OŠ Brinje Grosuplje
Ljubljanska cesta 40a
1290 Grosuplje

Žiga Remic

PAMETNI KONTROLNIK AKVARIJA

raziskovalna naloga

področje: elektrotehnika,
elektronika in robotika

Mentor: Matej Kastelic prof. fizike in tehnike
Somentor: Petra Maršić dipl. inž. rač. in inf.

Grosuplje, marec 2021

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Zahvala

Rad bi se zahvalil Zavodu 404 za material, prostor in orodje za izdelavo kontrolnika za akvarij.

Posebna zahvala gre somentorici Petri Maršić za vso pomoč, usmeritve, spodbude in potrpežljivost.

Mentorju Mateju Kastelicu se zahvaljujem za njegov čas in koordinacijo pri izdelavi raziskovalne naloge.

Hvala!

Kazalo

Kazalo slik

Kazalo tabel

Povzetek

Naslov: PAMETNI KONTROLNIK AKVARIJA

Avtor: Žiga Remic

Namen raziskovalne naloge je izdelava pametnega kontrolnika za sladkovodni akvarij. Kontrolnik, ki upravlja z akvarijem, je krmiljen s pomočjo daljinskega nadzora preko mobilne ali spletne aplikacije. V raziskovalni nalogi je najprej raziskan trg sladkovodnih akvarijskih kontrolnikov. Glede na ugotovitve so zastavljeni cilji za izdelavo kontrolnika z glavnimi funkcijami komercialnih cenovno dostopnejših kontrolnikov. Kontrolnik upravlja s filtrom, grelcem in lučjo v akvariju ter meri temperaturo vode v akvariju in temperaturo in vlažnost zraka v prostoru. Narejen je s pomočjo mikrokrmilnika ESP-WROOM-32. Program zanj je napisan v jeziku C++, prilagojenemu za Arduino programsko okolje. Uporabljena je knjižnica Blynk in pripadajoča aplikacija za Android telefone. Zasnovano in izdelano je bilo tiskano vezje na katerega so nameščeni mikrokrmilnik, elektronski senzorji in aktuatorji. V ogrodju Flask je izdelana tudi spletna stran za upravljanje s kontrolnikom. Testirana je bila zanesljivost delovanja kontrolnika.

Ključne besede: ESP32, Arduino IDE, akvarijski kontrolnik, tiskana vezja, IoT

1 Uvod

Akvariji so zapleten sistem živih bitjih, v katere je potrebno vložiti veliko truda, časa in tudi nekaj finančnih sredstev. Poskrbeti je treba, da vse naprave, ki so v ozadju akvarija, pravilno in nemoteno delujejo. S težavami vzdrževanja akvarija sem se pri mojem sladkovodnem akvariju srečal tudi sam. Te so še večje ob morebitni odsotnosti od doma. Različna podjetja so zato razvila tako imenovane akvarijske kontrolnike, ki upravljajo in nadzirajo akvarijske naprave ter omogočajo različne meritve.

Akvarijski kontrolniki pomagajo lastniku akvarija, kar se da pravilno upravljati s akvarijem in v njem vzdrževati zdrav ekosistem za vse akvarijske organizme. Problem s komercialnimi kontrolniki je, da praviloma stanejo več kot je povprečen lastnik akvarija pripravljen odšteti za to kar nudijo. Poleg tega večino obstoječih kontrolnikov v Sloveniji trenutno ni možno kupiti oziroma so le redki dobavljivi iz tujine.

Cilj raziskovalne naloge je izdelati akvarijski kontrolnik, ki upravlja in nadzira filter, luč in grelec v akvariju. Izvajati mora tudi nekaj osnovnih meritev in sicer: merjenje temperature zraka in vode ter merjenje relativne vlažnosti zraka.

V raziskovalni nalogi so najprej raziskane osnovne potrebe sladkovodnega akvarija. Nato je pregledan trenutni trg komercialnih akvarijskih kontrolnikov in njihovih funkcij. Glede na potrebe akvarija in funkcij komercialnih kontrolnikov je podana osnovna ideja, kakšen mora akvarijski kontrolnik biti. Na podlagi te sem se nato lotil izdelave svojega kontrolnika.

2 Teoretični del

2.1 O akvarijih

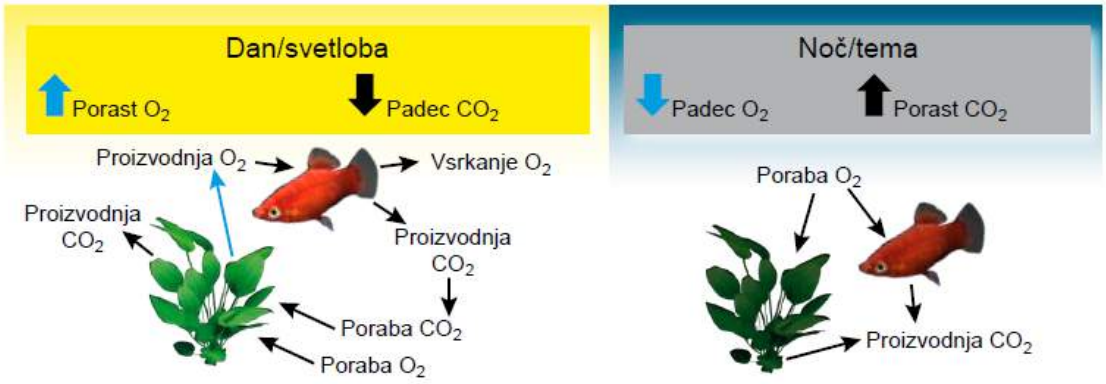
Akvariji (slika 2.1) so več kot le posoda vode in nekaj rib. Z njimi si lahko del narave prinesemo v naš dom in če je akvarij pravilno sestavljen, se njegovi prebivalci v njem lahko počutijo kot v naravi. Za vzpostavitev ustreznega okolja v akvariju moramo upoštevati kar nekaj stvari [17].

Slika 2.1: Sladkovodni akvarij [28]

Raziskovalna naloga se osredotoča na sladkovodne akvarije.

2.1.1 Osvetlitev akvarija

Rastline in organizmi v akvariju potrebujejo svetlobo. S pomočjo nje se dogaja eden izmed najpomembnejših procesov v akvariju – fotosinteza. S fotosintezo rastline v akvariju zmanjšujejo vsebnost ogljikovega dioksida (CO_2) in povečujejo raven kisika (O_2), ki ga uporabijo drugi organizmi v akvariju, na primer ribe in polži. Osvetlitev akvarija je pomembna tudi za ostale organizme, saj so, ko je prižgana osvetlitev bolj aktivni, ob ugasnjenih lučeh pa so bolj mirni.



Slika 2.2: Kroženje plinov v akvariju [17]

Poznamo več vrst luči za akvarije:

- fluorescentne,
- LED in
- halogenske.

Pri upravljanju luči v akvariju je pomembno, da so vsak dan prižgane enako časa in približno ob enaki uri. Koliko časa naj bi bile luči prižgane, je odvisno od vrste rastlin v akvariju, če se v akvarij dodaja ogljikov dioksid (CO_2) in svetilnosti luči. Na splošno se priporoča, da luči niso nikoli prižgane več kot 8 ur dnevno. Če so luči prižgane preveč časa oziroma so premočne, lahko to močno poveča rast alg, ki v akvarijih niso zaželene [2].

2.1.2 Filtriranje akvarija

„Zmogljiv filtrirni sistem je srce akvarija“ [10].

Akvarijski filtri delujejo s pomočjo vodne črpalke, ki poganja vodo skozi različne materiale nameščene v manjših posodah znotraj filtra.

Filter ima v akvariju tri glavne namene:

1. Pri razgradnji ribjih iztrebkov in odvečne hrane v akvariju nastane amonijak (NH_3), ki je strupen in lahko škoduje vsem organizmom v akvariju. Znotraj filtra so naseljene bakterije, ki amonijak spremenijo v nitrit in nato v nitrat, ki ga lahko uporabijo rastline pri svoji rasti.

2. Filter vodo poganja po akvariju, zato da lahko vse rastline dostopajo do enake količine mineralov.

3. Filter s svojim tokom premika odpadke, ki se lahko naberejo na dnu akvarija in jih posrka, da se jih lažje očisti iz akvarija.

Filter naj bi deloval brez prestanka oziroma brez večjih premorov [13].

2.1.3 Gretje akvarija

Grelec v akvariju vzdržuje temperaturo. Večina grelcev, ki so trenutno dostopni na trgu ima že vgrajen termostat, to pomeni, da sami prižigajo in ugašajo grelni element, da se doseže želena temperatura. Ta je odvisna od tega, kakšni organizmi živijo v akvariju. Tropskim ribam najbolj ugaja temperatura vode 24-26 °C, medtem ko nekatere hladnovodne ribe lahko živijo že na 15 °C [22, 33, 14].

2.1.4 Hranjenje organizmov v akvariju

Medtem ko vsa tehnika v akvariju skrbi za primerno okolje za organizme, morajo ti od nekje dobiti tudi hrano. Večina rib potrebuje 16-24 ur, da v celoti prebavi hrano, zato je ribe priporočljivo hraniti enkrat do dvakrat dnevno. Količina hrane naj bi bila takšna, da jo ribe pojejo v manj kot petih minutah. Če smo v dilemi kolikšna količina hrane je zadostna, se vedno odločimo za manj hrane. Preostanek hrane, ki je ribe ne pojedo, onesnažuje akvarij in poveča raven strupenega amoniaka (NH_3), zato se je treba temu izogibati [34].

2.2 Pregled akvarijskih kontrolnikov na trgu

Na spletu sem na različnih spletnih straneh poiskal nekaj komercialnih kontrolnikov za akvarije, ki so zmožni upravljati luč, filter in grelec akvarija. Vse kontrolniki so iz tujine, saj v Sloveniji nisem našel nobene trgovine, ki bi jih prodajala.

2.2.1 GHL Profilux 4e

GHL Profilux 4e je eden izmed najbolj popularnih kontrolnikov za akvarije. Zasnovan je na modularen način. To pomeni, da je sistem sestavljen iz centralnega krmilnika in dodatnih modulov, na katerih so senzorji, električne vtičnice, itd.. Sposoben je upravljati z veliko napravami in beležiti mnogo meritev. Tako lahko:

- nadzira do 32 luči,

- meri temperaturo, trdoto in pH vode, oksidacijsko redukcijski potencial, prevodnost vode, količino raztopljenega kisika (O_2) v vodi, vlažnost in temperaturo zraka v prostoru, nivo vode v akvariju,

- zazna puščanje akvarija,
- nadzira do 16 različnih črpalk ali filtrov, s čimer je možna simulacija nevihte, plime, itd. in
- nanj je možno dodati razširitvene module in še dodatno povečati število naprav, ki jih je možno nadzirati.

Slika 2.3: GHL Profilux 4e [12]

Za vse naprave, ki jih je mogoče nadzirati, je možno nastaviti, kdaj se prižgejo in ugasnejo ter ustvariti urnike. Profilux 4e ima vgrajeno brezžično povezavo WiFi, preko katere se poveže v internetno omrežje. Meritve vseh senzorjev je možno spremljati preko aplikacije »GHL Connect« za pametni telefon in računalnik, kjer je mogoče tudi upravljati z napravami in drugimi nastavitvami. Profilux 4e ima na ohišju tudi LCD zaslon in nekaj gumbov, s katerimi se tudi lahko upravlja določene nastavitve in preveri meritve. Vgrajen ima tudi majhen zvočnik za zvočne alarme in uporabnika lahko obvešča preko e-mail-a in SMS sporočila [12].

GHL Profilux 4e stane približno 450 €, vendar ne vključuje senzorjev in modulov z vtičnicami, zato je treba te dokupiti posebej, kar pomeni, da je končna cena precej višja. Tako imenovan Start set, ki vsebuje Profilux 4e, modul s 6 vtičnicami ter senzorjema za temperaturo in pH stane približno 750 €. Kar pomembno se mi zdi omeniti tudi, da pH senzor dovolj natančno deluje le 6 mesecev. Po tem času proizvajalec GHL priporoča nakup novega senzorja, ki stane 75 €.

2.2.2 HYDROS Control 2

HYDROS Control 2 je nekoliko bolj preprost kontrolnik, ki je prav tako narejen po modularnem dizajnu. Na njegovem ohišju je nameščenih 6 različnih priključkov, in sicer:

- 2 priključka na katera lahko priklopimo različne senzorje;
- 2 izhodna priključka, na katere lahko priklopimo naprave, ki delujejo na 12 V_{DC}, kot so majhne črpalke, luči itd.. HYDROS Control 2 je zmožen postopno prižigati in ugašati te naprave in meriti njihovo porabo toka in
- 2 nadzorna priključka. Preko enega izmed njih je HYDROS Control 2 priključen na napajalnik, drugega pa je možno uporabiti za povezavo z drugimi HYDROS napravami.

Poleg fizičnih priključkov, se lahko HYDROS Control 2 preko brezžične povezave WiFi poveže z do 8 drugimi napravami, na primer moduli z vtičnicami, na katere se lahko priključi naprave.

Slika 2.4: HYDROS Control 2 [5]

Podobno kot pri GHL kontrolniku je možno vse nastavitve spreminjati in preverjati meritve preko aplikacije za pametne telefone. Kontrolnik v primeru, da meritev iz nekega senzorja preseže nastavljeno mejo, pošlje uporabniku e-mail. HYDROS Control 2 je certificiran pod oznako IP65, kar pomeni, da je do določene mere vodoodporen [5].

Cena kontrolnika HYDROS Control 2 je približno 160 $ (130 €). Tako imenovani Starter set, ki vključuje kontrolnik, napajalnik, temperaturni senzor in modul s 4 vtičnicami ter 4 USB priključki za napajanje naprav stane približno 200 $ (165 €).

2.2.3 Felix Smart

Felix Smart je najnovejši kontrolnik, ki je predstavljen v raziskovalni nalogi. Zamišljen je na nekoliko bolj preprost način kot drugi kontrolniki. V primerjavi z večino drugih, ki so narejeni na modularen način, ima Felix Smart kontrolnik večino komponent že vgrajenih v ohišje. Vsebuje 8 električnih vtičnic in 6 USB priklopov. Na njih se lahko priklopi 360 ° podvodno kamero in 1 sam senzorski modul, ki ima vgrajenih več senzorjev in meri: temperaturo vode, osvetljenost (v enoti LUX), vsebnost amoniaka (NH_3), količino uporabne svetlobe za fotosintezo, pH vode, barvno temperaturo svetlobe (v enoti kelvin) ter nivo vode. Kontrolnik Felix Smart se v omrežje poveže preko povezave WiFi, upravljanje z napravami in branje meritev je na voljo v aplikaciji za pametne telefone. V tej se lahko nastavi tudi alarme, scenarije za prižiganje in ugašanje naprav itd. Zanimivost tega kontrolnika v primerjavi z drugimi je, da se ga lahko upravlja tudi preko glasovnih ukazov pametnih pomočnikov Siri, Google Home Assistant in Amazon Alexa [9].

Slika 2.5: Felix Smart kontrolnik s senzorjem in kamero [9]

Kontrolnik Felix Smart stane približno 282 €. Lahko se ga kupi tudi v kompletu

s senzorjem (cena je približno 485 €), kompletu s kamero (cena je približno 404 €) ali v kompletu s kamero in senzorjem (cena je približno 647 €). Poleg tega je potrebno senzorju enkrat mesečno zamenjati merilni list (water monitoring slide), ki je naprodaj v kompletu treh za približno 30 €, kar pomeni dodaten strošek približno 10€ na mesec.

2.3 Osnovna ideja kontrolnika

Glede na to kakšni so komercialni kontrolniki in potrebe akvarijev sem dobil približno idejo kakšne funkcije naj ima moj kontrolnik. Mora biti sposoben:

1. Upravljati in nadzirati najmanj naslednje naprave:
    - filter,
    - grelec in
    - luč v akvariju.
2. Izvajati najmanj naslednje meritve:
    - temperaturo vode
    - temperaturo zraka v prostoru in
    - relativno vlažnost zraka v prostoru

Kontrolnik naj naprave upravlja tako, kot je najbolje za zdravje akvarija in njegovih prebivalcev:

- luč naj bo prižgana toliko kot je potrebno,
- grelec naj ne pregreva akvarija in
- filter naj deluje vedno, razen kadar ga želi uporabnik izklopiti zaradi hranjenja rib ali katerega drugega razloga (npr. čiščenje akvarija).

Ves nadzor nad zgoraj navedenimi napravami in pregled meritev naj bo mogoče brezžično izvajati preko pametnega telefona ali računalnika.

Razmišljal sem tudi, da bi kontrolniku dodal sistem za hranjenje rib, vendar sem ugotovil, da je samodejne hranilnike možno kupiti za tako nizko ceno, da se to ne splača.

2.4 Raziskovalna vprašanja in hipoteze

Z osnovno idejo, kakšen naj bo kontrolnik, sem si postavil nekaj hipotez:

HIPOTEZA 1: Možno je izdelati kontrolnik, ki bo nadzoroval filter, grelec ter luč v akvariju in stane manj kot komercialni kontrolniki.

HIPOTEZA 2: Vse funkcije kontrolnika (nadzor naprav in ogled meritev) bo uporabnik lahko upravljal in nadziral na daljavo preko pametnega telefona.

HIPOTEZA 3: Kontrolnik bo akvarij obvaroval pred pregrevanjem v primeru okvare grelca.

HIPOTEZA 4: V primeru napake pri branju oziroma nedelovanju senzorjev bo kontrolnik napako sporočil uporabniku.

3 Eksperimetalni del

3.1 Metode dela

V mojem raziskovanju sem prvo raziskal potrebe akvarija, kakšni kontrolniki so že na voljo na trgu in kako zadostijo tem potrebam. Nato sem si zastavil osnovno idejo in cilje, kakšen mora biti kontrolnik ter razmišljal o različnih načinih doseganja teh ciljih. Tako sem prišel do prvege različice kontrolnika, od tam pa sem ga razvijal naprej do končne različice.

Čez celoten postopek zasnove in izdelave kontrolnika me je vodila literatura, ki mi je prinesla veliko odgovorov na vprašanja in rešitve na razne probleme, ki pa jih kljub temu ni bilo malo.

3.2 Materiali

Po določitvi ciljev za izdelavo kontrolnika je čas za izbiro komponent. Za izdelavo kontrolnika sem uporabil naslednje materiale :

- mikrokrmilno ploščico DOIT Esp32 DevKit,
- rele modul,
- temperaturni senzor DS18B20,
- digitalni senzor DHT22 za merjenje temperature in relativne vlažnosti zraka,
- modul realnega časa DS3231,
- tiskano vezje,
- nadometna doza,
- vtičnice, nosilec in okvir,
- pleksi steklo 4 mm ,
- priključni kabel,

- tesnilno uvodnico,
- vijake,
- nosilec za varovalko in cevna varovalko,
- vtič za napajanje in
- žice.

3.2.1 DOIT Eap32 DevKit

DOIT Esp32 DevKit (slika 3.1) je mikrokrmilna ploščica, ki jo poganja mikrokrmilnik ESP32 WROOM. Zanimivo je to, da ima kljub nizki ceni (3 €) podporo za povezavi Wifi in Bluetooth, kar bo uporabno za nadziranje kontrolnika na daljavo. DOIT Esp32 DevKit ima skupaj 32 pinov (priključkov), od katerih je:

- 25 digitalnih vhodno-izhodnih (I/O) pinov z oznako DIO,
- 6 vhodov z analogno digitalnim pretvornikom,
- 2 izhodna pina z analogno digitalnim pretvornikom,
- 3 povezave UART,
- 2 povezave SPI in
- 3 povezave I2C.

Slika 3.1: Mikrokrmilna ploščica DOIT Esp32 DevKit v1 [7]

Procesor mikrokrmilnika je Tensilica 32-bit dve-jedrni Xtensa LX6, ki doseže frekvenco do 240 Mhz. V Esp32 DOIT DevKti je vgrajen tudi regulator napetosti, ki sprejme napetost 5-12 V_{DC} in jo zniža na 3,3 V_{DC}, na kateri deluje mikrokrmilnik. To mikrokrmilniško ploščico se lahko programira v programskih jezikih Arduino C++ in MicroPython [37]. V raziskovalni nalogi je uporabljen programski jezik Arduino C++.

3.2.2 Rele modul

Releji so električna stikala. Poznamo dve vrsti relejev: elektromehanske in polprevodniške releje. Elektromehanski releji delujejo s pomočjo elektromagneta, ki v njih povzroči, da se kontakti v releju stikajo ali pa so ločeni med sabo. Polprevodniški releji v sebi ne vsebujejo premikajočih mehanskih delov, temveč imajo vsebi polprevodnik, ki postane prevoden pri določeni napetosti [11].

Slika 3.2: Polprevodni štirikanalni rele modul [35]

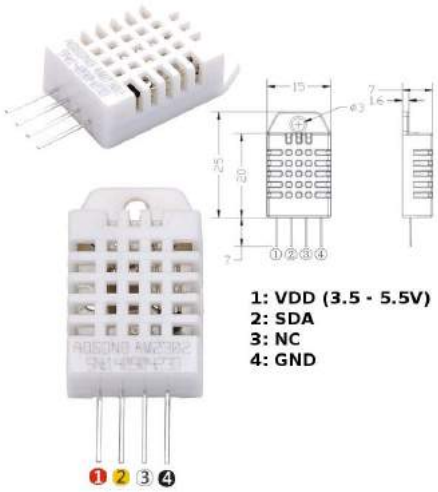
3.2.3 Temperaturni senzor DS18B20

Temperaturni senzor DS18B20 deluje preko povezave 1-Wire, ki omogoča komunikacijo med senzorjem in mikrokrmilnikom preko ene same žice. DS18B20 ima natančnost +- 0,5 °C (v območju -10-85 °C). Temperaturo bere v 9 do 12 bitni resoluciji. Za pravilno delovanje mora biti njegov priključek za prenos signala DQ preko dvigovalnega upora 4.7 kΩ priključen na priključek VCC. DS18B20 deluje na napetosti 3-5,5 V. Senzor lahko deluje tudi v parazitnem načinu, v katerem je njegov VCC priključek povezan na ozemljitev GND namesto na 3-5,5 V_{DC}, napajanje pa dobi preko DQ priključka. DS18B20 je naprodaj v ohišju tehnologije TO-92 in vodotesni različici [20].

Slika 3.3: Temperaturni senzor DS18B20 [31]

3.2.4 Senzor DHT22

DHT22 je digitalni senzor za vlago in temperaturo zraka. Temperaturo meri v 8-bitni resoluciji z natančnostjo +-0,5 °C (v -40-80 °C) in relativno vlago z natančnostjo +-2 % RH. DHT22 deluje na napetosti od 3,3 do 6 V_{DC} [1].

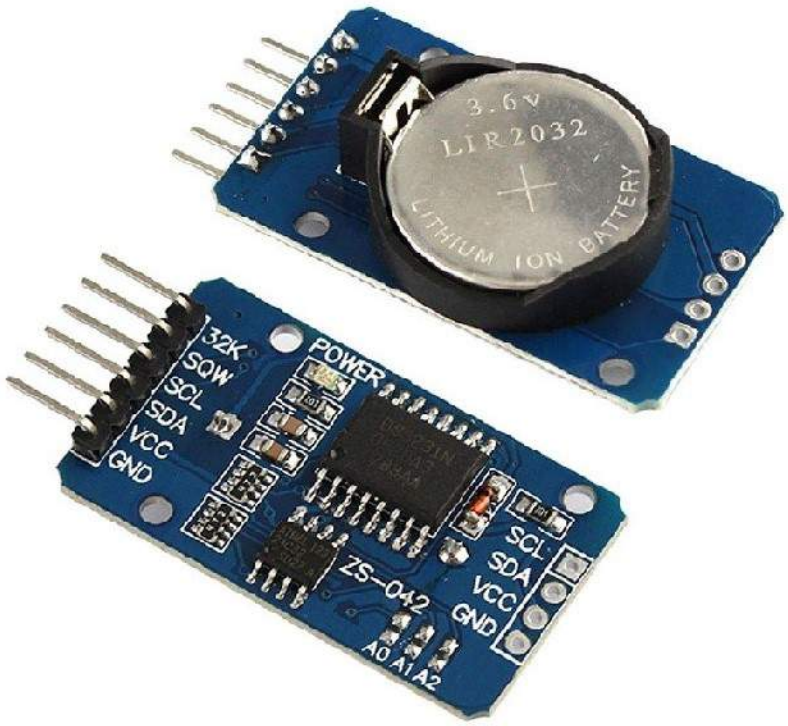


Slika 3.4: Senzor DHT22 [6]

3.2.5 Modul realnega časa DS3231

Modul realnega časa DS3231 služi kot ura. Nastavimo ga na določen datum in čas, ki si ju zapomni. Modul prepozna različne dolžine mesecev in leta. Poganja ga čip DS3231, kar mu omogoča natančnost +- 2 minuti na leto. DS3231 deluje na

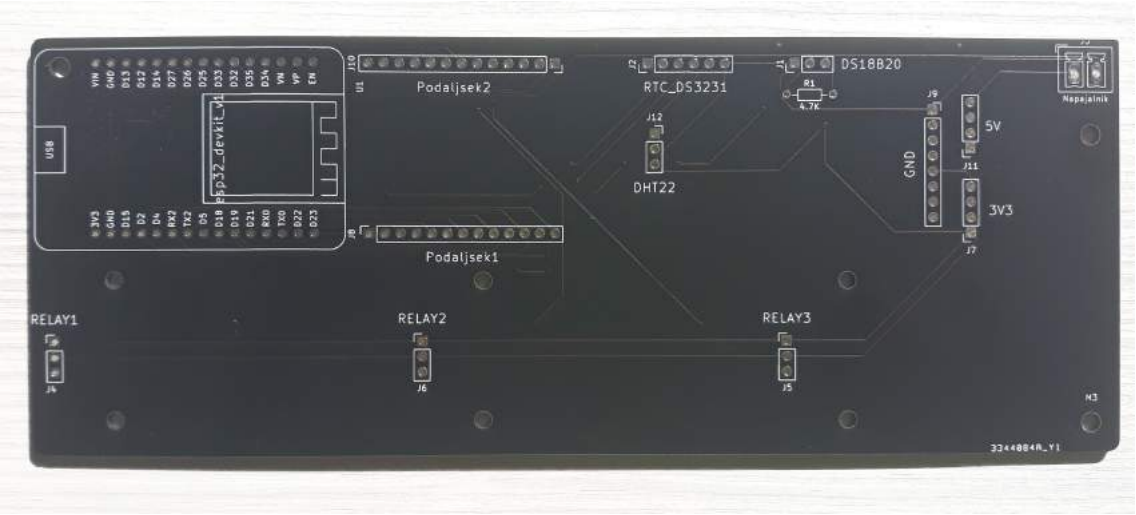
napetosti 3,3-5 V_{DC}. Komunicira lahko preko povezave I2C. Ima vgrajen nosilec za baterijo CR2032, s pomočjo katere ne izgubi časovnih podatkov v primeru, da odpove glavno napajanje [18].



Slika 3.5: Modul realnega časa DS32321 [8]

3.2.6 Tiskano vezje

Za povezavo in ogrodje med nekaterimi komponentami je bilo uporabljeno tiskano vezje, zasnovano posebej za akvarijski kontrolnik. Izdelava tiskanega vezja je opisana v poglavju 3.3.2 Izdelava tiskanega vezja.



Slika 3.6: Tiskano vezje

3.2.7 Napajalnik

Za napajanje sem uporabil AC/DC pretvornik LS25-5 proizvajalca TDK-Lambda, ki 220 V_{AC} pretvori na 5 V_{DC}. Ima učinkovitost 77 % in pretovori lahko do 5 A toka, kar je več kot dovolj za vse druge komponente [30].

Slika 3.7: TDK-Lambda AC/DC pretvornik LS25-5 [29]

3.2.8 Material za izdelavo ohišja kontrolnika

3.2.8.1 Nadometna doza

Za izdelavo ohišja sem se odločal med več vrstami materialov. Prva zamisel je bila, da bi bilo ohišje 3D natisnjeno. Ugotovil sem da se to skoraj ne splača, saj bi 3D tiskalnik za tiskanje dovolj velikega ohišja porabil veliko filamenta in časa. Nato sem začel razmišljati, da bi ohišje naredil iz lesa, vendar tudi to ni najboljša rešitev, ker je kontrolnik blizu akvarija in obstaja možnost, da pride v stik z vodo. To bi les vpil in ohišje bi verjetno razpadlo, poleg tega pa bi lahko prišlo do kratkih stikov. Tako sem prišel do ideje, da bi uporabil že standardno plastično ohišje. Na spletu sem našel nadometno dozo, ki je ustrezala potrebam kontrolnika (slika 3.8).

Slika 3.8: Nadometna doza ELETTROCANALI EC400C8 dimenzije 300x220x120mm [23]

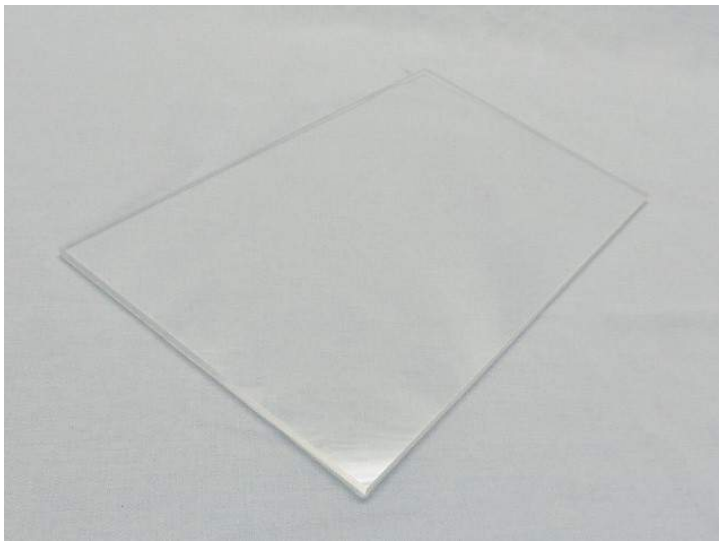
3.2.8.2 Vtičnice, nosilec in okvir

Za priklop naprav na kontrolnik sem izbral vtičnice, nosilec in okvir namenjene za montažo na steno (slika 3.9).

Slika 3.9: Vtičnica SCHUKO+KS 2P+E 16A 250V 2M PW, vtičnica EURO+KS 2P 10A 250V 1M PW, nosilec z vijaki 7M in okvir LINE 7M PW[36]

3.2.8.3 Pleksi steklo

V notranjosti nadometne doze sem za pritrditev naprav uporabil 4mm debelo pleksi steklo (slika 3.10).



Slika 3.10: Pleksi steklo 4mm [26]

3.2.8.4 Priključni kabel

Za povezavo kontrolnika v električno omrežje je potreben priključni kabel. Za kontrolnik je uporabljen 3 žilni(nula – modra, faza – rdeča, zemlja – zeleno-rumena), 1.5 mm2 debel kabel.

Slika 3.11: Priključni kabel [27]

3.2.8.5 Tesnilna uvodnica

Da bi priključni kabel trdno stal v ohišju in da je luknja skozi katero pride dobro zatesnjena, sem se odločil, da ga v ohišje vpeljem s pomočjo tesnilne uvodnice.

Slika 3.12: Tesnilna uvodnica [32]

3.2.8.6 Vijaki

Za pritrditev komponent na ohišje sem uporabil vijake z navojem M3 in M4.

Slika 3.13: M3 vijaki [19]

3.2.9 Ostali elektromaterial

3.2.9.1 Nosilec za varovalko in cevna varovalka

Zaradi varnostnih razlogov sem v kontrolnik želel vgraditi tudi varovalko. Odločil sem se za mini cevno varovalko (2 A). Ta je bila najbolj primerna ravno zaradi svoje velikosti.

Slika 3.14: Nosilec za mini cevno varovalko [25]

3.2.9.2 Vtič za napajanje

Želel sem, da je senzor za temperaturo vode možno odklopiti iz kontrolnika na preprost način in ga nato tudi priklopiti nazaj, ob tem pa ni potrebno razmišljati, katero žico se kam priklopi. Zato sem se odločil, da bom za povezavo senzorja za temperaturo vode uporabil kar preprost vtič, ki je običajno namenjen za napajanje, vendar ustreza potrebi.

Slika 3.15: Napajalni vtič [24]

3.2.9.3 Žice

Seveda morajo biti komponente med sabo povezane z žicami. Za komponente, ki delujejo na 5 VDC in 3,3 VDC sem uporabil žice DUPONT (slika 3.16). Za komponente, ki so povezane na 220 VAC, pa sem uporabil 1 mm2 debele žice.

Slika 3.16: Jumper žice [15]

3.2.10 Pripomočki

Pri izdelavi kontrolnika sem uporabil naslednje pripomočke:

- računalnik s programi Arduino IDE, Microsoft Visual Studio Code in Onshape,
- laserski rezalnik,
- izvijače,
- ročni vrtalnik in svedre,
- ščipalne klešče in klešče za kabelske kontakte,
- spajkalnik in
- multimeter.

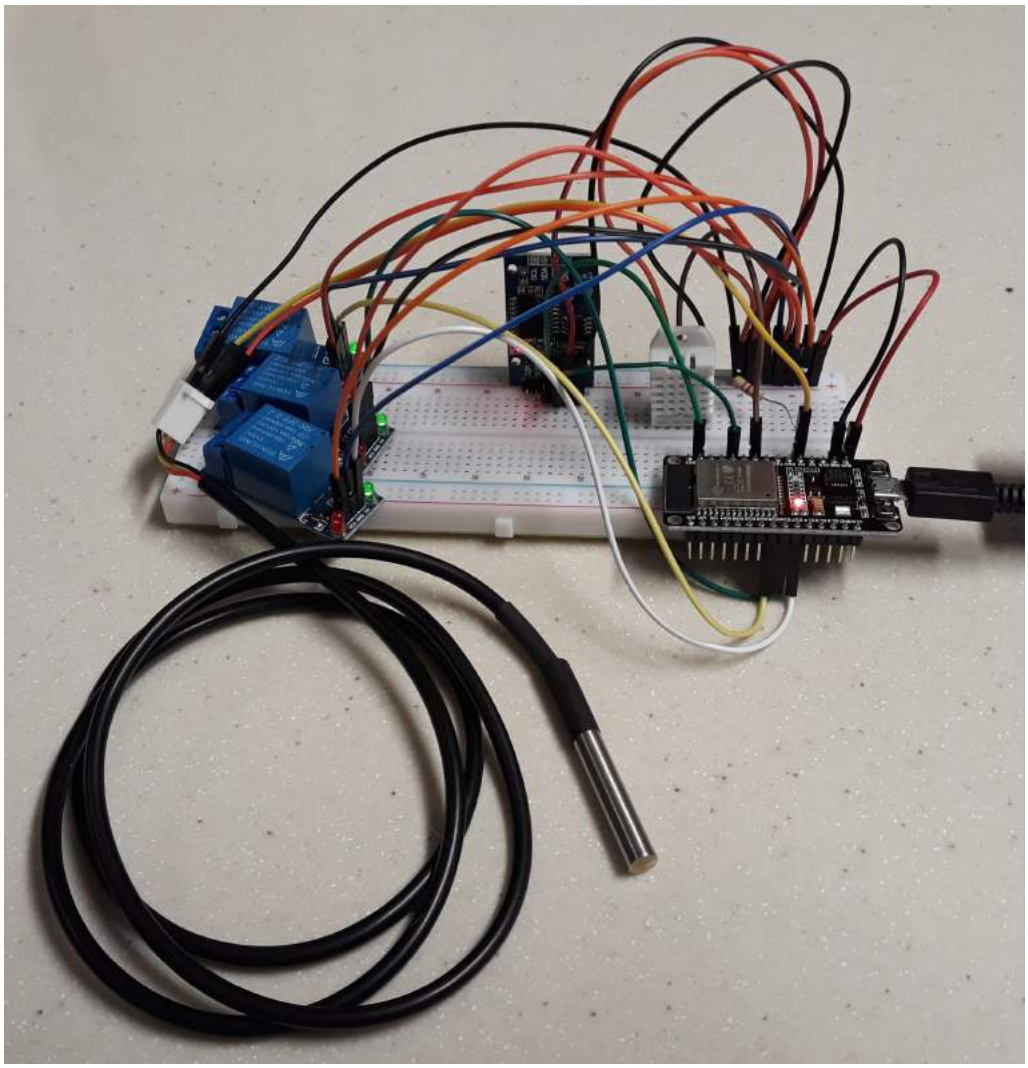
Glavne značilnosti programskega okolja Arduino IDE so opisane v poglavju 3.4.1 Arduino IDE.

3.3 Izdelava kontrolnika

3.3.1 Začetki izdelave kontrolnika

Izbrane komponente za kontrolnik sem skupaj s pomočjo jumper žic in prototipne ploščice povezal, kot je vidno na sliki 3.17

Mikrokrmilna ploščica ESP32 je povezana na USB napajalnik (polnilec za telefon ali računalnik). Druge komponente pa so nato preko prototipne ploščice povezane na mikrokrmilno ploščico ESP32. Uporabil sem 3 ločene rele module, vsak je bil priključen na 3,3 V (priključek VCC na releju), na zemljo (priključek GND na releju) in na signalni priključek IN. Ta je bil priključen na enega izmed pinov 25, 26 ali 27 na mikrokrmilni ploščici, preko katerih so releji krmiljeni. Temperaturni senzor DS18B20 sem priključil na 3,3 V (VDD), zemljo (GND) in še na pin 4 za komunikacijo (DQ na senzorju). Senzor DHT22 za temperaturo zraka ter relativno vlažnost sem povezal na 3,3 V (VCC), zemljo (GND) in na pin 18 (Data). Modul realnega časa sem povezal s uporabo povezave I2C. Priključek VCC sem povezal na 3,3 V, GND na zemljo, SCL na pin 22 in SDA na pin 21. Ko sem imel vse to povezano med sabo, so se povezave zaradi velikega števila nepritrjenih žic velikokrat prekinile, zato se je pogosto dogajalo, da se je katera izklopila. Izklopljene žice je bilo zapleteno povezati nazaj na pravo mesto. Prav tako ni bilo možno fiksno pritrditi relejev. Zaradi teh slabosti sem se odločil izdelati tiskano vezje za kontrolnik. Tiskano vezje

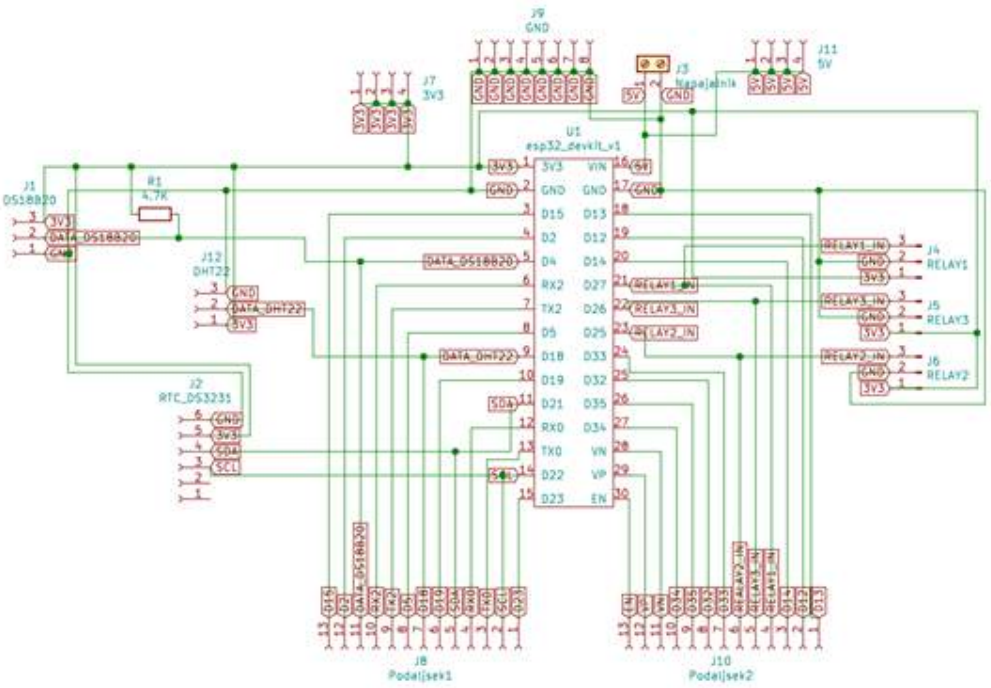


Slika 3.17: Povezava komponent za kontrolnik na prototipni ploščici

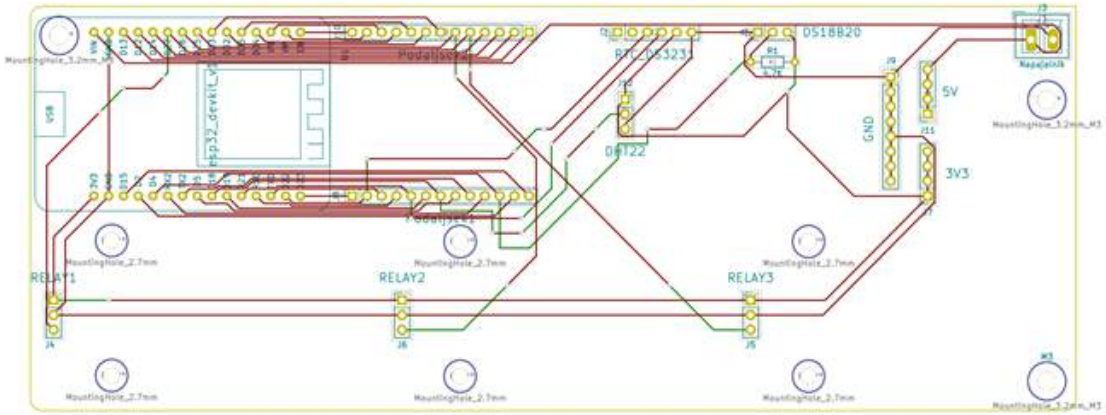
je stabilnejše, saj se z njim odpravi morebitne zaplete zaradi izklopa žic. Na njem sem predvidel tudi odprtine za pritrditev relejev z vezicami.

3.3.2 Izdelava tiskanega vezja

Vezje in logično shemo sem načrtoval v programu KiCad. Najprej sem narisal shematiko, ki vključuje vse povezave navedene v poglavju 3.3.1 z dodanim priključkom za napajanje (5 V_{DC}), razširitvami za napajanje (4-krat +5 V_{DC} in +3.3 V_{DC}, 8-krat GND) in podaljški za lažji dostop do pinov. Na podlagi shematike sem nato narisal tiskano vezje.



Slika 3.18: Shematika tiskanega vezja (Priloga 1: Shematika tiskanega vezja)

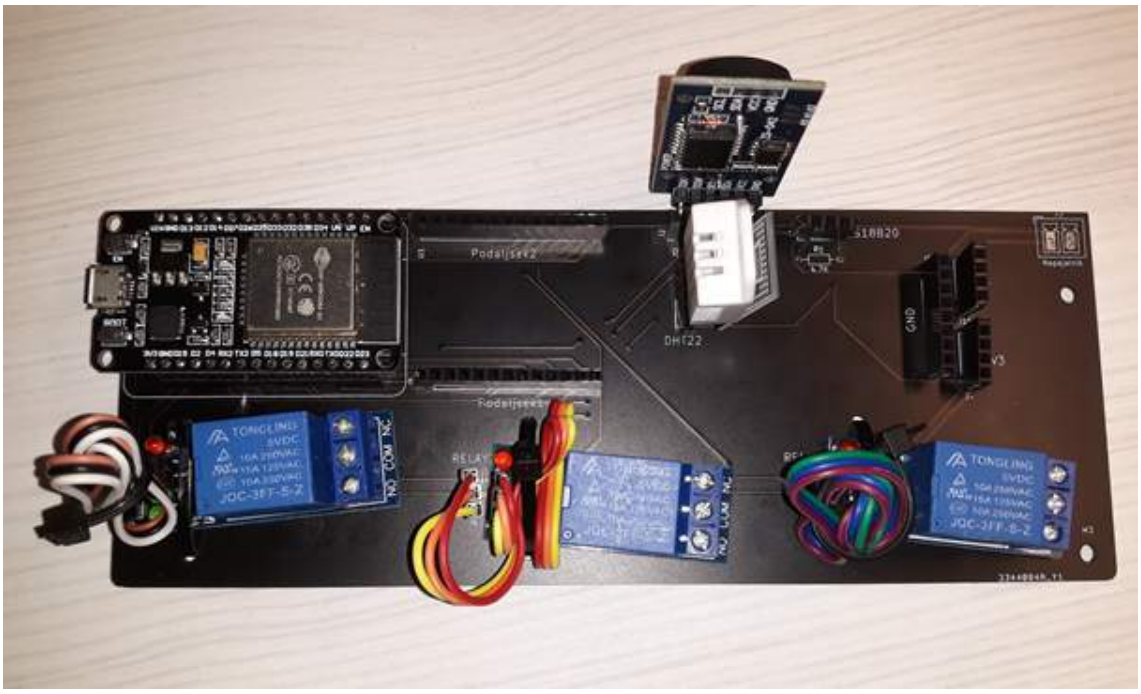


Slika 3.19: Načrt tiskanega vezja (Priloga 2: Načrt tiskanega vezja)

Na sliki 3.21 je vidno vezje na katerega so prispajkane vse komponente. Na sliki ni senzorja DS18B20, ki se na vezje poveže preko povezovalnih žic in priklopa za zunanje napajanje, ker je bil za napajanje sprva uporabljen USB kabel.

Slika 3.20: Rele modul, uporabljen za prvo različico kontrolnika [21]

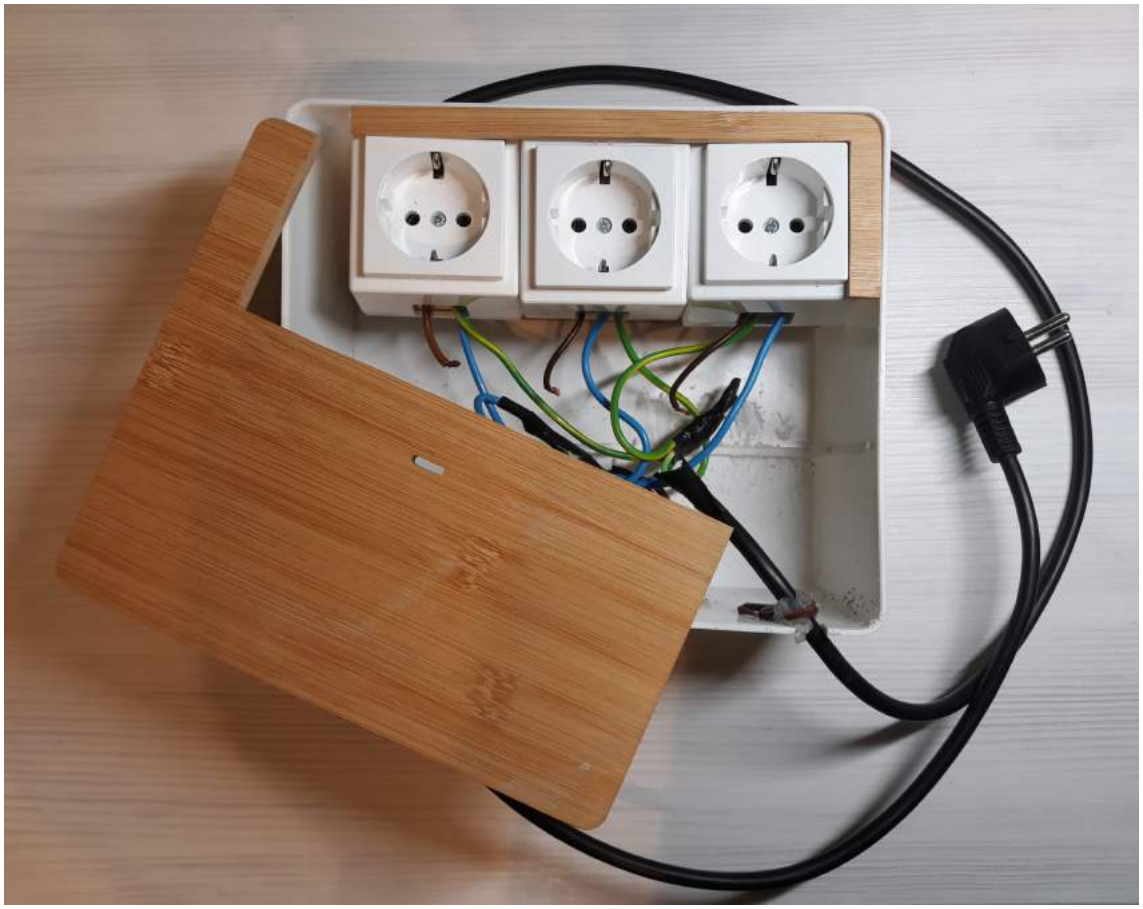
Vezje sprva ni delovalo. Pri risanju vezja sem storil napako. Enega izmed prehodov med dvema slojema tiskanega vezja sem postavil preblizu druge poti na vezju, zato sta bila pina D2 in D21 (uporabljen za SDA pri komunikaciji I2C) povezana. To je povzročilo, da se je ESP32 ves čas ponovno zaganjal. To sem rešil tako, da sem z nožem prerezal povezavo pina D2, ki ga nisem potreboval.



Slika 3.21: Končano vezje s komponentami za prvo različico kontrolnika

3.3.3 Izdelava prvega ohišja in vgradnja komponent

Prvo ohišje kontrolnika sem izdelal iz stare plastične škatle z lesenim pokrovom v katerega sem izrezal odprtine za vtičnice. Odrezal sem tudi predelne plastične stene, ki so bile v škatli in naredil odprtino za napajalni kabel. V prvem ohišju so bile za priklop akvarijskih naprav uporabljene nadometne vtičnice (slika 3.22), za napajanje pa 200 mA polnilec USB za telefon brez ohišja (na sliki pod lesenim pokrovom).



Slika 3.22: Prvo ohišje

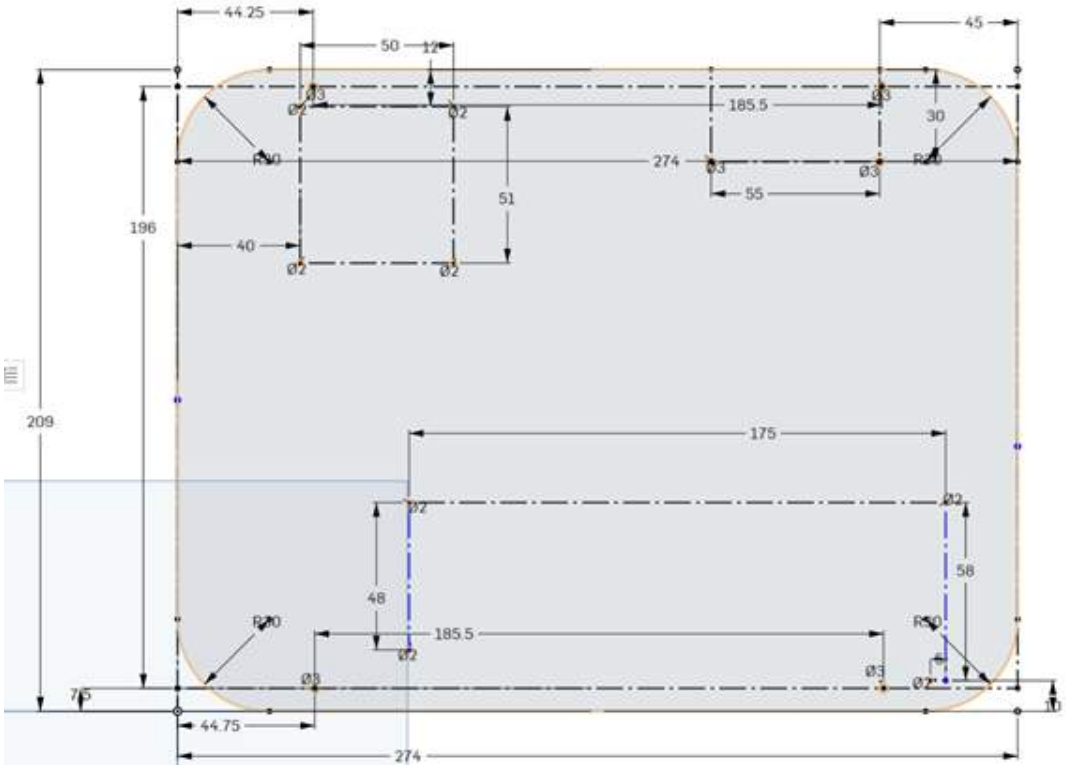
Vsaka žica iz napajalnega kabla (faza, nula in zemlja) je v ohišju razdeljena na štiri dele: trije deli za vtičnice in en del za napajalnik. Nula in zemlja sta priključeni direktno na vtičnice, faza pa je priključena preko relejev. Glavne težave tega ohišja so bile nestabilnost (pokrov ni bil stabilen, napajalni kabel in vtičnice so bile slabo pritrjene itd.), prostorska stiska (težek dostop do komponent) in neurejenost.

3.3.4 Izdelava drugega ohišja in vgradnja komponent

Z drugim ohišjem sem poskušal odpraviti vse napake prvega, zato sem želel, da je večje, urejeno in trdno.

Novo ohišje mora imeti dovolj prostora za vgradnjo vseh komponent, njihovo fiksno pritrditev kot tudi možnost nadgradnje z dodatnimi komponentami. Za ohišje sem zato izbral nadometno dozo dimenzije 300 mm x 220 mm x 120 mm.

Za pritrditev komponent v notranjosti sem iz pleksi stekla izdelal ploščo. Ploščo sem narisal v programu Onshape in jo izrezal na laserskem rezalniku. Na plošči sem predvidel odprtine za pritrditev komponent. Ploščo sem z vijaki pritrdil na dno ohišja.

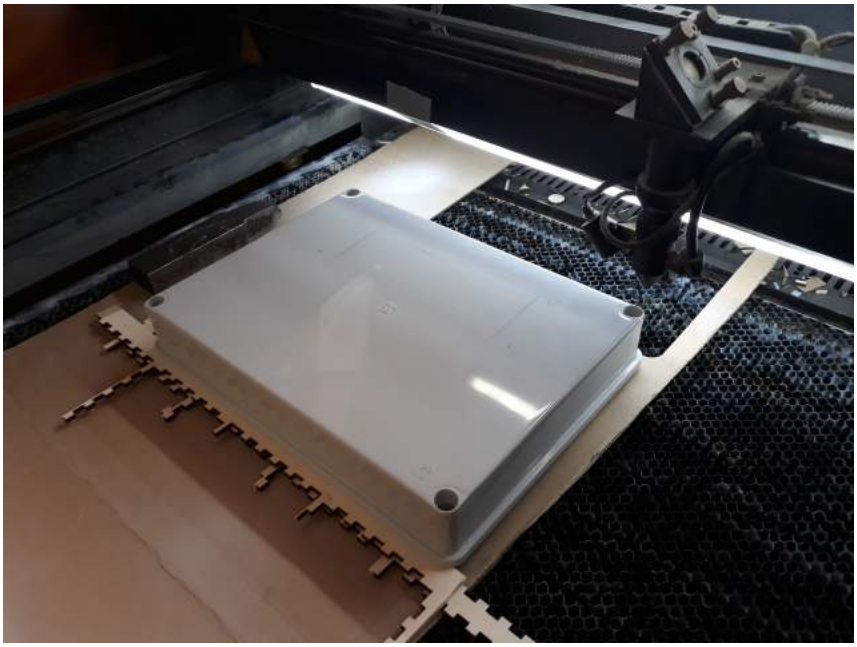


Slika 3.23: Skica za izrez plošče v programu Onshap

V odprtine na plošči sem z navojnim svedrom M3 urezal navoj in komponente z vijaki pritrdil na ploščo.

Slika 3.24: Izrez plošče za pritrditev komponent

Kot je opisano v poglavju 3.2.8 *Material za izdelavo ohišja* kontrolnika so za priklop akvarijskih naprav na kontrolnik uporabljene navadne vtičnice namenjene za montažo na steno. Za njihovo vgradnjo sem s pomočjo laserskega rezalnika izrezal odprtine v pokrov doze. Vtičnice sem pritrdil na pokrov z vijaki M4 in maticami.



Slika 3.25: Izrez lukenj za vtičnice

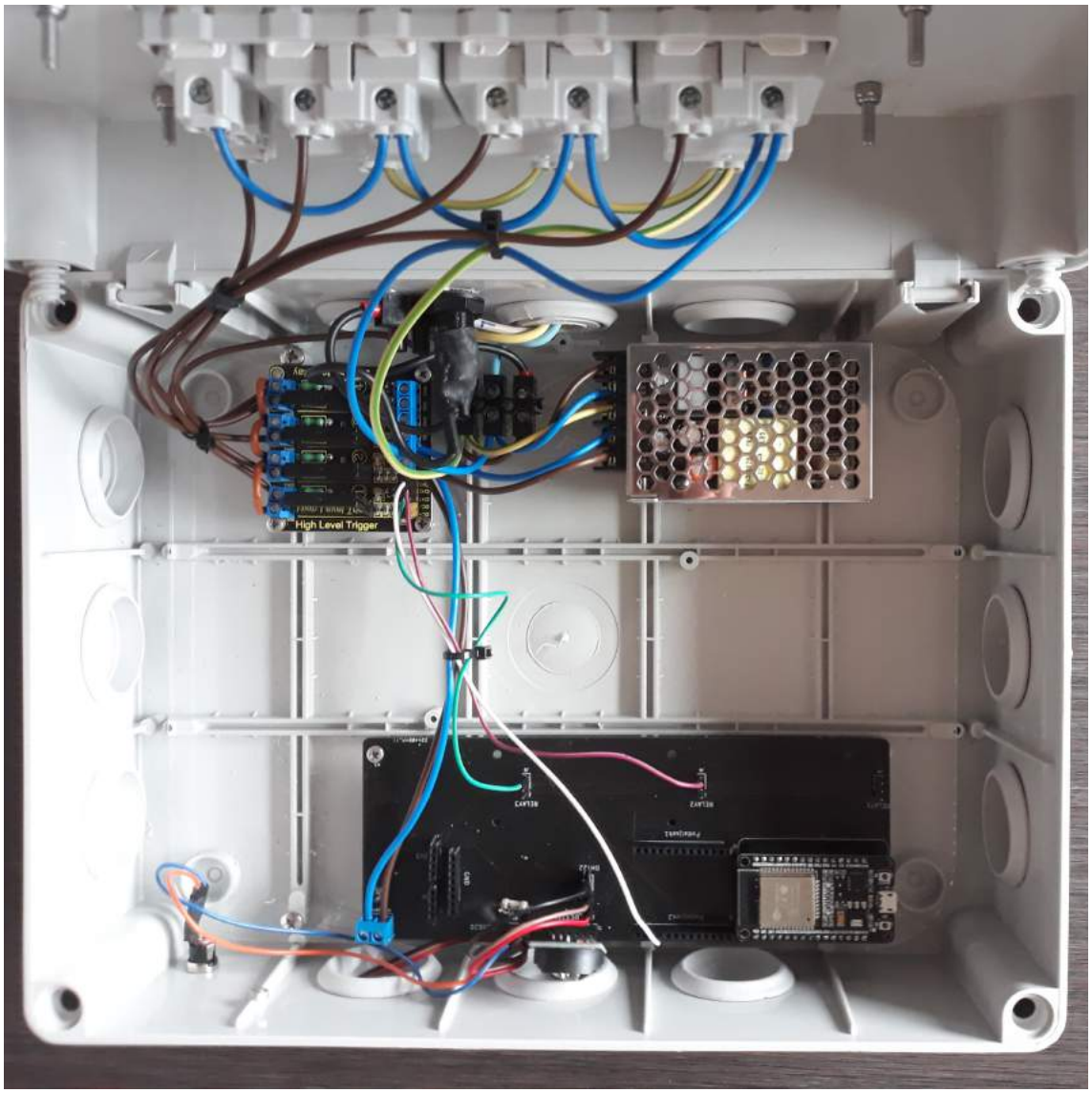
Na straneh ohišja sem zvrtal še nekaj manjših odprtin: za senzor DHT22, konektor DS18B20, ohišje varovalke in vhod priključnega kabla. Konektor in ohišje za varovalko sem pritrdil z maticama, senzor pa sem pritrdil z matico in vijakom M3. Napajalni kabel sem vpeljal v ohišje preko tesnilne uvodnice. Vse komponente sem povezal med sabo. Elektronika je bila tako varno pritrjena v ohišje, ki jo varuje pred prahom in vodo. S tem sem zaključil izdelavo končne različice kontrolnika.

Slika 3.26: Namestitev vtičnic in napajalnega kabla

Kot je vidno iz slike 3.26 so na kontrolniku 4 vtičnice, torej ena več, kot je potrebno za nadzor 3 akvarijskih naprav (grelec, filter in luč). Četrta vtičnica je dodana z namenom, da bi kontrolnik lahko nadziral še eno dodatno napravo na primer:

črpalko za gnojila ali ventil za dodajanje ogljikovega dioksida (CO_2). Prav tako ohišje omogoča vgradnjo še dodatnih vtičnic ter dodatnih naprav kot je črpalka za gnojila.

Na sliki 3.27 je vidna notranjost kontrolnika. Za razliko od prve različice kontrolnika sem v tem uporabil tudi nov rele modul, ki ima 4 manjše SSR releje. Uporabil sem tudi napajalnik, ki ga v prvem ohišju ni bilo, saj je bilo napajanje zagotovljeno preko USB kabla.

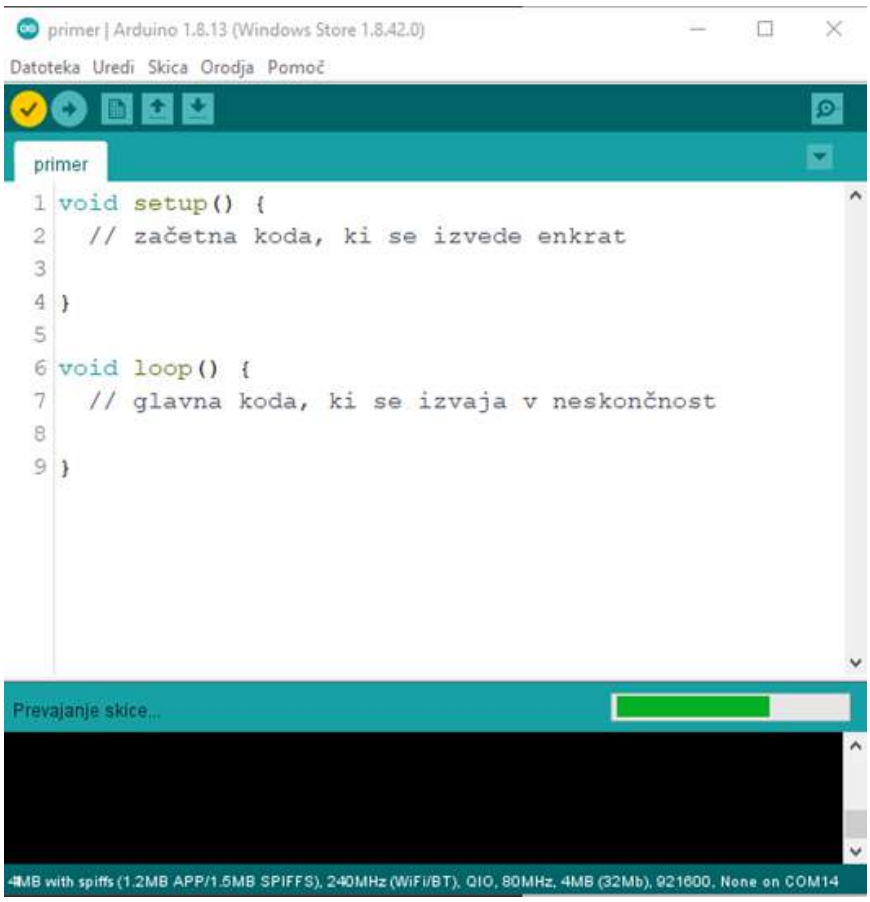


Slika 3.27: Namestitev elektronike v notranjost nadometne doze

3.4 Programski del

3.4.1 Arudino IDE

Kontrolnik sem programiral v razvojnem jeziku C++, prilagojenem za Arduino programsko okolje.To je odprtokodno in brezplačno okolje, ki deluje v Windows, MacOS in Linux operacijskih sistemih. Primarno je namenjeno za programiranje mikrokrmilnih ploščic Arduino. Zaradi njegove prilagodljivosti pa ga lahko uporabimo tudi za programiranje mikrokrmilnih ploščič drugih proizvajalcev, kot je na primer DOIT Esp32 DevKit, ki sem ga uporabil v kontrolniku. V Arduino IDE lahko glede na potrebe našega projekta dodamo veliko različnih knjižnic(zbirke že napisanih funkcij za lažje programiranje). Za pregled kode Arduino IDE vsebuje prevajalnik, ki nas obvesti o napakah v kodi. Slabost Arduino IDE je predvsem, da nima vgrajenih samodejnih popravkov kode med pisanjem in predlogov za lažje pisanje. Programiranje mikrokrmilnih ploščic v Arduino IDE običajno poteka preko kabla USB in serijskega pretvornika, vendar sem kontrolnik programiral na drugačen način (glej poglavje 3.4.10 *Način za posodobitev na daljavo*) [3, 16].

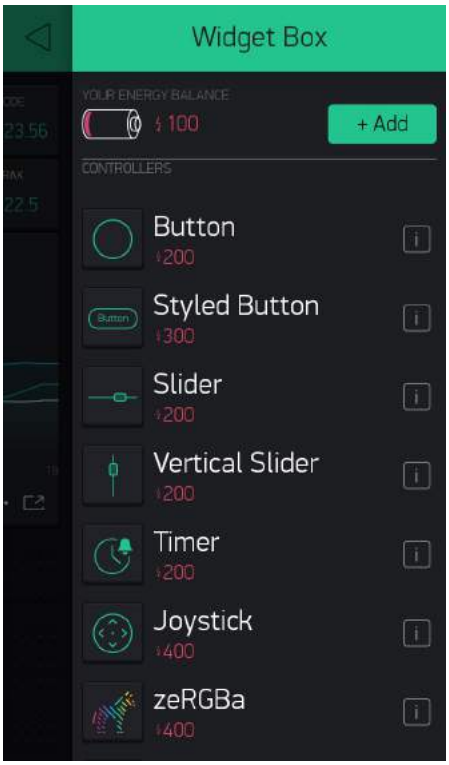


Slika 3.28: Razvojno okolje Arduino IDE

Projekti v Arduino IDE se imenujejo skice. Vsaka skica vsebuje najmanj dve osnovni funkciji *void setup()* in *loop()*. Funkcija *void setup()* (namenjena nastavitvi komunikacij, V/I enot) se izvede enkrat na začetku programa, medtem ko se funkcija *void loop()* (namenjena izvajanju glavnega programa) izvaja v neskončnost.

3.4.2 Blynk knjižnica

Blynk je odprtokodna platforma za internet stvari (ang. Internet of Things oziroma IoT). Skupaj za njo je narejena aplikacija za pametne telefone, ki je prilagodljiva uporabnikovim potrebam. V njej lahko nastavljamo tako imenovane male aplikacije (ang. widget), preko katerih lahko podatke pošiljamo (na primer: tipka, analogna ročica, itd.) ali pa jih beremo (na primer: prikazovalnik številk, grafi, itd.) Podatki, ki jih pošlje uporabnik ali kontrolnik, niso shranjeni lokalno, temveč na strežniku oblaka Blynk. Ta je brezplačen, plačljivo je le število widget-ov, ki se jih v aplikaciji lahko kupi z enoto "energija", ki je plačilno sredstvo v aplikaciji. Uporabnik ima brezplačno na voljo 2000 enot "energije". Dodatno lahko kupi "energijo"v različnih paketih, na primer 1000 enot za 3,3 €. Druga možnost je, da ima uporabnik na svojem računalniku lokalni strežnik, kar omogoči neomejeno količino "energije". Slabost v tem primeru je, da je strežnik dostopen le iz domačega omrežja. Za primer upravljanja z akvarijskim kontrolnikom je 2000 enot "energije"ravno dovolj.



Slika 3.29: Meni za dodajanje widgetov v Blynk aplikaciji za pametne telefone

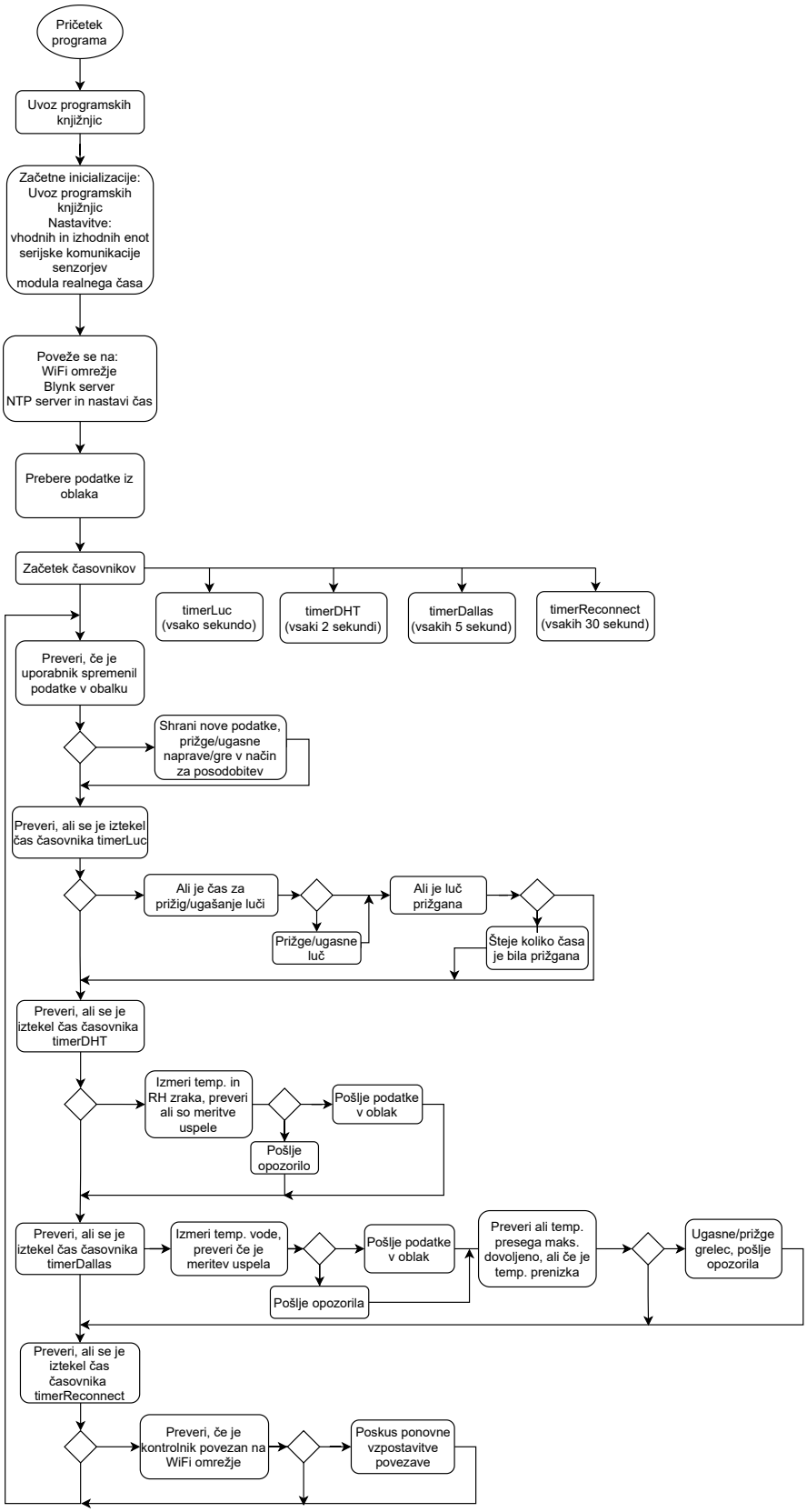
Podatki so shranjeni v oblaku Blynk v tako imenovanih virtualnih pin-ih. V njih lahko shranimo različne vrste podatkov. Identifikacija za dostop do podatkov poteka s tako imenovanimi avtorizacijskimi ključi (ang. Auth Token-ni), ki so drugačni za vsak projekt. Uporabnik jih vidi v nastavitvah aplikacije Blynk in jih prejme po e-pošti. Za Blynk so napisane knjižnice v programskih jezikih C++, JS in Python, poleg tega pa podpira tudi HTTP zahteve.

Za nadziranje kontrolnika je uporabljena aplikacija Blynk, v kodi pa sem uporabil knjižnico Blynk. Ta omogoča preprosto spreminjanje in branje podatkov iz oblaka Blynk. Koda za komunikacijo z oblakom Blynk je opisana v poglavju 3.4.5 Preverjanje, če je uporabnik spremenil podatke v oblaku, 3.4.6 Merjenje temperature in relativne vlage zraka timerDHT in 3.4.7 Merjenje temperature vode in upravljanje z grelcem timerDallas. Glavna slabost knjižnice Blynk je verjetno, da mora v funkciji *void loop()* biti funkcija *Blynk.run()*, vse druge funkcije pa morajo biti klicane preko tako imenovanih časovnikov imenovanih BlynkTimer. Tem določimo ime, funkcijo, ki jo morajo poklicati, in čas, ki mine med dvema klicema funkcije. Nato v *void loop()* ustavimo funkcijo *imeČasovnika.run()*, da časovnik pravilno deluje [4].

3.4.3 Potek programa za kontrolnik

Program za kontrolnik deluje tako, da so posamezni deli razdeljeni na funkcije void, ki se jih v *void loop()* kliče s časovniki. Poleg tega so nekatere funkcije poklicane tudi ob spremembah podatkov v oblaku (na primer uporabnik prižge luč).

Potek programa je prikazan na diagramu na sliki 3.30. Posamezni deli diagrama so opisani v naslednjih poglavjih, celoten program za kontrolnik je priložen v prilogi 3 (Priloga 3: Celotna koda za kontrolnik).



Slika 3.30: Diagram poteka programa kontrolnika

3.4.4 Začetne inicializacije in nastavitve

V vsakem programu se na začetku vključi knjižnice in definira spremenljivke. Zato sem tudi v programu za kontrolnik naredil tako. Spodaj je prikazana koda, ki vključi knjižnice, definira spremenljivke za vzpostavitev povezave z WiFi omrežjem in oblakom ter definira pine, na katere so priključeni releji.

```
//knjiznjice
#include <OneWire.h>
#include <DallasTemperature.h>
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
#include "DHTesp.h"
#include "RTClib.h"
#include <TimeLib.h>
#include <WiFiUdp.h>
#include "ESP32_MailClient.h"

//knjiznice za posodobitev preko interneta
#include <WebServer.h>
#include <ESPmDNS.h>
#include <Update.h>

const char* host = "esp32";
char auth[] = ""; // auth token za blynk
char ssid[] = ""; //ime wifija
char password[] = "";  //geslo za wifi

#define luc_pin 27 //pini na katere so prikljuceni releji za
   akvarijske naprave
#define grelec_pin 25
#define filter_pin 26
```

Koda je napisana pred *void setup()* in *void loop()*. Poleg definiranja spremenljivk sem tukaj ustvaril tudi objekte za senzorje, modul realnega časa in pripravil časovnike. V *void setup()* nato kontrolnik prebere te spremenljivke ter se poveže na omrežje WiFi, vzpostavi povezavo s senzorji in modulom realnega časa, nastavi modul realnega časa na uro, ki jo prebere iz strežnika NTP, se poveže na server Blynk, nastavi pine za releje na izhodne, začne časovnike, prebere spremenljivke iz oblaka in jih shrani ter nastavi serijsko komunikacijo.

3.4.5 Preverjanje, če je uporabnik spremenil podatke v oblaku

Virtualne pine oblaka Blynk je možno prebrati s funkcijo *BLYNK_WRITE(številka virtualnega pina)*, ki se samodejno izvede ob vsaki spremembi podatkov v oblaku. Spodaj je viden primer branja virtualnega pina 20 preko katerega se nastavlja ura ob kateri se prižge akvarijska luč.

```
BLYNK_WRITE(V20){
  prizig_h = param.asInt();
  String t = "prizig_h je sedaj ";
  String s = t + prizig_h;
  terminal.println(s);
  terminal.flush();
}
```

Program najprej pretvori pridobljene podatke v podatkovni tip int (številka), za to je uporabljena funkcija *.asInt()*, jih shrani v spremenljivko prizig_h in pošlje odgovor preko terminala, ki je viden v aplikaciji. Na podoben način lahko kontrolnik tudi upravlja z napravami, takrat, ko jih prižge ali ugasne uporabnik:

```
  BLYNK_WRITE(V4)
{
  filter = param.asInt();
  String t = "Filter je sedaj ";
  String s = t + filter;
  terminal.println(s);
  terminal.flush();
  digitalWrite(filter_pin, filter);
}
```

V tem primeru kontrolnik le nastavi pin, ki nadzira rele za filter na prejeto vrednost. Podatke iz virtualnih pinov se lahko prebere tudi s funkcijo *Blynk.syncVirtual(številka virtualnega pina)*, ki aktivira funkcijo *BLYNK_WRITE()*, kot če bi se podatki v oblaku spremenili v tistem trenutku. Kontrolnik pošilja in bere veliko podatkov iz oblaka. Celoten seznam uporabljenih virtualnih pinov, spremenljivk in njihovih namenov je prikazan v tabeli 3.1.

Tabela 3.1: Seznam uporabljenih virtualnih pinov

Št. V. pina	**Ime spremenljivke**	**Namen**
V1		Pošiljanje sporočil/opozoril uporabniku
V2	luc	Nadzor luči
V3	grelec	Nadzor grelca
V4	filter	Nadzor filtra
V5	programing_mode	Aktivacija načina za posodobitev na daljavo
V10	newValues.temperature	Pošiljanje temperature zraka v oblak
V11	newValues.humidity	Pošiljanje RH zraka v oblak
V12	temperatura_vode	Pošiljanje temperature vode v oblak
V20	prizig_h	Nastavitev ure ob kateri se prižge luč
V21	prizig_min	Nastavitev minute v uri ob kateri se prižge luč
V22	maks_time_h	Koliko ur na dan je lahko luč prižgana
V23	maks_time_min	Koliko minut po preteku zadnje ure je luč prižgana
V24	luc_timer	Koliko minut bo luč prižgana, če jo prižge uporabnik in je maks. čas že presežen
V25	maks_voda_temp	Temperatura pri kateri kontrolnik izklopi grelec

3.4.6 Merjenje temperature in vlage zraka timerDHT

Kot je opisano v poglavju 3.2.4 Senzor DHT22 se za merjenje temperature in relativne vlage zraka uporablja senzor DHT22. Meritve se izvedejo v funkciji *DHT_meritev*, ki jo časovnik timerDHT pokliče vsaki 2 sekundi. Za komunikacijo s senzorjem je uporabljena knjižnica DHTesp. Kontrolnik najprej poskusi prebrati podatke iz senzorja, jih shrani v spremenljivko newValues ter nato preveri, če je bila meritev uspešna. V tem primeru podatke pošlje v oblak s funkcijo *Blynk.virtualWrite(številka*

pina, podatek), v nasprotnem primeru pa uporabnika opozori, da meritev ni uspela.

```
void DHT_meritev(){
  //pridobi temperaturo in vlago od DHT22 senzorja
  TempAndHumidity newValues = dht.getTempAndHumidity();
  //preveri ce je bilo branje uspesno in poslje podatke blynku, ce
   branje ni bilo uspesno poslje opozorilo na terminal
  if(String(newValues.temperature) != "nan"){
  Blynk.virtualWrite(V10, newValues.temperature);
  Blynk.virtualWrite(V11, newValues.humidity);
  }
  else{ terminal.println("dht senzor ne deluje!"); }
}
```

3.4.7 Merjenje temperature vode in upravljanje z grelcem timerDallas

Meritve temperature vode se izvajajo s senzorjem DS18B20 in pripadajočo proizvajalčevo knjižnico DallasTemperature. Ta omogoča, da s funkcijo *ds18b20_meritev*, ki jo vsakih 5 sekund pokliče časovnik z imenom timerDallas, kontrolnik najprej prebere temperaturo senzorja, preveri če je branje uspelo in v tem primeru temperaturo pošlje v oblak. Če je temperatura večja od maksimalne dovoljene temperature (spremenljivka maks_voda_temp), kontrolnik ugasne grelec ter pošlje uporabniku obvestilo o previsoki temperaturi v terminal in na e-poštni naslov. Za pošiljanje sporočila na e-poštni naslov je uporabljena knjižnica ESP32_MailClient, za kontrolnik pa sem ustvaril Gmail račun.

V nadaljevanju je prikazan del kode, s katero se izmeri temperatura vode in kontrolnik preveri, če je branje uspelo (če ni, bo temp. enaka -127 ali 85). Nato kontrolnik preveri, če je temperatura previsoka in, če je, ugasne grelec ter pošlje opozorilo na e-poštni naslov in na terminal.

```
sensors.requestTemperatures();
float temperatura_vode = sensors.getTempCByIndex(0);
Serial.print(temperatura_vode);
Serial.println("°C");
if(temperatura_vode != -127 && temperatura_vode != 85){
Blynk.virtualWrite(V12, temperatura_vode);
if(temperatura_vode >= maks_voda_temp && grelec == 1){
  Serial.println("temperatura je prevelika, ugasanje grelca");
  terminal.println("temperatura je prevelika, ugasanje grelca");
  terminal.flush();
  smtpData.setLogin(smtpServer, smtpServerPort,
 emailSenderAccount, emailSenderPassword);
  smtpData.setSender("Akvarij", emailSenderAccount);
  smtpData.setPriority("1");
  smtpData.setSubject("Akvarij se pregreva!");
  String t = "<div style=\"color:#2f4468;\"><h1>V akvariju je
 bila izmerjena neobicajno visoka temperatura. <br>Temperatura je
 : ";
  String b = "°C <br>Grelec je bil ugasnjen</h1><p>- Avtomatsko
 poslal kontrolnik akvarija</p></div>";
  String s = t + temperatura_vode + b;
  smtpData.setMessage(s, true);
  smtpData.addRecipient(emailRecipient);
  smtpData.setSendCallback(sendCallback);
if (!MailClient.sendMail(smtpData)){
  Serial.println("Error sending Email, " + MailClient.
 smtpErrorReason());
  terminal.println("Error sending Email, " + MailClient.
 smtpErrorReason());
  terminal.flush();
}
smtpData.empty();
t = "";
b = "";
s = "";
  digitalWrite(grelec_pin, LOW);
  Blynk.virtualWrite(V3, LOW);
  grelec = 0;
}
```

Poleg zgoraj navedene kode kontrolnik preveri tudi, če je temperatura vode padla nazaj pod maksimalno dovoljeno. Če je temperatura padla pod maksimalno dovoljeno in je grelec ugasnjen, ga prižge nazaj oziroma pošlje opozorilo, če meritev ni uspela.

3.4.8 Upravljanje z lučjo timerLuc

Časovnik z imenom timerLuc upravlja z akvarijsko lučjo in se izvede vsako sekundo. Najprej kontrolnik prebere čas iz modula realnega časa. Za komunikacijo z modulom je uporabljena knjižnica RTClib. Nato program prebere še temperaturo iz modula realnega časa. Če je ta večja od 40 °C pošlje opozorilo, da je nekaj narobe. Za tem je del programa, ki vsak dan ob polnoči ponastavi spremenljivko, v kateri je shranjeno koliko časa je bila prižgana luč. Ta program je prikazan v prvem if stavku v spodnji kodi:

```
if(now.hour() == 00 && now.minute() == 01 && (total_h_turned_on != 0 || total_min_turned_on != 0 || total_sec_turned_on != 0)){
    total_h_turned_on = 0;
    total_min_turned_on = 0;
    total_sec_turned_on = 0;
    maks_time_dosezen = false;
    terminal.println("casovne spremenljivke resetirane");
    terminal.flush();
  }

  if(now.hour() == prizig_h && now.minute() == prizig_min && maks_time_dosezen == false && luc != 1){

    digitalWrite(luc_pin, HIGH);
    Blynk.virtualWrite(V2, HIGH);
    sec_on = now.second();
    min_on = now.minute();
    h_on = now.hour();
    luc = 1;

    Serial.println("luc prizgana po urniku");
    terminal.println("luc prizgana po urniku");
    terminal.flush();
  }

   if(luc == 1){
   luc_stetje();
   }
}
```

Naslednji je if stavek, v katerem se preveri ali je nastopil čas za prižig luči. Če je temu tako in luč še ni prižgana, jo kontrolnik prižge. Ko je luč prižgana se izvede še funkcija *luc_stetje*, s katero kontrolnik šteje koliko časa je luč prižgana. To sporoča uporabniku in ugasne luč, če je presežen maksimalen čas za delovanje luči. Če upo-

rabnik prižge luč, ko je maksimalni čas za delovanje luči tisti dan že bil presežen, se luč samodejno ugasne po določenem nastavljenem času (na primer 10 min). Ta funkcija je uporabna na primer za hranjenje rib, ali pa če želi uporabniki le malo pogledati v akvarij.

3.4.9 Ponovno povezovanje timerReconnect z omrežjem WiFi

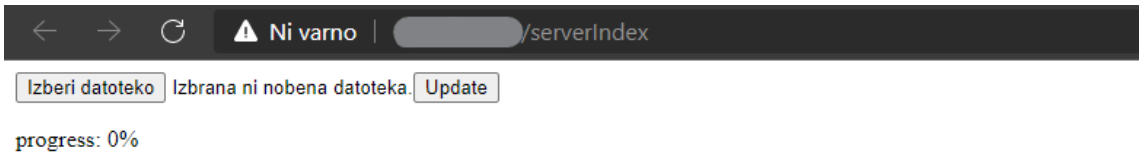
V primeru, če zaradi neznanega razloga omrežje WiFi preneha delovati in se nato nazaj vklopi, kontrolnik ponovno vzpostaviti povezavo. To naredi s funkcijo *reconnectBlynk*, ki jo vsake pol minute sproži časovnik timerReconnect. Koda za to je prikazana v nadaljevanju.

```
void reconnectBlynk() {
   //ce izgubi wifi povezavo se poskusa povezati nazaj
   if (WiFi.status() != WL_CONNECTED) {
     WiFi.disconnect();
     Serial.println("Povezovanje v WiFi omrezje...");
     WiFi.mode(WIFI_AP_STA);
     WiFi.begin(ssid, password);
     delay(1000);
     if(WiFi.status() != WL_CONNECTED) {
      Serial.println("Wifi povezava ni ponovno vzpostavljena");
     } else {
       Serial.println("WiFi povezava ponovno vzpostavljena");
     }
   }
}
```

3.4.10 Način za posodobitev kontrolnika na daljavo

Posodabljanje programa kontrolnika preko kabla USB je nepraktično, saj je kontrolnik za testiranje najbolje imeti priklopljen na akvarijske naprave in senzor za temperaturo vode. Da mi ne bi bilo treba za vsako posodobitev kontrolnika le tega premeščati ali pa povezovati preko zelo dolgega kabla, sem se odločil, da bom v kodo implementiral način za posodobitev preko interneta. To sem storil z knjižnicami WebServer, ESPmDNS in Update.

Ko uporabnik nastavi virtualni pin 5 na vrednost 1, kontrolnik preneha z vsemi drugimi dejanji in zažene strežnik, ki servira spletno stran. Nato ima uporabnik eno minuto časa, da se prijavi, naloži prevedeno kodo (.bin datoteka) in pritisne gumb Update. Uporabniški vmesnik je viden na sliki 3.31.



Slika 3.31: Uporabniški vmesnik za posodobitev kontrolnika na daljavo

Nato kontrolnik prenese datoteko, jo shrani in se ponovno zažene s posodobljenim programom. Če uporabnik v 1 min ne naloži nobene datoteke, se kontrolnik le ponovno zažene s starim programom. Kodo z dodatkom za posodabljanje na daljavo sem sicer moral prvič naložiti preko kabla, sedaj pa dodatek omogoča posodabljanje na daljavo vse dokler ne izbrišem dela kode za posodobitev na daljavo.

3.4.11 Programiranje spletne strani za nadzor kontrolnika

Aplikacija Blynk za pametne telefone deluje dobro, vendar je praktično imeti dostop do nadzora naprav in meritev tudi preko računalnika. Ker Blynk podpira HTPP zahteve, se lahko nadzor naprav izvaja preko obiska spletne strani. Tako na primer obisk povezave `http://blynk-cloud.com/tvojauthtoken/update/V2?value=1` prižge akvarijsko luč, obisk povezave `http://blynk-cloud.com/tvojauthtoken/get/V10` pa prikaže temperaturo zraka.

Da ne bi bilo vsakič potrebno tipkati ali kopirati povezave, se lahko ustvari bližnjica, na katero dvokliknemo in nam odpre povezavo v brskalniku. Vendar to ni najbolj praktična rešitev. Zato sem se odločil, da bom ustvaril spletno stran na kateri bodo vidne meritve in kjer bo uporabnik lahko upravljal in nadziral akvarijske naprave. Spletno stran sem napisal v programskem jeziku Python s pomočjo programskega modula Flask. Za branje in pisanje na virtualne pine sem uporabil knjižnico Requests. V nadaljevanju je prikazan primer kode s katero spletna stran prebere virtualni pin 10 za temperaturo, odstrani odvečen del podatkov in podatek o temperaturi shrani v spremenljivko temp_zrak:

```
    response = requests.get("http://blynk-cloud.com/"+auth_token+"/get/V10")
    temp_zrak = response.text[2:-4]
```

Upravljanje z napravami poteka z dvostopnimi ročicami na spletni strani. Strežnik pregleda, katero ročico je uporabnik spremenil oziroma premaknil ter ugasne ali prižge napravo. Poleg tega še posodobi stanje vseh drugih naprav. V nadaljevanju je prikazan primer kode za upravljanje z lučjo:

```
    luc = request.form.get('luc')
    if request.form.get('luc_sprememba'):
        if luc == "on":
            response = requests.get("http://blynk-cloud.com/"+auth_token+"/update/V2?value=1")
            response = requests.get("http://blynk-cloud.com/"+auth_token+"/get/V2")
            luc_toggle = response.text[2:-2]
            response = requests.get("http://blynk-cloud.com/"+auth_token+"/get/V4")
            filter_toggle = response.text[2:-2]
            response = requests.get("http://blynk-cloud.com/"+auth_token+"/get/V3")
            grelec_toggle = response.text[2:-2]
        else:
            response = requests.get("http://blynk-cloud.com/"+auth_token+"/update/V2?value=0")
            response = requests.get("http://blynk-cloud.com/"+auth_token+"/get/V2")
            luc_toggle = response.text[2:-2]
            response = requests.get("http://blynk-cloud.com/"+auth_token+"/get/V4")
            filter_toggle = response.text[2:-2]
            response = requests.get("http://blynk-cloud.com/"+auth_token+"/get/V3")
            grelec_toggle = response.text[2:-2]
```

Prikazana koda od HTML obrazca prebere podatke za ročico za upravljanje luči, nato preveri, če je uporabnik ugasnil/prižgal luč ter če je to naredil podatke pošlje v oblak. Poleg tega, če je bila luč prižgana/ugasnjena posodobi podatke o stanju ostalih naprav.

Uporabniški vmesnik spletne strani je viden na sliki 3.32.

Slika 3.32: Uporabniški vmesnik spletne strani za nadzor kontrolnika

3.5 Cena kontrolnika

Sestavil sem seznam vseh komponent in njihovih cen. Cene posameznih komponent so prikazane preglednici 3.2.

Tabela 3.2: Cene komponent za izdelavo kontrolnika

Komponente	**Cena (EUR)**
DS18B20	6,15
ESP32	4,00
Releji	10,30
Vtičnice	18,50
N. O. doza	20,00
DHT22	3,00
DS3231	2,00
Napajalnik	17,80
Tiskano vezje	2,00
Priključni kabel	4,50
Pleksi steklo	8,00
Ostale komponente	5,00
Skupaj	101,25

Skupna cena materiala za kontrolnik znaša približno 100 €. Ceno bi lahko še dodatno znižal z izbiro cenejšega napajalnika ali ohišja. V ceno nisem vključil stroškov mojega dela za razvoj kontrolnika, saj bi bil pri serijski proizvodnji in v količinah v katerih so izdelani komercialni kontrolniki ta strošek zanemarljiv.

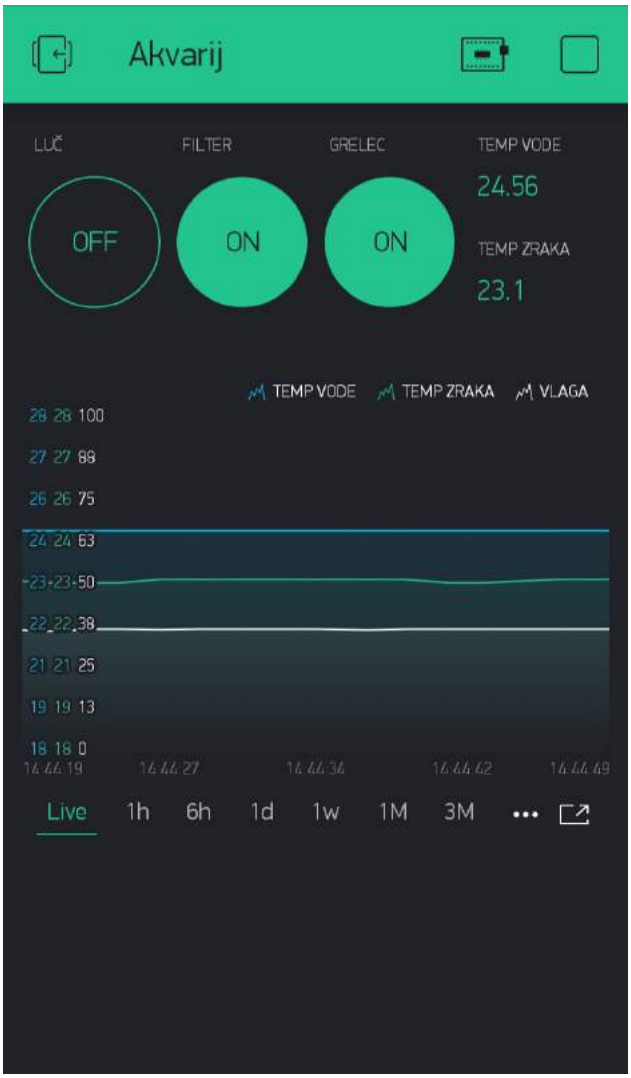
4 Rezultati in razprava

4.1 Opis končanega kontrolnika

Osnovna ideja za kontrolnik je, da je sposoben upravljati in nadzirati najmanj tri naprave (filter, grelec in luč) in izvajati najmanj tri meritve (temperatura zraka in vode ter relativna vlažnost zraka).

Slika 4.1: Končan kontrolnik med delovanjem

Izdelan kontrolnik je spodoben opravljati vsa navedena opravila. V aplikaciji za pametne telefone, ki je prikazana na sliki 4.2, so prikazane meritve in grafi. Temperatura vode in zraka sta prikazani posebej še s številsko vrednostjo. Relativna vlažnost zraka zaradi pomanjkanja "energije"v aplikaciji s številko ni prikazana.

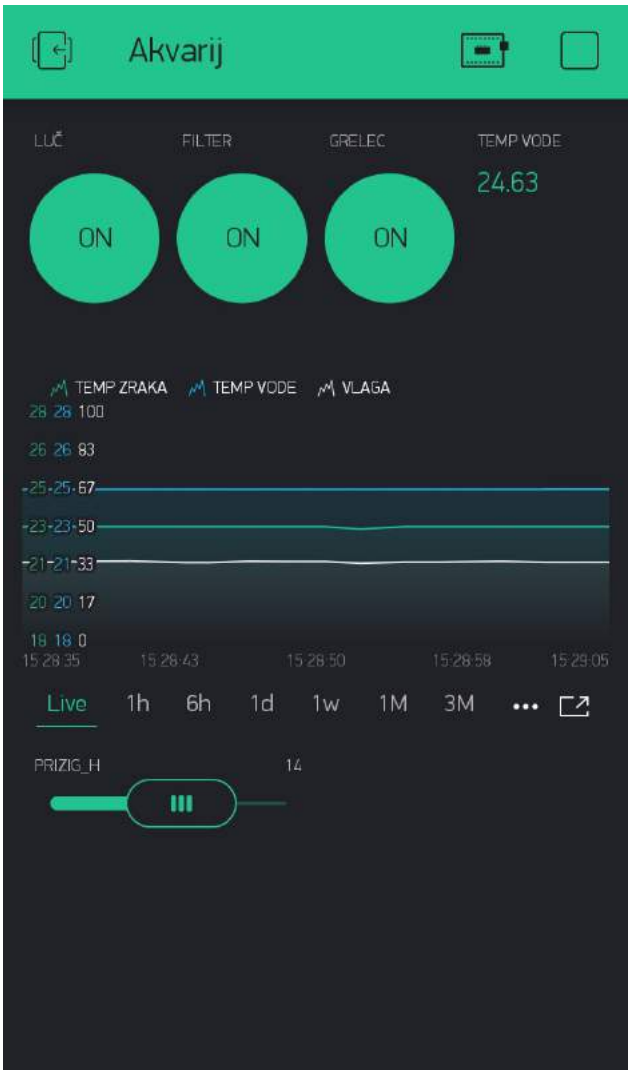


Slika 4.2: Aplikacija za nadzor kontrolnika

Če iz aplikacije odstranimo prikazovalnik temperature zraka imamo dovolj "energije", da lahko dodamo "terminal"– ukazno lupino. Ta nam omogoča prikaz opozoril, ki jih pošilja kontrolnik in sicer na primer informacije o napakah pri branju senzorjev ali pa informacijo koliko časa je bila prižgana luč. Na sliki 4.3 je prikazana informacija o času delovanja luči.

Slika 4.3: Aplikacije za nadzor kontrolnika s terminalom

Če želimo spreminjati vrednosti spremenljivk kot so na primer: ob kateri uri se prižge luč, maksimalno dovoljeno temperaturo vode itd., moramo odstraniti enega izmed »widget-ov« in dodati drugega za pošiljanje podatkov, kot je to prikazano na sliki 4.4. Na takšen način lahko iz aplikacije nadziramo vse vrednosti spremenljivk iz preglednice 1, ki so na virtualnem pinu s številko 20 ali več. Uporabnik lahko plača tudi 3,3 € in s tem kupi 1000 dodatne »energije« v Blynk aplikaciji. S to dodatno energijo je možno dodati še 4 dodatne ročice za nadziranje drugih nastavitev in prikazovalnik za temperaturo zraka.



Slika 4.4: Aplikacija za nadzor kontrolnika z ročico za spreminjanje ure prižiga luči

Filter, grelec in akvarijsko luč je možno ročno upravljati preko gumbov v aplikaciji. Z lučjo in grelcem kontrolnik upravlja tudi avtomatsko.

Luč v akvariju se samodejno prižge ob nastavljeni uri, če tisti dan še ni presežen njen maksimalni dovoljeni čas delovanja. Če je uporabnik luč že prižgal in jo nato ugasnil, bo luč prižgana za toliko manj časa, kot je tisti dan že bila. Če uporabnik prižge luč po tem, ko je presegla maksimalni čas delovanja, se bo po poljubno nastavljenem času samodejno ugasnila (če uporabnik te nastavitve ne spremeni je to 10 min).

Grelec se glede na temperaturo vode, ki jo izmeri s svojim senzorjem, samodejno prižiga in ugaša. Grelec lahko prižiga ali ugaša tudi kontrolnik, če temperatura vode, ki jo izmeri senzor DS18B20, pade pod oziroma preseže nastavljeno temperaturo.

Kontrolnik ponuja dve funkcionalnosti vezani na spremljanje temperature. Prva služi le kot dodatno varovalo pred pregrevanjem akvarija v primeru okvare grelca. Če senzor DS18B20 izmeri neobičajno visoko temperaturo vode, kontrolnik grelec ugasne. Ob tem pošlje uporabniku sporočilo na e-poštni naslov, kot je prikazano na sliki 4.5 .

Slika 4.5: Opozorilo ob ugasnitvi grelca na e-poštni naslov

Druga je reguliranje delovanje grelca v primeru, če je potrebno v akvariju vzdrževati oziroma nastaviti maksimalno temperaturo vode nižje (na primer 23 °C) od temperature na katero grelec normalno greje vodo (na primer 25 °C). V tem primeru kontrolnik prižiga in ugaša grelec tako, da voda ne preseže nastavljene maksimalne temperature (23 °C). Ko se temperatura vode zniža za 0,5 °C pod maksimalno nastavljeno (23 °C), se grelec ponovno prižge. Uporabnik lahko tudi ročno izklopi grelec, vendar se ta ob naslednji meritvi temperature vode ponovno prižge, če je temperatura vode prenizka.

Včasih se pri komunikaciji med kontrolnikom in senzorji zgodi napaka. Kontrolnik to zazna. Pri napaki komunikacije s senzorjem DHT22 se namesto izmerjene temperature in relativne vlažnosti zraka izpiše sporočilo »nan«. Pri napaki komunikacije s senzorjem DS18B20 pa se namesto dejanske temperature vode izpiše številka »85« ali »-127«. V primeru napake pri komunikaciji z navedenima senzorjema kontrolnik pošlje opozorilo v terminal in meritve ne pošlje v oblak, zato da grafični prikazi ne bi prikazovali napak.

Če se zgodi napaka pri branju senzorja DS18B20, kontrolnik uporabniku dodatno pošlje tudi sporočilo na e-poštni naslov. To je zelo pomembno, saj v primeru, če bi odpovedala senzor za temperaturo vode in senzor znotraj grelca hkrati, akvarij ni zaščiten pred pregrevanjem. Uporabnik tako ve, da mora preveriti stanje senzorjev oziroma naprav v akvariju. Sporočilo, ki ga prejeme uporabnik na e-poštni naslov je prikazano na sliki 4.6.

Slika 4.6: Opozorilo na e-poštni naslov v primeru napake pri branju senzorja temperature vode

Kot je opisano tudi v poglavju 3.4.11 *Programiranje spletne strani za nadzor kontrolnika*, je akvarij možno upravljati in nadzirati tudi preko za ta namen izdelane spletne strani. Na vrhu spletne strani so prikazane meritve. Naprave upravljamo s premikanjem digitalnih ročic. Spletna stran je narejena za uporabo na različnih napravah (računalnikih, pametnih telefonih, tablicah), saj se prilagodi velikosti zaslona naprave. Na naslednji sliki je prikazana spletna stran kot se jo vidi na pametnem telefonu.
Glavna trenutna slabost spletne strani je, da se prikazane informacije (meritve, stanje naprav) posodobijo le, ko uporabnik osveži spletno stran ali pa prižge/ugasne eno izmed naprav. To nameravam v prihodnosti popraviti z uporabo razvojnega okolja Ajax. Poleg tega nameravam na spletno stran dodati še nadzor nad preostalimi nastavitvami kontrolnika in prikazovalnik grafov.

Slika 4.7: Mobilna spletna stran

Izdelan kontrolnik in programsko kodo zanj sem preizkušal na mojem akvariju 4 tedne in v tem času nisem opazil nepričakovanega vedenja kontrolnika oziroma njegovega napačnega delovanja. V tem času sem opazil tudi nekaj pozitivnih učinkov kontrolnika in sicer prihranek mojega časa za vzdrževanje akvarija ter manjšo rast alg, kar pripisujem predvsem primerni osvetlitvi in temperaturi vode v akvariju.

4.2 Pregled hipotez

V prejšnjem podpoglavju so bile opisane vse glavne funkcije kontrolnika. Sedaj, ko jih poznamo, lahko preverimo hipoteze.

HIPOTEZA 1: *Možno je izdelati kontrolnik, ki bo nadzoroval filter, grelec ter luč v akvariju in stane manj kot komercialni kontrolniki.*

Cena izdelave kontrolnika je približno 100 € (glej poglavje 3.5 *Cena kontrolnika*). Cena primerljivega komercialnega kontrolnika s senzorji HYDROS Control 2 pa je približno 165 € (glej poglavje 2.2 Pregled akvarijskih kontrolnikov na trgu). Pri temu je potrebno upoštevati še, da je naprodaj le v ZDA. Pri njegovem nakupu bi morali plačati še davke, carino ter stroške poštnine. Za omenjeni komercialni kontrolnik bi zato odšteli več kot 250 €. Tako je moj kontrolnik več kot dvakrat cenejši od obstoječih primerljivih komercialnih rešitev, zato je HIPOTEZA 1 potrjena.

HIPOTEZA 2: *Vse funkcije kontrolnika (nadzor naprav in ogled meritev) bo uporabnik lahko upravljal in nadziral na daljavo preko pametnega telefona.*

Uporabnik lahko vse funkcije upravlja in nadzira preko aplikacije za pametne telefone, če je telefon povezan na omrežje WiFi ali mobilno omrežje s prenosom podatkov. Poleg tega lahko uporabnik do določene mere kontrolnik upravlja tudi preko spletne strani. HIPOTEZA 2 je tako potrjena.

HIPOTEZA 3: *Kontrolnik bo akvarij obvaroval pred pregrevanjem v primeru okvare grelca.*

Uporabnik lahko v aplikaciji nastavi maksimalno dovoljeno temperaturo, pri kateri bo kontrolnik samodejno izklopil grelec in tako akvarij obvaroval pred pregrevanjem. Poleg tega kontrolnik v primeru napake pri branju senzorja za temperaturo vode DS18B20 uporabniku na e-poštni naslov pošlje obvestilo. Uporabnik tako ve, da mora preveriti stanje senzorjev oziroma naprav v akvariju. HIPOTEZA 3 je s tem potrjena.

HIPOTEZA 4: *V primeru napake pri branju oziroma nedelovanju senzorjev bo kontrolnik napako sporočil uporabniku.*

Kadar pri branju podatkov iz senzorjev pride do napake, kontrolnik to zazna in uporabniku pošlje obvestilo na terminal ter, če je prišlo do napake pri branju senzorja za temperaturo vode DS18B20, uporabniku tudi pošlje obvestilo na e-poštni naslov. Tako je HIPOTEZA 4 potrjena.

4.3 Možnosti za izboljšave in nadgradnje

Izdelani kontrolnik lastniku akvarija močno olajša skrb za akvarij in omogoča večjo varnost v primeru okvare grelca. Vendar je kljub temu možnih še veliko nadgradenj tako samega kontrolnika kot tudi njegovih pripadajočih programov (aplikacije, spletne strani in programa za kontrolnik).

V prihodnosti nameravam kontrolniku dodati še podporo za nadzor doziranja gnojil preko peristaltične črpalke in sistema za dovod CO_2. Za implementacijo navedenega mi je zmanjkalo časa pred rokom oddaje raziskovalne naloge, prav tako pa sistema za dovajanje CO_2 v mojem akvariju še nimam.

Poleg navedenih nadgradenj kontrolnika nameravam na spletni strani za nadzor kontrolnika dodati možnosti spreminjanja nastavitev, ki trenutno še niso podprte, ter prikazovalnike grafov meritev. Spletno stran nameravam izboljšati tudi tako, da se bodo podatki, ki jih prikazuje, samodejno posodabljali ne da bi bilo potrebno spletno stran osvežiti. Za nadzor kontrolnika bi lahko razvil tudi aplikacijo za pametne telefone, narejeno posebej za nadzor kontrolnika in opustil trenutno Blynk aplikacijo.

5 Zaključek

V raziskovalni nalogi je opisana izdelava akvarijskega kontrolnika, ki je zmožen upravljati in nadzirati filter, luč in grelec v akvariju. Kontrolnik lastniku akvarija znatno olajša njegovo vzdrževanje ter omogoča njegovo nemoteno delovanje in nadzor tudi ob daljši odsotnosti lastnika. Akvarijska luč se samodejno prižiga in ugaša, tako da je vsak dan prižgana ob določnem času za omejeno časovno obdobje, zato ne prihaja do pretirane rasti alg. Kontrolnik grelec upravlja tako, da je temperatura vode v akvariju primerna in preprečuje pregrevanje akvarija. Kontrolnik meri temperaturo vode in zraka ter relativno vlažnost zraka.

Naprave, ki jih nadzira kontrolnik, je možno upravljati v aplikaciji za pametne telefone. V tej so na ogled tudi vse meritve, preko nje je omogočeno spreminjanje raznih nastavitev. Ustvaril sem tudi spletno stran preko katere je možen pregled meritev in upravljanje z napravami. Izdelani kontrolnik ima vse navedene funkcije za več kot dvakrat manjšo ceno od obstoječih primerljivih komercialnih akvarijskih kontrolnikov.

Izdelan kontrolnik in programsko kodo zanj sem preizkušal na mojem akvariju 4 tedne in v tem času nisem opazil nepričakovanega vedenja kontrolnika oziroma njegovega napačnega delovanja. V tem času sem opazil tudi nekaj pozitivnih učinkov kontrolnika in sicer prihranek mojega časa za vzdrževanje akvarija ter manjšo rast alg, kar pripisujem predvsem primerni osvetlitvi in temperaturi vode v akvariju.

Izdelan kontrolnik že ima veliko funkcij, vendar ga je možno še nadgraditi in izboljšati. V prihodnosti nameravam kontrolniku dodati vsaj še podporo za nadzor doziranja gnojil preko peristaltične črpalke in sistema za dovod CO_2. Prav tako želim izboljšati spletno stran za upravljanje in nadzor kontrolnika.

Kljub vsem nadgradnjam in izboljšavam, ki jih še nameravam dodati kontrolniku, sem že sedaj zelo zadovoljen z izdelanim kontrolnikom in njegovim delovanjem.

Literatura

[1] Aosong Electronics Co.,Ltd, Digital-output relative humidity temperature sensor/module DHT22 (DHT22 also named as AM2302). Dosegljivo: https://datasheetspdf.com/pdf-file/792211/Aosong/DHT22/1, 2021. [Dostopano: 14. 2. 2021].

[2] Aquariadise, Full lighting guide. Dosegljivo: https://www.aquariadise.com/planted-aquarium-full-lighting-guide/, 2021. [Dostopano: 13. 2. 2021].

[3] Arduino - Home. Dosegljivo: https://www.arduino.cc/, 2021. [Dostopano: 6. 3. 2021].

[4] Blynk IoT platform: for businesses and developers. Dosegljivo: https://blynk.io/, 2021. [Dostopano: 6. 3. 2021].

[5] CoralVue Aquarium Products, HYDROS Control 2. Dosegljivo: https://www.coralvuehydros.com/control/control-2/, 2021. [Dostopano: 13. 2. 2021].

[6] DHT22. Dosegljivo: https://shop.mchobby.be/en/environnemental-press-temp-hum-gas/1535-dht22-extra-temperature-himidity-sensor-3232100015357-adafruit.html, 2021. [Dostopano: 14. 2. 2021].

[7] DOIT Esp32 DevKit v1 . Dosegljivo: https://makeradvisor.com/wp-content/uploads/2017/10/esp32-board-bg.jpg, 2021. [Dostopano: 14. 2. 2021].

[8] DS3231. Dosegljivo: https://robu.in/product/ds3231-rtc-module-precise-real-time-clock-i2c-at24c32/, 2021. [Dostopano: 17. 2. 2021].

[9] Felix Smart, Home. Dosegljivo: https://www.felixsmart.com/, 2021. [Dostopano: 13. 2. 2021].

[10] Fishbox, Postavitev akvarija. Dosegljivo: https://www.fishbox.pet/blogs/blog-o-akvaristiki/projekt-akvarij/, 2021. [Dostopano: 13. 2. 2021].

[11] Galco Industrial Electronics, How Relays Work. Dosegljivo: https://www.galco.com/comp/prod/relay.htm, 2021. [Dostopano: 14. 2. 2021].

[12] GHL, Profilux 4e. Dosegljivo: https://www.aquariumcomputer.com/products/profilux-aquarium-controller/profilux-4e/, 2021. [Dostopano: 13. 2. 2021].

[13] GreenAqua, Aquarium filter guide. Dosegljivo: https://greenaqua.hu/en/green-aqua-szuro-valasztasi-tajekoztato, 2021. [Dostopano: 13. 2. 2021].

[14] Axel Gutjahr. *Akvarijske ribe*. Ljubljana, Mladinska knjiga, 2007.

[15] Jumer Žice. Dosegljivo: http://ba.serial-cable.com/, 2021. [Dostopano: 6. 3. 2021].

[16] Peter Krkoč. *Arduino: programirajmo Arduino z lahkoto*. Ljubljana, Ax Elektronika D.O.O., 2014.

[17] Kroženje plinov v akvariju. Dosegljivo: https://www.mrpet.si/2-prvi-koraki-osvetlitev-in-ogrevanje-akvarija-p-18934.aspx, 2021. [Dostopano: 7. 2. 2021].

[18] LastMinuteEngineers.com, Interface DS3231 Precision RTC Module withArduino. Dosegljivo: https://lastminuteengineers.com/ds3231-rtc-arduino-tutorial/, 2021. [Dostopano: 17. 2. 2021].

[19] M3 vijaki. Dosegljivo: https://www.sparkfun.com/products/13227, 2021. [Dostopano: 18. 2. 2021].

[20] Maxim Integrated Products, DS18B20. Dosegljivo: https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf, 2021. [Dostopano: 14. 2. 2021].

[21] Mehanski rele modul. Dosegljivo: https://www.diyelectronics.co.za/store/electro-mechanical-relays/266-1-channel-5v-relay-module-10amp-250v.html, 2021. [Dostopano: 6. 3. 2021].

[22] Mr.Pet, Prvi koraki: Osvetlitev in ogrevanje akvarija. Dosegljivo: https://www.mrpet.si/2-prvi-koraki-osvetlitev-in-ogrevanje-akvarija-p-18934.aspx, 2021. [Dostopano: 13. 2. 2021].

[23] Nadometna doza ELETTROCANALI EC400C8. Dosegljivo: https://https://www.amazon.co.uk/Junction-IP55-300X220X120-ELECTROCHANNELS-EC400C8/dp/B06XC1QHX/, 2021. [Dostopano: 17. 2. 2021].

[24] Napajalni vtič . Dosegljivo: https://www.aliexpress.com/, 2021. [Dostopano: 18. 2. 2021].

[25] Nosilec za mini varovalko. Dosegljivo: https://etrgovina.elektronabava.si/sl/shop/mini-varovalka-nosilec-37404, 2021. [Dostopano: 18. 2. 2021].

[26] Pleksi steklo 4mm . Dosegljivo: https://www.pos-plastika.si/pleksi-steklo4, 2021. [Dostopano: 17. 2. 2021].

[27] Priključni kabel. Dosegljivo: https://www.conrad.si/, 2021. [Dostopano: 17. 2. 2021].

[28] Primer akvarija. Dosegljivo: https://dissolve.com/stock-photo/Boy-watching-fish-tunnel-huge-tank-Blue-Planet-rights-managed-image/102-D869-99-079, 2021. [Dostopano: 7. 2. 2021].

[29] TDK-Lambda AC/DC pretvornik LS25-5. Dosegljivo: https://www.ic-elect.si/napajalniki/industrijski-napajalniki/napajalniki-v-ohisju/ac-dc-pretvornik-220v-5v-25w-ls25-5.html, 2021. [Dostopano: 6. 3. 2021].

[30] TDK-Lamda, LS25-150 Series. Dosegljivo: https://www.ic-elect.si/fileuploader/download/download/?d=0&file=custom%2Fupload%2FFile-0-11489900-1470748725.pdf, 2021. [Dostopano: 6. 3. 2021].

[31] Temperaturni senzor DS18B20. Dosegljivo: https://ram-e-shop.com/product/wire-ds18b20/, 2021. [Dostopano: 14. 2. 2021].

[32] Tesnilna uvodnica. Dosegljivo: https://etrgovina.elektronabava.si/sl/shop/, 2021. [Dostopano: 18. 2. 2021].

[33] The Spruce Pets, Correct Aquarium Water Temperature. Dosegljivo: https://www.thesprucepets.com/aquarium-water-temperature-1381896, 2021. [Dostopano: 13. 2. 2021].

[34] The SprucePets, Determining How Much to Feed Aquarium Fish. Dosegljivo: https://www.thesprucepets.com/how-much-should-i-feed-my-fish-1378746, 2021. [Dostopano: 13. 2. 2021].

[35] Trdni štirikanalni rele modul. Dosegljivo: https://www.keyestudio.com/products/free-shipping-keyestudio-four-channel-solid-state-relays-module-for-arduino, 2021. [Dostopano: 14. 2. 2021].

[36] Vtičnica SCHUKO+KS 2P+E 16A 250V 2M PW, vtičnica EURO+KS 2P 10A 250V 1M PW , nosilec z vijaki 7M in okvir LINE 7M PW. Dosegljivo: https://www.elektrotrgovina.si/, 2021. [Dostopano: 17. 2. 2021].

[37] Zerynth Documentation, DOIT Esp32 DevKit v1. Dosegljivo: https://docs.zerynth.com/latest/reference/boards/doit_esp32/docs/, 2021. [Dostopano: 14. 2. 2021].

Priloge

Priloga 1: Shematika tiskanega vezja

Priloga 2: Načrt tiskanega vezja

Priloga 3: Celotna koda za kontrolnik

Priloga 1: Shematika tiskanega vezja

Priloga 2: Načrt tiskanega vezja

Priloga 3: Celotna koda za kontrolnik

```
//knjiznjice
#include <OneWire.h>
#include <DallasTemperature.h>
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
#include "DHTesp.h"
#include "RTClib.h"
#include <TimeLib.h>
#include <WiFiUdp.h>
#include "ESP32_MailClient.h"

//knjiznice za posodobitev preko interneta
#include <WebServer.h>
#include <ESPmDNS.h>
#include <Update.h>

const char* host = "esp32";
char auth[] = ""; // auth token za blynk
char ssid[] = ""; //ime wifija
char password[] = "";  //geslo za wifi

#define luc_pin 27 //rele 1 luc, rele 2 grelec in rele 2 filter
#define grelec_pin 25
#define filter_pin 26

#define emailSenderAccount    ""  //e-postni naslov za kontrolnik
#define emailSenderPassword   ""  // geslo e-postnega naslova
#define emailRecipient        ""  //e-postni naslov za prejemanje
    obvestil
#define smtpServer            "smtp.gmail.com"
#define smtpServerPort        465
SMTPData smtpData;
```

```
33
34 #define DHTTYPE DHT22
35 #define dhtPin 18  //pin za DHT22 senzor
36 DHTesp dht;
37
38 const int oneWireBus = 4; // pin za DS18B20 senzor
39 OneWire oneWire(oneWireBus);
40 DallasTemperature sensors(&oneWire);
41 int maks_voda_temp = 27; // temperatura pri kateri se bo izklopil
      grelec
42
43 RTC_DS3231 rtc;
44 char daysOfTheWeek[7][24] = {"Sunday", "Monday", "Tuesday", "
      Wednesday", "Thursday", "Friday", "Saturday"};
45
46 #define BLYNK_PRINT Serial
47
48 //timerji za vse razlicne dogodke
49 BlynkTimer timerDHT;
50 BlynkTimer timerLuc;
51 BlynkTimer timerReconnect;
52 BlynkTimer timerDallas;
53 BlynkTimer timerAirUpdate;
54
55 WidgetTerminal terminal(V1);
56
57 const char* ntpServer = "pool.ntp.org";
58 const long  gmtOffset_sec = 2*3600;
59 const int   daylightOffset_sec = 3600;
60
61 //spremenljivke ki povejo, ce so releji prizgani/ugasnjeni
62 int luc;
63 int grelec;
64 int filter;
65
66 bool maks_time_dosezen = false;
67
68 //############## URNIK ##############
69 //cas ob katerem se luc prizge, ce je maks_time_dosezen == false
70 int prizig_h = 15;
71 int prizig_min = 30;
72
73 //koliko casa je luc prizgana preden se maks_time_dosezen spremeni
      na true
```

```
74 int maks_time_h = 6;
75 int maks_time_min = 30;
76
77 // za koliko minut se prizge luc, ce jo prizge uporabnik in je
      maks_time_dosezen == true
78 int luc_timer = 10;
79 //###################################
80
81 //spremenljivke za belezenje koliko casa je bila luc prizgana
82 int h_turned_on;
83 int min_turned_on;
84 int sec_turned_on;
85 int total_h_turned_on;
86 int total_min_turned_on;
87 int total_sec_turned_on;
88 int h_on;
89 int min_on;
90 int sec_on;
91
92 int programing_mode; // ce je programing_mode == true, se bo
      prizgal server za posodobitev preko interneta
93
94 //timer id, ki so potrebni da lahko timerje izklopimo preden
     za[U+FFFD]emo server za posodobitev, da ga slucajno ne bi kaj motili
95 int timerLucId;
96 int timerDHTId;
97 int timerReconnectId;
98 int timerDallasId;
99
100 int ura_opozorjeno;
101
102 static const char ntpServerName[] = "us.pool.ntp.org"; // ime NTP
      serverja preko katerega dobimo tocen cas
103 const int timeZone = 1;  // casovni pas v katerem je Slovenija
104
105 WiFiUDP Udp;
106 unsigned int localPort = 8888;  // port preko katerega se poveze na
       NTP server
107
108 time_t getNtpTime();
109 void digitalClockDisplay();
110 void printDigits(int digits);
111 void sendNTPpacket(IPAddress &address);
112
```

```
113 // html koda za server za posodobitev
114 WebServer server(80);
115
116 /*
117  * Login page
118  */
119 const char* loginIndex =
120  "<form name='loginForm'>"
121     "<table width='20%' bgcolor='A09F9F' align='center'>"
122         "<tr>"
123             "<td colspan=2>"
124                 "<center><font size=4><b>ESP32 Login Page</b></font
    ></center>"
125                 "<br>"
126             "</td>"
127             "<br>"
128             "<br>"
129         "</tr>"
130         "<td>Username:</td>"
131         "<td><input type='text' size=25 name='userid'><br></td>"
132         "</tr>"
133         "<br>"
134         "<br>"
135         "<tr>"
136             "<td>password:</td>"
137             "<td><input type='password' size=25 name='pwd'><br></td
    >"
138             "<br>"
139             "<br>"
140         "</tr>"
141         "<tr>"
142             "<td><input type='submit' onclick='check(this.form)'
    value='Login'></td>"
143         "</tr>"
144     "</table>"
145 "</form>"
146 "<script>"
147     "function check(form)"
148     "{"
149     "if(form.userid.value=='admin' && form.pwd.value=='admin')"
150     "{"
151     "window.open('/serverIndex')"
152     "}"
153     "else"
```

```
    "{"
    " alert('Error password or Username')/*displays error message*/
    "
    "}"
    "}"
"</script>";

/*
 * Server Index Page
 */

const char* serverIndex =
"<script src='https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/
   jquery.min.js'></script>"
"<form method='POST' action='#' enctype='multipart/form-data' id='
   upload_form'>"
   "<input type='file' name='update'>"
        "<input type='submit' value='Update'>"
    "</form>"
 "<div id='prg'>progress: 0%</div>"
 "<script>"
  "('form').submit(function(e)""e.preventDefault();""var form =('#upload_form')
   [0];"


  "var data = new FormData(form);"
  "
   .ajax(""url :' /update',""type :' POST',""data : data,"čontentType : false,""processData : false,""xhr : f
   ('#prg').html('progress: ' + Math.round(per*100) + '%');"









  "}"
  "}, false);"
  "return xhr;"
  "},"
```

```
  "success:function(d, s) {"
  "console.log('success!')"
 "},"
 "error: function (a, b, c) {"
 "}"
 "});"
 "});"
 "</script>";


  BLYNK_WRITE(V1)
{
}

  BLYNK_WRITE(V2)
{
  DateTime now = rtc.now();
  luc = param.asInt();
  Serial.print("Relay 1 je sedaj ");
  Serial.println(luc);
  terminal.print("Relay 1 je sedaj ");
  terminal.println(luc);
  terminal.flush();
  digitalWrite(luc_pin, luc);
  if(millis() < 10000){
    terminal.println("luc se ni prizgala ker je od zagona minilo
   premalo casa");
    Blynk.virtualWrite(V2, 0);
    digitalWrite(luc_pin, 0);
    luc = 0;
    return;
  }
  else if(luc == 1){
   h_on = now.hour();
   min_on = now.minute();
   sec_on = now.second();
  }
  else if(luc == 0){
    luc_stetje();
  }
}
  BLYNK_WRITE(V3)
{
  grelec = param.asInt();
```

```
235   String t = "Grelec je sedaj ";
236   String s = t + grelec;
237   terminal.println(s);
238   terminal.flush();
239   digitalWrite(grelec_pin, grelec);
240 }
241   BLYNK_WRITE(V4)
242 {
243   filter = param.asInt();
244   String t = "Filter je sedaj ";
245   String s = t + filter;
246   terminal.println(s);
247   terminal.flush();
248   digitalWrite(filter_pin, filter);
249 }
250   BLYNK_WRITE(V5)
251 {
252   programing_mode = param.asInt();
253   if(programing_mode = 1){
254 
255     Serial.println("Update nacin prizgan, zaganjanje serverja");
256     terminal.println("Zaganjanje serverja, Blynk se bo disconnectal
      ");
257     terminal.print("IP number assigned by DHCP is ");
258     terminal.println(WiFi.localIP());
259     terminal.flush();
260     delay(500);
261 
262     timerLuc.disable(timerLucId);
263     timerDHT.disable(timerDHTId);
264     timerReconnect.disable(timerReconnectId);
265     timerDallas.disable(timerDallasId);
266 
267     Blynk.disconnect();
268     if (WiFi.status() != WL_CONNECTED) {
269      WiFi.disconnect();
270      Serial.println("Connecting to WiFi...");
271      WiFi.mode(WIFI_AP_STA);
272      WiFi.begin(ssid, password);
273      delay(1000);
274      if(WiFi.status() != WL_CONNECTED) {
275       Serial.println("Not Reconnected");
276      } else {
277        Serial.println("Reconnected");
```

```
278       }
279     }
280         if (!MDNS.begin(host)) { //http://esp32.local
281       Serial.println("Error setting up MDNS responder!");
282       while (1) {
283         delay(1000);
284       }
285     }
286     Serial.println("mDNS responder started");
287     /*return index page which is stored in serverIndex */
288     server.on("/", HTTP_GET, []() {
289       server.sendHeader("Connection", "close");
290       server.send(200, "text/html", loginIndex);
291     });
292     server.on("/serverIndex", HTTP_GET, []() {
293       server.sendHeader("Connection", "close");
294       server.send(200, "text/html", serverIndex);
295     });
296     /*handling uploading firmware file */
297     server.on("/update", HTTP_POST, []() {
298       server.sendHeader("Connection", "close");
299       server.send(200, "text/plain", (Update.hasError()) ? "FAIL" : "
      OK");
300       ESP.restart();
301     }, []() {
302       HTTPUpload& upload = server.upload();
303       if (upload.status == UPLOAD_FILE_START) {
304         Serial.printf("Update: %s\n", upload.filename.c_str());
305         if (!Update.begin(UPDATE_SIZE_UNKNOWN)) { //start with max
      available size
306           Update.printError(Serial);
307         }
308       } else if (upload.status == UPLOAD_FILE_WRITE) {
309         /* flashing firmware to ESP*/
310         if (Update.write(upload.buf, upload.currentSize) != upload.
      currentSize) {
311           Update.printError(Serial);
312         }
313       } else if (upload.status == UPLOAD_FILE_END) {
314         if (Update.end(true)) { //true to set the size to the current
       progress
315           Serial.printf("Update Success: %u\nRebooting...\n", upload.
      totalSize);
316         } else {
```

```
317         Update.printError(Serial);
318       }
319     }
320   });
321   server.begin();
322   // server je pripravljen 1 minuto, ce v tem casu ne prejme updata
      , se bo esp32 samo ponovno zagnal
323   for(int i = 0; i < 60000; i++){
324     server.handleClient();
325     delay(1);
326   }
327
328   Serial.println("esp ni prejel updata, ponovni zagon");
329
330   abort();
331
332
333   }
334 }
335 //spreminjanje raznih nastavitev preko blynka
336 BLYNK_WRITE(V20){
337   prizig_h = param.asInt();
338   String t = "prizig_h je sedaj ";
339   String s = t + prizig_h;
340   terminal.println(s);
341   terminal.flush();
342 }
343 BLYNK_WRITE(V21){
344   prizig_min = param.asInt();
345   String t = "prizig_min je sedaj ";
346   String s = t + prizig_min;
347   terminal.println(s);
348   terminal.flush();
349 }
350 BLYNK_WRITE(V22){
351   maks_time_h = param.asInt();
352   String t = "maks_time_h je sedaj ";
353   String s = t + maks_time_h;
354   terminal.println(s);
355   terminal.flush();
356 }
357 BLYNK_WRITE(V23){
358   maks_time_min = param.asInt();
359   String t = "maks_time_min je sedaj ";
```

```
360   String s = t + maks_time_min;
361   terminal.println(s);
362   terminal.flush();
363 }
364 BLYNK_WRITE(V24){
365   luc_timer = param.asInt();
366   String t = "luc_timer je sedaj ";
367   String s = t + luc_timer;
368   terminal.println(s);
369   terminal.flush();
370 }
371 BLYNK_WRITE(V25){
372   maks_voda_temp = param.asInt();
373   String t = "maks_voda_temp je sedaj ";
374   String s = t + maks_voda_temp;
375   terminal.println(s);
376   terminal.flush();
377 }
378
379 void sendCallback(SendStatus info);
380
381 void sendCallback(SendStatus msg) {
382   // Print the current status
383   Serial.println(msg.info());
384
385   // Do something when complete
386   if (msg.success()) {
387     Serial.println("----------------");
388   }
389 }
390
391
392 void DHT_meritev(){
393   //pridobi temperaturo in vlago od DHT22 senzorja
394   TempAndHumidity newValues = dht.getTempAndHumidity();
395   //preveri ce je bilo branje uspesno in poslje podatke blynku, ce
      branje ni bilo uspesno poslje opozorilo na terminal
396   if(String(newValues.temperature) != "nan"){
397   Blynk.virtualWrite(V10, newValues.temperature);
398   Blynk.virtualWrite(V11, newValues.humidity);
399   }
400   else{ terminal.println("dht senzor ne deluje!"); }
401 }
402
```

```
403 void digitalClockDisplay()
404 {
405   // pridobi cas od NTP serverja in ce ga je sprejel posodobi cas
       na RTC modulu in napise trenutni cas
406   getNtpTime();
407   Serial.print(hour());
408   terminal.print(hour());
409   printDigits(minute());
410   printDigits(second());
411   Serial.print(" ");
412   Serial.print(day());
413   Serial.print(".");
414   Serial.print(month());
415   Serial.print(".");
416   Serial.print(year());
417   Serial.println();
418   int leto = year();
419   int dan = day();
420   int mesec = month();
421   int ura = hour();
422   int minut = minute();
423   int sekunde = second();
424   Serial.println(ura);
425   Serial.println(minut);
426   if(leto != 1970){
427   rtc.adjust(DateTime(leto, mesec, dan, ura, minut, sekunde));
428   //rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
429   Serial.println("cas posodobljen, rtc nastavljen");
430   terminal.println("cas posodobljen, rtc nastavljen");
431   terminal.flush();
432   }
433 }
434  void printDigits(int digits)
435 {
436   // del digitalClockDisplay-a, cas preuredi v lepsi format(pred
       stevila s eno stevko doda 0)
437   Serial.print(":");
438   terminal.print(":");
439   if (digits < 10)
440     Serial.print('0');
441     terminal.print('0');
442   Serial.print(digits);
443   terminal.print(digits);
444   terminal.flush();
```

```
445 }
446
447 void rtc_steka(){
448   //v primeru, da bi bil RTC nastavljen na nemogoc cas (vec kot je
      ur v dnevu), popravi cas.
449   DateTime now = rtc.now();
450   Serial.println(now.hour(), DEC);
451   if(now.hour() > 23){
452     Serial.println("rtc_steka resetira rtc....");
453     terminal.println("rtc_steka resetira rtc....");
454     terminal.flush();
455     digitalClockDisplay();
456   }
457 }
458
459 void luc_stuff(){
460   // upravlja z lucjo. preveri tudi temperaturo RTC senzorja in ce
      je previsoka poslje opozorilo resetira tudi casovne
      spremneljivke ob polnoci
461   rtc_steka();
462   DateTime now = rtc.now();
463
464   if(rtc.getTemperature() >= 40){
465     Serial.println("temperatura je previsoka, nekaj se pregreva!");
466     terminal.println("temperatura je previsoka, nekaj se pregreva!"
      );
467     terminal.flush();
468   }
469
470   if(now.hour() == 00 && now.minute() == 01 && (total_h_turned_on
      != 0 || total_min_turned_on != 0 || total_sec_turned_on != 0)){
471     total_h_turned_on = 0;
472     total_min_turned_on = 0;
473     total_sec_turned_on = 0;
474     maks_time_dosezen = false;
475     terminal.println("casovne spremenljivke resetirane");
476     terminal.flush();
477   }
478
479   if(now.hour() == prizig_h && now.minute() == prizig_min &&
      maks_time_dosezen == false && luc != 1){
480
481     digitalWrite(luc_pin, HIGH);
482     Blynk.virtualWrite(V2, HIGH);
```

```
483     sec_on = now.second();
484     min_on = now.minute();
485     h_on = now.hour();
486     luc = 1;
487 
488     Serial.println("luc prizgana po urniku");
489     terminal.println("luc prizgana po urniku");
490     terminal.flush();
491   }
492 
493    if(luc == 1){
494    luc_stetje();
495    }
496 }
497 
498 void reconnectBlynk() {
499    //ce izgubi wifi povezavo se poskusa povezati nazaj
500    if (WiFi.status() != WL_CONNECTED) {
501      WiFi.disconnect();
502      Serial.println("Povezovanje v WiFi omre[U+FFFD]je...");
503      WiFi.mode(WIFI_AP_STA);
504      WiFi.begin(ssid, password);
505      delay(1000);
506      if(WiFi.status() != WL_CONNECTED) {
507       Serial.println("Wifi povezava ni ponovno vzpostavljena");
508      } else {
509        Serial.println("WiFi povezava ponovno vzpostavljena");
510      }
511    }
512 }
513 
514 void ds18b20_meritev(){
515   // prebere temperaturo DS18B20 senzorja, jo poslje blynku in
       glede na temperaturo upravlja grelec
516   sensors.requestTemperatures();
517   float temperatura_vode = sensors.getTempCByIndex(0);
518   Serial.print(temperatura_vode);
519   Serial.println("[U+FFFD]C");
520   if(temperatura_vode != -127 && temperatura_vode != 85){
521   Blynk.virtualWrite(V12, temperatura_vode);
522   if(temperatura_vode >= maks_voda_temp && grelec == 1){
523     Serial.println("temperatura je prevelika, ugasanje grelca");
524     terminal.println("temperatura je prevelika, ugasanje grelca");
525     terminal.flush();
```

```
526     smtpData.setLogin(smtpServer, smtpServerPort,
        emailSenderAccount, emailSenderPassword);
527     smtpData.setSender("Akvarij", emailSenderAccount);
528     smtpData.setPriority("1");
529     smtpData.setSubject("Akvarij se pregreva!");
530     String t = "<div style=\"color:#2f4468;\"><h1>V akvariju je
        bila izmerjena neobicajno visoka temperatura. <br>Temperatura je
        : ";
531     String b = "°C <br>Grelec je bil ugasnjen</h1><p>- Avtomatsko
        poslal kontrolnik akvarija</p></div>";
532     String s = t + temperatura_vode + b;
533     smtpData.setMessage(s, true);
534     smtpData.addRecipient(emailRecipient);
535     smtpData.setSendCallback(sendCallback);
536   if (!MailClient.sendMail(smtpData)){
537     Serial.println("Error sending Email, " + MailClient.
        smtpErrorReason());
538     terminal.println("Error sending Email, " + MailClient.
        smtpErrorReason());
539     terminal.flush();
540   }
541   smtpData.empty();
542   t = "";
543   b = "";
544   s = "";
545     digitalWrite(grelec_pin, LOW);
546     Blynk.virtualWrite(V3, LOW);
547     grelec = 0;
548   }
549   else if(grelec == 0 && temperatura_vode <= (maks_voda_temp - 0.5)
        ){
550     Serial.println("temperatura je padla, ponovno priziganje grelca
        ");
551     terminal.println("temperatura je padla, ponovno priziganje
        grelca");
552     terminal.flush();
553     digitalWrite(grelec_pin, HIGH);
554     Blynk.virtualWrite(V3, 1);
555     grelec = 1;
556   }
557   }
558   else{
559     Serial.println("Napaka pri branju temperature vode");
560     terminal.println("Napaka pri branju temperature vode");
```

```
561
562     DateTime now = rtc.now();
563     if(ura_opozorjeno != now.hour()){
564      ura_opozorjeno = now.hour();
565      terminal.flush();
566      smtpData.setLogin(smtpServer, smtpServerPort,
        emailSenderAccount, emailSenderPassword);
567      smtpData.setSender("Akvarij", emailSenderAccount);
568      smtpData.setPriority("1");
569      smtpData.setSubject("Senzor v akvariju ne deluje");
570      smtpData.setMessage("<div style=\"color:#2f4468;\"><h1>
        Kontrolnik ni mogel prebrati temperature vode v akvariju.</h1> <
        p>- Avtomatsko poslal kontrolnik akvarija</p></div>", true);
571      smtpData.addRecipient(emailRecipient);
572      smtpData.setSendCallback(sendCallback);
573      if (!MailClient.sendMail(smtpData)){
574        Serial.println("Error sending Email, " + MailClient.
        smtpErrorReason());
575        terminal.println("Error sending Email, " + MailClient.
        smtpErrorReason());
576        terminal.flush();
577      }
578      smtpData.empty();
579      }
580    }
581 }
582
583 void luc_stetje(){
584     // izracina koliko casa je bila luc prizgana in to poslje
585     DateTime now = rtc.now();
586     if(millis() > 10000){
587
588      if(luc == 0){
589      Serial.println("Luc je bila ugasnjena");
590      terminal.println("Luc je bila ugasnjena.");
591      terminal.flush();
592       int t_total_h_turned_on = total_h_turned_on + h_turned_on;
593       int t_total_min_turned_on = total_min_turned_on +
        min_turned_on;
594       int t_total_sec_turned_on = total_sec_turned_on +
        sec_turned_on;
595       if(t_total_sec_turned_on >= 60){
596          t_total_min_turned_on ++;
597          t_total_sec_turned_on = total_sec_turned_on - 60;
```

```
598       }
599     if(t_total_min_turned_on >= 60){
600        t_total_h_turned_on ++;
601        t_total_min_turned_on = total_min_turned_on - 60;
602       }
603     if(t_total_sec_turned_on < 0){
604      t_total_min_turned_on --;
605      t_total_sec_turned_on = sec_turned_on + 60;
606     }
607     if(t_total_min_turned_on < 0){
608       h_turned_on --;
609       min_turned_on = min_turned_on + 60;
610     }
611     if(t_total_h_turned_on < 0){
612       t_total_h_turned_on --;
613       t_total_min_turned_on = min_turned_on + 60;
614     }
615     terminal.println("skupaj je bila prizgana:");
616     terminal.print(t_total_h_turned_on);
617     terminal.print(" ur  ");
618     terminal.print(t_total_min_turned_on);
619     terminal.print(" min  ");
620     terminal.print(t_total_sec_turned_on);
621     terminal.println(" sec  ");
622     terminal.flush();
623     Serial.println("skupaj je bila prizgana:");
624     Serial.print(t_total_h_turned_on);
625     Serial.print(" ur  ");
626     Serial.print(t_total_min_turned_on);
627     Serial.print(" min  ");
628     Serial.print(t_total_sec_turned_on);
629     Serial.println(" sec  ");
630     total_h_turned_on = t_total_h_turned_on;
631     total_min_turned_on = t_total_min_turned_on;
632     total_sec_turned_on = t_total_sec_turned_on;
633   }
634   else{
635    sec_turned_on = now.second() - sec_on;
636    min_turned_on = now.minute() - min_on;
637    h_turned_on = now.hour() - h_on;
638
639    if(sec_turned_on < 0){
640     min_turned_on --;
641     sec_turned_on = sec_turned_on + 60;
```

```
642   }
643   if(min_turned_on < 0){
644     h_turned_on --;
645     min_turned_on = min_turned_on + 60;
646   }
647   if(h_turned_on < 0){
648     h_turned_on --;
649     min_turned_on = min_turned_on + 60;
650   }
651   if(sec_turned_on < 0){
652    min_turned_on --;
653    sec_turned_on = sec_turned_on + 60;
654   }
655   if(min_turned_on < 0){
656     h_turned_on --;
657     min_turned_on = min_turned_on + 60;
658   }
659   if(h_turned_on < 0){
660     h_turned_on --;
661     min_turned_on = min_turned_on + 60;
662   }
663
664   Serial.println("luc je bila prizgana:");
665   Serial.print(h_turned_on);
666   Serial.print(" ur  ");
667   Serial.print(min_turned_on);
668   Serial.print(" min  ");
669   Serial.print(sec_turned_on);
670   Serial.println(" sec  ");
671   Serial.flush();
672   terminal.println("luc je bila prizgana:");
673   terminal.print(h_turned_on);
674   terminal.print(" ur  ");
675   terminal.print(min_turned_on);
676   terminal.print(" min  ");
677   terminal.print(sec_turned_on);
678   terminal.println(" sec  ");
679   terminal.flush();
680   int t_total_h_turned_on = total_h_turned_on + h_turned_on;
681   int t_total_min_turned_on = total_min_turned_on + min_turned_on;
682   int t_total_sec_turned_on = total_sec_turned_on + sec_turned_on;
683   if(t_total_sec_turned_on >= 60){
684      t_total_min_turned_on ++;
685      t_total_sec_turned_on = total_sec_turned_on - 60;
```

```
686       }
687     if(t_total_min_turned_on >= 60){
688        t_total_h_turned_on ++;
689        t_total_min_turned_on = total_min_turned_on - 60;
690       }
691     if(t_total_sec_turned_on < 0){
692      t_total_min_turned_on --;
693      t_total_sec_turned_on = sec_turned_on + 60;
694     }
695     if(t_total_min_turned_on < 0){
696       h_turned_on --;
697       min_turned_on = min_turned_on + 60;
698     }
699     if(t_total_h_turned_on < 0){
700       t_total_h_turned_on --;
701       t_total_min_turned_on = min_turned_on + 60;
702     }
703     terminal.println("skupaj je bila prizgana:");
704     terminal.print(t_total_h_turned_on);
705     terminal.print(" ur  ");
706     terminal.print(t_total_min_turned_on);
707     terminal.print(" min  ");
708     terminal.print(t_total_sec_turned_on);
709     terminal.println(" sec  ");
710     terminal.flush();
711     Serial.println("skupaj je bila prizgana:");
712     Serial.print(t_total_h_turned_on);
713     Serial.print(" ur  ");
714     Serial.print(t_total_min_turned_on);
715     Serial.print(" min  ");
716     Serial.print(t_total_sec_turned_on);
717     Serial.println(" sec  ");
718
719     if(t_total_h_turned_on >= maks_time_h && t_total_min_turned_on
      >= maks_time_min && maks_time_dosezen == false){
720        digitalWrite(luc_pin, LOW);
721        Blynk.virtualWrite(V2, LOW);
722        luc = 0;
723        Serial.println("Luc je presegla maksimalni cas. Sedaj je
     ugasnjena.");
724        terminal.println("Luc je presegla maksimalni cas. Sedaj je
     ugasnjena.");
725        terminal.flush();
726        total_h_turned_on = t_total_h_turned_on;
```

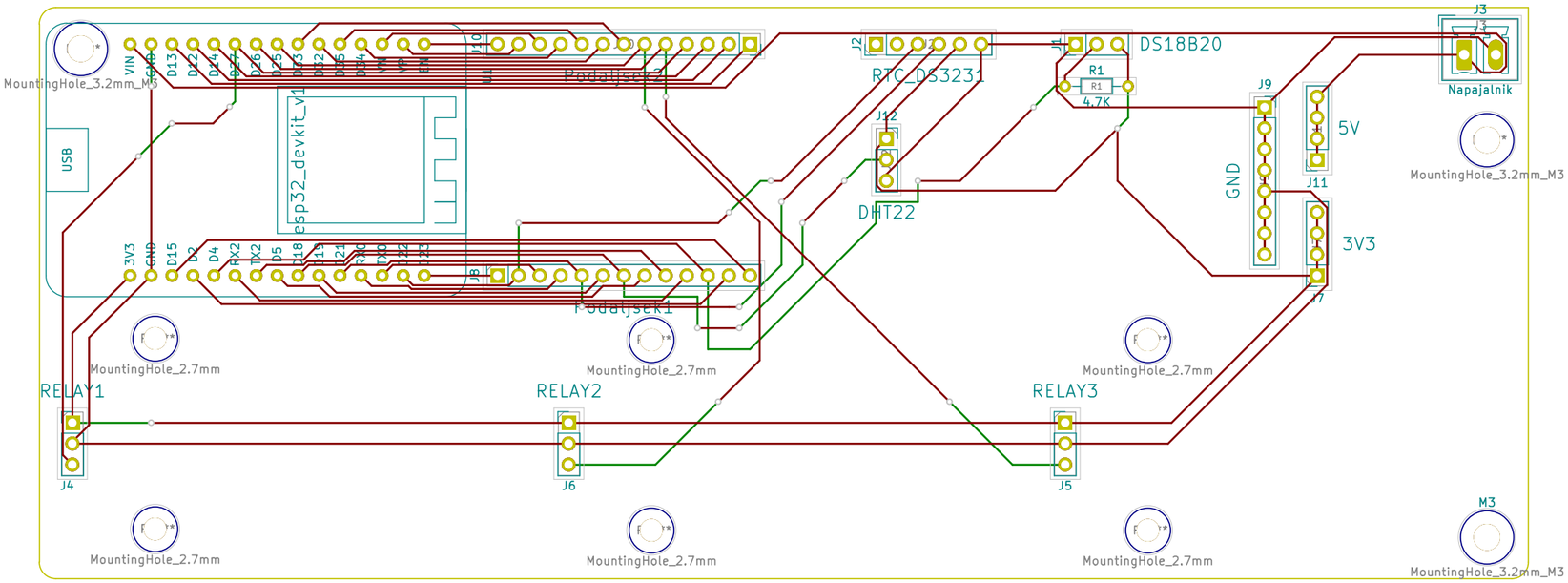
```
727       total_min_turned_on = t_total_min_turned_on;
728       total_sec_turned_on = t_total_sec_turned_on;
729       maks_time_dosezen = true;
730     }
731     else if(maks_time_dosezen == true && min_turned_on >= luc_timer
    ){
732       digitalWrite(luc_pin, LOW);
733       Blynk.virtualWrite(V2, LOW);
734       luc = 0;
735       total_h_turned_on = t_total_h_turned_on;
736       total_min_turned_on = t_total_min_turned_on;
737       total_sec_turned_on = t_total_sec_turned_on;
738     }
739    }
740    }
741 }
742
743 void setup() {
744   //nastavitev za serijsko komnikacijo z racinalnikom
745   Serial.begin(9600);
746   //nastavitev razlicnih pinov
747   pinMode(luc_pin, OUTPUT);
748   pinMode(grelec_pin, OUTPUT);
749   pinMode(filter_pin, OUTPUT);
750   pinMode(dhtPin, INPUT);
751   // poveze se na wifi
752   Serial.print("Connecting to ");
753   Serial.println(ssid);
754   WiFi.begin(ssid, password);
755   // caka dokler se wifi ne poveze oziroma mine 5 sekund
756   int i;
757   while (WiFi.status() != WL_CONNECTED && i < 10){
758     delay(500);
759     i++;
760     Serial.print(".");
761   }
762
763   // poveze se na NTP server
764   Serial.print("IP number assigned by DHCP is ");
765   Serial.println(WiFi.localIP());
766   Serial.println("Starting UDP");
767   Udp.begin(localPort);
768   Serial.print("Local port: ");
769   Serial.println("waiting for sync");
```

```
770  setSyncProvider(getNtpTime);
771
772  //poveze se na blynk
773  Blynk.config(auth);
774  terminal.print("IP number assigned by DHCP is ");
775  terminal.println(WiFi.localIP());
776  terminal.flush();
777
778  // pripravi DHT22 in DS18B20
779  dht.setup(dhtPin, DHTesp::DHT22);
780  sensors.begin(); //ds18b20
781
782   // pripravi RTC, ce ga ne najde poslje sporocilo in se resetira
783  rtc.begin();
784   if (! rtc.begin()) {
785    Serial.println("Couldn't find RTC");
786    terminal.println("Ne najdem RTCja! abort");
787    terminal.flush();
788    Serial.flush();
789    abort();
790  }
791
792  digitalClockDisplay();
793
794  Blynk.syncVirtual(V1);
795  Blynk.run();
796  delay(300);
797  Blynk.syncVirtual(V2);
798  Blynk.run();
799  delay(300);
800  Blynk.syncVirtual(V3);
801  Blynk.run();
802  delay(300);
803  Blynk.syncVirtual(V4);
804  Blynk.run();
805  delay(300);
806  Blynk.syncVirtual(V20);
807  Blynk.run();
808  delay(300);
809  Blynk.syncVirtual(V21);
810  Blynk.run();
811  delay(300);
812  Blynk.syncVirtual(V22);
813  Blynk.run();
```

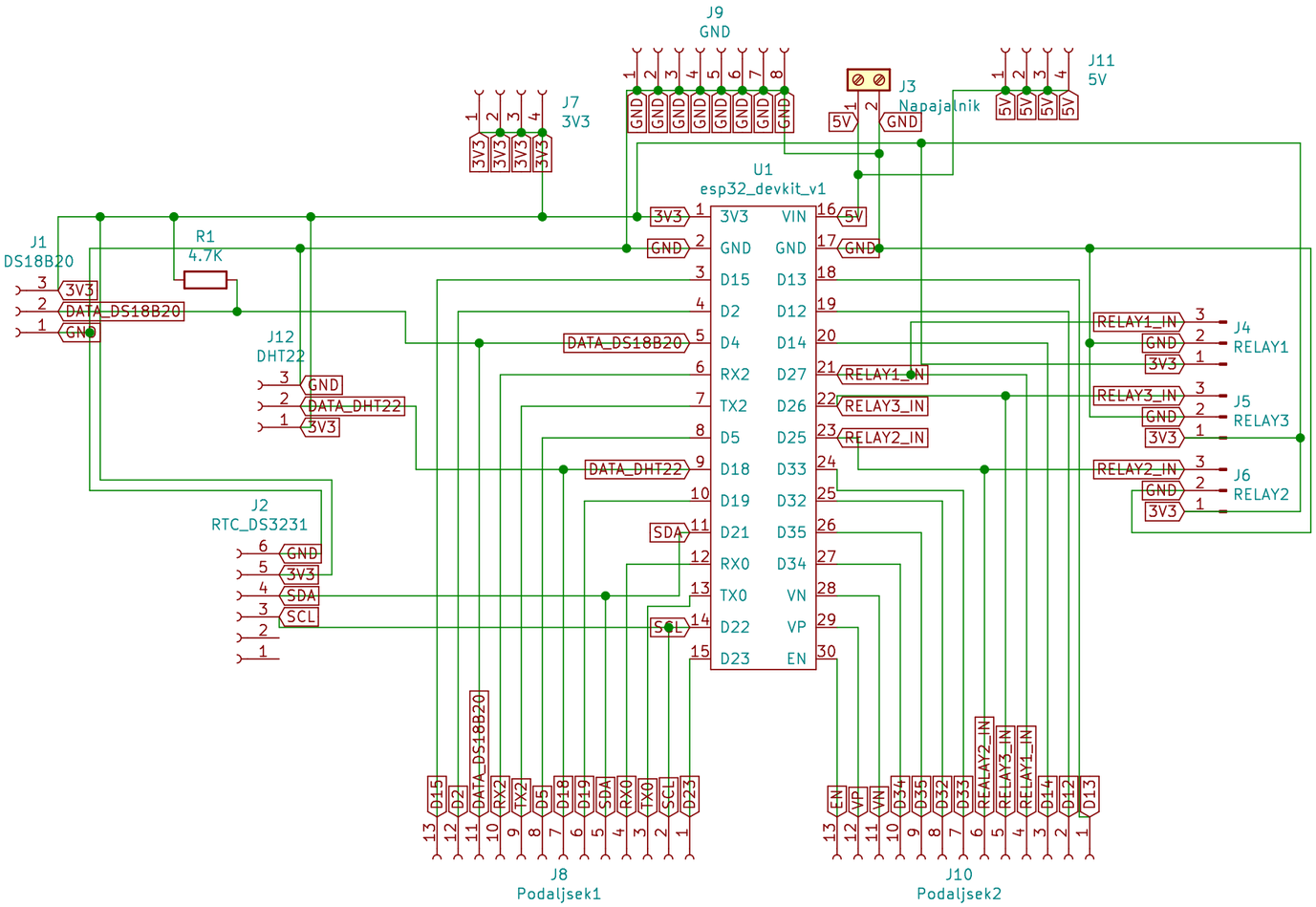
```
814   delay(300);
815   Blynk.syncVirtual(V23);
816   Blynk.run();
817   delay(300);
818   Blynk.syncVirtual(V24);
819   Blynk.run();
820   delay(300);
821   Blynk.syncVirtual(V25);
822   Blynk.run();
823   delay(300);
824
825   // zacne timerje
826   timerLucId = timerLuc.setInterval(1000L, luc_stuff);
827   timerDHTId = timerDHT.setInterval(2000L, DHT_meritev);
828   timerDallasId = timerDallas.setInterval(5000L, ds18b20_meritev);
829   timerReconnectId = timerReconnect.setInterval(30000L,
       reconnectBlynk);
830
831   // resetira koliko casa je bila luc prizgana, ce se je slucajno
       prizgala ob zagonu
832   total_h_turned_on = 0;
833   total_min_turned_on = 0;
834   total_sec_turned_on = 0;
835   maks_time_dosezen = false;
836
837 }
838
839 void loop() {
840   // izvaja Blynk
841   Blynk.run();
842   // in vse timerje
843   timerDallas.run();
844   timerLuc.run();
845   timerDHT.run();
846   timerReconnect.run();
847 }
848
849 // Za povezavo z NTP serverjem
850 const int NTP_PACKET_SIZE = 48; // NTP time is in the first 48
       bytes of message
851 byte packetBuffer[NTP_PACKET_SIZE]; //buffer to hold incoming &
       outgoing packets
852
853 time_t getNtpTime()
```

```
{
  IPAddress ntpServerIP; // NTP server's ip address

  while (Udp.parsePacket() > 0) ; // discard any previously
   received packets
  Serial.println("Transmit NTP Request");
  WiFi.hostByName(ntpServerName, ntpServerIP);
  Serial.print(ntpServerName);
  Serial.print(": ");
  Serial.println(ntpServerIP);
  sendNTPpacket(ntpServerIP);
  uint32_t beginWait = millis();
  while (millis() - beginWait < 1500) {
    int size = Udp.parsePacket();
    if (size >= NTP_PACKET_SIZE) {
      Serial.println("Receive NTP Response");
      Udp.read(packetBuffer, NTP_PACKET_SIZE);  // read packet into
    the buffer
      unsigned long secsSince1900;
      // convert four bytes starting at location 40 to a long
   integer
      secsSince1900 =  (unsigned long)packetBuffer[40] << 24;
      secsSince1900 |= (unsigned long)packetBuffer[41] << 16;
      secsSince1900 |= (unsigned long)packetBuffer[42] << 8;
      secsSince1900 |= (unsigned long)packetBuffer[43];
      return secsSince1900 - 2208988800UL + timeZone *
   SECS_PER_HOUR;
    }
  }
  Serial.println("No NTP Response :-(");
  terminal.println("No NTP Response :-(");
  terminal.flush();
  return 0; // return 0 if unable to get the time
}

// send an NTP request to the time server at the given address
void sendNTPpacket(IPAddress &address)
{
  // set all bytes in the buffer to 0
  memset(packetBuffer, 0, NTP_PACKET_SIZE);
  // Initialize values needed to form NTP request
  // (see URL above for details on the packets)
  packetBuffer[0] = 0b11100011;   // LI, Version, Mode
  packetBuffer[1] = 0;     // Stratum, or type of clock
```

```
  packetBuffer[2] = 6;      // Polling Interval
  packetBuffer[3] = 0xEC;  // Peer Clock Precision
  // 8 bytes of zero for Root Delay & Root Dispersion
  packetBuffer[12] = 49;
  packetBuffer[13] = 0x4E;
  packetBuffer[14] = 49;
  packetBuffer[15] = 52;
  // all NTP fields have been given values, now
  // you can send a packet requesting a timestamp:
  Udp.beginPacket(address, 123); //NTP requests are to port 123
  Udp.write(packetBuffer, NTP_PACKET_SIZE);
  Udp.endPacket();
}
```



Sheet:		
File: esp32_akvarij.kicad_pcb		
Title: Načrt tiskanega vezja pametnega kontrolnika akvarija		
Size: A4	Date:	**Rev:**
KiCad E.D.A. kicad (5.1.8)-1		Id: 1/1



Sheet: /
File: esp32_akvarij.sch

Title: Shema za tiskano vezje pametnega kontrolnika akvarija

Size: A4	Date:	Rev:
KiCad E.D.A. kicad (5.1.8)-1		Id: 1/1