Sustav za navodnjavanje upravljan Raspberry računalom

Danko, Sven

Sveučilište
Sjever

Završni rad br. 484/EL/2021

Sustav za navodnjavanje upravljan Raspberry računalom

Sven Danko, 2036/336

Varaždin, lipanj 2021. godine

Sveučilište
Sjever

Odjel za elektrotehniku

Završni rad br. 484/EL/2021

Sustav za navodnjavanje upravljan Raspberry računalom

Student

Sven Danko, 2036/336

Mentor

Miroslav Horvatić, dipl.ing.

Varaždin, lipanj 2021. godine

Sažetak

Ideja ovog završnog rada bila je izraditi sustav za automatsko navodnjavanje koji se može koristiti za razne primjene, od velikih poljoprivrednih sustava pa sve do malih kućnih biljka.

Glavni dio završnog rada čini malo računalo Raspberry Pi. Navedeno računalo temelj je cijelog projekta, te se na samom Raspberry Pi-u odvija cijeli proces automatizacije. Raspberry Pi prati stanje senzora vlage kako bi utvrdio da li je potrebno uključiti pumpu za navodnjavanje. Senzor vlage mjeri otpor između dvije elektrode, te daje signal ovisno o otporu koji je unaprijed podešen. U opisanom sustavu koristi se i senzor temperature kako bi se mogla nadgledati temperatura same vode.

Kako bi korisnik mogao dobiti povratne informacije, izrađena je i aplikacija za dobivanje povratnih informacija. U aplikaciji je prikazano ono što senzori očitavaju - kolika je trenutna temperatura vode, da li je trenutno u zemlji prisutno dovoljno vlage te da li sustav trenutno navodnjava ili ne. Za izradu aplikacije korišten je React Native, programski okvir za izradu mobilnih aplikacija koji radi na bazi programskog jezika JavaScript.

Ključne riječi: Automatsko upravljanje, pumpa, Raspberry Pi, React Native, senzor

Abstract

The idea of this final thesis was to make an automated irrigation system wich would serve for various applications, from big agricultural systems to small house plants.

The main part of this final thesis is made up of the Raspberry Pi microcomputer. It is the foundation of the whole project, and the whole automation process. The Raspberry Pi tracks the state of the moisture sensor to determine if it is necessary to activate the irrigation pump. The moisture sensor measures the resistance between two electrodes, and it gives a signal based on the resistance wich is adjusted in advance. In the described system a temperature sensor is also used so that the water temperature can also be monitored.

An application is made so that the end user can have feedback into the process. The application shows what the sensors measure – the current water temperature, if there is enough water present in the soil and if the system is being irrigated or not. The React Native framework was used to make the application, wich is used to make mobile apps based on the programming language JavaScript.

Key words: Automated irrigation, pump, Raspberry Pi, React Native, sensor

Popis korištenih kratica

RPi	Raspberry Pi
GPIO	General Purpose Input/Output
RN	React Native

Sadržaj

1. Uvod

U današnjem užurbanom vremenu, često se događa da ljudi nemaju vremena za svoje hobije. Posao i obaveze zauzimaju većinu ljudskog vremena te dolazi pitanje, kako se brinuti za stvari za koje ljudi nemaju vremena? Odgovor je jednostavan: automatsko upravljanje.

Automatsko upravljanje daje mogućnost da monotone zadatke koji ne zahtijevaju ljudski nadzor obrađuje elektronika. Ovo je vidljivo u mnogim industrijskim postrojenjima - ponajviše u automobilskoj industriji u kojoj su velik dio posla preuzele robotske ruke. Razvojem tehnologije, sustavi automatskog upravljanja vide se i u kućnim primjenama - sustavi pametnih kuća, robotski usisavači i slično.

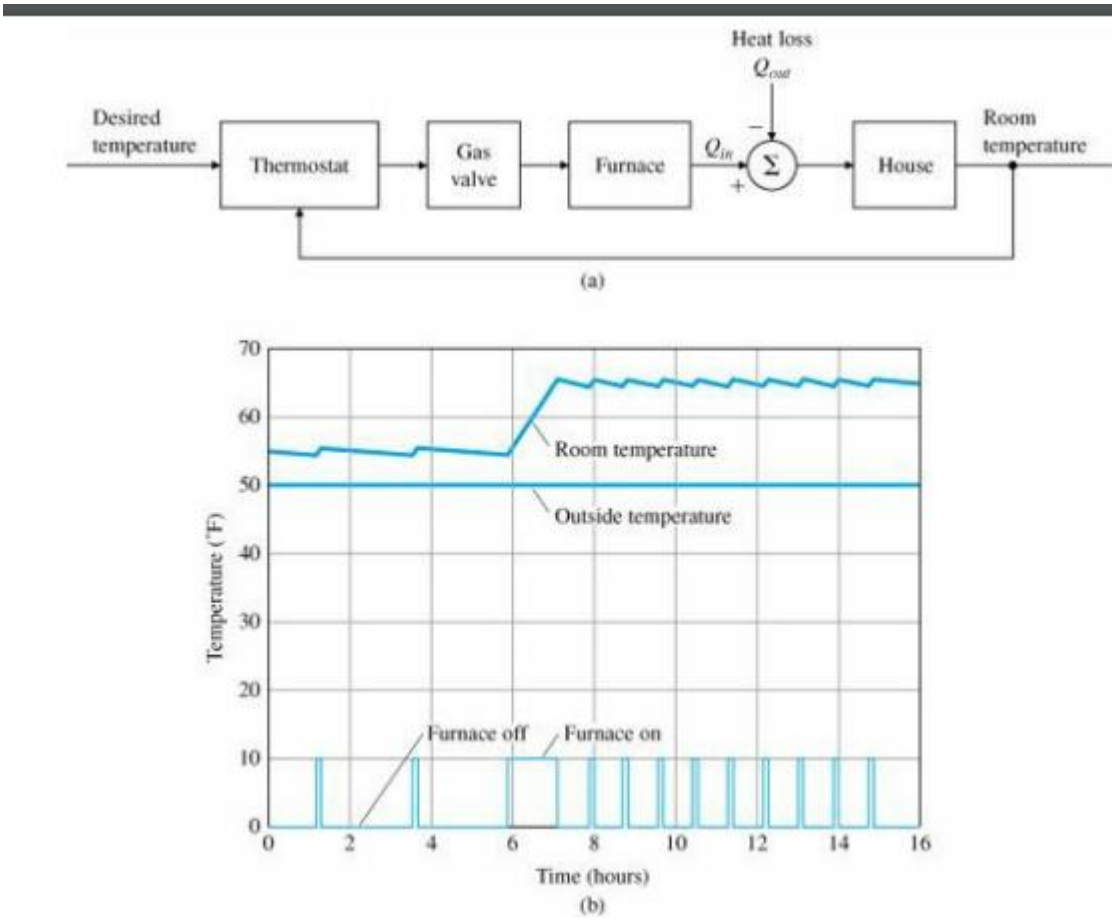
Kako bi prikazali mogućnost izrade relativno jednostavnog i praktičnog sustava automatskog upravljanja za kućnu upotrebu, napravljen je sustav automatskog navodnjavanja kućnih biljka koji sadrži korisničko sučelje za prikaz informacija. Sustav je osmišljen kao demonstracija kako se i najbanalnije stvari ljudskog života mogu olakšati primjenom sustava automatskog upravljanja. Prikazani sustav može se proširiti te primijeniti i na veća postrojenja i sustave navodnjavanja.

Kako bi se mogao realizirati kvalitetan automatizirani sustav navodnjavanja u sljedećem poglavlju će se analizirati problemi koje sustav mora riješiti te odrediti struktura samog sustava. Nakon prikaza strukture sustava može se pristupiti praktičnoj izvedbi koje će biti opisano u poglavlju 3. U poglavlju 3 opisat će se princip rada svakog pojedinog elementa sustava navodnjavanja te njihovo međusobno povezivanje u funkcionalnu cjelinu. Na kraju će se ispitati rad sustava te će dobiveni podaci biti obrađeni, na temelju čega će se izvesti zaključak o radu sustava.

2. Sustav automatskog navodnjavanja

Automatsko upravljanje je disciplina koja se bavi teorijskom i praktičnom analizom i sintezom automatiziranih sustava. Automatsko upravljanje daje oruđe za pravilno i efikasno izrađivanje automatiziranih sustava. Automatsko upravljanje promatra dvije osnovne vrste sustava: sustave bez povratne veze te sustave s povratnom vezom.

Sustavi bez povratne veze su sustavi koji rade bez ikakve povratne informacije o stanju procesa i vanjskih uvjeta vezanih uz sam proces. Najčešće se sastoje od dijelova koji svoj posao rade cijelo vrijeme jednako, bez obzira na vanjske uvjete i čimbenike.



Slika 2.1 - Blok dijagram sustava s povratnom vezom [1]

Sustavi s povratnom vezom su sustavi koji upravljanje procesom obavljaju ovisno o vanjskim uvjetima. Najbolji primjer sustava s povratnom vezom jest termostat koji nadzire temperaturu u kućanstvu te tu temperaturu mijenjaju prema zadanoj referentnoj vrijednosti. Nakon procesa zagrijavanja on nadzire stvarnu vanjsku temperaturu i uspoređuje je sa željenom temperaturom. Regulator tada računa razliku između željene i stvarne temperature te može podesiti rad izvršnih

elementa po potrebi. [1] Na slici 2.1 može se vidjeti simbolički prikaz takvog procesa. Sustav automatskog upravljanje se najčešće sastoji od četiri glavna elementa:

- Mjerni elementi
- Izvršni elementi
- Regulator
- Proces.

Navodnjavanje je proces dovođenja vode iz spremnika tekućine do bilo koje biljke, pri čemu se koristi sustav cijevi, kanala ili bilo kojih ljudski napravljenih sredstava. Svrha navodnjavanja je da se izbjegnu nepredvidljivi vremenski uvjeti te da se uspostavi pouzdan sustav opskrbe biljke vodom. Sustavi navodnjavanja koriste se već od davnih vremena, gdje su drevne civilizacije kao Egipat i Rim gradili kanale koje su koristili kako ne bi morali ručno nositi vodu. [2]

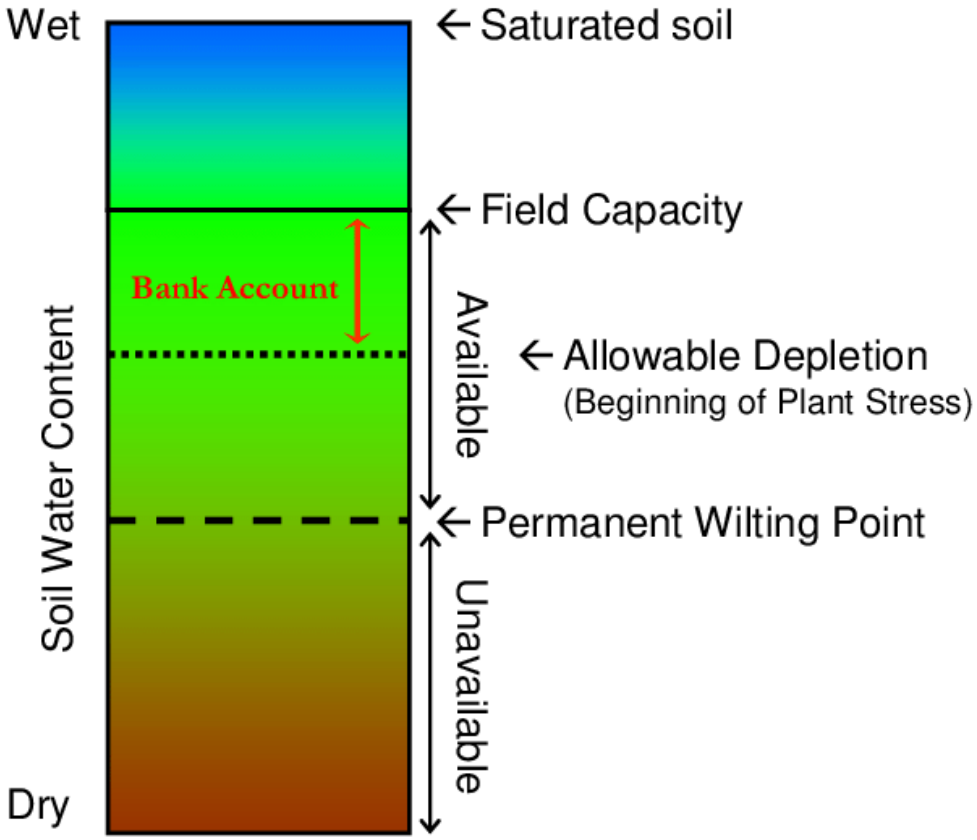
Automatsko upravljanje daje mogućnost poboljšanja sustava navodnjavanja. Kada bi se uložilo dovoljno resursa u razvijanje takvih sustava, potreban trud za uzdržavanje poljoprivrednih imanja drastično bi se smanjio. Ne samo to, već bi se kvaliteta uroda potencijalno poboljšala zbog specijaliziranog navodnjavanja svake vrste ploda koji zahtijevaju drugačije količine vode za najefikasniji urod. No, možda i najvažnije od svega, moguće je osigurati da se ne koriste velike količine vode za nepotrebno navodnjavanje, te se mogu koristiti za kasniju upotrebu čime bi se uštedilo na svjetskoj količini vode. Kao primjer koliko je ovo bitno, može se spomenuti da je ruska vlada 1918. godine preusmjerila dvije rijeke “Amu Darya” i “Syr Darya” kako bi se navodnjavala agrikulutralna polja. Rezultat je bio nestanak većine Aralskog mora, koje se danas naziva Aralsko jezero. [2]

U automatiziranom sustavu navodnjavanja potrebno je osigurati i korisničko sučelje kako bi korisnik imao povratne informacije i uvid u unutarnji rad samog procesa i sustava automatskog navodnjavanja.

2.1. Mjerni elementi

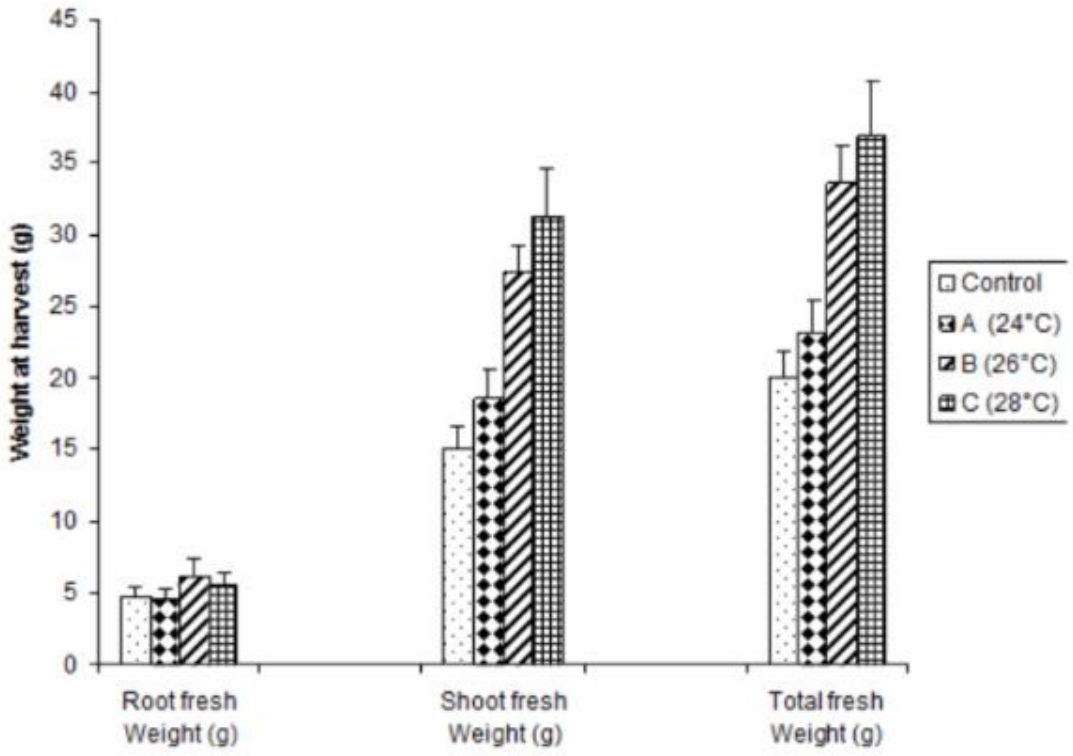
Kako bi sustav bio kvalitetniji i efikasniji, treba odrediti najvažnije procesne varijable koji se moraju regulirati. Regulacijom se treba osigurati da proces i produkt procesa ostane kvalitetan i dosljedan. Za postizanje kvalitetnijih i dosljednijih rezultata pri uzgoju biljka ili usjeva potrebno je regulirati vlažnost zemlje u kojoj biljka raste i temperaturu vode u spremniku koja služi za zalijevanje same biljke.

No, takva regulacija se ne može postići bez povratnih informacija, jer bi to bio sustav bez povratne veze, sustav koji izvršava slijed naredbi bez uvažavanja promjene vanjskih utjecaja. Takva vrsta sustava nije odgovarajuća jer su biljke osjetljive pa promjene vanjskih uvjeta imaju velik utjecaj na promjenu rasta biljke. Stoga treba postojati povratna veza, a tu povratnu vezu pružaju mjerni elementi. Dakle, potrebni su mjerni elementi koji mjere razinu vlage u zemlji te temperaturu vode u spremniku, a za još kvalitetniji uzgoj biljaka mogu se dodati i mjerni elementi koji mjere kemijski sastav tekućine.



Slika 2.2 - Važnost vlage u uzgoju biljaka [3]

Na slici 2.2. vidi se koliko je zapravo važno održavati određenu razinu vlage u zemlji u kojoj biljka raste. Ako je previše vlage u zemlji, biljka počinje trunuti. Ako je vlage premalo, biljka počinje doživljavati stres te je znatno ugrožen njen rast pa biljka može čak i uvenuti. Biljka mora biti u "zelenoj zoni" kao što prikazuje slika 2.2. kako bi se postigli najbolji rezultati. [3] No, sve biljke nemaju jednaku karakteritiku kao biljka u slici 2.2, već je ta karakteristika simbolična i korištena u svrhu pojašnjavanja važnosti održavanja biljka u određenim zonama vlažnosti.



Slika 2.3 - Urod biljke u odnosu na temperaturu vode [4]

Na slici 2.3. vidljivo je kako je urod također ovisan o temperaturi vode s kojom je zalijevan. Unutar određenih granica temperature, može se zaključiti da veća temperatura vode daje bogatiji urod te je potrebno pratiti i temperaturu vode. [4]

Ovim podacima lakše je definirati proces te varijable o kojima je proces ovisan, varijable poput vlažnosti zemlje i temperature vode za navodnjavanje.

2.2. Izvršni elementi i proces

Izvršni elementi su elementi sustava automatskog upravljanja koji djeluju na proces i tim djelovanjem proces pokušavaju dovesti u željeno stanje. Kako bi se mogli odabrati izvršni elementi, mora se poznavati sam proces kojim se želi upravljati. Dakle ako se promatra sustav za navodnjavanje, proces mora osigurati dovoljan dotok vode u zemlju u određenim intervalima kako bi se održavala određena vlaga u samoj zemlji te postigao primjeren iznos temperature vode u spremniku i time osigurao najveći urod same biljke.

U ovom radu se promatra biljka u kućnom uzgoju gdje je prosječna temperatura 23°C te se može pretpostaviti da izvršni element za zagrijavanje vode nije potreban jer će voda poprimiti temperaturu okoline. Kad bi rad bio baziran na vanjskom uzgoju ili uzgoju sa drugačijim temperaturnim uvjetima, bilo bi potrebno osigurati još jedan izvršni element koji bi osigurao da voda bude primjerene temperature.

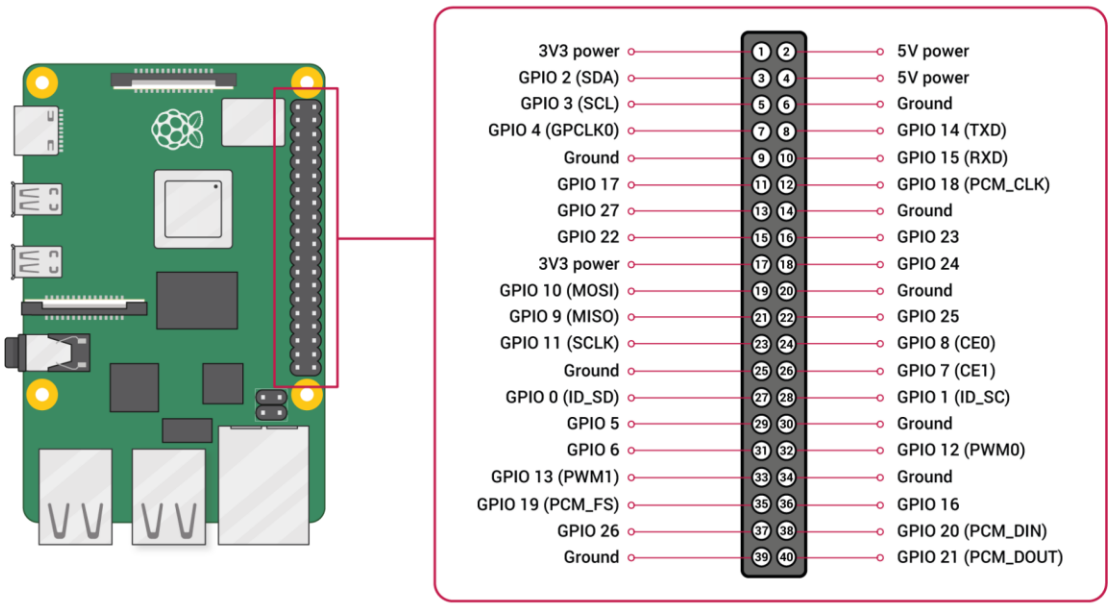
Budući da nije nužan izvršni element za reguliranje temperature vode, potrebno je samo odabrati izvršni element za upravljanje navodnjavanjem zemlje. Izvršni element mora imati sposobnost prenijeti dovoljnu količinu vode za zadovoljavanje uvjeta zasićenosti zemlje vlagom, te on sam mora biti otporan na vodu da ne bi dolazilo do nepotrebnih kvarova.

Također, budući da regulator na svojim izlazima može dati signale koji imaju relativno male iznose napona i struje, a izvršnim elementima je često potrebna veća količina snage u odnosu na potrebe regulatora, bilo bi poželjno osigurati zaštitu samog regulatora od štetnih utjecaja struje i napona nad njim.

2.3. Regulator - Raspberry Pi

Slika 2.4 - Prikaz Raspberry Pi 2 model B

Raspberry Pi (u daljnjem tekstu: RPi) je mini računalo izgrađeno na jednoj pločici. Sve što je prisutno u stolnom računalu prisutno je i u RPi-u. Stvoreno je u Velikoj Britaniji u svrhu učenja osnovnih računalnih znanja bez potrebe za kupnjom mnogo skupljih računala. Raspberry Pi se brzo proširio kao jeftina ali moćna računalna platforma koja se može prilagoditi za automatizaciju. Zahvaljujući svojem višejezgrenom procesoru, RPi može vršiti više zadataka automatizacije i u isto vrijeme vršiti ostale radnje potrebne za rad računala. Operativni sustav na kojem radi RPi najčešće je Raspbian OS, operativni sustav koji je baziran na Debian distribuciji Linux-a, no postoje i besplatne verzije Windows sustava te još nekolicina ostalih operativnih sustava. Glavni programski jezik u kojem se RPi programira je Python, te se pomoću njega mogu koristiti ugrađeni General Purpose Input/Output (u daljnjem tekstu: GPIO) pinovi. GPIO je skupina pinova koji služe za komunikaciju sa vanjskim senzorima - preko njih RPi dobiva informacije koje su mu potrebne za nadzor i upravljanje procesa.



Slika 2.5 - Prikaz GPIO pinova [5]

GPIO pinovi su grupirani prema funkcijama koje obavljaju. Iako većina GPIO pinova može obrađivati jednostavne digitalne signale, neki od pinova imaju sposobnost izvršavati dodatne posebne zadatke koji nisu dostupni u "običnim" pinovima. Na primjer, pin 32 je namijenjen za modulaciju širine impulsa. Pin 7 je konfiguriran za korištenje sa tzv. "*1-wire interface*" što zapravo omogućuje rad senzora koji podržavaju taj režim rada da rade samo preko jedne žice. Ta žica je žica za prijenos podataka. Budući da RPi ima sposobnost napajanja manjim strujama putem samih GPIO pinova ovo je moguće ostvariti (GPIO pinovi mogu generirati i do 50 mA struje). Neki senzori podržavaju "*1-wire interface*" za komunikaciju s RPi-em putem jedne žice. To je moguće samo na pinu 7. Postoje i ostali pinovi koji mogu primati podatke u oba smjera no postoji pravilo kod kojeg gore spomenuto ne radi. To su takozvani MISO (master in slave out, GPIO 10 ili pin 19 na pločici) ili MOSI (master out slave in, GPIO 9 ili pin 21 na pločici) koji podržavaju samo određene smjerove prijenosa podataka. Ovi režimi rada se najčešće ne koriste za rad sa senzorima nego za rad sa drugim uređajima, jer se radi najviše o prijenosu podataka, a ne i samoj obradi. MISO je režim u kojem "slave", uređaj koji je podređen glavnom mikrokontroleru, šalje informacije na "master" uređaj, uređaj koji je nadređeni cijelom sustavu. MOSI je obrnuti režim rada gdje "slave" uređaj prima podatke sa "master" uređaja. Ostali pinovi mogu primati podatke samo u jednom smjeru kako je određeno u programu uz par izuzetaka koji imaju određeni smjer

strujanja podataka (kao npr. pin 38 gdje podaci mogu samo “izlaziti”, tj. mikrokontroler može samo slati naredbe i podatke).

Svi GPIO pinovi koji se nalaze na RPi-u primaju isključivo digitalne signale, RPi nema mogućnost čitanja i obrade analognih signala. Iz ovog razloga je potrebno odabrati senzore koji komuniciraju pomoću digitalnih signala te ako primaju analogne signale, potrebno je osigurati analogno-digitalne pretvornike za te senzore.

Kako bi se moglo pristupati GPIO pinovima nužno je definirati same pinove, a to se radi na sljedeći način:

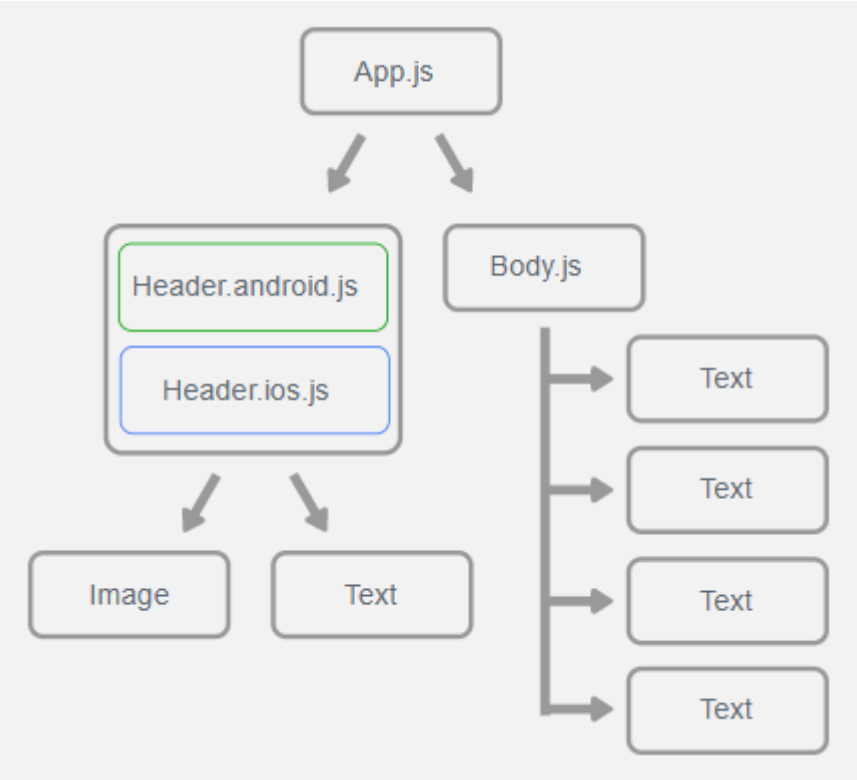
```
motor = 37
vlaga = 40
GPIO.setmode(GPIO.BOARD)
GPIO.setup(vlaga, GPIO.IN)
GPIO.setup(motor, GPIO.OUT)
```

Slika 2.6 - Prikaz definiranja GPIO pinova u programskom jeziku Python

Pri svakom definiranju pinova najprije je potrebno definirati način prepoznavanja pinova na samom RPi-u. Postoje dvije konfiguracije - “*board*” i “*bcm*”. Konfiguracija “*board*” koristi fizičku numeraciju na samom RPi-u, dok “*bcm*” koristi naziv GPIO pina kako bi program mogao prepoznati o kojem pinu se radi. Na slici 2.5 brojevi koji se nalaze u sredini (brojevi napisani na RPi-u) označavaju numeraciju na “*board*” konfiguraciji, dok su brojevi vezani uz opis svakog pina na slici zapravo imena pojedinog pina korištenog u “*bcm*” konfiguraciji. Nakon definiranja konfiguracije pinova (u ovom radu se koristi “*board*” konfiguracija), poželjno je definirati pinove. Pridruživanje imena pinovima, kao što je napravljeno u prva dva reda na slici 2.6, nije potrebno, no poželjno je jer daje referencu na element koji se koristi kada se pišu naredbe u kasnijim dijelovima programa. Da je u programu definiran “*bcm*” način rada, oznake pinova bile bi 26 za varijablu motor, te 21 za varijablu vlaga. Na kraju postavljanja pinova za njihovu određenu svrhu, nužno je definirati je li određeni pin korišten za unos ili izlaz podataka kako bi RPi znao postupati s tim pinovima.

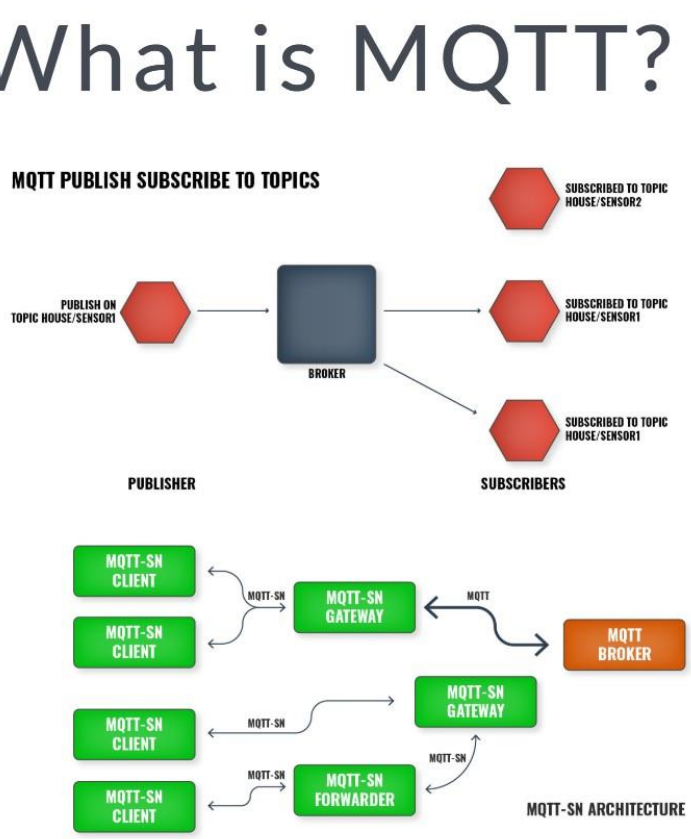
2.4. Korisničko sučelje i MQTT

Za izradu korisničkog sučelja potrebno je izraditi web aplikaciju ili mobilnu aplikaciju. Budući da u današnje vrijeme skoro svatko ima mobilni telefon te su oni prijenosni, napraviti će se mobilna aplikacija. Za nju će se koristiti programski okvir React Native (u daljnjem tekstu: RN) koji se bazira na programskom jeziku JavaScript. RN je platforma na kojoj je izgrađeno mnogo današnjih najuspješnijih aplikacija, dvije od koje su Instagram i Facebook. No najimpresivniji dio ovog okvira jest da je on tzv. *"cross-platform"*, tj. isti programski kod optimiziran je za rad na Android i iOS operativnim sustavima te kasnije i za Windows operativne sustave, no i dalje se najviše koristi za izgradnju mobilnih aplikacija. [6]



Slika 2.7 - Dijagram rada programskog okvira React Native [6]

Kako bi se uspostavila komunikacija između RPi-a i okvira RN, potreban je poseban protokol pomoću kojeg će se uspostaviti komunikacijska veza. U ovu svrhu koristiti će se MQTT protokol. MQTT je protokol dizajniran za rad u sporim i nestabilnim internetskim uvjetima.



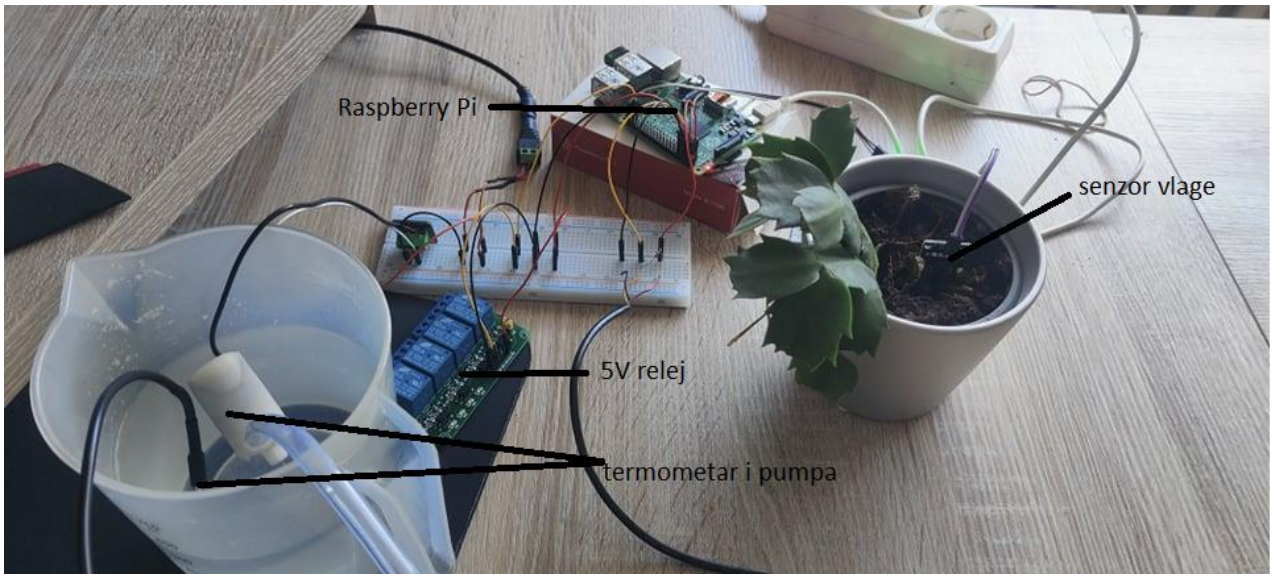
Slika 2.8 - Princip rada MQTT protokola [7]

MQTT je protokol koji radi na principu izmjene podataka kroz centralni "broker". Uz brokera, postoje *"publisher"* (objavitelj) i *"subscriber"* (pretplatitelj). Objavitelj može slati svoje podatke preko određene teme centralnom uređaju koji je postavljen kao broker. Broker može biti i jedinstvena web stranica sa postavljenim MQTT protokolom. U svrhe završnog rada, kao centralni broker koristiti će se hivemq.com, što će i biti demonstrirano u daljnjem tekstu. Ako neki uređaj želi prikupljati podatke od tih istih objavitelja, određeni uređaj se mora spojiti kao klijent na centralni broker preko njegove jedinstvene adrese, te se po želji pretplatiti na teme od kojih želi dobivati informacije. Za primjer: senzor vlage objavi prikupljene podatke brokeru pod temom "vlaga". Korisnik se zatim spaja na brokera, pretplati se na temu "vlaga", te dobiva podatke od svih objavitelja koji objavljuju na temu "vlaga".

3. Praktični dio

3.1. Maketa sustava i kod

Maketa sustava omogućava jednostavan prikaz izgleda sustava i testiranje funkcionalnosti samog sustava. Takav izgled sustava nije krajnji te nije optimiziran za kućnu upotrebu ili prodaju kao krajnji proizvod. U maketi sustava korišteno je 5 glavnih elemenata te su svi elementi sustava za navodnjavanje prikazani na slici 3.1, a njihov rad je objašnjen u daljnjem dijelu poglavlja 3.



Slika 3.1 - Maketa sustava

U svrhe izrade završnog rada korišten je mali kaktus zbog svoje dostupnosti. Svi senzori su na RPi spojeni korištenjem eksperimentalne pločice. RPi se napaja preko punjača za mobitel jer je za pravilan rad RPi-a potrebno osigurati barem 2A struje i 5V napona kako bi svi elementi pravilno radili.

Na slici 3.2 će ukratko biti objašnjen slijed glavnog dijela programa. Prije glavnog dijela programa izvršeno je definiranje pinova i konfiguriranje svih senzora za korištenje u glavnom dijelu programa.

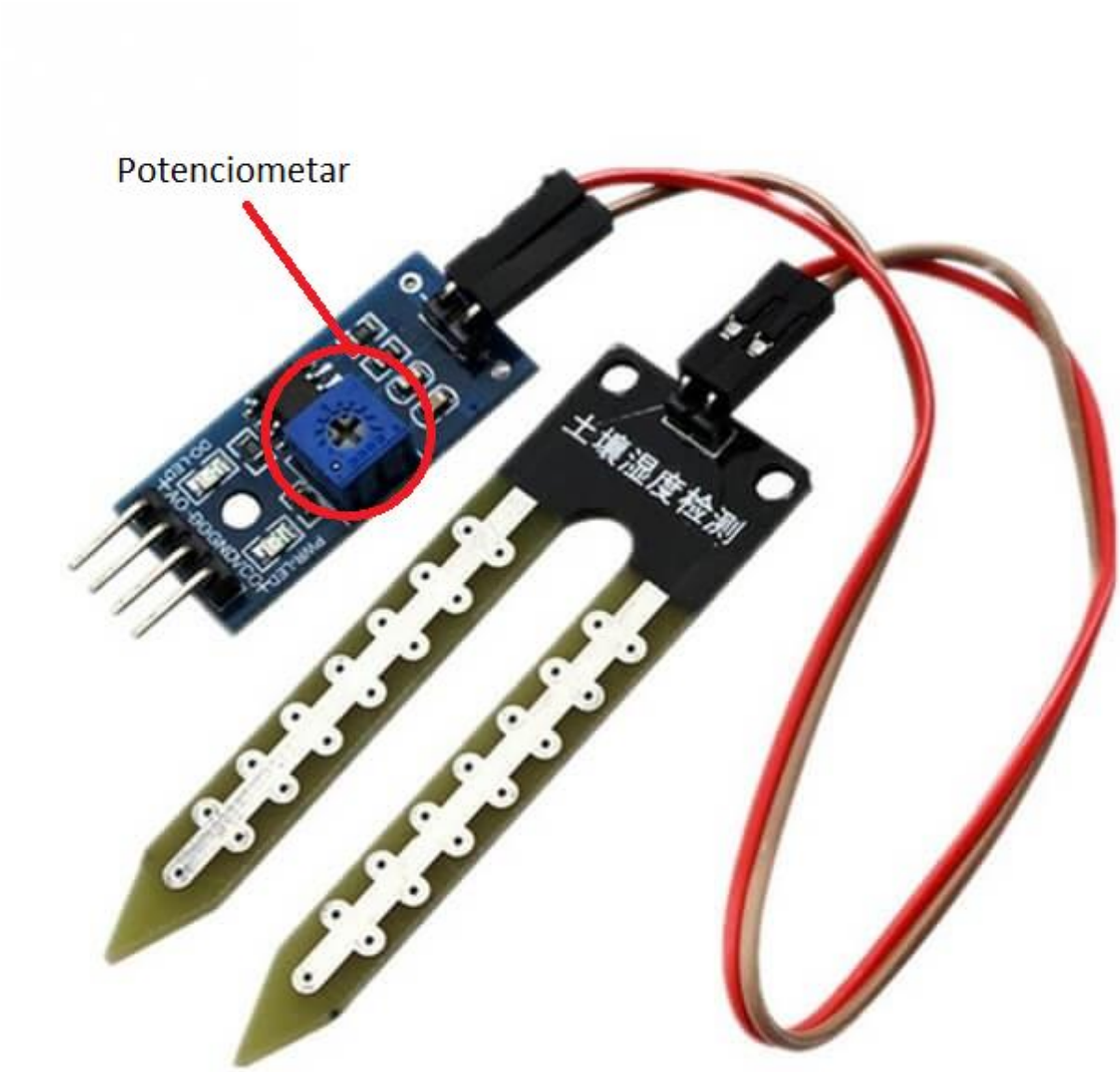
```
publish.single("Motor", "Motor ne radi", hostname="broker.hivemq.com")

while True:
        publish.single("Temperatura", str(read_temp()), hostname="broker.hivemq.com")
        print(read_temp())
                #testiranje za vodu

        if GPIO.input(vlaga) == 0: #ako je voda detektirana
                publish.single("Vlaga", "Zemlja je vlazna", hostname="broker.hivemq.com")
                time.sleep(10)

        elif GPIO.input(vlaga) == 1: #ako voda nije detektirana
                publish.single("Vlaga", "Zemlja je suha", hostname="broker.hivemq.com")
                time.sleep(1)

                publish.single("Motor", "Motor radi", hostname="broker.hivemq.com")
                try:
                        GPIO.output(motor, GPIO.LOW)
                        time.sleep(2)   #vrijeme rada pumpe
                except KeyboardInterrupt:
                        pass

                publish.single("Motor", "Motor ne radi", hostname="broker.hivemq.com")
                try:
                        GPIO.output(motor, GPIO.HIGH)
                        time.sleep(1)
                except KeyboardInterrupt:
                        pass
                time.sleep(1800) #vrijeme cekanja prije novog testiranja
                time.sleep(0.1)

GPIO.cleanup()
```

Slika 3.2 - Glavni dio programa

Program počinje slanjem poruke na aplikaciju da pumpa ne radi pošto je pri početnom pokretanju aplikacije pumpa u stanju mirovanja. Petlja "*while True*" je standardna petlja za beskonačno ponavljanje nekog dijela programa do prekida od strane korisnika. Na početku svake iteracije mjeri se temperatura vode u spremniku te se šalje na aplikaciju. Zatim se testira postojanje vlage u zemlji te se grana na jedan od dva načina rada. Ako je vlaga detektirana program se vraća na početak. No ako vlaga nije detektirana program nastavlja sa daljnjim dijelom programa te se pokreće postupak zalijevanja. Nakon izvršavanja tog dijela programa program počinje ispočetka te se ovaj proces ponavlja.

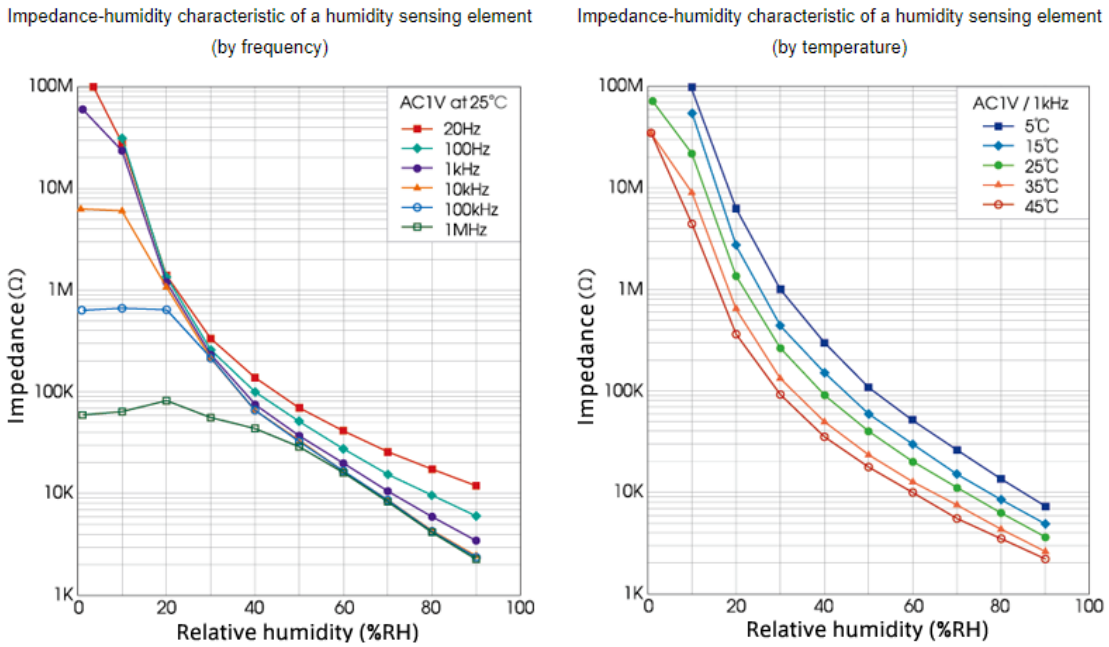
3.2. Senzori

U svrhu ovog završnog rada, korištena su dva senzora: otporni senzor vlage te senzor temperature DS18B20.



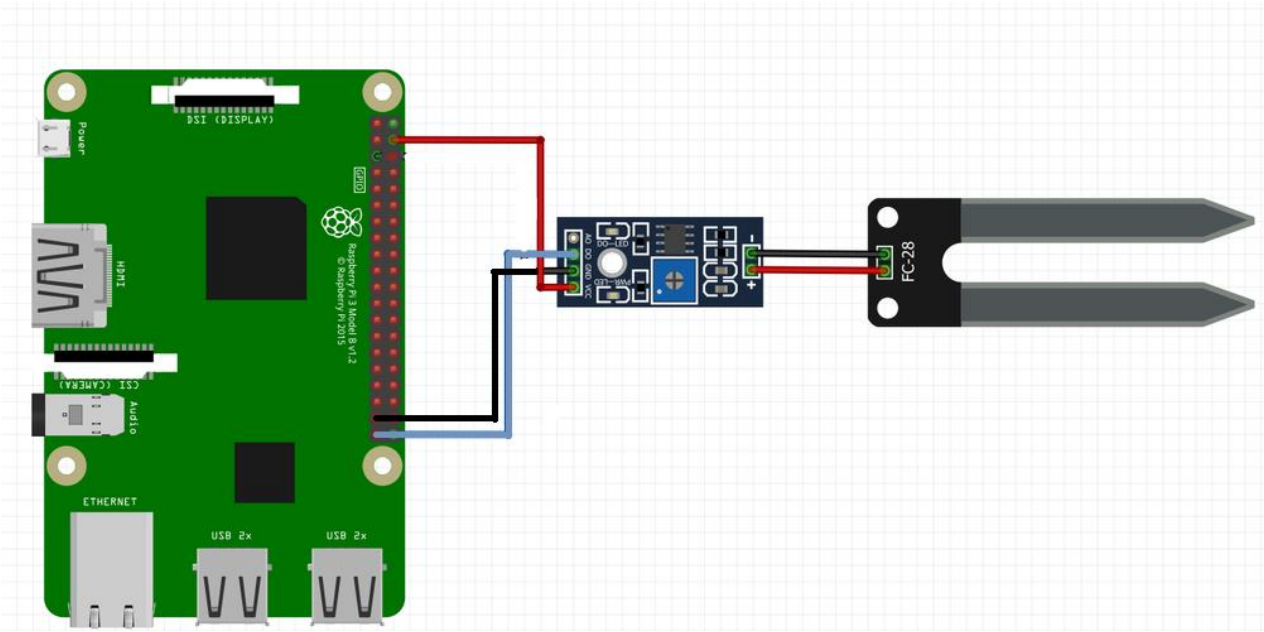
Slika 3.3 - Otporni senzor vlage

Senzor vlage koji se koristi u svrhu završnog rada radi na principu mjerenja impedancije između dvije elektrode. Promjenom vlage mijenja se impedancija zemlje u kojoj se senzor nalazi. Kako se impedancija mijenja, mijenja i odziv samog senzora. Impedancija je također ovisna o temperaturi mjerene tvari te frekvenciji signala korištenog za samo mjerenje, no frekvencija u uvjetima mjerenja je konstantna, pa je ona neovisna o tom dijelu karakteristike. [8]



Slika 3.4 - Ovisnost impedancije o frekvenciji, temperaturi i relativnoj vlazi [8]

Otporni senzor vlage korišten u ovom završnom radu ima podesivo područje rada, područje u kojem će senzor davati izlazni signal. Pomoću potenciometra prikazanog na slici 3.3, moguće je podesiti njegovu osjetljivost, tj. područje impedancije u kojem senzor daje izlazni signal (digitalni 1) i područje impedancije u kojem senzor ne daje izlazni signal (digitalna 0). Ovo je vrlo korisno kako bi se lakše definirala granična vrijednost vlage pri kojoj zemlja nije dovoljno vlažna te se treba pokrenuti pumpa za zalijevanje biljke. Takvo podešavanje senzora puno je lakše i jednostavnije te je izabrano u svrhu ovog završnog rada. Za pravilan rad ovog senzora potrebno ga je priključiti na istosmjerni napon 5V koji se može pronaći na pinu 2 ili 4 na RPi-u (vidi sliku 3.5).



Slika 3.5 - Spajanje senzora vlage na RPi

Očitanje stanja senzora vlage je vrlo jednostavno. Potrebno je definirati na kojem će se pinu spojiti senzor vlage (vidi sliku 2.6) te iskoristiti jednostavnu naredbu. Na slici 3.6 prikazan je kod za konfiguraciju senzora vlage.

```
if GPIO.input(vlaga) == 0: #ako je voda detektirana
        publish.single("Vlaga", "Zemlja je vlazna", hostname="broker.hivemq.com")
        time.sleep(10)

elif GPIO.input(vlaga) == 1: #ako voda nije detektirana
        publish.single("Vlaga", "Zemlja je suha", hostname="broker.hivemq.com")
        time.sleep(1)
```

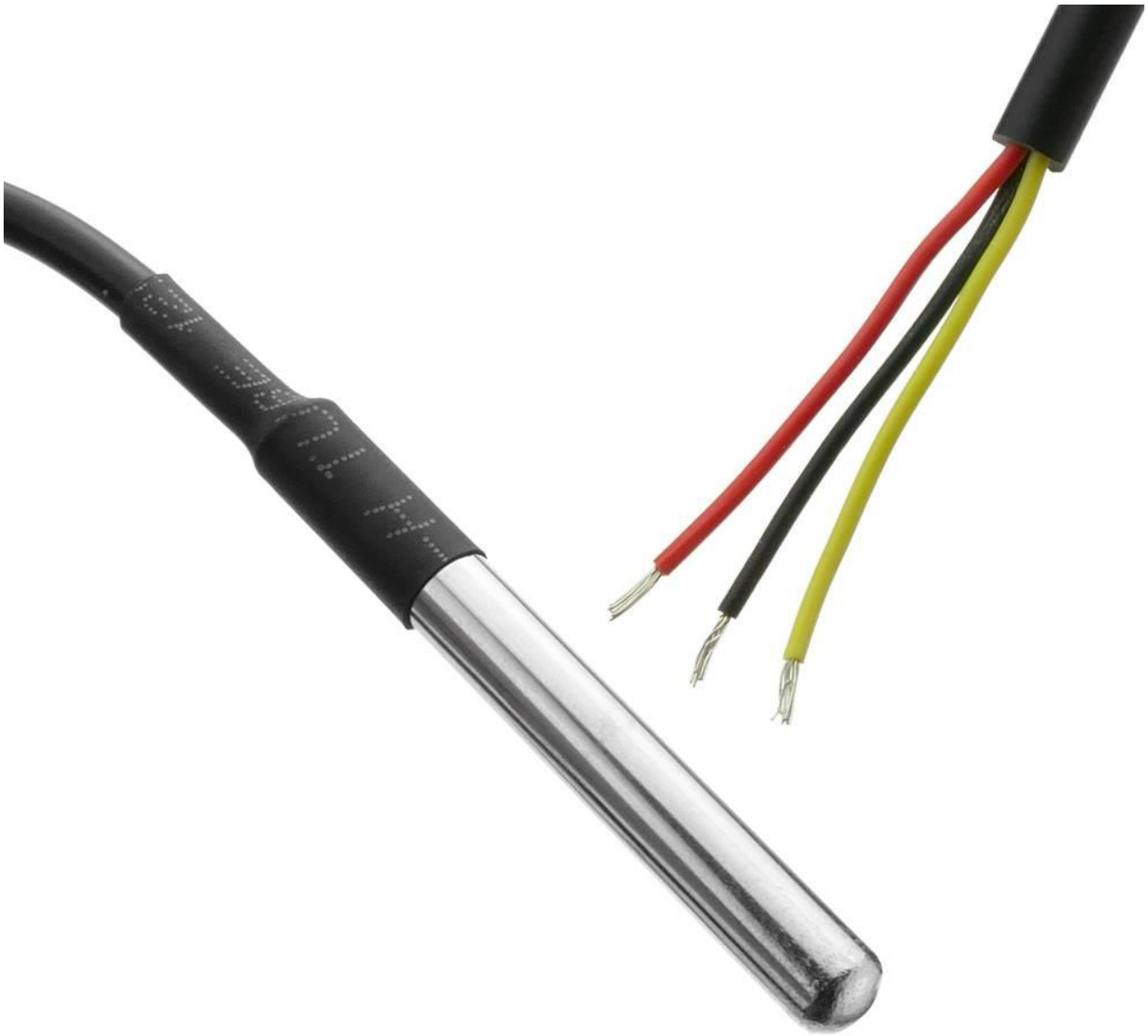
Slika 3.6 - Kod za senzor vlage

Nakon definiranja pina 40 kao pina koji koristi senzor vlage (vidi sliku 2.6) postavljaju se uvjeti za daljnji rad programa. GPIO.input označava da se radi o ulaznoj varijabli te se u zagradi označava da se radi o senzoru vlage. Program se tada grana na "*if* " ili "*elif (else if)*" funkcije od kojih se može u bilo kojem trenutku ostvariti samo jedna, ovisno o odzivu senzora vlage. Ako senzor vlage detektira vodu tada se šalje poruka na aplikaciju da je zemlja vlažna, program čeka 10 sekundi te se vraća na početak programa. Program čeka 10 sekundi kako ne bi došlo do slučajnih pogreška na strani RPi-a tijekom prebrzog izvršavanja koda. Ovo je moguće napraviti jer za ovaj proces nije potreban brzi odziv, budući da se sve vezano uz navodnjavanje događa sporo (voda se sporo upija te se zemlja sporo suši). Ako senzor detektira vlagu, on šalje poruku na aplikaciju da je zemlja vlažna te se program nastavlja dalje do završetka cijelog koda.

Osjetljivost senzora vlage namješta se korištenjem potenciometra za fiksno određivanje granice pa je prije upotrebe senzor potrebno kalibrirati na točno određenu poziciju. Kada je zemlja u kojoj se nalazi biljka dovoljno suha senzor se stavlja u samu zemlju te se potenciometar podesi do kad svjetlo za odziv zasvijetli. To će biti granična vrijednost te u tom području program mora proraditi. Na slici 3.7 vidi se odziv senzora koji je podešen na graničnu vrijednost.

Slika 3.7 - Kalibriranje senzora vlage

Senzor ima dvije LED svjetiljke za označavanje stanja rada. Kada je upaljena samo jedna LED svjetiljka, znači da senzor ne detektira vodu te da voda nije prisutna. Kada su upaljene dvije LED svjetiljke, senzor detektira vodu te daje svjetlosni signal za korisnika. Nakon kalibriranja senzor je spreman za upotrebu. Kad god se zemlja odvlaži do postavljene granice ostatak programa će proraditi. Ovakav način uporabe ima prednost u tome da je ovaj senzor puno lakše i brže postaviti za određene biljke, za razliku od ostalih senzora koji bi zahtijevali posebno podešavanje koda u samom RPi-u za pravilan rad.

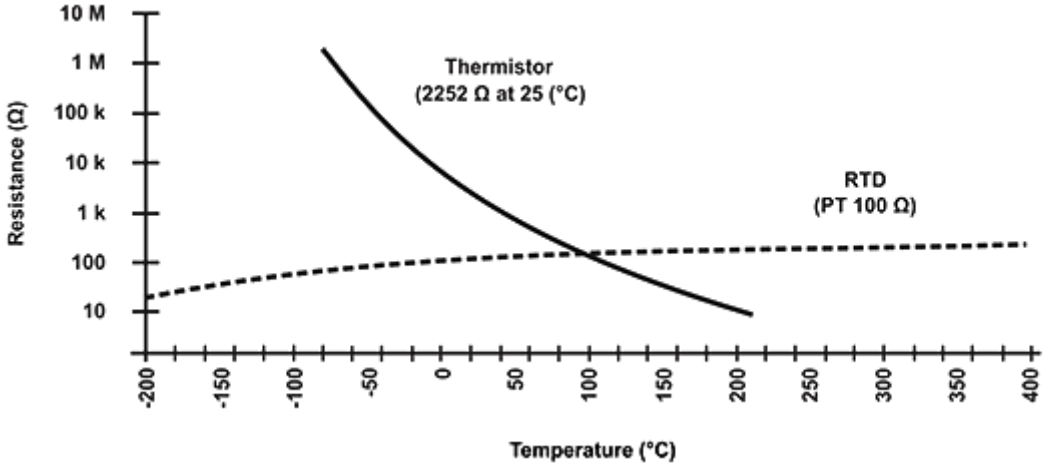


Slika 3.8 - Vodootporni senzor temperature DS18B20

DS18B20 je termistor koji zbog svoje jednostavnosti upotrebe, velike dostupnosti, male cijene, izdržljive građe te mjernog područja temperature od -55°C do +125°C s razinom točnosti od ±5°C (±0.5°C u rasponu od -10°C do +85°C) predstavlja jedan od najkorištenijih osjetnika temperature. [9]

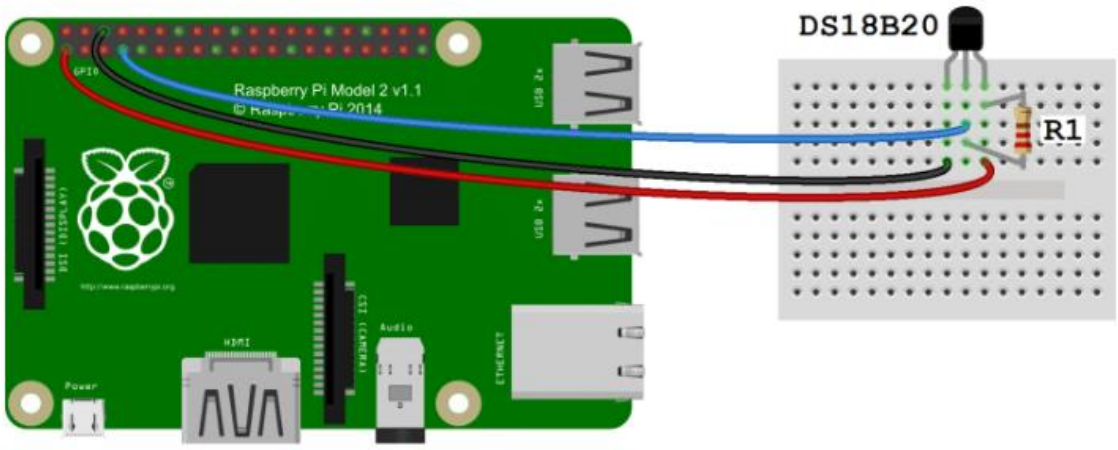
Termistori su uređaji kojima se unutarnji otpor mijenja ovisno o temperaturi. Ovo ih čini vrlo pogodnim za mjerenja temperature, kako je i prikazano na izabranom senzoru temperature. Termistori se dijele na termistore sa pozitivnim temperaturnim koeficijentom i termistore sa negativnim temperaturnim koeficijentom. DS18B20 je termistor sa negativnim temperaturnim

koeficijentom. Termistorima sa negativnim temperaturnim koeficijentom smanjuje se unutarnji otpor s povećanjem temperature, kao što se može vidjeti u slici 3.9, dok je kod termistora sa pozitivnim temperaturnim koeficijentom obrnuto.



Slika 3.9 - Karakteristika termistora sa negativnim temperaturnim koeficijentom [10]

Za stabilan i precizan rad senzora potrebno je, kao i kod ostalih senzora, napajanje. Napajanje za ovaj senzor mora biti 3.3V kao što je označeno od strane proizvođača samog senzora. Na slici 3.10 prikazano je spajanje senzora DS18B20 na RPi.



Slika 3.10 - Spajanje senzora DS18B20 na Raspberry Pi

Pri spajanju senzora potrebno je između VCC i podatkovne žice staviti 4.7kΩ ili 10kΩ otpornik. Taj otpornik se zove *"pull-up otpornik"* te ima važnu ulogu za rad ovog senzora. Pull-up otpornici služe za postavljanje napona na visoku razinu signala kada je to potrebno za određeni senzor ili za stabilan rad uređaja koji su vrlo osjetljivi na vanjske utjecaje elektromagnetskih valova (npr. kontroliranje LED diode sa tipkom). Kada kod takvih osjetljivih elemenata ne bi bio

osiguran "*pull-up otpornik*", elektromagnetni valovi uzrokovali bi na tim elementima nagle uspone i padove u signalu te bi elementi koji ovise o tom signalu naglo i nekontrolirano reagirali. DS18B20 nije osjetljiv na vanjske utjecaje elektromagnetnih signala, no za pravilan rad obavezno mu je spajanje otpornika od 4.7kΩ jer je potrebno osigurati visok signal na podatkovnoj vezi pri čitanju temperatura. [10] Kod za pravilan rad senzora prikazan je na slici 3.11.

```
os.system('modprobe w1-gpio')
os.system('modprobe w1-therm')

base_dir = '/sys/bus/w1/devices/'
device_folder = glob.glob(base_dir + '28*')[0]
device_file = device_folder + '/w1_slave'

def read_temp_raw():
    f = open(device_file, 'r')
    lines = f.readlines()
    f.close()
    return lines

def read_temp():
    lines = read_temp_raw()
    while lines[0].strip()[-3:] != 'YES':
        time.sleep(0.2)
        lines = read_temp_raw()
    equals_pos = lines[1].find('t=')
    if equals_pos != -1:
        temp_string = lines[1][equals_pos+2:]
        temp_c = float(temp_string) / 1000.0
        return temp_c
```

Slika 3.11 - Kod za senzor DS18B20

Ovaj senzor se konfigurira malo drugačije nego ostali senzori. Većina senzora koja se postavlja na RPi-u spoji se na sam RPi te oni automatski rade preko GPIO pinova. Prije nego se planira koristiti DS18B20 se mora konfigurirati unošenjem posebnih naredbi te konfiguriranjem nekih postavki u samom kernelu Raspbian operacijskog sustava. U konfiguracijskoj datoteci koja se nalazi na putanji "*/boot/config.txt*" potrebno je na kraju datoteke dodati sljedeće:

```
dtoverlay = w1_gpio
```

Nakon toga potrebno je u terminalu napisati sljedeće dvije naredbe:

```
sudo modprobe w1-gpio

sudo modprobe w1-therm
```

Nakon izvršavanja spomenutih naredbi senzor je postavljen, no budući da ovaj senzor radi preko *"1-wire bus"* konfiguracije te se koristi pull-up otpornik, on zapravo cijelo vrijeme mjeri temperaturu te sprema podatke u datoteku koja se mora prvo pronaći. Folder se može pronaći na putanji *"/sys/bus/w1/devices/28*/w1_slave"* gdje zvjezdica ("*") označava da se ispred broja 28 mogu nalaziti bilo koji znakovi, tj. traži se folder sa početkom imena "28". Zatim se u kodu pronalazi varijabla "t=" te se čita i dijeli s 1000 kako bi se dobila vrijednost sa pravilnom pozicijom decimalne točke.

Iako je temperatura same vode koja se nalazi u spremniku najmanje bitan dio ovog završnog rada, svakako je prije samog korištenja senzora temperature potrebno osigurati da senzor radi ispravno. Ljudske greške su vrlo česte u polju elektronike, pa tako i u procesu izrade samog senzora, pisanju koda za rad senzora ili samom spajanju senzora. Krivim mjerenjima dolazi do krivih povratnih informacija, a krive povratne informacije vode ka ponovnim ljudskim greškama koje su vođene krivim informacijama. Zbog toga je od velike važnosti vidjeti da li nakon pisanja koda i spajanja taj senzor radi u granicama zadanim od strane proizvođača senzora. Senzor temperature DS18B20 ima toleranciju od ±0.5°C u području u kojem se senzor koristi za svrhe ovog završnog rada. Najlakši način osiguranja pravilnog rada senzora je pomoću eksperimenta sa ledenom kupkom. Ledena kupka je voda u kojoj je stavljen led kako bi temperatura vode poprimila 0°C. Tada se senzor može testirati te mora očitati temperaturu od 0°C ili temperaturu u zadanim granicama postavljenim za taj određeni senzor. Na slici 3.12 prikazana je izvedba pokusa ledene kupke.

Slika 3.12 - Pokus ledene kupke

Pri pokusu ledene kupke važno je mjeriti temperaturu na površini same kupke jer se na dnu posude u kojoj se radi ledena kupka taloži topliji dio vode te bi moglo doći do krivog zaključka da senzor krivo radi. Nakon što je osigurano pravilno smještanje senzora u ledenu kupku potrebno je pokrenuti program za mjerenje temperature te tu temperaturu i izmjeriti. Na slici 3.13 prikazana je mobilna aplikacija sa izmjerenom temperaturom ledene kupke.

Slika 3.13 - Prikaz izmjerene temperature pokusom ledene kupke

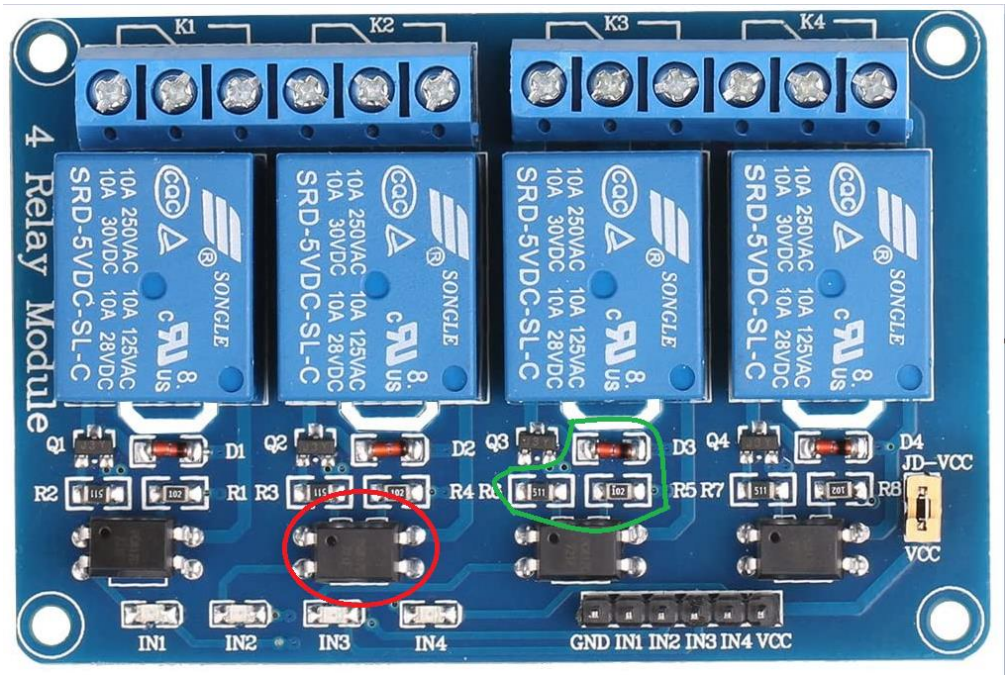
Izmjerena temperatura ledene kupke iznosi 0.25°C što je unutar granica dane od strane proizvođača senzora, koja iznosi ±0.5°C. Nakon ovoga korisnik se može sa sigurnošću pouzdati da neće doći do greške pri mjerenju.

3.3. Izvršni element

Prema zahtjevi za odabir izvršnog elementa, odabrana je 12VDC vodootporna pumpa (slika 3.14). Izabrana je 12VDC pumpa zbog lakoće nabave 12VDC napajanja, te zbog niske cijene i efikasnosti same pumpe. No, kako bi se kontrolirala pumpa potrebna je neka vrsta sučelja za upravljanje radom pumpe, budući da RPi na svojim pinovima ne može dati dovoljno veliku struju koja je potrebna za napajanje pumpe.

Slika 3.14 - 12VDC vodootporna pumpa

Pri korištenju pumpe sa RPi-om potrebno je osigurati neku vrstu kontrole pumpe. Postoje razni microchipovi koji služe za upravljanje smjerom rada motora i za pojačanje signala kako bi se zadovoljili uvjeti rada za te određene motore. No, pošto se u ovom završnom radu koristi pumpa te joj nije potrebno određivanje smjera rada već je potrebno osigurati da pumpa samo radi, za kontrolu pumpe izabran je 5V relej (slika 3.15). Relej je izabran zbog lakoće konfiguriranja na samom RPi-u te zbog svoje već ugrađene zaštite od štetnih utjecaja struje i napona.



Slika 3.15 - 5V relej

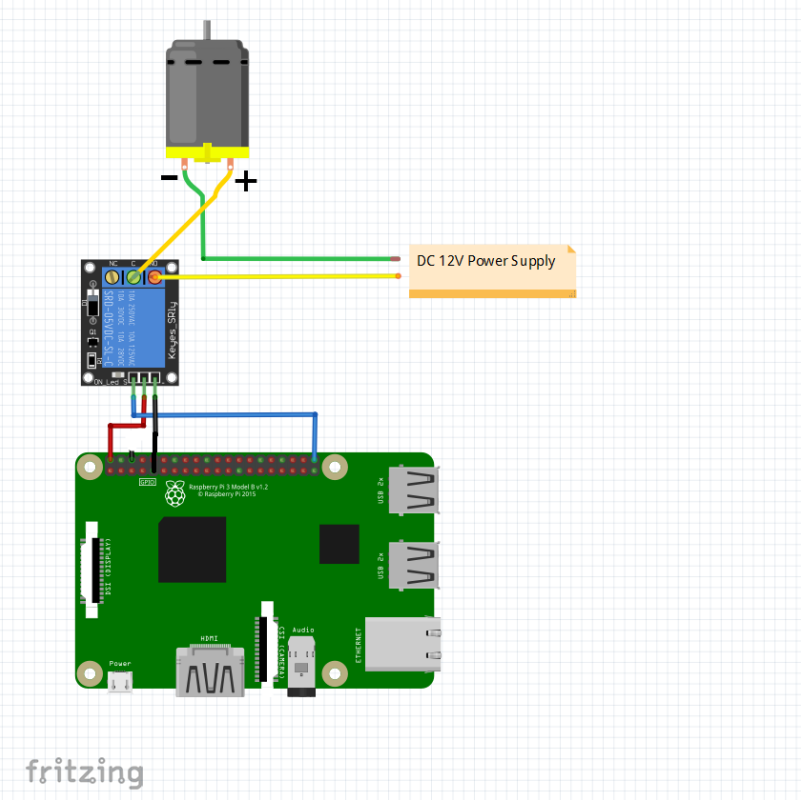
Crvenom bojom prikazan je optoizolator, dok je zelenom bojom prikazana tzv. *"flyback dioda"*. Optoizolator ima zadaću zaštiti izlazne pinove SOC čipa RPi računala od prevelikih struja (optoizolator posjeduje sloj zraka između ulaznih pinova i samog releja), dok *"flyback dioda"* štiti od naglih porasta napona na namotu releja.

Programski kod za uključenje pumpe je vrlo jednostavan. Nakon definiranja pinova pumpa se može vrlo lagano pokrenuti postavljanjem na HIGH ili LOW vrijednost (slika 3.16), ali radom sa ovim relejem logika rada je invertirana (LOW se koristi za rad pumpe).

```
try:
        GPIO.output(motor, GPIO.LOW)
        time.sleep(5)    #vrijeme rada pumpe
except KeyboardInterrupt:
        pass

publish.single("Motor", "Motor ne radi", hostname="broker.hivemq.com")
try:
        GPIO.output(motor, GPIO.HIGH)
        time.sleep(1)
except KeyboardInterrupt:
        pass
```

Slika 3.16 - Kod za rad pumpe

Na RPi-u ne postoji 12V napajanje, pa je zbog toga nužno osigurati vanjsko napajanje od 12V. Na slici 3.17 prikazano je pravilno spajanje pumpe na RPi i 5V relej.



Slika 3.17 - Spajanje 12VDC pumpe na RPi

Ovdje se savršeno vidi kako je RPi izoliran od 12V napajanja. Iako je ovo 1-kanalni relej, princip rada je jednak kao i na 4-kanalnom releju samo što je raspored ulaznih pinova, dioda i optoizolatora različit.

Rad izvršnog elementa ništa ne znači ako se taj izvršni element ne koristi pravilno. Za pravilan rad izvršnog elementa potrebno je odrediti vrijeme rada samog izvršnog elementa. No pitanje je koliko je zapravo vode potrebno za rast same biljke? Kratkim eksperimentom može se odrediti količina vode potrebna za svako zalijevanje. Cvijet se može staviti u posudu u kojoj se nalazi određena količina vode te se nakon 30 minuta može izmjeriti količina vode koja je ostala u posudi. Oduzimanjem konačne količine vode od početne količine vode može se dobiti količina vode potrebna da se zemlja zasiti, a u ovom slučaju ta količina je 100 mililitara.

Za ulijevanje izračunate količine vode također je potrebno odrediti vrijeme rada pumpe kako se cvijet ne bi previše ili premalo zalijao. Uranjanjem pumpe u vodu te pokretanjem može se

izmjeriti količina vode koja se premjesti iz jedne posude u drugu. U 60 sekundi rada pumpa je napumpala 3L vode. To znači da je za 100 mililitara vode potrebno 2 sekunde.

Na slici 3.16, "*time.sleep*" naredba označuje vrijeme pauze gdje program ne izvršava sljedeću naredbu sve dok zadano vrijeme istječe. Pored prve "*time.sleep*" naredbe vidi se komentar (označen sa znakom #) u kojem se napominje da je to polje u kojem se upisuje vrijeme rada pumpe. Naredba prima brojke te su one izražene u sekundama. Dakle, pumpa je konfiguriran da radi 2 sekunde te da se onda isključi. No, program nakon toga ne smije odmah nastaviti detektirati vlagu. Naime, kad bi to bilo tako, zemlja ne bi imala vremena upiti dovoljnu količinu vode te bi senzor ponovo detektirao da je zemlja suha, a zbog toga bi sustav dozirao previše vode u zemlju.

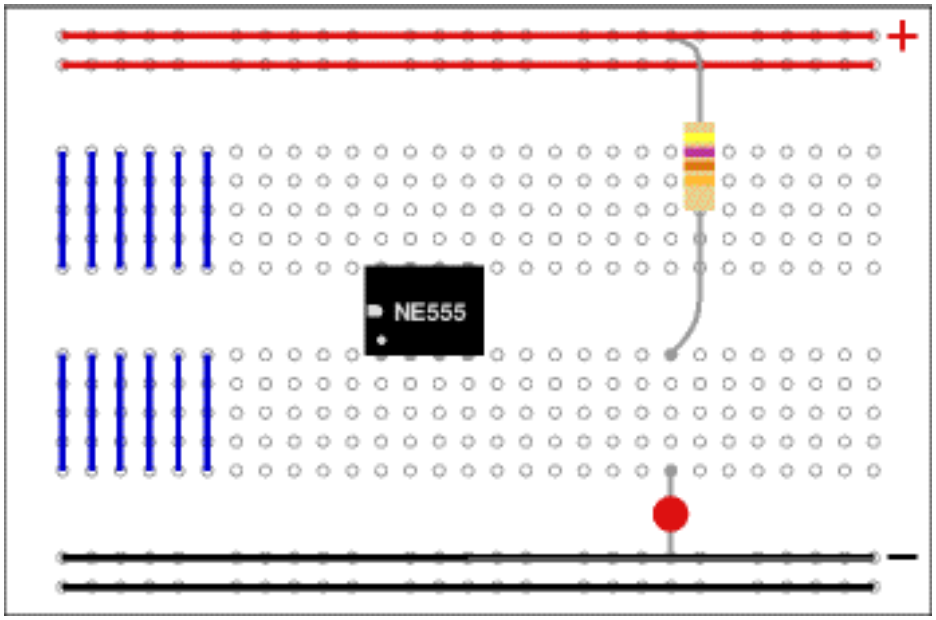
```
try:
        GPIO.output(motor, GPIO.HIGH)
        time.sleep(1)
except KeyboardInterrupt:
        pass
time.sleep(1800) #vrijeme cekanja prije novog testiranja
time.sleep(0.1)
```

Slika 3.18 - Čekanje prije ponovnog testiranja

Na slici 3.18 definirano je da nakon gašenja motora program mora čekati 1800 sekundi (30 minuta) kako bi zemlja mogla u potpunosti upiti vodu. Nakon toga program se vraća u početno stanje.

3.4. Eksperimentalna pločica

Eksperimentalna pločica je pločica koja se koristi kao element preko kojeg se međusobno spajaju svi elementi nekog sustava. Pri eksperimentalnom radu nije uvijek najučinkovitije odmah sve elemente trajno spojiti već se može koristiti eksperimentalna pločica kako bi se ti elementi mogli kratkoročno spojiti do kraja trajanja eksperimenta te naknadno odspojiti. Na slici 3.19 prikazana je shema eksperimentalne pločice.

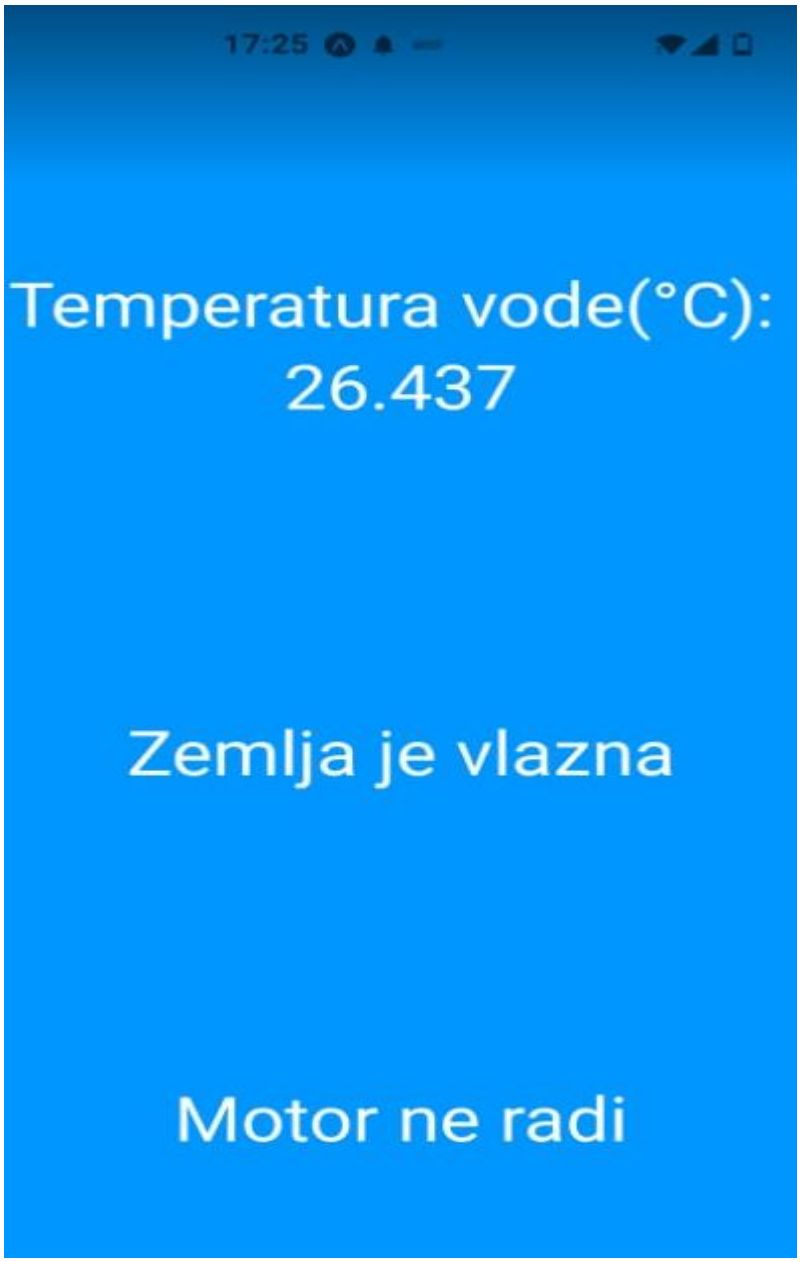


Slika 3.19 - Shema eksperimentalne pločice

Kao što se vidi na slici 3.19, eksperimentalna pločica je podijeljena na redove i stupce koji su spojeni na isti potencijal. Spajanjem dva različita elementa na isti stupac dobiva se serijski spoj tih elemenata, te se također može i lagano postići paralelni spoj elemenata tako da se dva elemenata spoje u dva ista stupca (npr. prvi element spoji se na stupac 1 i 3 te se drugi element također spoji na te stupce, samo u drugu rupicu).

3.5. Mobilna aplikacija

Iako su automatizirani sustavi posve sposoban odraditi njima zadane zadatke bez problema, ponekad je potrebna ljudska intervencija. Za potrebe ljudske intervencije nužno je imati uvid u varijable koje se golim okom ne mogu vidjeti, tj. uvid u vrijednosti koje čitaju sami senzori. Za to je potrebna platforma na kojoj će se same vrijednosti prikazivati. Za ovo se može izgraditi mobilna aplikacija na bazi okvira React Native. Na slici 3.20 može se vidjeti izgled aplikacije napravljene u sklopu ovog završnog rada.



Slika 3.20 - Prikaz aplikacije

Za određivanje izgleda aplikacije potrebno je napraviti određena polja u kojima se definiraju svi atributi za izgled aplikacije (Slika 3.21).

```
const styles = StyleSheet.create ({
  container: {
    flex: 1,
    backgroundColor: 'rgba(0, 150, 255, 1)',
    alignItems: 'center',
    justifyContent: 'center',
    paddingTop: Platform.OS == "android" ? StatusBar.currentHeight : 0,
  },

  text: {
    fontSize: 30,
    color: '#fff',
    fontSize: 32,
  },

  app:{
    marginBottom: 100,
    marginVertical: 70
  }
})
```

Slika 3.21 - Kod za definiranje aplikacije

No, izgled aplikacije nije sve što je potrebno kako bi aplikacija bila završena. Za ostvarenje cilja postavljenog pri definiranju elemenata potrebnih za pravilno funkcioniranje ovog rada, potrebno je i osigurati funkcionalnost aplikacije. Aplikacija je osmišljena tako da prikazuje stanje temperature vode u spremniku, stanje vlage u zemlji i stanje rada pumpe te je to sve podijeljeno na 3 cjeline. Pri dolasku nove poruke sa senzora povezanih na RPi, poruka vezana za određeno polje se briše pa tako nova poruka zauzima njezino mjesto.

3.6. MQTT protokol

Kako bi se uspostavila veza između senzora i aplikacije potrebno je izraditi vezu između njih. Za to će se koristiti MQTT protokol.

Kao što je već rečeno, za pravilan rad MQTT protokola potreban je centralni *"broker"*. Za njega će se koristiti hivemq.com. No prije rada sa MQTT-om na RPi-u potrebno je instalirati MQTT biblioteku jer nije uključena u operacijski sustav Raspbian koji se koristi u ovom završnom radu. Instalacija se vrši pomoću sljedeće naredbe:

```
sudo pip -y install-mqtt
```

Pošto se na RPi-u ne prima niti jedna varijabla, postavljanje klijenta nije potrebno. Potrebno je napisati naredbu za slanje podataka na broker.

```
publish.single("Motor", "Motor ne radi", hostname="broker.hivemq.com")

while True:
        publish.single("Temperatura", str(read_temp()), hostname="broker.hivemq.com")
        print(read_temp())
                #testiranje za vodu

        if GPIO.input(vlaga) == 0: #ako je voda detektirana
                publish.single("Vlaga", "Zemlja je vlazna", hostname="broker.hivemq.com")
                time.sleep(10)

        elif GPIO.input(vlaga) == 1: #ako voda nije detektirana
                publish.single("Vlaga", "Zemlja je suha", hostname="broker.hivemq.com")
                time.sleep(1)

                publish.single("Motor", "Motor radi", hostname="broker.hivemq.com")
                try:
                        GPIO.output(motor, GPIO.LOW)
                        time.sleep(5)   #vrijeme rada pumpe
                except KeyboardInterrupt:
                        pass

                publish.single("Motor", "Motor ne radi", hostname="broker.hivemq.com")
                try:
                        GPIO.output(motor, GPIO.HIGH)
                        time.sleep(1)
                except KeyboardInterrupt:
                        pass
                time.sleep(20) #vrijeme cekanja prije novog testiranja
                time.sleep(0.1)
```

Slika 3.22 - MQTT kod na Raspberry Pi-u

Simbolična naredba za objavljivanje poruke pomoću MQTT protokola izgleda ovako:

```
publish.single("tema", poruka, hostname="link brokera")
```

Analizom koda može se zaključiti da se MQTT-em šalju poruke na 3 teme - jedna za vlagu pod nazivom "Vlaga", jedna za temperaturu pod nazivom "Temperatura" te jedna za stanje rada pumpe pod nazivom "Motor". Poruka o temperaturi se šalje u određenom intervalu, dok se poruke o vlazi i pumpi šalju ovisno o vlažnosti zemlje. Kada je zemlja dovoljno zasićena vlagom, na aplikaciju dolazi poruka o vlazi. Kada zemlja postane dovoljno suha, šalje se poruka da je zemlja suha te da trenutno pumpa radi. Pumpa zatim radi određeno vrijeme, te je onemogućen dugi rad pumpe koji bi zemlju mogao zasititi vodom.

Sljedeći korak je dohvaćanje podataka na samu aplikaciju. Kako bi aplikacija imala sposobnost koristiti MQTT protokol također je potrebno instalirati biblioteku u direktorij u kojem se nalazi aplikacija. Kad je to učinjeno, može se nastaviti dalje.

Prvi dio koda odnosi se na postavljanje klijenta za MQTT na samoj aplikaciji. Kako bi neki uređaj primao poruke od brokera on se mora na njega spojiti kao klijent preko njegove IP adrese te određenog porta.

```
constructor(){
  super();
  this.onMessageArrived = this.onMessageArrived.bind(this)
  this.onConnectionLost = this.onConnectionLost.bind(this)


  const client = new Paho.MQTT.Client('broker.hivemq.com', 8000, 'pi@njolac');
  client.onMessageArrived = this.onMessageArrived;
  client.onConnectionLost = this.onConnectionLost;
  client.connect({
    onSuccess: this.onConnect,
    useSSL: false ,
    onFailure: (e) => {console.log("here is the error" , e); }
```

Slika 3.23 - Spajanje na brokera preko aplikacije

Preko aplikacije se spaja na port 8000 brokera s IP adresom 'broker.hivemq.com'. "Client" je potreban kako bi aplikacija znala da se radi o samom MQTT kodu. Također se sve poruke vežu na "this" varijablu za daljnju uporabu. Zatim je potrebno pretplatiti se na sve teme od kojih se želi dobivati podatke (slika 3.24):

```
onConnect = () => {
  const { client } = this.state;
  client.subscribe('Temperatura');
  client.subscribe('Vlaga');
  client.subscribe('Motor')
  this.setState({isConnected: true, error: ''})
};
```

Slika 3.24 - Pretplata na željene teme

Ovim kodom se pretplaćuje na teme “Temperatura”, “Vlaga” i “Motor”. Uvijek kad se na web pošalje poruka koja je namjenjena za jednu od ovih tri tema, aplikacija će imati mogućnost dohvatiti tu poruku za daljnju obradu.

Kako bi se poruka mogla prikazati u samoj aplikaciji, mora se prvo pohraniti (Slika 3.25).

```
onMessageArrived(entry) {
    this.setState((state) => ({
        ...state.items,
        tema: {
            ...state.tema,
            [entry.destinationName]: entry.payloadString
        }
    }));
}
```

Slika 3.25 - Pohranjivanje dolazećih poruka

Nakon što neka poruka dođe na jednu od tri preplaćenih tema, ona se sprema u *“entry”* objekt. Poruka sa sobom nosi mnogo atributa od kojih je nužno vidjeti gdje je pohranjena sama poruka (slika 3.26).

```
Object {
  "_getDestinationName": [Function anonymous],
  "_getDuplicate": [Function anonymous],
  "_getPayloadBytes": [Function anonymous],
  "_getPayloadString": [Function anonymous],
  "_getQos": [Function anonymous],
  "_getRetained": [Function anonymous],
  "_setDestinationName": [Function anonymous],
  "_setDuplicate": [Function anonymous],
  "_setQos": [Function anonymous],
  "_setRetained": [Function anonymous],
}
```

Slika 3.26 - Prikaz “entry” objekta

Nakon ispisivanja vrijednosti za svakog od ponuđenih atributa, tema se može pronaći u "*destinationName*" atributu. Pomoću "*setState*" možemo odabrani atribut entry objekta pohraniti u neku varijablu, koja se u ovom kodu zove "*tema*". "*payloadString*" pretvara svaku poruku koja dolazi na samu aplikaciju u "*string*" (tekstualni oblik podataka) kako bi se omogućio kasniji ispis poruke, a pomoću "*entry.destinationName*" može se filtrirati poruka ovisno kojoj temi pripada.

Na kraju se poruka mora ispisati u samoj aplikaciji. Za ispis se koriste već definirane varijable (slika 3.27).

```
render() {
  return(
    <SafeAreaView style={{ flex: 1}}>
    <View style={styles.container}>
      <View style={styles.app}>
        <Text style={styles.text}>Temperatura vode: </Text>
        <Text style={styles.text}>{this.state.tema.Temperatura}</Text>
      </View>
      <View style={styles.app}>
        <Text style={styles.text}>{this.state.tema.Vlaga}</Text>
      </View>

      <View style={styles.app}>
        <Text style={styles.text}>{this.state.tema.Motor}</Text>
      </View>
    </View>
    </SafeAreaView>
  );
}
}
```

Slika 3.27 - Ispis podataka

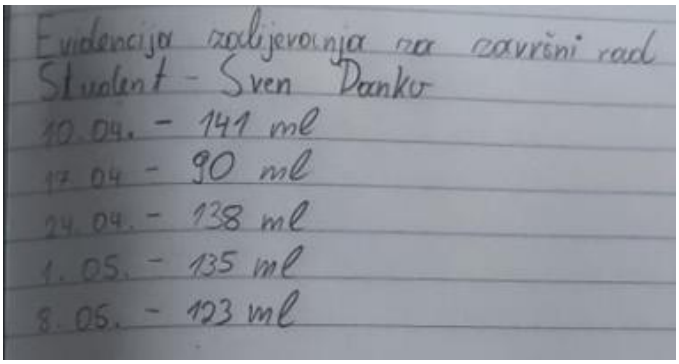
U svakom "*text*" polju mora se navesti tema poruke kako bi se ispisivale poruke vezane samo uz tu temu. U suprotnom bi se sve poruke naizmjence ispisivale na jednom mjestu. "*this.state.tema*" je varijabla preko koje se pohranjuju poruke, dok je sve iza ".*tema*" zapravo oznaka za pojedinu temu koja se ispisuje.

4. Analiza rezultata

Analiza rezultata radila se u periodu od 2 mjeseca. Prvih mjesec dana biljka je zalijevana svakih tjedan dana tako da se stavila u posudu, u tu posudu se stavila biljka i ostavila tako pola sata da upije vodu. Količina vode koja je zatim preostala zapisana je kako bi se u kasnijem periodu prikazali relevantni rezultati. Drugih mjesec dana cvijet je zalijevan pomoću sustava izrađenog za završni rad. Nakon perioda od mjesec dana preostala voda u posudi za zalijevanje zabilježena je te se izračunala količina potrošene vode u tom periodu. Eksperiment je započeo u subotu 10.04.2021, druga faza je nastavljena 08.05.2021 kad je prva faza eksperimenta završila, te je cjelokupni eksperiment zaustavljen 12.06.2021. Na slici 4.1 može se vidjeti stanje cvijeta prije početka pokusa.

Slika 4.1 - Cvijet prije eksperimenta

Ručno zalijevanje vode radilo se tako da se u posudu natočila voda, biljka se ostavila u vodi te pustila da upije vodu. Količina preostale vode u posudi se zatim izmjerila i zapisala u zapisnik na slici 4.2.



Slika 4.2 - Zapisnik količine neiskorištene vode

U posudu su prije stavljanje biljke natočene količine vode koje su za 100ml veće od zapisanih veličina na slici 4.2, a zapisane količine vode predstavljaju količinu vode koja nije potrošena već je bačena. Količine su nasumično natočene za oponašanje nasumičnosti pri stvarnim zalijevanjima biljka. Zbrajanjem vrijednosti bačene vode dobiva se da se u periodu od mjesec dana bacilo 627 ml vode dok je ukupna količina utrošene vode iznosila cca. 1130 ml. U drugom dijelu eksperimenta bijela posuda napunjena je do zadnje crtice koja iznosi 700 ml. Naknadno nije dodavana ni oduzimana voda iz posude. Nakon perioda od mjesec dana u posudi je ostalo cca. 210 ml vode, te su zalijevanja također raspoređena na 5 zalijevanja. U oba slučaja je količina utrošene vode iznosila cca. 500 ml. Oduzimanjem količine bačene vode od količine vode utrošene za zalijevanje biljke pomoću automatiziranog sustava zalijevanja vidi se da je bačeno 117 ml više no što je uopće potrebno za zalijevanje same biljke. Tih 117 ml moglo se iskoristiti u neku drugu svrhu, a ostalih 500 ml moglo se koristiti za zalijevanje biljke. Računica pokazuje kako je zapravo neefikasno zalijevati biljke ručno, već je ekološki i ekonomski efikasnije koristiti sustav navodnjavanja. Preostalih 210 ml vode koja se nalazi u lijevoj posudi i dalje će ostati u sustavu navodnjavanja, dok se neiskorištena vode pri ručnom zalijevanju u većini slučajeva jednostavno baci.

Slika 4.3 - Usporedba iskorištene vode

Na kraju se može promatrati razlika između početnog stanja cvijeta i završnog stanja cvijeta. Na početku je cvijet visio te nije bio u najboljem stanju, dok se nakon eksperimenta cvijet lagano uspravio te također i dodatno narastao. Na slici 4.4 vidi se krajnje stanje cvijeta.

Slika 4.4 - Krajnje stanje cvijeta

Nakon analize rezultata može se zaključiti da su obje metode zalijevanja cvijeta kroz period od 2 mjeseca bile uspješne, no automatizirani sustav ne zahtijeva ljudski angažman kako bi se obavilo zalijevanje te ima sposobnost štedjeti vodu kao što je bilo prikazano tijekom ispitivanja sustava. Sustav je u periodu od mjesec dana uštedio 417 ml vode u odnosu na proces ručnog zalijevanja, no preostalih 210 ml i dalje se može iskoristi u sustavu navodnjavanja, pa je potencijalna ušteda čak i 627 ml vode. To ga čini financijski i ekološki boljom dugoročnom opcijom.

5. Zaključak

21. stoljeće je jednim dijelom obilježeno kao stoljeće u kojoj Zemlja pati. Prirodne katastrofe, pandemije i globalno zatopljenje ostavile su svoj trag u ljudskoj i zemaljskoj povijesti. No, jedna tema koja se nigdje ne spominje je nestanak pitke vode. Mali broj ljudi je svjestan koliko je malo pitke vode danas dostupno te da se ta voda troši kao da je beskonačna. Ovo se najviše vidi u kućnom uzgoju biljka, gdje se biljke prekomjerno zalijevaju te se suvišna voda naknadno baca. Automatizirani sustavi navodnjavanja jedno su od kvalitetnih rješenja za očuvanje vode jer se količina potrošene vode može precizno odrediti za svaku biljku, te nema bacanja. Također, sam urod i kvaliteta uroda se poboljšava zbog čega su ovakvi sustavi veoma poželjni ne samo za velika postrojenja, već i za kućnu upotrebu.

Izrada automatskog sustava navodnjavanja započela je jednostavnom misli, a ta misao je bila kako i gdje je moguće uštedjeti vodu? Nakon osmišljavanja koncepta osmišljen je sustav te je pronađena platforma koja će biti glavni dio sustava. Pošto je ovo bilo novo iskustvo, najveći izazov je bio prilagođavanje poprilično novoj platformi i operacijskom sustavu te kreiranje korisničkog sučelja. Srećom, internet je prepun ljudi koji su spremni prenijeti svoje znanje te je svladavanje problema ovisilo o volji za pronalaskom rješenja. Nabavka i spajanje elemenata svodilo se na pronalazak najjeftinijih rješenja. Najveće olakšanje u procesu izrade sustava bilo je implementacija PuTTY emulatora koji je omogućio bežično spajanje i rad na RPi-u. Pomoću svega ovoga moguće je bilo izraditi sustav koji ima potencijal štedjeti značajne količine vode, pod uvjetom da se pravilno koristi. Iako trenutna verzija nije sposobna odraditi to, izgradnjom ovog sustava izgrađena je podloga za izradu sustava namijenjenih za veća postrojenja.

6. Literatura

[1] G.F. Franklin, J.D. Powell, A. Emami-Naeini (2002.), *Feedback Control of Dynamical Systems*, Prentice Hall; 6th edition

[2] https://www.nationalgeographic.org/encyclopedia/irrigation/, dostupno 29.05.2021.

[3] T. Maughan, D. Drost, L. N. Allen: Vegetable Irrigation: Squash and Pumpkin, UtahState University, Logan, 2015.

[4] S. Nxawe, C. P. Laubscher, P.A. Ndakidemi: Effect of regulated irrigation water temperature on hydroponics production of Spinach (Spinacia oleracea L), Cape Peninsula University of Technology, Cape Town, 2009

[5] https://www.raspberrypi.org/documentation/usage/gpio/, dostupno 25.05.2021.

[6] https://reactnative.dev/, dostupno 23.05.2021.

[7] https://www.pcwdld.com/what-is-mqtt, dostupno 25.05.2021.

[8] https://product.tdk.com/en/techlibrary/productoverview/numidity-sensors.html, dostupno 23.05.2021.

[9] https://eepower.com/resistor-guide/resistor-types/ntc-thermistor/#, dostupno 27.05.2021.

[10] https://cdn.sparkfun.com/datasheets/Sensors/Temp/DS18B20.pdf

Popis slika

Sveučilište Sjever
Sveučilišni centar Varaždin
104. brigade 3, HR–42000 Varaždin

Prijava završnog rada

Definiranje teme završnog rada i povjerenstva

ODJEL Odjel za elektrotehniku

STUDIJ preddiplomski stručni studij Elektrotehnika

PRISTUPNIK Sven Danko | JMBAG

DATUM 21.06.2021. | KOLEGIJ Automatsko upravljanje

NASLOV RADA Sustav za navodnjavanje upravljan Raspberry računalom

NASLOV RADA NA ENGL. JEZIKU Irrigation system controlled by a Raspberry computer

MENTOR Miroslav Horvatić, dipl. ing. | ZVANJE predavač

ČLANOVI POVJERENSTVA

1. mr. sc. Ivan Šumiga, dipl. ing., viši predavač
2. mr. sc. Matija Mikac, dipl. ing., viši predavač
3. Miroslav Horvatić, dipl. ing., predavač
4. dr. sc. Ladislav Havaš, docent, rezervni član
5.

Zadatak završnog rada

BROJ 484/EL/2021

OPIS

Potrebno je realizirati sustav za navodnjavanje koji mjeri vlagu u tlu i automatski vrši zalijevanje. Upravljanje sustavom realizirati korištenjem Raspberry računala.

U radu je potrebno:
- osmisliti i objasniti princip rada sustava
- odabrati komponente i realizirati sklopovlje sustava
- osmisliti i realizirati programsku podršku za Raspberry računalo
- osmisliti i realizirati mobilnu aplikaciju koja prikazuje informacije o radu sustava
- ispitati rad realiziranog sustava.

ZADATAK URUČEN 25.6.2021. | POTPIS MENTORA

REPUBLIKA HRVATSKA 5 SVEUČILIŠTE SJEVER KOPRIVNICA SVEUČILIŠTE SJEVER

Sveučilište
Sjever

SVEUČILIŠTE
SJEVER

IZJAVA O AUTORSTVU
I
SUGLASNOST ZA JAVNU OBJAVU

Završni/diplomski rad isključivo je autorsko djelo studenta koji je isti izradio te student odgovara za istinitost, izvornost i ispravnost teksta rada. U radu se ne smiju koristiti dijelovi tuđih radova (knjiga, članaka, doktorskih disertacija, magistarskih radova, izvora s interneta, i drugih izvora) bez navođenja izvora i autora navedenih radova. Svi dijelovi tuđih radova moraju biti pravilno navedeni i citirani. Dijelovi tuđih radova koji nisu pravilno citirani, smatraju se plagijatom, odnosno nezakonitim prisvajanjem tuđeg znanstvenog ili stručnoga rada. Sukladno navedenom studenti su dužni potpisati izjavu o autorstvu rada.

Ja, SVEN DANKO *(ime i prezime)* pod punom moralnom, materijalnom i kaznenom odgovornošću, izjavljujem da sam isključivi autor/ica završnog/~~diplomskog~~ *(obrisati nepotrebno)* rada pod naslovom SUSTAV ZA NAVODNJAVANJE UPRAVLJAN RASPBERRY RAČUNALOM *(upisati naslov)* te da u navedenom radu nisu na nedozvoljeni način (bez pravilnog citiranja) korišteni dijelovi tuđih radova.

Student/ica:
(upisati ime i prezime)

Sven Danko
(vlastoručni potpis)

Sukladno Zakonu o znanstvenoj djelatnost i visokom obrazovanju završne/diplomske radove sveučilišta su dužna trajno objaviti na javnoj internetskoj bazi sveučilišne knjižnice u sastavu sveučilišta te kopirati u javnu internetsku bazu završnih/diplomskih radova Nacionalne i sveučilišne knjižnice. Završni radovi istovrsnih umjetničkih studija koji se realiziraju kroz umjetnička ostvarenja objavljuju se na odgovarajući način.

Ja, SVEN DANKO *(ime i prezime)* neopozivo izjavljujem da sam suglasan/na s javnom objavom završnog/diplomskog *(obrisati nepotrebno)* rada pod naslovom SUSTAV ZA NAVODNJAVANJE UPRAVLJAN RASPBERRY RAČUNALOM *(upisati naslov)* čiji sam autor/ica.

Student/ica:
(upisati ime i prezime)

Sven Danko
(vlastoručni potpis)