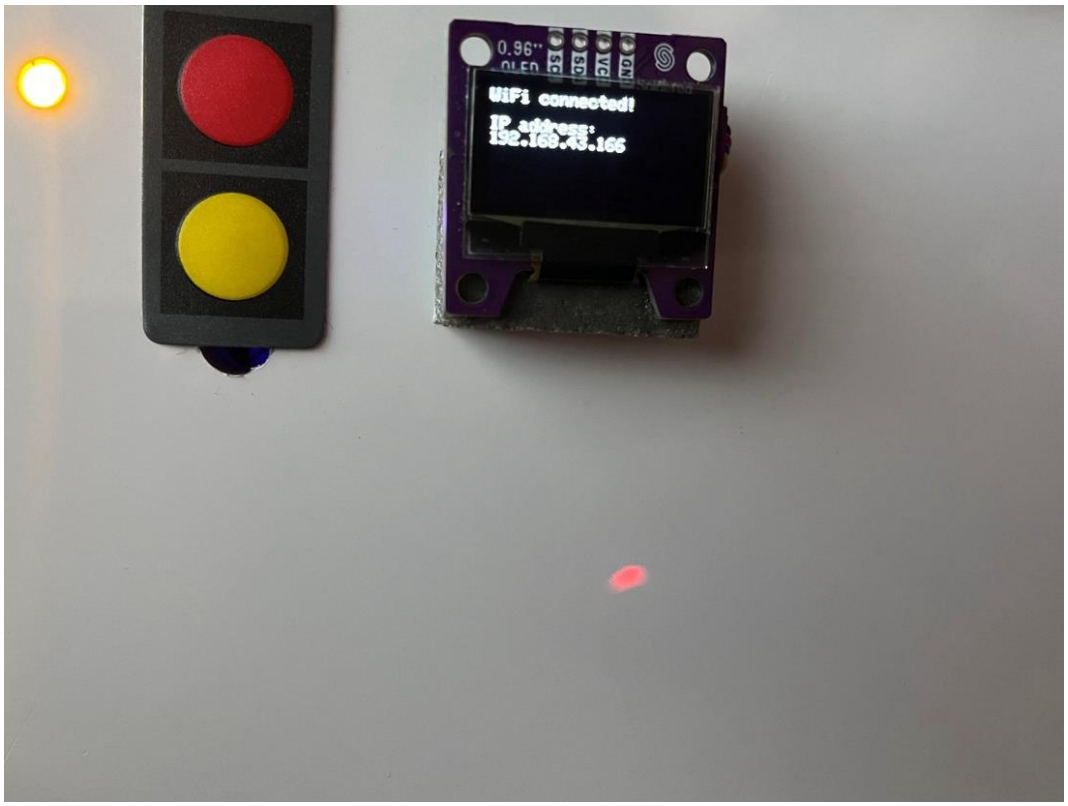


Slika 4.9 *Blok dijagram komunikacije*

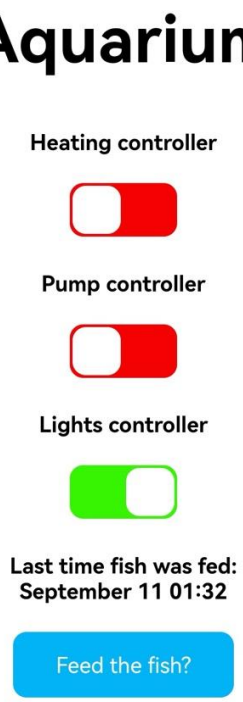
Upravljanje sustavom je omogućeno na dva načina: putem membranske tipkovnice i OLED ekrana (Slika 4.10..) i putem web server-a (4.11.).

Putem OLED ekrana korisnik na uvid dobiva trenutni status parametara kao što su grijanje (koje reprezentira LED dioda), osvjetljenje, pumpa za vodu, povezanost sa internetom i zadnje vrijeme kada su ribice nahranjene. Membranska tipkovnica omogućuje mijenjanje prikaza na OLED ekranu sa jednom tipkom, a drugom omogućuje pokretanje akcije u odnosu koji je trenutni prikaz na OLED ekranu. U slučaju da je na ekranu prikaz koji se odnosi na pumpu za vodu, grijanje ili osvjetljenje, donja tipka pali/gasi tu komponentu. Omogućeno je ručno pokretanje hranjenja putem donje tipke ukoliko je na ekranu prikaz zadnjeg vremena hranjenja.

Putem web server-a korisniku je prikazano i omogućeno upravljanje istim stvarima kao i putem membranske tipkovnice i OLED ekrana.



Slika 4.10. *Izgled korisničkog sučelja na kućištu*



Slika 4.11. *Izgled korisničkog sučelja na web server-u*

5. ZAKLJUČAK

U ovom diplomskom radu izrađen je mikroupravljački sustav za IoT autonomni akvarij. U radu je prikazano kako pomoću Dasduino CONNECTPLUS mikroupravljača možemo upravljati akvarijem na 2 načina, pomoću membranske tipkovnice i asinkronog web server-a. Izrađena je maketa, napravljen PCB i 3D isprintan dio koji predstavlja hranilicu koja se upravlja stepper motorom. Zbog velikog broja pinova na mikroupravljaču mogući su dodatni napredci u smislu dodavanja novih komponenta, a samim time i novih funkcionalnosti. Glavni problem uočen prilikom izrade diplomskog rada je pad napona u sustavu ukoliko je pumpa za vodu uključena zbog toga što se ona napaja pomoću step-up modula koji se napaja preko Dasduina. Taj problem bi se riješio da se osmisli drugačiji (vanjski) izvor napajanja za pumpu za vodu.

LITERATURA

[1] Overview of the Arduino IDE, dostupno na: https://docs.arduino.cc/software/ide-v1/tutorials/Environment/

[2] EasyEDA, dostupno na: https://easyeda.com/page/about

[3] Autodesk Fusion, dostupno na: https://www.autodesk.com/solutions/what-is-fusion-360.html

[4] Dasduino CONNECTPLUS, Soldered Electronics, 2022., dostupno na: https://github.com/SolderedElectronics/Dasduino-CONNECTPLUS-hardware-design

[5] Stepper motor, dostupno na: https://lastminuteengineers.com/28byj48-stepper-motor-arduino-tutorial/

[6] SSD1306, Solomon Systech, 2008., dostupno na: https://cdn-shop.adafruit.com/datasheets/SSD1306.pdf

[7] XL6009, dostupno na: https://www.haoyuelectronics.com/Attachment/XL6009/XL6009-DC-DC-Converter-Datasheet.pdf

[8] 2 kanalni relej, dostupno na: https://lastminuteengineers.com/two-channel-relay-module-arduino-tutorial/

[9] James Fuller, 12V Water Pump Control with Arduino, 2023., dostupno na: https://www.circuits-diy.com/12v-water-pump-control-with-arduino/

[10] WS2812B Intelligent control LED integrated light source, Worldsemi, dostupno na: https://cdn-shop.adafruit.com/datasheets/WS2812B.pdf

[11] Arduino Membrane Keypad, dostupno na: https://randomnerdtutorials.com/arduino-membrane-keypad-tutorial/

[12] WiFi.h biblioteka, dostupno na: https://github.com/espressif/arduino-esp32/blob/master/libraries/WiFi/src/WiFi.h

[13] time.h biblioteka, dostupno na: https://mikaelpatel.github.io/Arduino-RTC/de/df7/time_8h_source.html

[14] Adafruit_NeoPixel.h biblioteka, dostupno na: https://github.com/adafruit/Adafruit_NeoPixel/blob/master/Adafruit_NeoPixel.h

[15] Keypad.h biblioteka, dostupno na: https://github.com/Chris--A/Keypad

[16] Wire.h biblioteka, dostupno na: https://github.com/esp8266/Arduino/blob/master/libraries/Wire/Wire.h

[17] Adafruit_SSD1306.h biblioteka, dostupno na: https://github.com/adafruit/Adafruit_SSD1306

[18] Adafruit_GFX.h biblioteka, dostupno na: https://github.com/adafruit/Adafruit-GFX-Library

[19] Stepper.h biblioteka, dostupno na: https://github.com/arduino-libraries/Stepper

[20] ESPAsyncWebServer.h biblioteka, dostupno na: https://github.com/me-no-dev/ESPAsyncWebServer

SAŽETAK

Mikroupravljački sustav za IoT akvarij

Cilj rada je bio ostvariti mikroupravljački sustav za IoT autonomni akvarij. Prikazano je kako, uz pmoć Dasduino CONNECTPLUS mikroupravljača, upravljati sustavom putem membranske tipkovnice i asinkronog web server-a. Dodatno je izrađeno kućište, PCB pločica i 3D isprintana hranilica koju pokreće stepper motor. Korisnik, putem OLED ekrana i web server-a, dobiva na pregled trenutni status grijanja, pumpe, osvjetljenja i zadnjeg puta kad su ribice nahranjene.

ABSTRACT

Microcontroller system for IoT Autonomous Aquarium

The aim of the project was to create a microcontroller system for an IoT autonomous aquarium. It demonstrates how to control the system using a membrane keypad and an asynchronous web server, powered by the Dasduino CONNECTPLUS microcontroller. Additionally, a custom enclosure, PCB board, and a 3D-printed feeder driven by a stepper motor were developed. The user can view the current status of heating, pump, lighting, and the last feeding time of the fish via the OLED screen and the web server.

ŽIVOTOPIS

Fran Štrasser rođen je 28.07.1997. u Osijeku. Osnovnu školu je završio u Višnjevcu. 2016. godine završava III. gimnaziju u Osijeku. Iste godine upisuje preddiplomski sveučilišni studij Računarstva na Fakultetu Elektrotehnike, Računarstva i Informacijskih tehnologija u Osijeku koji završava 2021. godine. Iste godine upisuje diplomski studij Računarstva smjer Informacijske i podatkovne znanosti na istom fakultetu. Praksu odrađuje u International Value Services-u 2023. godine te završetkom prakse se zapošljava u istoj firmi kao Flutter developer.

PRILOZI

1. Programski kod

```
#include <WiFi.h> //Korištene biblioteke
#include "time.h"
#include <Adafruit_NeoPixel.h>
#include<Keypad.h>
#include <Wire.h>
#include <Adafruit_SSD1306.h>
#include <Adafruit_GFX.h>
#include "Stepper.h"
#include "ESPAsyncWebServer.h"

#define OLED_WIDTH 128
#define OLED_HEIGHT 64
#define OLED_ADDR   0x3C

const char* ssid = "xxxxxx"; //Parametri za spajanje na Wi-Fi
const char* password = "xxxxxx";

const char* ntpServer = "pool.ntp.org"; //Server za dohvaćanje vremena
const long  gmtOffset_sec = 7200;
const int   daylightOffset_sec = 0;

const byte ROWS = 2; //Parametri za korištenje membranske tipkovnice
const byte COLS = 1;

char keys[ROWS][COLS] =
{
  'A',
  'B',
};
byte rowPins[ROWS] = {18, 19};
byte colPins[COLS] = {23};

const int ledStripPin = 2; //Globalne varijable
const int numOfLeds = 60;
int red = 255;
int green = 255;
int blue = 255;

```

```
39  int pumpPin = 25;
40  int heatingPin = 26;
41  bool heatingOn = false;
42  bool pumpOn = true;
43  bool lightsOn = true;
44
45  enum DisplayScreenEnum { //Enum korišten za prikaz na OLED ekranu
46    WIFI,
47    HEATING,
48    PUMP,
49    LIGHTS,
50    FEEDER,
51  };
52
53  struct tm lastFeedingTime;
54  bool serverFeedingTrigger = false;
55
56  DisplayScreenEnum displayScreen = WIFI;
57
58  Adafruit_SSD1306 display(OLED_WIDTH, OLED_HEIGHT); //Instanciranje objekata
59  Adafruit_NeoPixel pixels = Adafruit_NeoPixel(numOfLeds, ledStripPin, NEO_GRB + NEO_KHZ800);
60  Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );
61  Stepper myStepper(2048, 12, 13, 14, 15);
62  AsyncWebServer server(80);
63
64  const char* PARAM_INPUT_1 = "output";
65  const char* PARAM_INPUT_2 = "state";
```

```
//Html kod za web server
const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML><html>
<head>
  <title>Aquarium</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" href="data:,">
  <style>
    html {font-family: Arial; display: inline-block; text-align: center;}
    h2 {font-size: 3.0rem;}
    p {font-size: 3.0rem;}
    body {max-width: 600px; margin:0px auto; padding-bottom: 25px;}
    .switch {position: relative; display: inline-block; width: 80px; height: 40px}
    .switch input {display: none}
    .slider {position: absolute; top: 0; left: 0; right: 0; bottom: 0;
    background-color: #f50202; border-radius: 10px}
    .slider:before {position: absolute; content: ""; height: 36px; width: 36px;
      left: 2px; bottom: 2px; top: 2px; background-color: #fff;
      -webkit-transition: .4s; transition: .4s; border-radius: 10px}
    input:checked+.slider {background-color: #37f502}
    input:checked+.slider:before {-webkit-transform: translateX(40px);
     -ms-transform: translateX(40px); transform: translateX(40px);}
    .button { background-color: #02b4f5; border: none; color: white;
      padding: 15px 32px; text-align: center; text-decoration: none;
      display: inline-block; font-size: 16px; border-radius: 10px}
  </style>
</head>
<body>
  <h2>Aquarium</h2>
  %BUTTONPLACEHOLDER%
<script>function toggleCheckbox(element) {
  var xhr = new XMLHttpRequest();
  if(element.checked){ xhr.open("GET", "/update?output="+element.id+"&state=1", true); }
  else { xhr.open("GET", "/update?output="+element.id+"&state=0", true); }
  xhr.send();
}
function feedFishRequest () {
  var xhttp = new XMLHttpRequest();
  xhttp.open("GET", "/feedFish", true);
  xhttp.send();
}
</script>
</body>
</html>
)rawliteral";
```

```
112  void setup() { // Setup funkcija
113    Serial.begin(115200);
114    pixels.begin();
115    pixels.setBrightness(100);
116    myStepper.setSpeed(5);
117    display.begin(SSD1306_SWITCHCAPVCC, OLED_ADDR);
118    display.clearDisplay();
119    display.setTextSize(0.5);
120    display.setTextColor(WHITE);
121    connectToWiFi();
122    pinMode(pumpPin, OUTPUT);
123    pinMode(heatingPin, OUTPUT);
124    ledStripController();
125    configTime(gmtOffset_sec, 0, ntpServer);
126    if (getLocalTime(&lastFeedingTime)) {
127      feedTheFish(lastFeedingTime, true);
128    }
129    digitalWrite(pumpPin, HIGH);
130    digitalWrite(heatingPin, LOW);
131    server.on("/", HTTP_GET, [](AsyncWebServerRequest * request) { //Rukovanje korisničkim zahtjevima sa web server-a
132      request->send_P(200, "text/html", index_html, processor);
133    });
134
135    server.on("/update", HTTP_GET, [] (AsyncWebServerRequest * request) {
136      String inputMessage1;
137      String inputMessage2;
138      if (request->hasParam(PARAM_INPUT_1) && request->hasParam(PARAM_INPUT_2)) {
139        inputMessage1 = request->getParam(PARAM_INPUT_1)->value();
140        inputMessage2 = request->getParam(PARAM_INPUT_2)->value();
141        serverChangeValues(inputMessage1.toInt(),inputMessage2.toInt());
142      }
143      request->send_P(200, "text/html", index_html, processor);
144    });
145    server.on("/feedFish", HTTP_GET, [](AsyncWebServerRequest *request){
146     serverFeedingTrigger = true;
147     struct tm timeInfo;
148     if (getLocalTime(&timeInfo)) {
149      lastFeedingTime = timeInfo;
150    }
151     request->send_P(200, "text/html", index_html, processor);
152    });
153    server.begin();
154  }
155
```

```
156  void loop() { // Loop funkcija
157    if (WiFi.status() != WL_CONNECTED) {
158      connectToWiFi();
159    }
160    char key = keypad.getKey();
161    struct tm timeInfo;
162    if (key)
163    {
164      if (key == 'A') {
165        changeScreen();
166        changeDisplayScreens();
167      }
168      else {
169        changeDisplayValues();
170      }
171    }
172    if (getLocalTime(&timeInfo)) {
173      feedTheFish(timeInfo, false);
174    }
175  }
176
177  void serverChangeValues(int firstValue, int secondValue){ //Funkcija za pokretanje akcija s obzirom na korisnički zahtjev sa web server-a
178    if(firstValue == 1){
179      if ((secondValue == 0 && heatingOn == true) || (secondValue == 1 && heatingOn == false)){
180              displayScreen = HEATING;
181              changeDisplayValues();
182              return;
183            }
184    }
185    if(firstValue == 2){
186      if ((secondValue == 0 && pumpOn == true) ||  (secondValue == 1 && pumpOn == false)) {
187              displayScreen = PUMP;
188              changeDisplayValues();
189              return;
190            }
191    }
192    if(firstValue == 3){
193      if ((secondValue == 0 && lightsOn == true) || (secondValue == 1 && lightsOn == false)) {
194              displayScreen = LIGHTS;
195              changeDisplayValues();
196              return;
197            }
198    }
199  }
```

```
201    void ledStripController() { //Paljenje/Gašenje RGB LED trake
202      pixels.clear();
203      for (int i = 0; i < numOfLeds; i++) {
204        pixels.setPixelColor(i, lightsOn ? pixels.Color(0, 0, 0) : pixels.Color(red, green, blue));
205        pixels.show();
206      }
207      lightsOn = !lightsOn;
208
209    }
210
211    void changeScreen() { //Funkcija za promjenu prikaza na OLED ekranu
212      switch (displayScreen) {
213        case WIFI:
214          displayScreen = HEATING;
215          return;
216        case HEATING:
217          displayScreen = PUMP;
218          return;
219        case PUMP:
220          displayScreen = LIGHTS;
221          return;
222        case LIGHTS:
223          displayScreen = FEEDER;
224          return;
225        case FEEDER:
226          displayScreen = WIFI;
227          return;
228      }
229    }
```

```
227  void changeDisplayScreens() {  //Funkcija koja obavlja prikaz na OLED ekranu
228    switch (displayScreen) {
229      case WIFI:
230        if (WiFi.status() != WL_CONNECTED) {
231          display.clearDisplay();
232          display.setCursor(0, 0);
233          display.println("WiFi not connected!");
234          display.println("Press down button to reconnect.");
235          display.display();
236        } else {
237          display.clearDisplay();
238          display.setCursor(0, 0);
239          display.println("WiFi connected!");
240          display.println("\nIP address: ");
241          display.println(WiFi.localIP());
242          display.display();
243        }
244        return;
245      case HEATING:
246        display.clearDisplay();
247        display.setCursor(0, 0);
248        display.print("Heating: ");
249        display.println(heatingOn ? "ON" : "OFF");
250        display.print("Turn heating ");
251        display.println(heatingOn ? "OFF?" : "ON?");
252        display.display();
253        return;
254      case PUMP:
255        display.clearDisplay();
256        display.setCursor(0, 0);
257        display.print("Pump: ");
258        display.println(pumpOn ? "ON" : "OFF");
259        display.print("Turn pump ");
260        display.println(pumpOn ? "OFF?" : "ON?");
261        display.display();
262        return;
263      case LIGHTS:
264        display.clearDisplay();
265        display.setCursor(0, 0);
266        display.print("Lights: ");
267        display.println(lightsOn ? "ON" : "OFF");
268        display.print("Turn lights ");
269        display.println(lightsOn ? "OFF?" : "ON?");
270        display.display();
271        return;
272      case FEEDER:
273        display.clearDisplay();
274        display.setCursor(0, 0);
275        display.println("Last feeding time:");
276        display.println(&lastFeedingTime, "\n%B %d\n%H:%M");
277        display.display();
278        return;
279    }
280  }
```

```
282  void changeDisplayValues() {  //Pokretanje akcije u odnosu na trenutni prikaz na OLED ekranu
283    switch (displayScreen) {
284      case FEEDER:
285        struct tm timeInfo;
286        if (getLocalTime(&timeInfo)) {
287          feedTheFish(timeInfo, true);
288        }
289        return;
290      case WIFI:
291        return;
292      case HEATING:
293        heatingOn ? digitalWrite(heatingPin, LOW) : digitalWrite(heatingPin, HIGH);
294        heatingOn = !heatingOn;
295        changeDisplayScreens();
296        return;
297      case PUMP:
298        pumpOn ? digitalWrite(pumpPin, LOW) : digitalWrite(pumpPin, HIGH);
299        pumpOn = !pumpOn;
300        changeDisplayScreens();
301        return;
302      case LIGHTS:
303        ledStripController();
304        changeDisplayScreens();
305        return;
306    }
307  }
308
309  void connectToWiFi() {  // Funkcija za spajanje na Wi-Fi
310    WiFi.begin(ssid, password);
311    display.clearDisplay();
312    display.setCursor(0, 0);
313    display.println("Connecting to ");
314    display.println(ssid);
315    display.display();
316    while (WiFi.status() != WL_CONNECTED) {
317      display.print(".");
318      display.display();
319      delay(100);
320    }
321    delay(2000);
322    changeDisplayScreens();
323  }
324
325  void feedTheFish(struct tm timeInfo, bool manualCall) {
326    if (serverFeedingTrigger || manualCall ||
327        timeInfo.tm_hour - lastFeedingTime.tm_hour >= 12 ||
328        ((timeInfo.tm_hour < lastFeedingTime.tm_hour) &&
329          (timeInfo.tm_hour + 24 - lastFeedingTime.tm_hour >= 12))) {
330      display.clearDisplay();
331      display.setCursor(0, 0);
332      display.print("Feeding the fish...");
333      display.display();
334      myStepper.step(1024);
335      delay(3000);
336      lastFeedingTime = timeInfo;
337      serverFeedingTrigger = false;
338      changeDisplayScreens();
```

```
void feedTheFish(struct tm timeInfo, bool manualCall) {
  if (serverFeedingTrigger || manualCall ||
      timeInfo.tm_hour - lastFeedingTime.tm_hour >= 12 ||
      ((timeInfo.tm_hour < lastFeedingTime.tm_hour) &&
        (timeInfo.tm_hour + 24 - lastFeedingTime.tm_hour >= 12))) {
    display.clearDisplay();
    display.setCursor(0, 0);
    display.print("Feeding the fish...");
    display.display();
    myStepper.step(1024);
    delay(3000);
    lastFeedingTime = timeInfo;
    serverFeedingTrigger = false;
    changeDisplayScreens();
  }
}

String processor(const String& var) {  //Funkcija koja mijenja HTML-ov placeholder
  if (var == "BUTTONPLACEHOLDER") {
    String buttons = "";
    buttons += "<h4>Heating controller</h4><label class=\"switch\"><input type=\"checkbox\" onchange=\"toggleCheckbox(this)\" id=\"1\" "
     + outputState(1) + "><span class=\"slider\"></span></label>";
    buttons += "<h4>Pump controller</h4><label class=\"switch\"><input type=\"checkbox\" onchange=\"toggleCheckbox(this)\" id=\"2\" "
     + outputState(2) + "><span class=\"slider\"></span></label>";
    buttons += "<h4>Lights controller</h4><label class=\"switch\"><input type=\"checkbox\" onchange=\"toggleCheckbox(this)\" id=\"3\" "
     + outputState(3) + "><span class=\"slider\"></span></label>";
    buttons += "<h4>Last time fish was fed:<br />"
     + outputState(4) + "</h4><button class=\"button\" onclick=\"feedFishRequest();\">Feed the fish?</button>";
    return buttons;
  }
  return String();
}

String outputState(int output) {  //Funkcija za ispis trenutnih vrijednosti komponenti na web server-u
  switch (output) {
    case 1:
      if (heatingOn) {
        return "checked";
      } else {
        return "";
      }
    case 2:
      if (pumpOn) {
        return "checked";
      } else {
        return "";
      }
    case 3:
      if (lightsOn) {
        return "checked";
      } else {
        return "";
      }
    case 4:
      char timeStringBuff[50];  //50 chars should be enough
      strftime(timeStringBuff, sizeof(timeStringBuff), "%B %d %H:%M", &lastFeedingTime);
      return timeStringBuff;
```