SVEUČILIŠTE U SPLITU

FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

PRIMJENA IoT TEHNOLOGIJA U POLJOPRIVREDI ZA POBOLJŠANJE KVALITETE PROIZVODNJE

Ivan Križanović

Split, srpanj 2017.

SVEUČILIŠTE U SPLITU
FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE

Diplomski studij: **Računarstvo**

Smjer/Usmjerenje: **Računarstvo**

Oznaka programa: 250

Akademska godina: 2016./2017.

Ime i prezime: **IVAN KRIŽANOVIĆ**

Broj indeksa: 681-2015

ZADATAK DIPLOMSKOG RADA

Naslov: **PRIMJENA IoT TEHNOLOGIJA U POLJOPRIVREDI ZA POBOLJŠANJE KVALITETE PROIZVODNJE**

Zadatak: Razviti bežičnu mrežu senzorskih čvorova koji sadrže različite tipove osjetila/senzora za mjerenje i očitavanje podataka vezanih uz uzgoj poljoprivrednih kultura. Podaci prikupljeni putem podataka koristiti će se za unapređenje ekološke proizvodnje u poljoprivredi. Potrebno je proizvesti strukturiranu komunikaciju između čvorova koja rezultira spremanjem podataka mjerenja u predefiniranu bazu podataka. Kreirati *web* servis za detaljan grafički prikaz podataka iz baze podataka. Servis treba podržavati rad u realnom vremenu. Detaljno dokumentirati arhitekturu mreže kao i rezultate provedenih testova.

Prijava rada: 20.02.2017.

Rok za predaju rada: 07.09.2017.

Rad predan:

Predsjednik
Odbora za diplomski rad:

Mentor:

Prof. dr. sc. Sven Gotovac

Pof. dr. sc. Mario Čagalj

Sadržaj

1 Uvod

Koncept “pametnih” uređaja se počeo razvijati još od 1982. godine zahvaljujući istraživanjima i prikupljenim znanjima o *real-time* analizi podataka, *machine learning*-u, senzorima, ugradbenim sistemima te dotad već ukorijenjenom *wireless* spajanju na internet. Ideja se zadržala do danas i unazad dvadeset godina je postala pristupačna široj publici ljudi, ne samo IT kompanijama.

Zaživila je u mnogim kućanstvima sa različitom primjenom pa tako i omogućila različitim razvojnim platformama kreiranje kompleksnijih sustava i distribuiranih aplikacija. Većina je takvih platformi prihvatila koncept sa fokusom na različita real-time mjerenja i razmjenu podataka unutar velikih sistema, stoga danas imamo niz inačica hardverskih rješenja.

1.1 Internet of things

Pojam *IoT* (hrv. Internet stvari) danas najčešće koristimo kada opisujemo bežičnu komunikaciju različitih uređaja s ciljem međusobne razmjene podataka. U navedenom izrazu pojam *things* označava objekt fizičke ili virtualne okoline koji se može identificirati i integrirati u postojeće komunikacijske mreže. Takav skup objekata čini jedan sustav koji može biti povezan sa nekim drugim sustavom ili uređajem s ciljem međusobne razmjene podataka i informacija neovisno o razvojnoj platformi sustava. Predstavlja široki spektar različitih vrsta međusobno uparenih uređaja koji sakupljaju i dijele korisne podatke. Primarna ideja kreiranja takvih sustava je omogućiti udaljeno upravljanje sustavom putem postojeće mrežne infrastrukture te na taj način sačuvati njegovu autonomiju a pritom omogućiti integraciju sa drugim sustavima na mreži.

Termin se sve više počeo pojavljivati kao tema rasprave na akademskim skupovima kao vizija gdje kontrolni i automatizacijski sistemi (uključujući i kućne uređaje) sudjeluju u razvoju tog koncepta. Jedan od prvih takvih projekata bio je studenata američkog sveučilišta “Carnegie Mellon University” koji su htjeli modificirati „*Coca-Cola*“ automate s ciljem praćenja evidencije stanja proizvoda u automatima.

U svojoj prvotnoj interpretaciji termin je dobio na važnosti kada se razvila težnja da se svi elektronički objekti u svakodnevnoj okolini uključe u koncept kako bi se poboljšala kvaliteta i ubrzali procesi u svakodnevnom životu, te reduciralo utrošeno vrijeme. Ljudska bi se interakcija sa sistemom primjenjivala isključivo kada postoji trenutna potreba korisnika. Do 1999. godine kada

je koncept doživio prvi svoj veliki pomak velike IT kompanije su ulagale novac u razvoj tog područja te su ponudile prvotna rješenja poput Microsoftovog „*Microsoft at Work*“ i „NEST“ tvrtke Novell.

Termin „Internet stvari“ postaje sve raširenij i kod proizvođača elektroničkih uređaja pa je najširu primjenu pronašao kod proizvodnje uređaja bijele tehnike u kućanstvu, stoga ne čudi brojka da će do 2020. godine biti preko 20 milijardi aktivnih uređaja upravu u tom području. Ekspanzijom *IoT* sustava u budućnosti postoje određena funkcionalna očekivanja koja obuhvaćaju obradu velikog broja podataka na različitim lokacijama te njihovo procesuiranje i analizu kroz druge sustave. Pritom je važno postići maksimalnu efikasnost sustava u smislu brzine i potrošnje resursa. Takvi su sustavi danas primjenu pronašli u različitim poljima pa je tako zaživila ideja pametnih gradova i pametnih kućanstava gdje smo svakodnevno okruženi sa kućanskim uređajima koji međusobno komuniciraju.

1.2 Kratak opis projekta

Diplomski rad je dio projekta *E-Vineyard,* nastao kao dio *Ericsson Nikola Tesla Summer Camp* ljetne prakse za studente, a cilj je kreirati sustav za praćenje fizikalnih parametara poljoprivrednih kultura. Sustav je prvotno namijenjen isključivo za mjerenje fizikalnih parametara vinove loze no može se primijeniti u različitim poljoprivrednim kulturama kombinacijom senzora po potrebi. Sastoji se od skupa čvorova ili objekata, kako ih naziva koncept *Internet of things,* koji u određenom vremenskom intervalu aktiviraju skupinu senzora, formiraju mjerenja, te ih prosljeđuju u bazu podataka. Ti se podaci zatim detaljnije analiziraju i grafički prikazuju na internet stranici te u ovisnosti o vremenu prikazuju klimatske uvjete koji djeluju na poljoprivredne kulture. Analizom tih podataka i provjerom kvalitete i kvantitete poljoprivrednog proizvoda kroz određeni vremenski period moguće je dobiti procjene poljoprivrednog uroda koje možemo očekivati te preventivno djelovati kako bi postigli željene rezultate. Obzirom da se radi o elektroničkim uređajima koji sadrže skupinu senzora, dodavanjem ili reduciranjem senzora prilagođavamo sustav kako bi popratili bitne klimatske čimbenike za razvoj željene kulture.

Sustav je također moguće modificirati i kroz vrijeme kreirati automatizirani sustav koji bi krajnjem korisniku pritiskom na gumb internet aplikacije omogućio udaljenim putem pokretati neke

mehaničke radnje, na primjer navodnjavanje, te na taj način djelovati ovisno o utjecaju vremena i pritom spriječiti nepovoljne utjecaje na razvoj kulture.

Korištene su elektroničke inačice matičnih pločica *Raspberry Pi 3* i *Libelium Waspmote* koje imaju podjednaku funkcionalnost u sustavu ali su drugačije izvedene. *Raspberry Pi* je mini-računalo slabijih performansi i hardverskih karakteristika za razliku od današnjih izvedenica stolnih računala, a *Libelium Waspmote* je mikrokontroler ili programabilni sklop koji je izvedenica *Arduino* pločica. Koristili smo različite vrste senzora koje su prilagođene isključivo *Arduino* tipovima elektroničkih uređaja. Izmjereni podaci su bili prikupljni u relacijsku *MySQL* bazu podataka i dokumentiranu *NoSQL* bazu podataka *MongoDB.*

U drugom poglavlju kratko definiramo planirani model arhitekture sustava, način njegova izvođenja i funkcionalne zahtjeve koje sustav treba ispunjavati.

Treće poglavlje objašnjava korištenje modula za uspostavljanje komunikacije između čvorova, programskog alata *XCTU* za konfiguriranje fizikalnih parametara signala za komunikaciju i pojašnjava princip autonomnog napajanja čvorova.

U četvrtom poglavlju definiramo *Raspberry Pi* uređaj kroz dvije osnovne namjene. Pojašnjavamo principe rada operacijskog sustava, programskih kodova, senzora i pojašnjavamo postupnu komunikaciju između čvorova i servera.

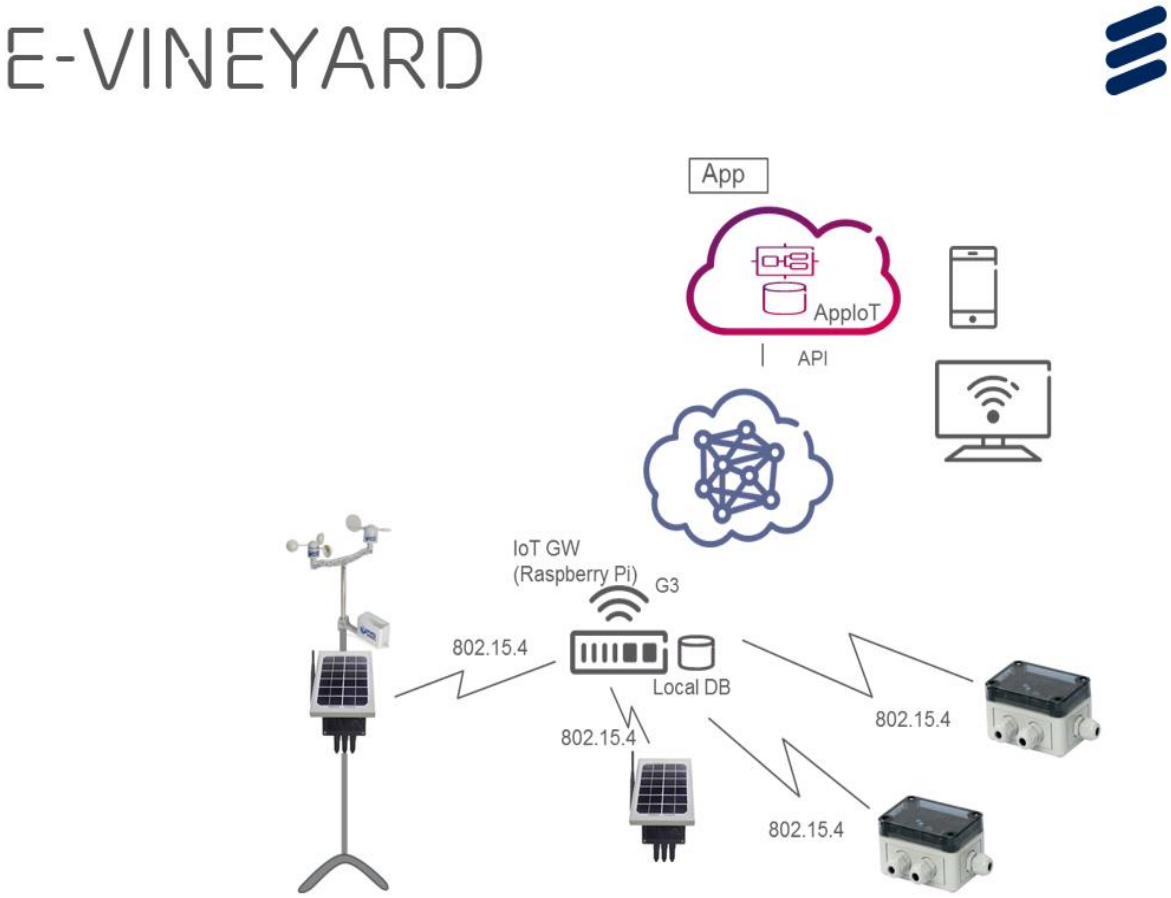
U petom poglavlju opisujemo izvedbu *Libelium Waspmote* uređaja, njegovu namjenu i primjenjeni programski kod uz korištene senzore. Uspoređujemo principe rada i razlike u performansama naspram *Raspberry Pi* uređaja.

U šestom poglavlju opisujemo konfiguraciju servera. Pojašnjavamo korištenje baza podataka *MySQL* i *MongoDB*, međusobne razlike u performansama i načine upravljanja sa podacima. Opisujemo izradu i način korištenja internet stranice za grafički prikaz pohranjenih podataka.

2 Model arhitekture sustava

Sustav je po svojoj prirodi modularan a njegovi čvorovi (uređaji) rade autonomno. Čvorovi su izvedeni u tri tipa sa dvije osnovne namjene:

- *Raspberry Pi* čvor za mjerenje
- *Libelium Waspmote* čvor za mjerenje
- *Raspberry Pi* čvor za prosljeđivanje podataka na server



Slika 2.1 Model arhitekture sustava

Proširivanje sustava se izvršava adicijski što znači da dodavanjem bilo kojeg od dodatnih čvorova u umreženi sustav, neovisno o njegovoj namjeni, ne utječemo na rad ostalih čvorova u mreži. Sustav je skalabilan i moguće ga je prenamijeniti za različite poljoprivredne kulture.

Obzirom da se radi o *point-to-point* arhitekturi sustava postoje određeni funkcionalni zahtjevi:

- Autonomija čvorova u mreži – podrazumijeva se da svaki od čvorova ima ulogu zasebnog podsustava koji radi sam za sebe
- Neovisnost čvorova u radu – neometani rad pojedinačnih čvorova, stanje ostalih čvorova ne bi trebalo ovisiti o stanju drugih čvorova u mreži
- Neprestano napajanje centralnog uređaja – ovisnost čvorova za mjerenje o centralnom čvoru koji prosljeđuje podatke na server, ukoliko dođe do kvara centralnog čvora podaci nisu u mogućnosti dospjeti na server u bazu podataka
- Neovisnost senzora – problemi jednog ili više senzora ne bi trebali utjecati na rad ostalih senzora
- Dostupnost podataka
- Nesmetana komunikacija među uređajima

Neki od tih funkcionalnih zahtjeva su postigli željene ciljeve no obzirom na različitost proizvodne namjene uređaja i načine njihove izvedbe zahtjevi nisu ispunjeni u potpunosti.

3 Komunikacija sustava

Komunikacija između uređaja je ključna stavka ovog projekta. Kako bi postigli maksimalnu autonomiju cjelokupnog sustava, smanjili potrebu da se ljudskim utjecajem regulira rad sustava i osigurali stabilnost povezanosti uređaja potrebno je jako puno testiranja. Vanjski utjecaji i reljefne karakteristike poljoprivredne kulture u kojoj se vrše mjerenja vidno utječu na performanse komunikacije sustava, stoga je potrebno pravilno i optimalno postaviti čvorove u sustavu kako bi reducirali pojavljivanje pogreški u radu sustava.

Obzirom na okolinu u kojoj se čvorovi nalaze potrebno je uspostaviti optičku vidljivost između čvorova. Prilikom testiranja primijetili smo da brzina slanja podataka i frekvencija vidno utječu na vrijeme odašiljanja poruka.

Izmjerene podatke sa čvorova za mjerenje je potrebno proslijediti u bazu podataka na serveru kako bi bili dostupni za prikaz krajnjem korisniku. Komunikaciju u sustavu stoga dijelimo u dva dijela:

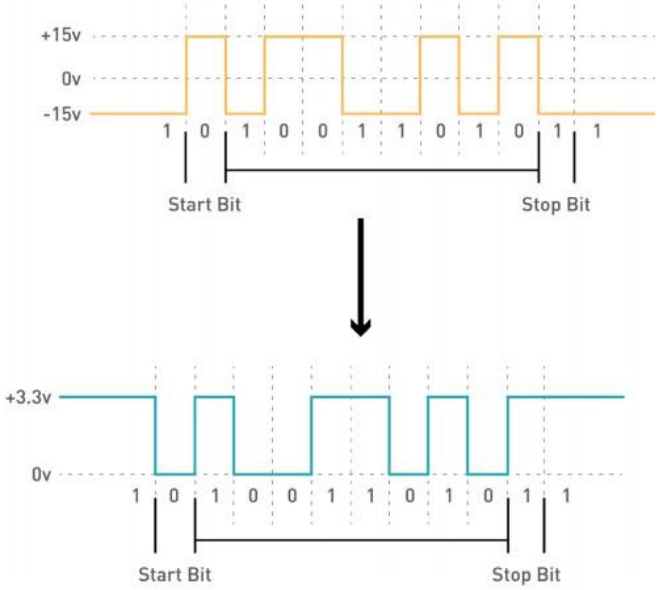
- Komunikacija od čvora za mjerenje prema centralnom čvoru za prosljeđivanje podataka na server
- Komunikacija od čvora za prosljeđivanje do servera

Svi čvorovi unutar umreženog sustava komuniciraju putem radijskog *Xbee* modula koji omogućuje bežićnu komunikaciju svih uređaja u mreži. Kako bi osigurali što bolju kvalitetu odašiljenog signala za prijenos poruka koristili smo dodatnu antenu jačine signala 5 decibela.

Slika 3.1 Xbee Pro series 2 i 5dBi Antenna [4]

Prilikom testiranja *Xbee* modula sa antenama primijetili smo da nije moguće uspostaviti komunikaciju između različith verzija modula istog proizvođača jer komponente nisu međusobno kompatibilne. Naime, *Xbee series 1* modul koristi isključivo 802.15.4. *firmware* (hrv. dodatak), a *Xbee series 2* koristi *ZigBee mesh firmware.*

Komunikacija je definirana bipolarnim protokolom RS-232 standarda za ostvarivanje serijske komunikacije a služi za sinkronizaciju prilikom slanja podataka. Podatkovni signal vrijednosti između -3V i -15V se smatra logičkim 1, a signal od 3V do 15V logičkim 0. Logička nula je startni bit nakon kojeg slijedi 8 podatkovnih bitova, zatim opcionalno paritetni bit i na kraju bit zaustavljanja.



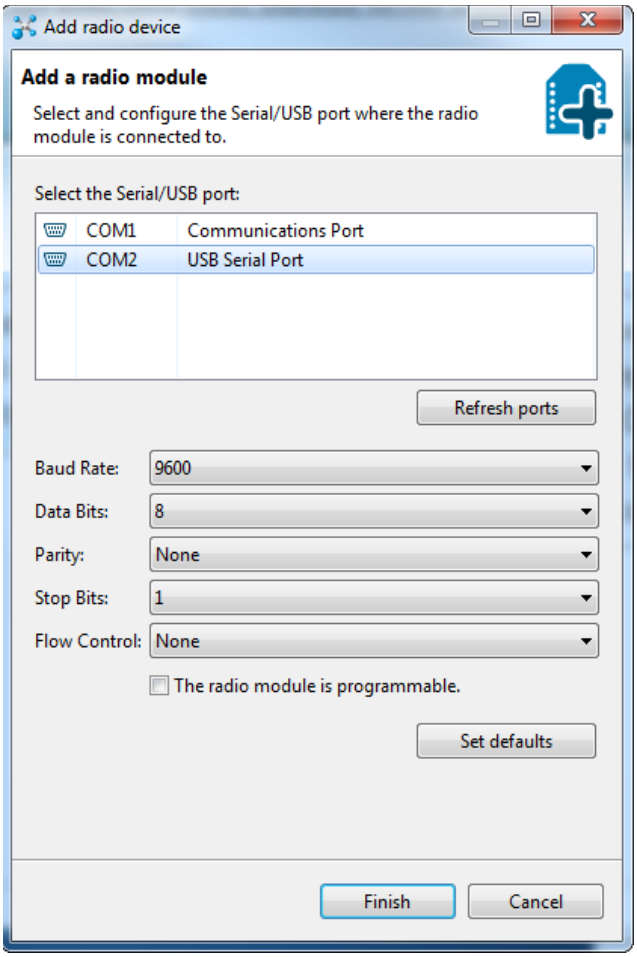
Slika 3.2 RS-232 konverzija voltaže [8]

3.1 XCTU

Za konfiguriranje *Xbee* modula koristili smo *XCTU* programski alat koji sadržava niz mogućnosti za definiranje parametara slanja podataka, modova za korištenje, nadogradnju *firmwarea*, definiranja imenovanja i niz ostalih postavki.

Xbee modul dolazi sa programabilnom matičnom pločicom (engl. „*Explorers*“) koja služi kao sučelje između računala i modula, sadržava procesor za konfiguraciju *Xbee* modula i slanje

podataka sa računala. Spajanje modula se izvršava putem USB priključka na računalo koje ima instaliran alat *XCTU*. Prilikom spajanja modula, alat prepoznaje USB pristupnu točku na koju je spojen modul.



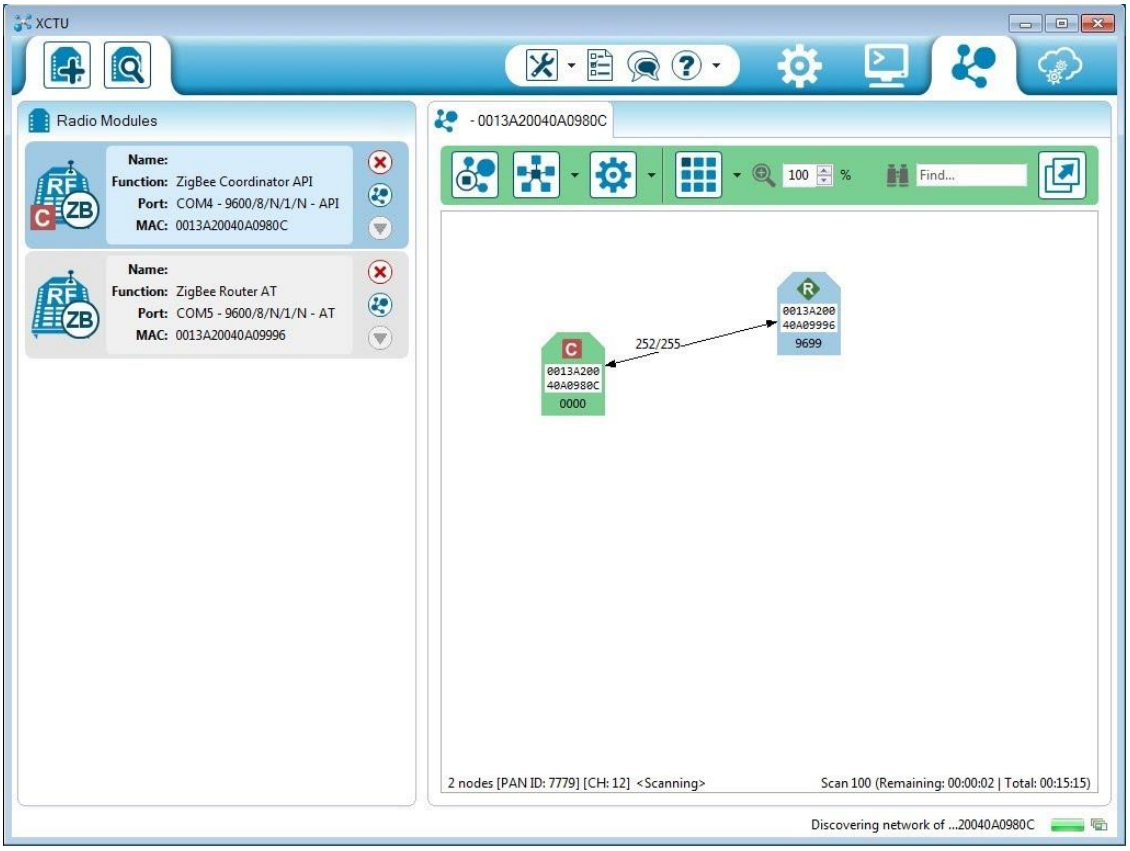
Slika 3.3 XCTU prepoznaje zauzeće USB portova, ručno dodajemo one zauzete Xbee modulom

Odabirom *porta* (hrv. pristupne točke) na koji je spojen modul, *XCTU* čita njegovu početnu konfiguraciju. Od fizikalnih karakteristika signala za nas je bitno postaviti isti kanal za komunikaciju na svim modulima (kanal je po početnim postavkama postavljen na vrijednost „C“), te promijeniti parametar PAN ID koji jedinstveno identificira mrežu za komunikaciju. Jako je bitno postaviti optimalnu brzinu slanja podataka (engl. *baud rate*), u našem slučaju *master clock* (hrv. upravljački sat) radi na frekvenciji od 8MHz što se pri brzini slanja od 9600 bps (8 podatkovnih bita uz jedan paritetni i jedan zaustavni bit) pokazalo kao optimalan omjer brzine, frekvencije i kvalitete signala pri čemu je potrošnja struje otprilike 1.72 mA.

Postoje tri osnovna tipa rada modula:

- C – *coordinator*
- R – *router*
- E – *end device*

U našem slučaju korisitmo dva *tipa*, E za čvorove koji mjere i C za one čvorove koji prosljeđuju podatke na server, koordiniraju komunikaciju i osluškuju odašiljanja čvorova za mjerenje.



Slika 3.4 Prikaz umreženih čvorova u programskom alatu XCTU za komunikaciju putem Xbee modula

3.2 Napajanje čvorova

Princip samostalnog napajanja čvorova temelji se na tri komponente:

- Akumulatorska baterija
- Kontroler za solarno napajanje

- Solarna ploča

Cilj je kreirati sustav napajanja čvorova koji će prikupljati sunčevu energiju pomoću solarnih ploča i preko kontrolera puniti akumulatorske baterije koje napajaju čvorove.

Za prikupljanje solarne energije koristili smo solarne ploče (eng. *panel*) od 6V i maksimalne snage 10W postavljene na svaki od čvorova. Usmjerene su direktno prema nebu tako da ploče najoptimalnije dohvaćaju solarnu energiju. Solarne ploče zatim spajamo na napajanje i uzemljenje solarnog kontrolera koji sadrži 3 priključka: za solarnu ploču (ulaz), za akumulatorsku bateriju i za *Raspberry Pi* (izlaz). Kasnije tijekom razvoja projekta smo shvatili da kontroler prima minimalno 6V napona na ulazu, a kada bi došlo do pada napona na ulazu, ne bi bio u mogućnosti dalje nastaviti napajati bateriju.

Slika 3.5 Kontroler solarnog napajanja

Akumulatorska baterija korištena na *Raspberry Pi* uređaju je bila kapaciteta 12Ah (amper-sati) i napona 6V što bi teoretski trebalo biti dovoljno no u praksi se nije pokazalo točnim. Unatoč razumijevanje pogreške prilikom odabira solarnog kontrolera, potrošnja *Raspberry Pi*-a je ostala

ista, baterija je mogla duže davati struju kada joj pada napon, no i dalje nedovoljno za potpuno samostalni napajanje *Raspberry Pi* čvora.

Libelium Waspmote je koristio bateriju napajanja 3.7V i kapaciteta 6600 mA (miliamper-sati) a sustav je autonomno neusporedivo duže od *Raspberry Pi* čvorova.

Kroz daljnji razvoj sustava čvor za prosljeđivanje podataka smo spojili na konstantno napajanje i čvor je u daljnjem nastavku rada bio u stabilnom stanju.

Slika 3.6 Akumulatorska baterija Waspmote uređaja

4 Raspberry Pi

Raspberry Pi je mini računalo veličine kreditne kartice koji primjenu nalazi u raznim elektroničkim projektima ali je namijenu pronašao i u manje zahtjevnim zadacima poput igranja jednostavnih igara, uredskim poslovima vezanim uz dokumente, surfanju internetom te ostalim zadacima koje obavljaju stolna računala.

Razvijen je u Velikoj Britaniji u sklopu „*Raspberry Pi Organization"* organizacije koja je krenula u proizvodnju s ciljem promicanja računalne znanosti u školama. Primjena se uskoro proširila izvan ciljanog tržišta a najviše u robotici, te je prodaja već u samim početcima dosegnula značajne rezutlate, proglašen je 2015. godine najprodavanijim računalom u Velikoj Britaniji.

Prva je verzija uređaja *Raspberry Pi 1 model B* proizvedena 2012. godine, pa zatim model *Raspberry Pi 1 A*, 2014. godine. S vremenom su proizvedeni modeli *Raspberry Pi Zero* sa reduciranim brojem pinskih ulaza i izlaza, te modeli 2 i 3 sa progresivno pobljšanim hardverskim komponentama. Posljednja distribucija je model *Raspberry Pi 3 B* koji se na tržištu pojavio početkom 2016. godine.

Prilikom izrade ovog projekta, koristili smo posljednje i hardverski najsnažnije izdanje iz serije *Raspberry Pi* pločica „*Raspberry Pi 3 model B"*.

U fazi eksperimentiranja i testiranja stabilnosti sustava testirali smo ponašanja pločica u raznim uvjetima i uzročno-posljedično djelovanje na ponašanje operacijskog sustava, senzora i skripti koje se izvode u radu. Bitan element ovog projekta jest reducirati fizičku interakciju korisnika sustava, odnosno automatizirati sve procese u sustavu tako da se sustav izvršava autonomno uz uvjet da su uređaji konstantno pri napajanju. Pritom postoje određena ograničenja koje ćemo navesti u nastavku dokumenta.

Hardverske značajke uređaja:

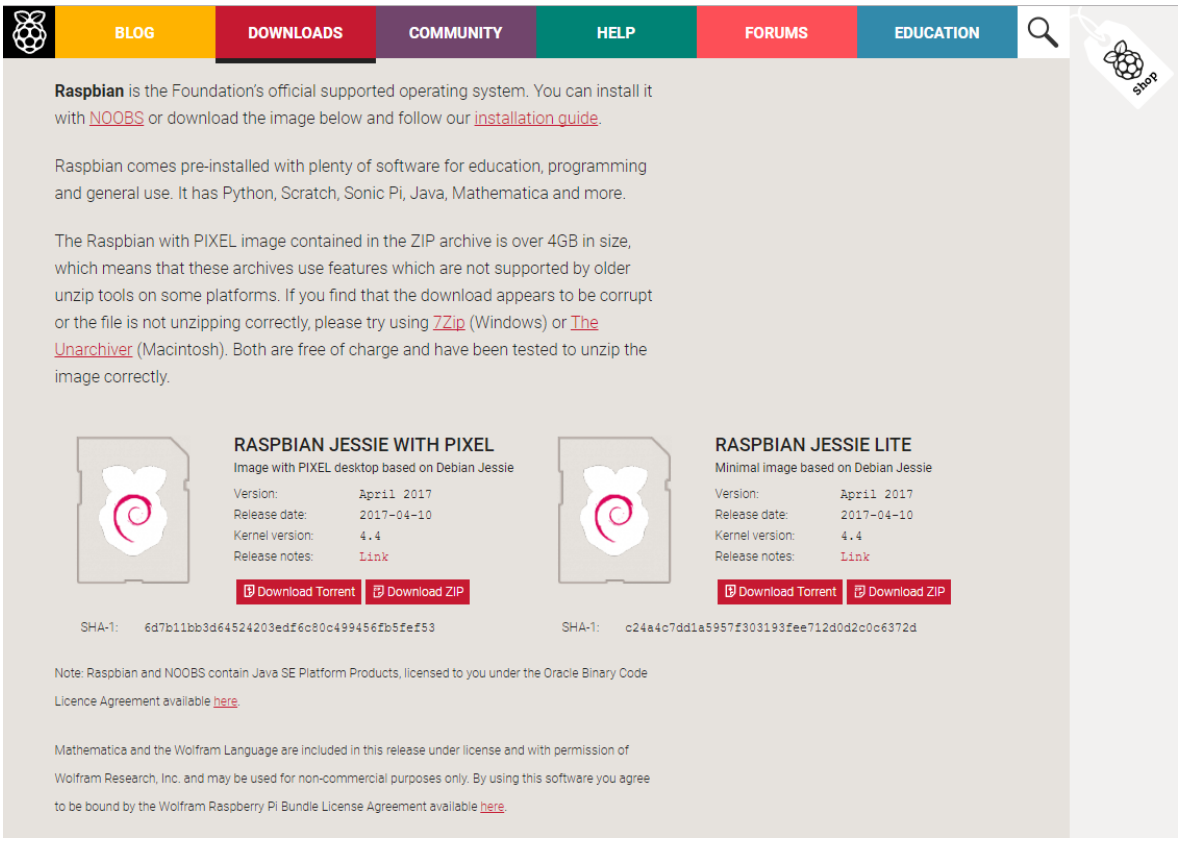
- 1 GB RAM memorije
- 1.2 GHz 64-bit četverojezgreni procesor ARMv8
- 802.11n Wireless LAN
- Bluetooth 4.1

- 4 USB *port*a
- 40 GPIO pinova
- Full HDMI *port*
- Ethernet *port*
- 3.5 mm priključak (*jack*) za audio i video
- Micro SD slot

Slika 4.1 Komponente Raspberry Pi 3 model B pločice [11]

Prvi u nizu zadataka prilikom konfiguriranja *Raspberry Pi* čvorova jest odabrati i instalirati optimalni operacijski sustav. Na službenim stranicama organizacije moguće je pronaći te preuzeti različite inačice operacijskih sustava koje su prilagođene *Raspberry Pi* uređajima u smislu niske potrošnje energije i manje zahtjevnih u smislu potrošnje hardverskih resursa uređaja. Obzirom da se napajanje našeg sustava temelji na litijskim baterijama koje se poprilično brzo troše odabrali smo *Raspbian Jessie with Pixel* operacijski sustav zbog vidljivog grafičkog sučelja (za razliku od *Lite* verzije koja kao prikaz ima isključivo i samo *Linux*-ov alat „*Terminal*“ za unos komandi) što nam je pomoglo prilikom pisanja skripti za izvođenje. To je jedan od najpopularnijih operacijskih sustava kod *Raspberry Pi* pločica i stoga ima jako dobru dokumentaciju na internetu te je jako

dobro podržan od strane ostalih korisnika što nam je jako pomoglo pri rješavanju problema na koje smo naišli prilikom izrade sustava.



Slika 4.2 Odabir inačice operaijskog sustava za Raspberry Pi pločice [12]

Postupak instalacije operacijskog sustava je također prisutan i jako dobro opisan na službenim stranicama organizacije a svodi se na:

- preuzimanje *.zip* datoteke koja sadrži *image* ili „sliku“ operacijskog sustava sa službene stranice *Raspberry Pi* organizacije
- preuzimanje jednog od ponuđenih softverskih paketa, na primjer Etcher SD
- umetanje *imagea* na SD karticu te njeno umetanje u *Raspberry Pi port*

Operacijski sustav se pokreće prilikom kontakta uređaja sa napajanjem a dolazi sa već predinstaliranim paketima poput programskih jezika *Python*, *Scratch*, alata *Terminal* i ostalih paketa i alata dovoljnim za početno korištenje uređaja. Prilikom pokretanja sustava potrebno je

upisati korisničke podatke (pi za *username* i raspberry za *password*), no moguće je reducirati unos tih podataka prilikom pokretanja sustava, što je nama jako bitno u smislu reduciranja korisničke interakcije i automatizacije daljnjih procesa, na način da se izmjeni konfiguracija uređaja pokretanjem komande u terminalu `raspi-config`. Tada nam se prikaže grafičko sučelje gdje odabiremo akcije „*Enable Boot to Desktop*“ a zatim „*Desktop Login as user pi at the Graphical Desktop*“, praktično tada operacijski sustav promijeni skriptu `/etc/inittab` odnosno primijeni dodatak `--autologin` za komandnu liniju `1:2345:respawn:/sbin/getty --autologin {USERNAME} --noclear 38400 tty1`. Tako se prilikom svakog sljedećeg pokretanja operacijski sustav pokreće automatski sve do faze upravljačke ploče pritom pokrečući inicijalizacijske skripte.

4.1 CRON

Uz automatizirano pokretanje operacijskog sustava koristimo i softverski alat *cron* (engl. *cron table*) koji nam služi za automatsko pokretanje procesa prilikom pokretanja operacijskog sustava. Kod oba *Raspberry Pi* čvora, onog za mjerenje koji sadrži senzore i onog za prosljeđivanje podataka, konfiguracija uređaja i operacijskog sustava je ista. Temelji se na pokretanju što je minimalnog broja pozadinskih procesa sustava kako bi se potrošnja energije napajanja svela na minimum.

Cron je alat koji poziva konfiguracijsku skriptu komandom u *bash*-u `crontab -e`. Prilikom definiranja skripti za izvođenje kod svakog pokretanja sustava potrebno je prije naredbe izvođenja skripte u *python*-u definirati naredbu `@reboot` kako bi se izvršila komanda, primjerice:

```
@reboot nohup python /home/pi/Desktop/skripta_za_mjerenje.py &.
```

Kako bi proces postavili da se izvršava u pozadini, a ne u prioritetnom terminalu, potrebno je unijeti naredbu `nohup` prije pokretanja naredbe za izvršavanje skripte, na početku i `&` znak na kraju komande.

Također, ukoliko se izvodi više takvih skripti, potrebno je pripaziti na poredak skripti kako bi redoslijed izvođenja bio valjan. Obzirom da je to skripta napisana u *bash* programskom jeziku potrebno je pripaziti da se ostaviti prazan redak na kraju skripte jer po prirodi tog programskog jezika on tako prepoznaje kraj komandi za izvršavanje.

4.2 Čvor za prosljeđivanje podataka na server

U ranim fazama izrade projekta pokušali smo modificirati čvor za prosljeđivanje da ima dvostruku namjenu, pridjeljujući mu pritom i onu čvora za mjerenje, no obzirom na probleme napajanja čvorova i daljnji koncept kod *Raspberry Pi* uređaja to nije bilo moguće. Ovo je prioritetni čvor koji služi kao poveznica cijelog sustava, stoga je ovaj čvor pri konstantnom napajanju kako bi bili sigurni, ukoliko je sve u redu sa njegovim radom, da će podaci sa čvorova za mjerenje stići na server.

Skripta sadrži biblioteke:

- *serial* – služi za korištenje Xbee modula i kreiranja serijske komunikacije
- *socket* – služi za kreiranje *HTTP* zahtjeva prema serveru za slanje podataka
- *datetime* – služi za dodavanje vremena i datuma na kraj primljenog JSON podatka
- *traceback* – služi za ispis logova i detaljniju analizu ukoliko dođe do pogreške u radu

Nakon uključivanja biblioteka, potrebno je definirati parametre za komunikaciju Xbee modula a to izvršavamo pridjeljujući varijabli `ser` izvršenu komandu `serial.Serial('dev/ttyUSB0', 9600, timeout=5)`. Zatim primamo podatke pomoću metode `ser.readline()`.

```
1.  reading = ser.readline()
2.  now = datetime.datetime.now().strftime("%Y-%m-%d %H:%M")
3.  reading = reading + ',"date": "%s"}' % now
```

Slika 4.3 Obrada JSON podatka i dodavanje parametra vremena

Potrebno je definirati varijable *host* (hrv. domaćin) *i port* koje prosljeđujemo kao parametre na mrežu, te da kreiramo *socket* (hrv. utičnica) za uspostavljanje *HTTP* konekcije sa serverom.

```
1.  while True:
2.      print json.loads(message)
3.      serversock = socket.socket()
4.      addr = (host, port)
5.      try:
6.          serversock.connect((addr))
7.          print 'Waiting For Connection..'
8.          print serversock.recv(1024)
9.
10.         print ('Connection Established From: ', addr)
11.         serversock.sendall(message)
12.     except Exception, e:
13.         traceback.print_exc()
```

Slika 4.4 Programski kod za prosljeđivanje mjerenja na server

Kao što se vidi na prethodnoj slici podaci su poslani tek nakon što dobijemo povratnu informaciju od strane servera.

4.2.1 UMTSkeeper i Sakis 3G

Čvor za prosljeđivanje podataka koristi internet konekciju putem USB mobilne kartice za spajanje na 3G mrežu i prosljeđivanje podataka na server. Iz tog razloga potrebno je konfigurirati da se prilikom podizanja operacijskog sustava kreira potrebna konekcija na internet, stoga korisitmo alate *UMTSkeeper* i *Sakis 3G* kako bi taj proces automatizirali.

Prvo je potrebno instalirati *UMTSkeeper* alat koji neprestano provjerava da li postoji konekcija na Internet i u slučaju gubljenja konekcije ponavljano pokušava uspostaviti ponovnu konekciju. Pomoću predinstaliranog alata *wget* dohvaćamo, a zatim raspakirajemo dohvaćeni paket i instaliramo navedeni alat komandama:

```
wget "http://mintakaconciencia.net/squares/umtskeeper/src/umtskeeper.tar.gz"
md5sum umtskeeper.tar.gz
tar -xzvf umtskeeper.tar.gz
chmod +x sakis3g umtskeeper resetusb
```

Kako bi inicijalno postavili uspostavljanje konekcije na internet prilikom podizanja sustava čvora moramo također unutar *Cron*-a definirati komandu:

```
sudo ./umtskeeper --sakisoperators "USBINTERFACE='0' OTHER='USBMODEM'
USBMODEM='12d1:140c' SIM_PIN='1234' APN='CUSTOM_APN' CUSTOM_APN='provider.com'
APN_USER='0' APN_PASS='0'" --sakisswitches --log --nat 'no'
```

gdje su definirani sakis operatori:

- USBMODEM – identifikacijski parametar USB priključka od mobilne kartice (nakon ukopčavanje moguće ga je pročitati `lsusb -v` komandom
- SIM_PIN – pin mobilne kartice unutar USB priključka
- APN – (engl. *Access Point Network*) ime pristupne točke između mobilne kartice i pružatelja usluge interneta

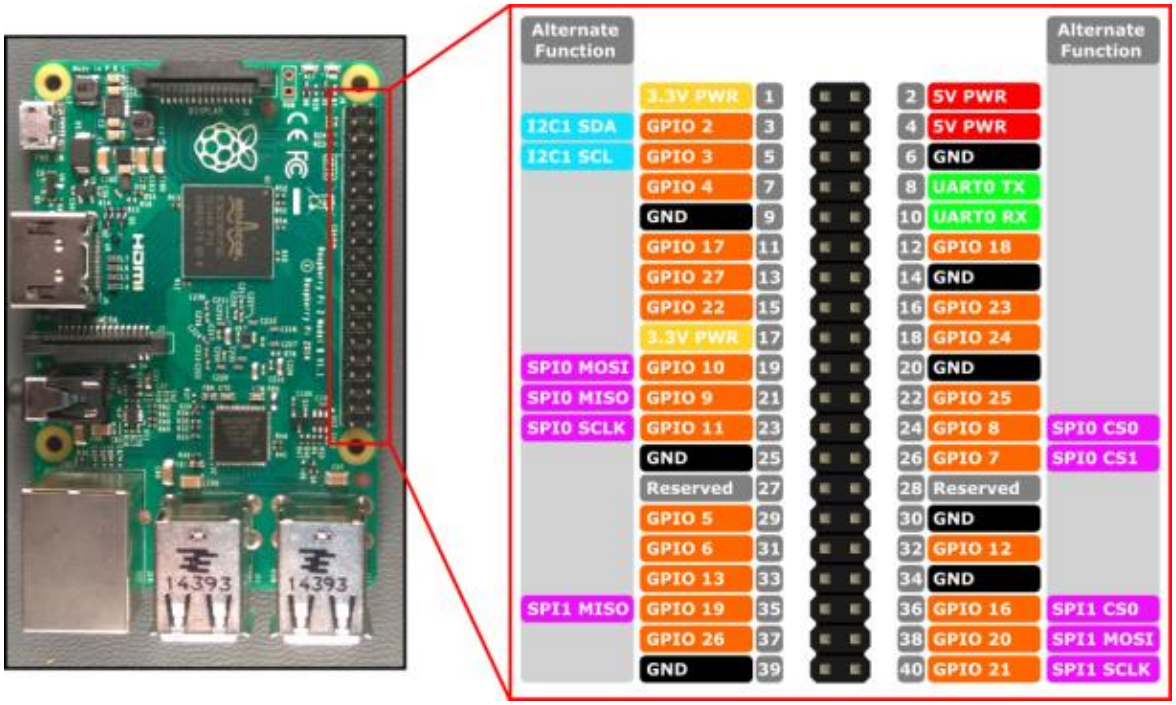
Ova se komanda stavlja na početak skripte za izvođenje u cron-u. Obzirom da se spajanje na internet izvršava poprilično brzo nema potrebe za kreiranjem potrebnog vremenskog razmaka između te komande i komande za pokretanje skripte za mjerenje i slanje podataka.

4.3 Čvor za mjerenje i njegovi senzori

Fizička interakcija između senzora i pločica uređaja se izvršava pomoću *pin*-ova kojih na našoj izvedenici Raspberry pi uređaja ima 40. Oni služe kao sučelje za spajanje dodataka na uređaj a često se definiraju kao GPIO (eng. „*general purpose input/output"*) pin-ovi. Obzirom na njihovu ulogu i način spajanja dijele se na:

- *Ground* – električno povezani pinovi za uzemljenje dodataka (0V), ima ih 8 na našem modelu Raspberry-a
- GPCLK (*general purpose clock*) – služe za postavljanje statičke frekvencije signala bez kontrole softverom, ima ih 3
- JTAG (*joint test action group*) – standardizirana sučelja za *debug*-iranje Raspberry Pi uređaja, ima ih 11
- 1-*wire* – služi za komunikaciju sa jednom žicom, uz uzemljenje, ima 1 takav na našem Raspberry pi modelu
- PCM (*pulse code modulation*) – služi za digitalno prezentiranje analognog signala, ima ih 4
- SDIO – SD *host* i eMMC sučelje za korištenje microSD kartica, ima ih 6
- I2C – koristimo ih u *WiringPi* ili BCM *mode*-u, korisni su zbog mogućnosti komuniciranja sa različitim tipovima perifernih uređaja i verificiranja adresa vanjskih uređaja, pokreću se korištenjem *smbus* biblioteke u programskom jeziku *python*, njihovi se pokretači (*driver*-i) omogućuju u `/boot/config.txt` datoteci dodavanjem linija `i2c-dev` i onom koja se odnosi na pokretač senzora „`i2c` – korišteni senzor", uz isključivanje komentiranjem linije „`blacklist i2c` – korišteni senzor" unutar `/etc/modprobe.d/raspi-blacklist.f` datoteke, također je potrebno instalirati korisnosti *python-smbus* i *i2c-tools* za korištenje u programskom jeziku *python*. ima ih 4

- SPI – poznati kao 4-*wire* pinovi za selektiranje *chip*-ova, koriste *spidev* biblioteku u programskom jeziku *python*, njihova se funkcionalnost pokretača definira unutar `/etc/modules` datoteke dodavanjem linije `dtparam=spi=on`, ima ih 11
- UART (universal asynchronus receiver/transmitter) – za korištenje protokola za serijsku asinkronu komunikaciju u sekvencijlanom *mode*-u, ima ih 2, TXD i RXD



Slika 4.5 GPIO pin-ovi Raspberry Pi 3 model B pločice [3]

Moguće je koristiti softverski paket *WiringPi* za referenciranje i jednostavnije korištenje BCM pinova kako bi sučelje bilo slično onom korištenom kod *Arduino* pločica. Prilikom kodiranja izvedbe mjerenja i slanja podataka koristili smo *python* programski jezik jer većina senzora ima gotove module i biblioteke za olakšano korištenje na *Raspberry Pi* pločicama.

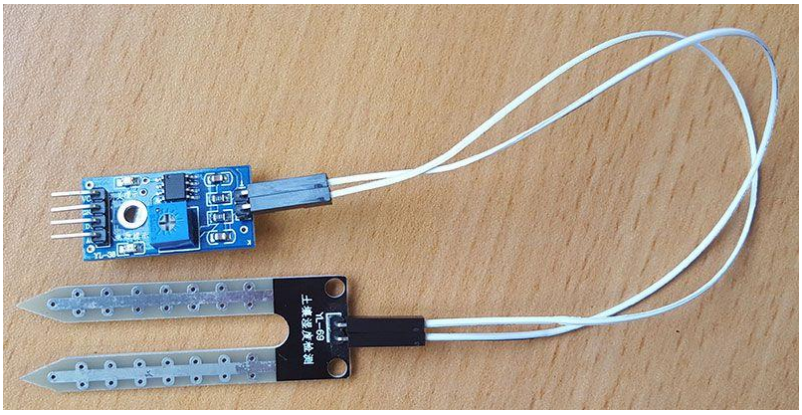
4.3.1 Korišteni senzori i njihova imenovanja

Prilikom izrade projekta eksperimentirali smo sa raznim senzorima i pokušavali postići izvedivu konfiguraciju. Senzori koriste mnoštvo *pin*-ova i tijekom izrade naišli smo na dosta problema vezanih uz nestabilnost cjelokupnog sustava, no bez obzira na njihovu različitost u implementaciji

povezanost je moguća ali stabilnost nije osigurana. *Raspberry Pi* nema ugrađene analogne ulaze stoga je potrebno koristiti ADC (eng. „*analog to digital converter*“) konverter.

Korišteni senzori:

- *YL-69* – Senzor za mjerenje vlažnosti tla čije se vrijednosti izražavaju u postotcima. To je analogni senzor koji koristi multiplekser MCP3008, te ADC konverter. Spaja se na pinove VCC +3.3V, GND i output.



Slika 4.6 YL-69 senzor za mjerenje vlažnosti tla [13]

Prilikom njegova uključivanja u programski kod potrebno je koristiti vanjsku biblioteku *spidev* koja aktivira SPI (eng. „*Serial to Parallel Interface*“) sučelje za definiranje korištenih kanala multipleksera. Varijabli pridijelimo metodu `spidev.Spidev()` te joj postavimo *Bus* i *Device* vrijednosti prilikom korištenja `spi.open()` metode. Zatim izvršavamo SPI transakciju i spremanje bitova, filtriranje podatkovnih bitova te vrijednost od 0 – 1023 prikazujemo u postotcima.

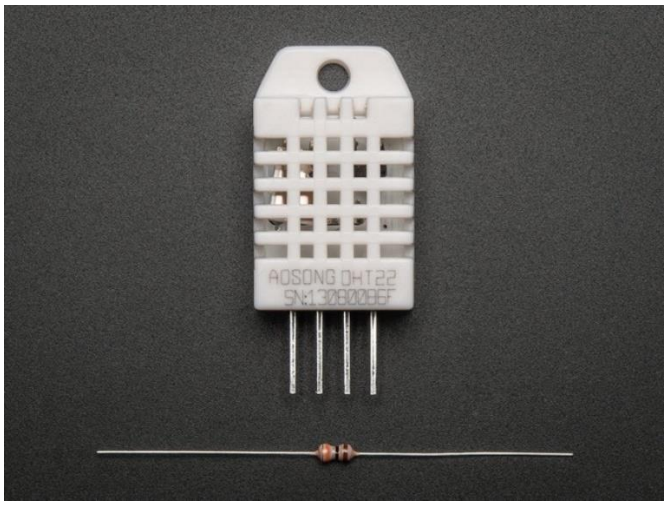
```
r = spi.xfer([1, (8+channel) << 4,0])

adcOut = ((r[1]&3) << 8) + r[2]
percent = int(round(adcOut/10.24))
return percent
```

Slika 4.7 Filtriranje podatkovnih bitova

- *DHT22* – Senzor koji sadrži senzor za temperaturu i vlažnost zraka. Koristi podatkovni pin, pin za uzemljenje i pin za napajanje VCC +3.3V. Potrebno je spojiti i otpornik od 4.7 kΩ ili 10 kΩ.

Za korištenje navedenog senzora moguće je preuzeti „Adafruit DHT“ biblioteku. Prije pozivanja naredbe za uključivanje senzora potrebno je inicijalizirati *GPIO_DHT22* varijablu koja označava pin za *output*. Potrebne su nam dvije varijable, kojima pridjeljujemo izmjerene vrijednosti temperature i vlažnosti zraka kao rezultate izvršenja naredbe `dht.read(dht.DHT22, GPIO_DHT22)`. Temperatura je iskazana u °C dok je vlažnost zraka iskazana u postotcima.

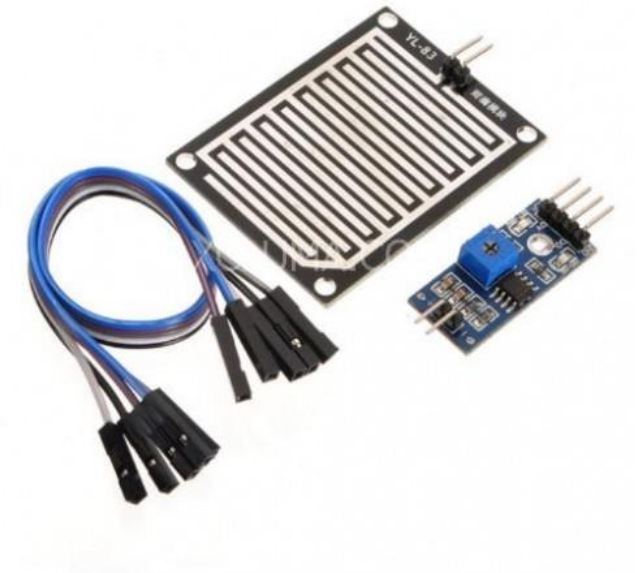


Slika 4.8 DHT22 senzor za mjerenje temperature i tlaka zraka [14]

- *TSL2561* –Digitalni 16-bitni senzor za mjerenje količine svijetlosti. Za korištenje ovog senzora potrebno je instalirati alat i2c koji sadrži pokretače (eng. *Driver*) za SCL i SDA pinove koje koristi osim VCC +3.3V pina i pina za uzemljenje.
  Nakon instaliranja alata ukoliko je spojen senzor, naredbom `i2c detect y -1` je moguće ispisati tablicu koja pokazuje adrese koje senzor koristi za korištenje. Uključimo li biblioteku *TSL2561* preuzetu sa *github*-a moguće je metodom `tsl.readLux()` dohvatiti valjane vrijednosti iskazane u mjernoj jedinici „lux“.

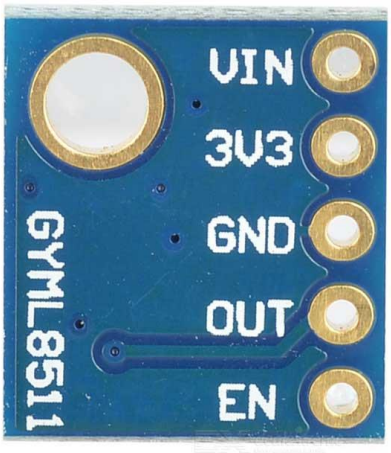
Slika 4.9 TSL2561 senzor za mjerenje količine svijetlosti [15]

- *YL-83* – Senzor za mjerenje vlažnosti lista. Također pripada skupini senzora koja koristi MCP3008 multiplekser stoga je izvedba u programskom kodu prilično jednaka onom kod *YL-69* uz korištenje drugog kanala. Mjerna jedinica je postotak.



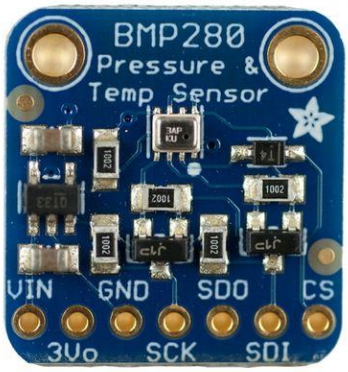
Slika 4.10 YL-83 senzor za mjerenje vlažnosti lista [16]

- *ML8511* – Senzor koji služi mjerenju UV zračenja. Također koristi pokretače i2c za upravljanje a koristili smo i vanjsku biblioteku *sensor.ML8511* preuzetu sa *github*-a. Mjerenje izvršavamo pomoću metode `ml.uv().mW_cm2`. Mjerna jedinica je mW/cm^2.



Slika 4.11 ML8511 senzor za mjerenje UV zračenja[17]

- *BMP280* – Senzor sadrži barometar i termometar. Od pinova može koristiti VCC +3.3V ili +5V, uzemljenje, te i2c pinove SCK i SDI za pokretanje *driver*-a. Koristili smo vanjsku biblioteku preuzete sa *Github*-a, *Adafruit_BME280*. Iako se radi o drugom senzoru namjena i princip korištenja su isti, pozivamo metode `sensor.read_temperature()` i `sensor.read_pressure()`.

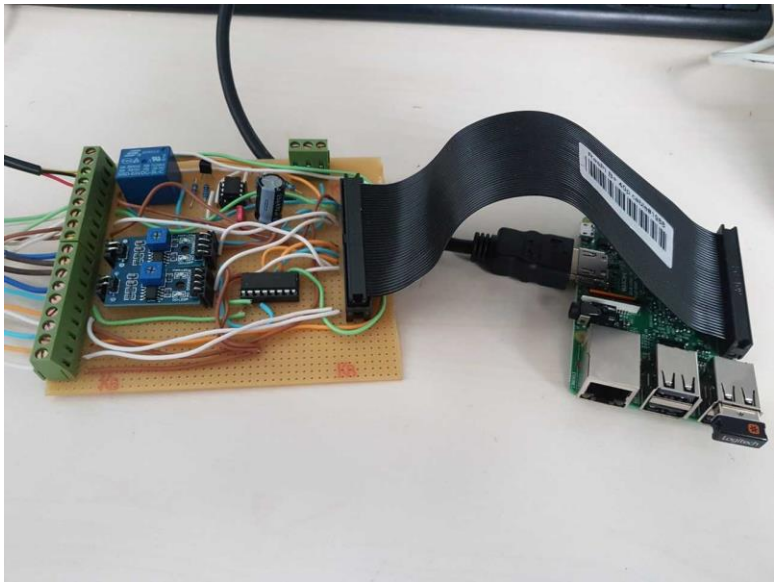


Slika 4.12 BMP280 senzor za mjerenje tlaka i temperature zraka [18]

4.3.2 Mjerenje i slanje podataka

Pomoću skripte napisane u *python*-u aktiviramo senzore koji obavljaju mjerenja, formiramo *JSON* tip podatka u obliku niza koji sadrži parove ID senzora i vrijednost, a zatim taj niz podataka prosljeđujemo dalje prema centralnom Raspberry Pi-u.

Skripta sadrži veliki broj senzora i pojedini senzori moraju koristiti iste *pin*-ove za pravilno funkcioniranje stoga smo bili prisiljeni koristiti multiplekser MCP3008 u izradi, a kasnije i u konfiguriranju skripte, kako bi stvorili kanale za naizmjenično korištenje pojedinih pinova. Osim multipleksera implementirali smo i tipku za ponovno pokretanje *Raspberry Pi* pločice u slučaju nekakve pogreške.



Slika 4.13 Model Raspberry Pi čvora za mjerenje sa svim spojenim senzorima

Početak skripte sadrži biblioteke koje služe kao funkcionalni dodaci za korištenje nekih alata i dodataka- Uključujemo ih pomoću naredbe `import` a to su:

- *serial* – služi za komunikaciju putem serijskog *porta*, komunikaciju uz pomoć *XBee* modula
- *time* - služi za kreiranje timeout-a prilikom mjerenja senzora i prosljeđivanja podataka
- *json* – omogućuje formiranje *JSON* podatka
- *os* i *sys* – kako bi mogli pokrenuti gašenje operacijskog sustava nakon slanja podatka
- *traceback* – kako bi u slučaju pogreške imali potrebne log-ove za jasnije razumijevanje nastalog problema
- *RPi.GPIO* – kako bi jednostavnije pozivali pinove na koje su spojeni senzori u programu

- *Adafruit_DHT, Adafruit_GPIO.I2C, spidev, TSL2561, w1thermsensor, sensor.MCP3004, sensor.ML8511, Adafruit_BME280* – biblioteke ili paketi biblioteka koje su prilikom korištenja senzora razvili korisnici s ciljem olakšavanja korištenja, preuzeti s Github servisa

Slika 4.14 Raspberry Pi čvor za mjerenje postavljen u vinogradu

Nakon definiranja biblioteka, inicijaliziranja varijabli za pohranu vrijednosti mjerenja senzora, te definiranja potrebnih vrijednosti pinova i kanala, pozivamo prethodno definirane metode za aktiviranje senzora. Nakon svakog izvršenog mjerenja provjeravamo valjanost svakog od mjerenja (isključujemo pojavljivanje „*None*“ vrijednosti i provjeravamo da li su svi senzori izvršili mjerenja) te pozivamo prethodno definiranu funkciju koja traži medijan vrijednost. Prilikom svake sesije mjerenja pokušavamo kod svakog aktiviranja svih senzora provjeritit da li su izmjerili valjanu vrijednost. Pamtimo pet valjanih rezultata mjerenja, te uzimamo medijan vrijednost za svaki od senzora, Razlog tomu je što kod niske razine napajanja *Raspberry Pi* drastično oscilira kod naponskih razina pinova, pogotovo kada je baterija za napajanje skoro prazna. Poremeti se

pravilan rad operacijskog sustava što može dovesti do nepravilnog mjerenja senzora a nerijetko i do pregorijevanja pojedinih senzora.

```
1.  def median(array):
2.      array.sort()
3.      return array[len(array)/2 + 1]
```

Slika 4.15 Funkcija za filtriranje medijan vrijednosti iz mjerenja

Nakon izvršenih mjerenja potrebno je formirati *JSON* tip podatka koji sadrži uređene parove ključ-vrijednost pomoću komande `json.dumps()`. Takav *JSON* podatak zatim proslijeđujemo na čvor za prosljeđivanje na server.

Vremenski interval za mjerenje i odašiljanje poruke je 15 minuta, stoga nakon poslane poruke, iz sigurnosnog razloga postavljamo vrijeme čekanja od 60 sekunda za gašenje svih senzora i stabiliziranje sustava a zatim pozivamo *bash* komandu `os.system(''shutdown -h now'')` pomoću koje „ublaženo“ gasimo sustav.

```
1.  try:
2.          send = json.dumps ({'RasID': 'a1',
3.                              'Temperature': "{0:0.1f}".format(t),
4.                              'Humidity': "{0:0.1f}".format(h),
5.                              'Wetness': "{0:0.1f}".format(rain),
6.                              'Soil_Humidity': "{0:0.1f}".format(soilHum),
7.                              'Soil_Temperature': "{0:0.1f}".format(soilTemp),
8.                              'Lux': "{0:0.1f}".format(lux),
9.                              'UV': "{0:0.1f}".format(uv),
10.                             'Air_Temp_BMP280': "{0:0.1f}".format(degrees),
11.                             'Pressure_BMP280': "{0:0.2f}".format(hectopascals)})
12.
13.         send = send[:-1]
14.         print send
15.         print ser.write
16.         ser.flush()
17.         time.sleep(60)
18.         os.system("shutdown -h now")
19.     except Exception:
20.         pass
```

Slika 4.16 Postupak za formiranje JSON podatka kojeg šaljemo na server i postupno gašenje čvora

Sustav se ponovno podiže pomoću releja koji šalje impuls, pozitivan napon, svakih 15 minuta i na taj način pokreće operacijski sustav.

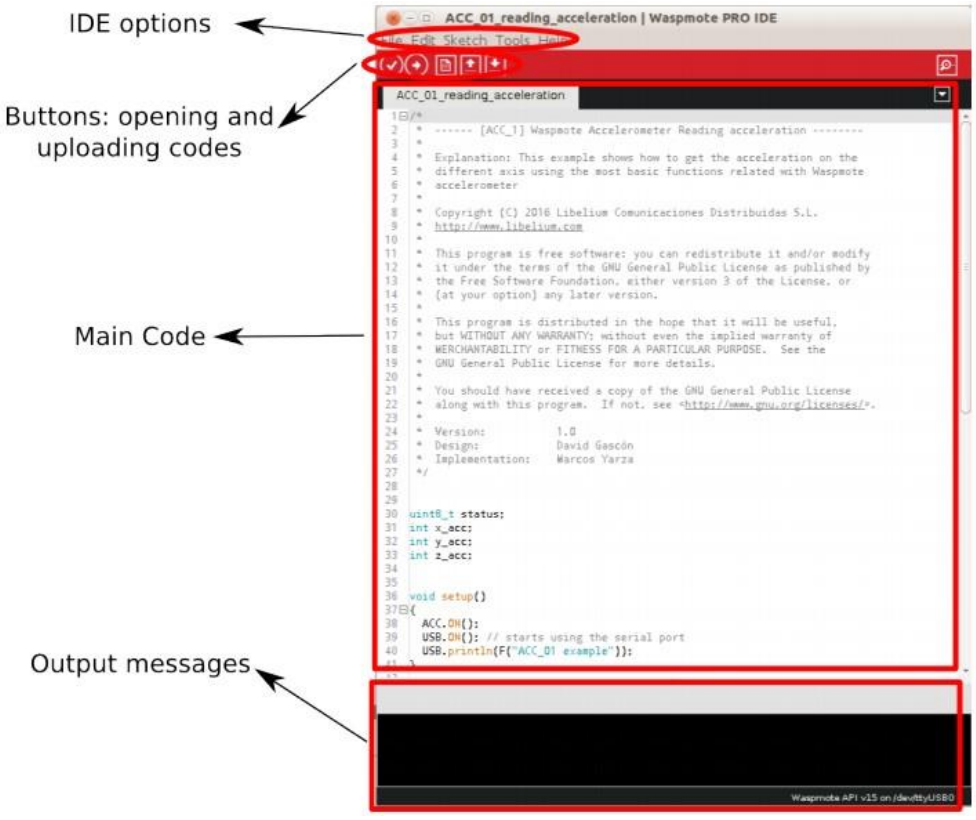
Na samom početku projekta razvila se ideja o serijalizaciji proizvodnje čvorova za mjerenje samo dodavanjem i spajanjem senzora. Nažalost, ideja nije bila moguća jer se kopiranjem takve slike

(eng. *image*-a) na drugi *Raspberry Pi* uređaj operacijki sustav nije mogao podignuti. Razlog nismo uspjeli dokučiti u tako kratkom vremenskom periodu izrade projekta no pretpostavljamo da je valjanim konfiguriranjem slike operacijskog sustava postupak moguć.

5 Libelium Waspmote

Osim *Raspberry Pi* pločica koristili smo iz eksperimentalnih razloga i jednu *Arduino* inačicu pločica, *Libelium Waspmote*, koja se po svemu sudeći pokazala kao prihvatljivije i optimalnije rješenje čvora za mjerenje, ali je cjenovno znatno skuplje iz razloga što je prilagođeno baš u svrhu kreiranja čvorova za mjerenje senzora. *Raspberry Pi* ima znatno širi spektar mogućnosti za uporabu, no zato troši iznimno više energije za napajanje.

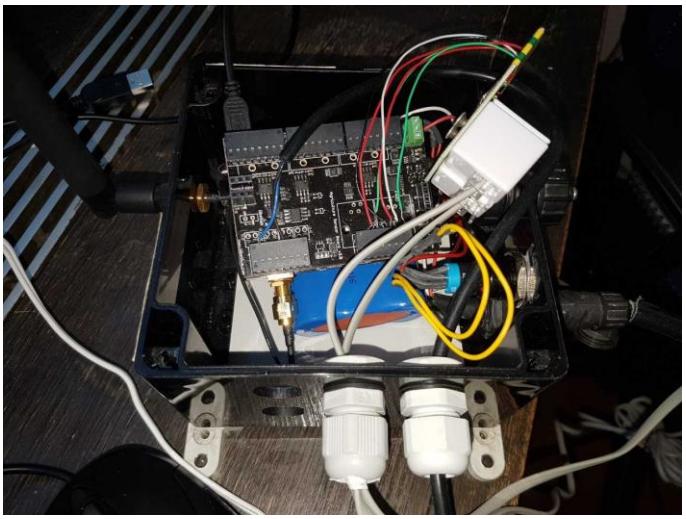
Arhitektura je bazirana na *Atmel ATmega1281* mikrokontroleru koji za razliku od *Raspberry Pi*-a obrađuje isključivo jedan C ili C++ programski kod spremljen u *flash* memoriju veličine 128 kB. *Waspmote Pro IDE* služi kao grafičko sučelje za kompajliranje programskog koda i dostupno je za *Linux, Windows* i *Mac* operacijske sustave. Programi su spremljeni u *sketch*-eve, datoteke tipa .pde namijenjene isključivo IDE-u, dok su vanjske datoteke i biblioteke koje pozivamo u *C* ili *C++* programskom jeziku.



Slika 5.1 Waspmote IDE [19]

5.1 Čvor za mjerenje i njegovi senzori

Postoji čitav niz paketa „*Plug & Sense!*“ koji dolaze sa zaštitnom kutijom i imaju set senzora sa različitom namjenom. Naš je čvor konfiguriran kao meteorološka stanica koja sadrži dodatak *Weather Station WS-3000* i senzor temperature tla. *WS-3000* se sastoji od senzora za mjerenje smjera vjetra, anemometra i pluviometra.



Slika 5.2 "Plug & Sense" konfigurirana Waspmote pločica

Programski kod je poprilično jednostavan a sastoji se od

- Pozivanje vanjskih biblioteka
- Funkcije `void setup()`
- Funkcije `void loop()`

Od vanjskih biblioteka koristimo `Wasp232.h` koja nam služi za korištenje metoda *Xbee* modula i uspostavljanje serijske konekcije, te `WaspSensorAgr_v20.h` koji sadrži sve potrebne metode za pravilan rad agrikulturalnih senzora.

Prije pozivanja glavne funkcije `void setup()` potrebno je definirati varijable u koje ćemo pohranjivati rezultate mjerenja. Unutar funkcije `void setup()` pozivamo metode `USB.ON()`, `W232.ON(SOCKET0)`, `W232.baudRateConfig(9600)`, `W232stopBitConfig(1)` pomoću kojih postavljamo potrebne parametre *Xbee* modula, a zatim pozivamo metode `SensorAgrv20.ON` za aktiviranje senzorskih pinova i `RTC.ON()` za brojač vremena prilikom postavljanja pločice u „*deep sleep mode*“ kako bi uštedjeli na potrošnji energije potrebne za napajanje.

```
1.  void setup()
2.  {
3.    // Turn on the USB and setup Xbee module
4.    USB.ON();
5.    W232.on(SOCKET0);
6.
7.    W232.baudRateConfig(9600);
8.    W232.stopBitConfig(1);
9.
10.   //Turn on the sensor board
11.   SensorAgrv20.ON();
12.
13.   //Turn on the RTC
14.   RTC.ON();
15. }
```

lika 5.3 Postavljanje početnih parametara

U sklopu `void loop()` funkcije pozivamo kod koji se izvršava nakon inicijaliziranja početnih postavki modula i dodataka programa. Pomoću metode `SensorAgrv20.setSensorMode` pozivamo svaki od senzora a metodom `SensorAgrv20.readValue` čitamo vrijednosti mjerenja i pridjeljujemo ih pojedinim varijablama.

```
1.  switch(SensorAgrv20.vaneDirection)
2.    {
3.    case  SENS_AGR_VANE_N   :  snprintf( vane_str, sizeof(vane_str), "N" );
4.                               break;
5.    case  SENS_AGR_VANE_NNE :  snprintf( vane_str, sizeof(vane_str), "NNE" );
6.                               break;
7.    case  SENS_AGR_VANE_NE  :  snprintf( vane_str, sizeof(vane_str), "NE" );
8.                               break;
9.    case  SENS_AGR_VANE_ENE :  snprintf( vane_str, sizeof(vane_str), "ENE" );
10.                              break;
11.   case  SENS_AGR_VANE_E   :  snprintf( vane_str, sizeof(vane_str), "E" );
12.                              break;
13.   case  SENS_AGR_VANE_ESE :  snprintf( vane_str, sizeof(vane_str), "ESE" );
14.                              break;
15.   case  SENS_AGR_VANE_SE  :  snprintf( vane_str, sizeof(vane_str), "SE" );
16.                              break;
17.   case  SENS_AGR_VANE_SSE :  snprintf( vane_str, sizeof(vane_str), "SSE" );
18.                              break;
19.   case  SENS_AGR_VANE_S   :  snprintf( vane_str, sizeof(vane_str), "S" );
20.                              break;
21.   case  SENS_AGR_VANE_SSW :  snprintf( vane_str, sizeof(vane_str), "SSW" );
22.                              break;
23.   case  SENS_AGR_VANE_SW  :  snprintf( vane_str, sizeof(vane_str), "SW" );
24.                              break;
25.   case  SENS_AGR_VANE_WSW :  snprintf( vane_str, sizeof(vane_str), "WSW" );
26.                              break;
27.   case  SENS_AGR_VANE_W   :  snprintf( vane_str, sizeof(vane_str), "W" );
28.                              break;
29.   case  SENS_AGR_VANE_WNW :  snprintf( vane_str, sizeof(vane_str), "WNW" );
30.                              break;
31.   case  SENS_AGR_VANE_NW  :  snprintf( vane_str, sizeof(vane_str), "WN" );
32.                              break;
```

```
33.   case  SENS_AGR_VANE_NNW :  snprintf( vane_str, sizeof(vane_str), "NNW" );
34.                              break;
35.   default                 :  snprintf( vane_str, sizeof(vane_str), "error" );
36.                              break;
37.   }
```

Slika 5.4 Switch-case petlja za definiranje smjera vjetra

Obzirom da podatke dalje prosljeđujemo u *JSON* potrebno je rezultate mjerenja pretvoriti u *string* format podataka metodom `Utils.float2String()`. Prilikom postavljanja smjera vjetra vršimo provjeru u sklopu *switch-case* petlje koja za svaki slučaj smjera vjetra definira *string* koji ga jednoznačno prikazuje.

```
1.  USB.println( vane_str );
2.    W232.send("{\"Tp\":");
3.    W232.send(tempStr);
4.    W232.send(",\"W\":");
5.    W232.send(anmStr);
6.    W232.send(",\"R\":");
7.    W232.send(pluStr);
8.    W232.send(",\"V\":\"");
9.    W232.send(vane_str);
10.   W232.send("\"}\r\n"); //remove } because datetime will be added on central RPi
11.
12.   //PWR.deepSleep("00:00:02:00",RTC_OFFSET,RTC_ALM1_MODE1,ALL_OFF); //Sleep and Wake up
      every 15 minutes.
```

Slika 5.5 Slanje podataka putem Xbee modula i stavljanje uređaja u „sleep mode“

Nakon pridruživanja vrijednosti mjerenja varijablama, formiramo *JSON* tip podatka sa parom ključ-vrijednost, te ih pomoću *Xbee* modula prosljeđujemo *Raspberry Pi*-u za slanje na server. Kako bi minimizirali potrošnju energije prilikom slanja podataka ne šaljemo identifikacijsku varijablu čvora (za razliku od *Raspberry Pi* čvorova gdje je šaljemo zbog više takvih čvorova), niti vrijednost o vremenu izmjerenog podatka nego ga pridružujemo nakon što dođe do čvora za prosljeđivanje.

Pomoću njega možemo grafički preko internet preglednika pristupiti podacima i organizirati konfiguriranje podatka u tablicama. Nedostatak korištenja je sporije dohvaćanje i obrađivanje podataka na klijentskoj strani servisa, u našem slučaju internet sučelja, od strane *JavaScript* programskog jezika kojeg smo koristilli zbog modernijeg pristupa prikaza.

Na serverskoj strani smo postavili *python* skriptu koja osluškuje zahtijeve centralnog čvora i prosljeđuje te podatke u bazu na način da ih razvrstava ovisno o preddefiniranom parametru *RasID*. Služi kao filter za spremanje i dohvaćanje podataka iz baze pri prikazivanju na internet stranici.

Imenovanja varijabli parametara mjerenja se definiraju u *JSON* notaciji prilikom izvršavanja kod čvorova, stoga smo zadržali za oba tipa baza podataka ista imenovanja i za spremanje tih podataka u bazu. Kod definiranja parametara koji ulaze u tablicu kod *MySQL* baze podataka moramo preddefinirati imenovanja parametara kako bi iste spremili, dok kod *MongoDB* baze podataka to nije potrebno.

Slika 5.6 Postavljeni Libelium Waspmote čvor u vinogradu

6 Konfiguracija servera

Za održavanje servera na kojem smo spremali podatke i prikazivali rezultate na internetu koristili smo usluge kompanije „*Hetzner online*“ pritom koristeći *Debian* inačicu operacijskog sustava *Linux* koja je jako slična onoj prisutnoj na *Raspberry Pi* uređajima. Operacijski sustav je imao predinstalirane pakete *Apache* i *PhpMyAdmin* za serversku namjenu prikaza internetskog sadržaja stoga je bilo potrebno kreirati skriptu koja će primati podatke i prosljeđivati ih u bazu podataka.

Skripta je konfigurirana kao pozadinski proces pomoću naredbe `nohup python /home/pi/Desktop/skripta_na_serveru.py &` koja osluškuje zahtjeve za konekcijom od strane čvora za prosljeđivanje na preddefiniranom *socketu* i pritom vrši ubacivanje tih podataka u bazu, inicijalna *MongoDB* verzija i *MySQL* verzija. U ranijim fazama izrade projekta koristili smo *MongoDB 3.1*, *NoSQL* bazu podataka, a zatim smo koristili *MySQL* bazu podataka koja dolazi kao predinstalirani paket u sklopu preuzetog operacijskog sustava.

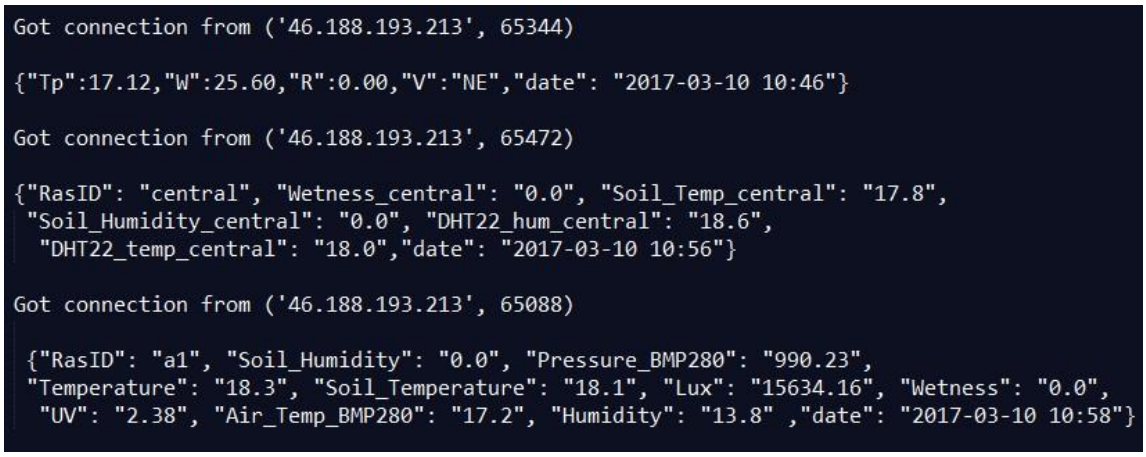
6.1 MongoDB

Obzirom na promjene prilikom korištenja baze podataka postoje dva korištena načina, za svaku od tipa baze podataka, kako ubacivati te podatke u bazu. *MongoDB* je „*open-source*“ (hrv. besplatan softver) platforma koja se definira kao *NoSQL* baza podataka dokumentiranog tipa, što znači da su podaci spremljeni u *JSON* notaciji, točnije *BSON* (eng. *binary-encoded format)* prilagođenoj baš za *MongoDB* koja je proširena implementacija *JSON*-a jer sadrži više podatkovnih tipova, sortiranja te je efikasnije za šifriranje i obrađivanje od klasične *JSON* notacije.

Iz tog razloga smo se i odlučili na početku koristiti dokumenitrani tip baze jer je brzina prilikom dohvaćanja podataka na bazu, obrade podataka pomoću programskih jezika za internet, kao npr. *JavaScript*, znatno brža što se na ovako veliki broj generiranih podataka primijeti ljudskim okom kada se radi podatkovno zahtjevniji upit na bazu. Prilikom dohvaćanja manje količine podataka razliku nije lako primijetiti, ali je osjetna razlika fizički vidljiva kod upita na bazu koji obuhvaćaju otprilike 10 tisuća podataka tipa *JSON*, a brzina se mjeri u par stotina milisekunda kašnjenja u prikazu na internetu.

Instalacija *MongoDB* servisa je besplatna i može se preuzeti sa službene stranice organizacije. U našem slučaju preuzeli smo inačicu *MongoDB Community Edition on Ubuntu* pomoću

predinstaliranog upravljačkog alata *apt* unošenjem terminalske komande `sudo apt-get install -y mongodb-org`. Servis se pokreće pomoću komande `sudo service mongod start` uz povratnu informaciju da proces očekuje podatke na tada već automatski postavljenom *portu* 27017, što se može modificirati u datoteci `/etc/mongod.conf`. Kako bi pristupili podacima i omogućili modificiranje i prikaz promjena u bazi moramo pokrenuti mongo *shell*, a to radimo pomoću komande `mongo`.



Slika 6.1 Primjeri JSON podataka koje spremamo u bazu (output log informacije)

6.1.1 Spremanje podataka u *MongoDB* bazu podataka

Skripta za spremanje podataka se sastoji od tri dijela:

- Pozivanje vanjskih biblioteka
- Inicijalizacija baze podataka i definiranje *socket*-a za primanje podataka
- Petlja za obrađivanje primljenih zahtjeva i prosljeđivanje podataka u bazu

Biblioteke stavljamo na početku skripte kako bi se mogle učitati prije izvođenja programa a to su:

- *socket* – služi za kreranje *socketa* za prihvaćanje zahtjeva
- *pymongo* – pomoću nje pozivamo metode potrebne za pristup i inicijalizaciju *MongoDB* bazi podataka
- *json* – služi za obradu *JSON* tipova podataka
- *datetime* – korištenje metoda za informiranje o vremenu izvođenja programa
- *pprint* – skraćeno od *pretty print* a služi za ljepši ispis logova i *JSON* objekata

- *traceback* – služi za razumijevanje logova i jasnije rješavanje problema prilikom izvođenja programa

Nakon uključivanja potrebnih biblioteka potrebno je inicijalizirati sučelje za pristup bazi podataka i kreirati preddefinirano sučelje za *socket* koji prima zahtjeve od strane čvora za prosljeđivanje. Kreiramo objekt koji poprima povratnu vrijednost pri pozivanju metode `pymongo.MongoClient().kolekcija` koja se spaja na kolekciju unutar baze podataka.

Zatim definiramo objekt koji poprima vrijednost pri pozivanju metode `socket.socket()` i pomoću nje vežemo par *host* i *port* vrijednosti.

```
1. db = pymongo.MongoClient().kolekcija
2. s = socket.socket()
3. host = socket.gethostname()
4. port = 12345
5. s.bind((host, port))
6. s.listen(1)
```

Slika 6.2 Kreiranje konekcija na MongoDB bazu i inicijalizacija socket konekcije za primanje podataka

Kao što je prikazano na slici vidimo da je objekt *socket*-a postavljen u *listen* način rada pritom osluškujući zahtjeve u periodima od jedne minute.

Nakon početnih inicijalizacija definiramo `while True:` petlju koja provjerava njihovu valjanost i ukoliko je sve u redu nastavlja sa primanjem zahtjeva. Kod primljene konekcije kreiramo novi objekt poput onoga za *socket*, koji poprima primljene vrijednosti.

Zatim dekodiramo primljeni *JSON* zapis pomoću naredbe `json.loads`, kreiramo novi objekt, te pomoću komandi `bulk.initialize_unordered_bulk_op()` i `bulk.insert()` postavimo podatke u kolekciju i zatvorimo konekciju prema čvoru za prosljeđivanje.

```
1.  while True:
2.      c, addr = s.accept()
3.      print 'Got connection from', addr
4.      c.send('Thank you for your connecting')
5.      ocitanje = c.recv(1024)
6.      print ocitanje
7.      try:
8.          ocitanje = json.loads(ocitanje)
9.      except ValueError, e:
10.         traceback.print_exc()
11.         continue
12.     print ocitanje
13.     bulk = db.testt.initialize_unordered_bulk_op()
```

```
14.     print bulk
15.     try:
16.         bulk.insert(ocitanje)
17.         result = bulk.execute()
18.         pprint (result)
19.     except Exception, e:
20.         traceback.print_exc()
21.     c.close()
```

Slika 6.3 Primanje konekcije i spremanje podataka u MongoDB bazu

6.1.2 Spremanje podataka u *MySQL* bazu podataka

Prilikom razvojnog testiranja u ranim fazama korištenja *MongoDB* servisa imali smo problema sa koruptiranjem podataka. To je ujedno i jedan od općenitih problema MongoDB servisa gdje je jedan *port* na serveru u *listen mode*-u, stanju konstantnog osluškivanja nadolazećih podataka na tom *portu* pa je ranjiv na utjecaj zlonamjernih vanjskih softvera.

U ranoj fazi se dogodio incident gdje smo primijetili da ne možemo uspostaviti *SSH* konekciju na server zbog prekomjernih pokušaja spajanja na server od strane vanjskog softvera. To je klasični primjer *DoS* (eng. „*Denial of Service*“) napada koji najčešće napada tako otvorene *portove* servera, najčešće velikih kompanija, s ciljem kreiranja veće količine prometa na mreži tako da softver nije u mogućnosti obraditi sve zahtjeve u kratkom vremenu. Tada dolazi do povećanog kašnjenja na usluzi stoga „normalni“ korisnici ne mogu pravilno i u razumnom vremenskom roku pristupiti servisu.

Problem smo primijetili tako što se nismo mogli spojiti na server zbog povećanog prometa, a i onda kada bi se spojili smo primijetili sporije izvršavanje komandi na serveru. Pokušaji su se mogli jasno vidjeti u ispisu komande `iptables` gdje možemo vidjeti sve oblike spajanja na otvorene *portove* u sustav. Napadi su vjerovatno izvršavani pomoću automatizirane skripte u omjeru tri napada po sekundi.

Obzirom da *MongoDB* sprema podatke u kolekcije koji imaju limitirani kapacitet (do 1 Gb podataka) i s vremenom bi se njihov kapacitet napunio te bi trebalo inicijalizirati nove kolekcije i definirati nova imenovanja u servisu pa tako i na internetskoj strani prikaza podataka, automatizacija dohvaćanja i spremanja podataka bi se znatno zakomplicirala. Uz to, *MongoDB* nije toliko pregledan a i zbog daljnjeg širenja sustava i promjene konfiguracije prebacili smo pohranu podataka na *MySQL* relacijsku bazu podataka koja je po svemu sudeći i sigurnija po pitanju pristupa

podacima, bolje je dokumentirana a i omogućeno je korištenje *phpmyadmin* alata koji dolazi predinstaliran u sklopu operacijskog sustava.

6.2 MySQL

MySQL je besplatan „*open source*“ sustav za upravljanje bazom podataka. Čest je izbor za različite projekte koji podrazumijevaju korištenje bazi podataka te se distribuira kao sastavni dio serverskih *Linux* distribucija. Vrlo je stabilna i ima dobro dokumentirane module, a uz to je podržana od strane raznim programskih jezika, uključujući *PHP* programski jezik korišten u našem projektu. Podaci se pohranjuju u tablice koje se sastoje od redova i stupaca koje još nazivamo poljima i atributima.

Postoji niz inačica različitih *MySQL* relacijskih baza podataka a podaci se u bazi definiraju kao entiteti između relacija koje mogu biti:

- jedan prema jedan
- jedan prema više
- više prema jedan
- više prema više

Obzirom da je sustav skalabilan, za razliku od *MongoDB* baze podataka koja ne mora imati predefinirane parametre za spremanje jer su svi podaci tipa *string*, našu smo bazu podataka *MySQL* definirali na način da se spremaju u jednu tablicu koja je imenovana po lokaciji na kojoj je sustav postavljen sa predefiniranim tipovima varijabli. Stupci predstavljaju senzore a podatak se sprema u jednom retku pritom popunjavajući polja prisutnih vrijednosti pojedinih senzora.

Skripta kojom stavaljamo podatke u *MySQL* bazu podataka se razlikuje u sintaksi i načinu inicijalizacije konekcije prema bazi, no ideja i princip su jako slični. Biblitoteku *pymongo* zamjenjujemo sa *MySQLdb* a konekciju prema bazi izvršavamo pridljeljujući nekoj varijabli, primjerice *db*, svojstva metode `MySQLdb.connect()` pritom prosljeđujeći informaciju o korisniku, lozinki, imenu baze i IP adresi (u našem slučaju je to *localhost*). Zatim inicijaliziramo novi objekt koji poprima pokazivačka svojstva konekcije pomoću naredbe `db.cursor()`. Prilikom povezivanja na bazu *MySQL* potrebno je pridijeliti parametre kao što su lozinka i korisničko ime, a kod *MongoDB* baze je dovoljno unijeti IP adresu i *port*, što pokazuje na višu razinu sigurnosti *MySQL* baza podataka.

```
db = MySQLdb.connect(host="localhost", user="user", passwd="password", db="mysql")
baza = db.cursor()
```

Slika 6.4 Spajanje na MySQL bazu podataka

Inicijalizacija *socketa* je ista kao i kod MongoDB baze podataka za primanje konekcija, kao i način dekodiranja *JSON*-a. Prilikom pohrane podataka u bazu postavili smo *SQL* naredbu na način da se svi postojeći parametri u zahtjevu za prijenos stavljaju u bazu. Naravno, postupak nije idealan i moguće je uvesti detaljnije provjere i analize ID-a čvora i ostataka parametara u svrhu filtriranja podataka i njihova stavljanja u bazu, no obzirom na uvjet da se podaci spremaju u jednu tablicu, uvjet je dovoljan.

```
baza.execute("""INSERT INTO vineyard (RasID, Soil_Humidity,
        Temperature, Lux, Wetness, UV, Humidity, Soil_Temperature,
        Pressure_BMP280, Air_Temp_BMP280, Tp, R, W, V,
        DHT22_temp_central, DHT22_hum_central, Wetness_central,
        Soil_Humidity_central, Soil_Temp_central, date) VALUES (%s, %s, %s, %s, %s, %s,
 %s, %s,
        %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)""",((ocitanje.get('RasID'),
         ocitanje.get('Soil_Humidity'), ocitanje.get('Temperature'), ocitanje.get('Lux'
),
          ocitanje.get('Wetness'), ocitanje.get('UV'), ocitanje.get('Humidity'),
           ocitanje.get('Soil_Temperature'), ocitanje.get('Pressure_BMP280'),
            ocitanje.get('Air_Temp_BMP280'), ocitanje.get('Tp'), ocitanje.get('R'),
             ocitanje.get('W'), ocitanje.get('V'), ocitanje.get('DHT22_temp_central'),

              ocitanje.get('DHT22_hum_central'), ocitanje.get('Wetness_central'),
               ocitanje.get('Soil_Humidity_central'), ocitanje.get('Soil_Temp_central')
, ocitanje.get('date'))))
        db.commit()
        c.close()
```

Slika 6.5 Spremanje izmjerenih podataka u prethodno definirane stupce u MySQL bazi

U daljnjem širenju sustava čvorovi se kod dodavanja osim u senzorima razlikuju i u parametru „*RasID*“, tako primjerice prilikom prikazivanja podataka na internet stranici možemo filtrirati čvorove neovisno o senzorima koje sadrže. Također osim indeksiranja po čvoru prilikom inicijalizacije *MySQL* baze podataka postavili smo da se parametar „*indeks*“ automatski inkrementira tako da se podaci indeksiraju redom kako su pristigli u bazu.

6.3 Prikaz podataka na internet stranici

Grafičko sučelje internet stranice za prikaz podataka kreirano je iz besplatnog predloška „*Gentellela Bootstrap 3*“ kojeg smo preuzeli i modificirali po našoj potrebi. Predložak je modularan i po prikazu je prilagođen za sve uređaje neovisno o veličini njihova ekrana.

Komunikaciju sa bazom podataka ostvarujemo pomoću *PHP* programskog jezika, a za moderniji prikaz koristili smo tehnologiju *jQuery* uz besplatne alate koji dolaze s njom. Koristimo isključivo *JavaScript* programski jezik za obradu dohvaćenih podataka i upravljanje sa strukturom *HTML* i *CSS* elemenata internet stranice.

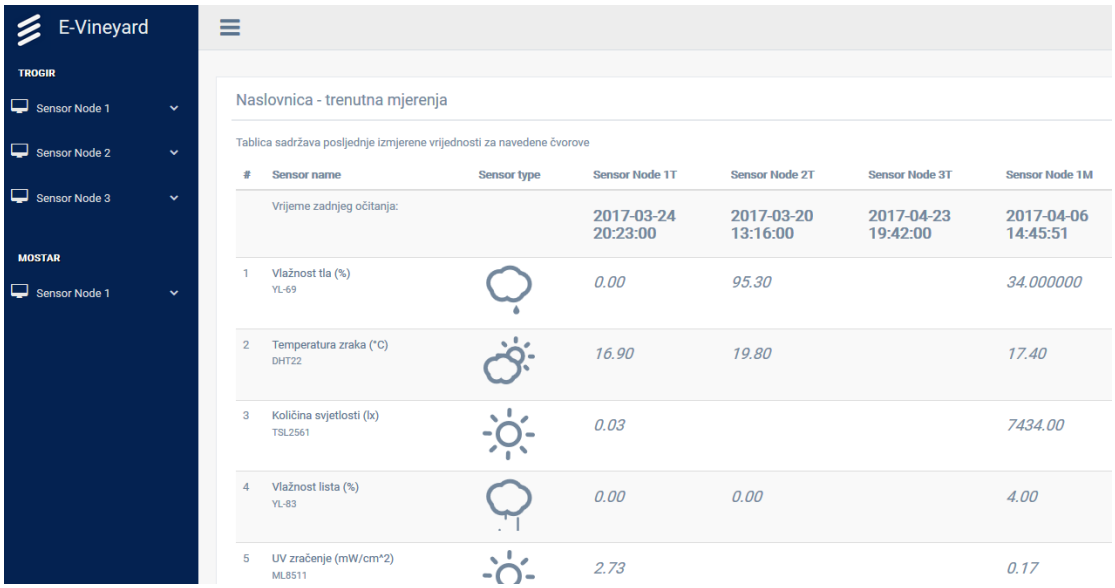
Ideja internet stranice je omogućiti intuitivno i jednostavno korisničko iskustvo prilikom analize podataka. Sadržaj stranice se dijeli u tri dijela ovisno o informacijama koje želimo analizirati:

- Naslovna stranica – prikazuje posljednja mjerenja na svim postavljenim čvorovima poljoprivredne kulture kako bi dobili „*real-time*“, trenutnu informaciju o mjerenjima i stanju u polju
- Stranica čvorova – prikazuje mjerenja svih senzora pojedinog čvora i prikazuje međusobne ovisnosti vrijednosti parametara, moguće je pregledavati mjerenja po određenim vremenskim intervalima
- Stranica senzora – prikazuje mjerenja svakog od senzora na čvoru, također je moguće pregledavati mjerenja po vremenskim intervalima

6.3.1 Naslovna stranica

Prilikom pristupa internet stranici korisnik prvo pristupa naslovnoj stranici. Naslovna stranica se sastoji od glavnog izbornika za navigaciju sa lijeve strane sučelja, koji je prisutan na svim stranicama upravo zbog lakšeg koordiniranja za korisnike, a u sredini se nalazi glavni sadržaj stranice koji prikazuje posljednje izmjerene parametre sa svih čvorova za svaki od senzora.

Pritiskom na gumb ikone stranice u gornjem lijevom kutu pozivamo osvježavanje naslovne stranice. Stranica ne sadrži dinamičke alate za samostalno ažuriranje podataka nego je potrebno poslati ponovni upit na bazu kako bi se pojavili novi izmjereni podaci.



Slika 6.6 Prikaz naslovne stranice

Kako bi prikaz naslovne stranice modernizirali i omogućili bolje razumijevanje senzora koristimo *jQuery* biblioteku *Skycons* koja nam služi za dinamički prikaz mehanizma ikona vremenskih uvjeta. Pomoću izbornika sa lijeve strane pristupamo pojedinim čvorovima i njihovim senzorima za detaljniji pregled mjerenja. Pristup bazi ostavarujemo putem *PHP* programskog jezika, a za *MySQL* to znači prvo definiranje potrebnih varijabli: *host*, ime baze, korisničko ime, lozinka i ime tablice kojoj pristupamo. Zatim varijabli pridružujemo vrijednost komande `$connection = new mysqli($host, $user, $password, $dbname)` kreiramo konekciju prema bazi. Ako je konekcija uspostavljena moguće je kreirati SQL upit na bazu za dohvaćanje redova iz *MySQL* baze i njihovo spremanje u varijablu tipa redak.

```
1.  <?php
2.
3.  $dbhost = 'localhost';
4.  $dbname = 'mysql';
5.  $user = 'root';
6.  $pass = 'Loza1911';
7.  $bas = 'vineyard';
8.
9.  $connection = new mysqli($dbhost, $user, $pass, $dbname);
10. $sql = $connection->query("SELECT * FROM vineyard ORDER BY indeks");
11. $series = array();
12. while($upit = $sql->fetch_assoc()){
13.         $series[]=$upit;
14.     }
15. ?>
```

Slika 6.7 Dohvaćanje svih izmjerenih podataka iz baze

Dohvaćeni redovi podataka se zatim pridjeljuju varijabli tipa redak u *JavaScript* kodu kao reprezentativni *JSON* tip podatka pomoću komande `json_encode()` koja *parsira* (eng. *parse* - raščlanjuje) podatke. Dohvaćene podatke razlikujemo po tipu pomoću parametra *RasID*, a poredak se definira pomoću parametra *date*.

```
1.  var varijabla = <?php echo json_encode($series); ?>
```

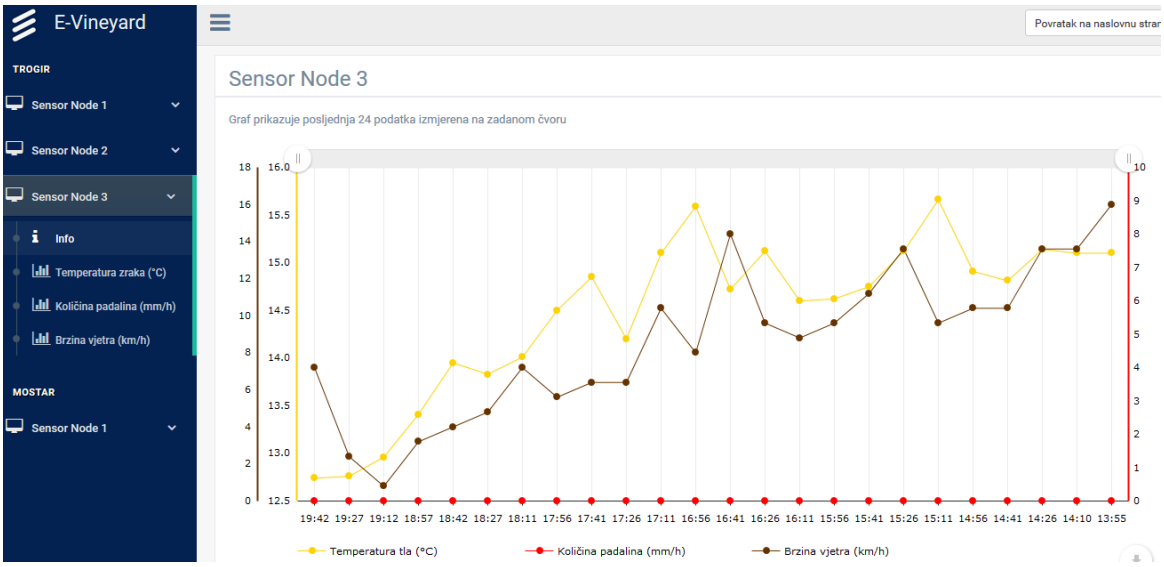
Slika 6.8 Parsiranje podataka iz PHP-a za korištenje u JavaScript-u

6.3.2 Stranica čvorova

Stranica čvorova služi za prikazivanje grafova koji sadrže usporedno sve senzore tog čvora kako bi korisnik mogao vidjeti međusobnu zavisnost mjerenja senzora. Grafovi pokazuju podatke izmjerene u posljenjih 24 sata i nije omogućeno prikazivanje željenog vremenskog intervala osim tog. Sa lijeve strane stranice se također nalazi izbornik za navigaciju.

Za grafički prikaz podataka na grafovima koristili smo *jQuery* alat „*amCharts"*. Alat sadrži biblioteke pomoću kojih možemo pozvati *Javascript* datoteke za mehanizam dinamičkog prikaza sadržaja. U našem slučaju koristili smo grafove tipa *serial* za stranicu čvorova i stranicu senzora. Prilikom analize dohvaćenih podataka moguće je po želji uključivati i isključivati prikaz željenog senzora pritiskom na definirana imenovanja ispod grafa. Alat također sadrži dodatak za preuzimanje slike grafa nakon upita na bazu podataka.

Ispod grafa se nalaze osnovne informacije o odabranom čvoru, vrijednost posljednjeg mjerenja na tom čvoru, te mapa koja prikazuje koordinate na kojem se taj čvor nalazi. Za prikaz mape koristili smo „*Google Maps*" API koji sadržava niz mogućnosti.



Slika 6.9 Prikaz stranice čvora

Podaci se u *JavaScript*-u raščlanjuju na isti način no jedina je razlika u definiranom upitu na bazu. U ovom slučaju dohvaćamo podatke ovisno o *RasID* parametru čvora kojeg prikazujemo.

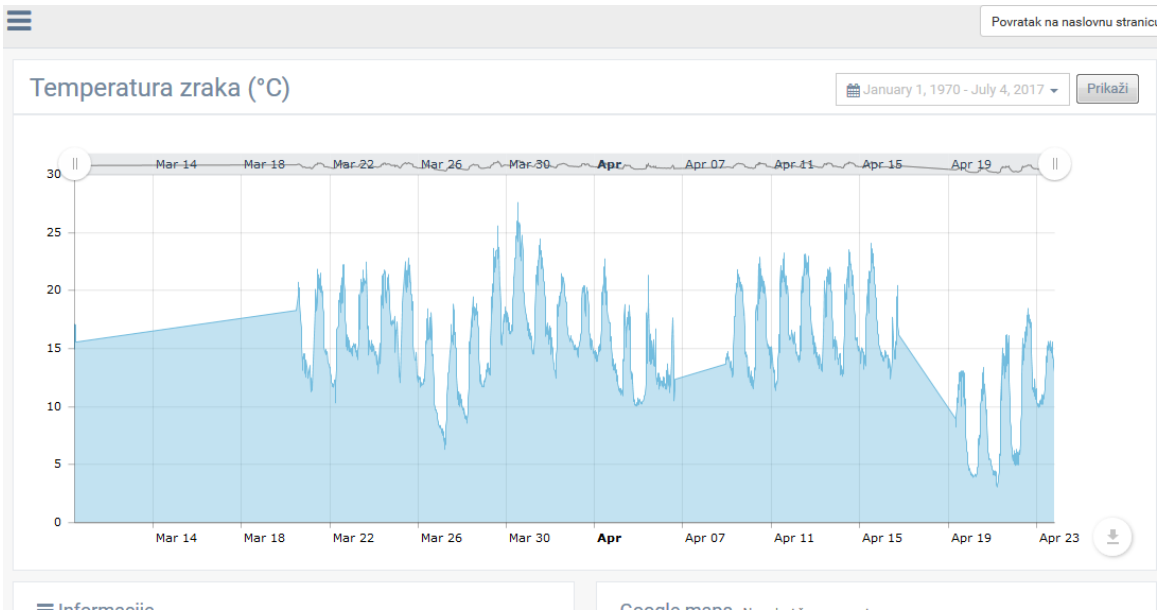
```
1.  $sql = $connection-
    >query("SELECT * FROM vineyard WHERE RasID = 'a1' ORDER BY date DESC LIMIT 96");
```

Slika 6.10 Dohvaćanje podataka u ovisnosti o željenom čvoru

6.3.3 Stranica senzora

Za svaki senzor koji vrši mjerenja moguće je popratiti njegove vrijednosti na individualnim stranicama senzora. Ovdje također koristimo *amCharts* biblioteku za prikaz vrijednosti i *daterangepicker* za odabir prikaza u određenom vremenskom intervalu. Korisnik odabirom željenog vremenskog intervala potvrđuje svoj unos i kao rezultat dobiva prikaz grafa izmjerenih podataka.

Odabir prikaza mjerenja nekog drugog senzora se izvršava isključivo preko glavnog izbornika sa lijeve strane. Za svaki od senzora je napisana zasebna skripta za dohvaćanje podataka.



Slika 6.11 Prikaz mjrenja pojedinog senzora

Na dnu stranice se također nalazi polje s informacijama o vrsti senzora i lokaciji čvora na kojem se senzor nalazi prikazana pomoću „*Google Maps*“ API-a. Iz prethodno opisanih stranica vidljivo je da su one poredane hijerarhijski počevši od naslovne stranice preko stranice čvorova do stranica pojedinih senzora. Osim naslovne stranice, sve ostale stranice sadrže u desnom gornjem kutu gumb za povratak na naslovnu stranicu koji smo kreirali kako bi pružili što bolje korisničko iskustvo.

7 Zaključak

Sustav mora biti energetski učinkovit i pritom mora minimizirati korištenje resursa. Prilikom izrade projekta smo zaključili da *Raspberry Pi* nije povoljan za izradu čvorova za mjerenje zbog velike potrošnje energije za napajanje. Za rad koristi operacijski sustav *Linux* koji u pozadini pokreće niz procesa koji su mu potrebni za funkcionalan rad, dok je *Libelium Waspmote* mikrokontroler koji pokreće samo jedan programski kod i zbog tog mu je potrošnja energije za napajanje znatno manja.

Usporedbe radi, *Raspberry Pi* uređaj je prilikom korištenja imao maksimalan vijek trajanja od 48 sati, unatoč povoljnim vremenskim uvjetima za punjenje solarne ploče napajanje sustava, za razliku od *Waspmote Libelium* uređaja koji je uz solarnu ploču i bateriju bio u aktivnom stanju i preko mjesec dana. Uz neprestano napajanje uređaja *Raspberry Pi* se pokazao kao dobro rješenje za čvor koji prosljeđuje podatke na server.

U nastavku razvoja projekta fokus razvoja je bio isključivo na *Arduino* inačicama platformi za razvoj sustava upravo zbog problema napajanja *Raspberry Pi* uređaja. Moguće rješenje bi bilo postaviti kapacitetno veće solarne ploče i veće litijske baterije za punjenje, no potreban je duži vremenski period testiranja kako bi bili sigurni da će sustav biti aktivan u svim vremenskim uvjetima.

Cilj ovog rada je bio kreirati energetski učinkovit i autonoman sustav koji je skalabilan i moguće ga je implementirati kod različitih poljoprivrednih kultura. Usporedili smo performanse *Raspberry Pi* i *Libelium Waspmote* platformi i proučili razlike u konfiguriranju čvorova. Proučili smo djelovanje vanjskog utjecaja na performanse i rad sustava. Obzirom na specifičnost okruženja u kojem će se ovakvi sustavi implementirati potrebno se posebno koncentrirati na rješenja autonomnog napajanja za daljnji razvoj sustava.

Obe platforme su pronašle primjenu u kućnoj uporabi i njihova popularnost s vremenom raste. Zbog niske cijene i jednostavnosti izrade postale su dostupne široj publici ljudi i kao takve možemo u budućnosti očekivati puno veći spektar primjene.

8 Literatura

[1] Wikipedia (2017), *Internet of things*, s Interneta, https://en.wikipedia.org/wiki/Internet_of_things

[2] Raspberry Pi Org. (2017), *FAQS*, s Interneta, https://www.raspberrypi.org/help/faqs/#introWhatIs

[3] Raspberry Pi Org. (2017), *GPIO: RASPBERRY PI MODELS A AND B*, s Interneta, https://www.raspberrypi.org/documentation/usage/gpio/

[4] Cooking Hacks (2015), *Bluetooth Low Energy Module 5dBi for Waspmote [Xbee socket]*, s Interneta, https://www.cooking-hacks.com/bluetooth-low-energy-module-5dbi

[5] Cooking Hacks (2015), *Xbee for Arduino and Raspberry Pi*, s Interneta, https://www.cooking-hacks.com/documentation/tutorials/xbee-arduino-raspberry-pi-tutorial/

[6] Display Plus (2013), *IBM and Libelium launch 6LoWPAN / IPV& Internet of Things platform*, s Interneta, http://www.displayplus.net/news/articleView.html?idxno=50034

[7] Libelium Comunicaciones Distribuidas S.L. (2017), *Waspmote Technical Guide v7.2*

[8] Libelium Comunicaciones Distribuidas S.L. (2017), *RS – 232 Module Communication Guide v7.1*.

[9] Wikipedija (2017), *MySQL*, s Interneta, https://hr.wikipedia.org/wiki/MySQL

[10] Bažant, Alen (2008), *Osnove prijenosa podataka*, Zavod za telekomunikacije, FER Zagreb

[11] Seenov Electronics for DYI, Educators and Everyone (2017), *Raspberry Pi 3 Model B Board*, s Interneta, https://www.seenov.com/product/raspberry-pi-3-model-b-board/

[12] Raspberry Pi Org. (2017), *Raspbian*, s Interneta, https://www.raspberrypi.org/downloads/raspbian/

[13] ModMyPi LTD (2015), *Raspberry Pi Plant Pot Moisture Sensor with Email Notification Tutorial*, s Interneta, https://www.modmypi.com/blog/raspberry-pi-plant-pot-moisture-sensor-with-email-notification-tutorial

[14] Adafruit (2017), *DHT22 temperature-humidity sensor + extras*, s Interneta, https://www.adafruit.com/product/385

[15] The e-Store @ MG Super LABS (2017), *TSL2561 digital luminosity/lux/light sensor*, s Interneta, https://www.mgsuperlabs.co.in/estore/TSL2561-digital-luminosity-lux-light-sensor

[16] Thomsen, Adilson (2014), *SENSOR DE CHUVA YL-83*, s Interneta, http://blog.filipeflop.com/sensores/sensor-de-chuva-yl-83.html

[17] dealextreme (2017), *GY-ML8511 Ultraviolet Light Sensor Breakout Test Module Detector, s Interneta*, http://www.dx.com/p/gy-ml8511-ultraviolet-light-uv-sensor-breakout-test-module-detector-blue-394373#.WV1i6oSGOCp

[18] Instructables (2016), *How to Use the Adafruit BMP280 Sensor – Arduin Tutorial*, s Interneta, http://www.instructables.com/id/How-to-Use-the-Adafruit-BMP280-Sensor-Arduino-Tuto/

[19] SmartSantander (2012), *Waspmote*, s Interneta, http://www.smartsantander.eu/wiki/index.php/Main/Waspmote

9 Sažetak/abstract i ključne riječi/keywords

Sažetak

U ovom dokumentu je prezentiran način rada sustava u realnom vremenu. Opisali smo pojam „Internet of things“ i što on označava. Definirali smo model arhitekture kreiranog sustava i njegove funkcionalne zahtjeve. Objasnili smo rješenja napajanja sustava i djelovanje vanjskog utjecaja. Opisali smo Xbee modul koji služi za ostvarivanje komunikacije između čvorova sustava i metode konfiguracije fizikalnih parametara signala pomoću alata XCTU. Prilikom opisivanja rada korištenih platformi Raspberry Pi i Libelium Waspmote definirali smo senzore u sustavu i metode njihova spajanja. Ponudili smo softversko rješenje za ostvarivanje mjerenja i prosljeđivanje tih podataka na server. Usporedno smo opisali razlike između MySQL i MongoDB baza podataka uz predočenje internet stranice za grafički prikaz ostvarenih mjerenja.

Ključne riječi

Internet of Things; Čvorovi sustava; Raspberry Pi; Libelium Waspmote; Xbee; Baza podataka; Prikaz na grafu;

Abstract

In this document we have presented the workflow of real-time system. The subject „Internet of Things“ is described and it is explained what it represents. We have defined an arhitecture model of created system and his requirements. Solution for system nodes feed is also described ahead of environmental influence. The communication between the nodes is accomplished with Xbee module and we described the configuration methods for physical parameters of signals by XCTU program. While describing used platforms Raspberry Pi and Libelium Waspmote we have defined the sensors we used and methods of connecting them on system nodes. Software solution for measuring the data and sending them to the server is also offered. Comparison between MongoDB and MySQL databases is also explained and their usage is shown on internet site, through serial graphs.

Keywords

Internet of Things; System nodes; Raspberry Pi; Libelium Waspmote; Xbee; Database; Serial graph;

10 Popis oznaka i kratica

IOT	Internet of Things
USB	Universal Serial Bus
HTTP	Hyper Text Transfer Protocol
JSON	JavaScript Object Notation
ID	Identification
SSH	Secure Shell
PHP	Hypertext Predprocessor
IP	Internet Protocol
API	Application Programming Interface