Mrežno upravljanje pokretnim laboratorijskim objektima

Breški, Mario

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

MARIO BREŠKI

Zagreb, 2018.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentor: Student:

Prof. dr. sc. Mario Essert, dipl. ing. Mario Breški

Zagreb, 2018.

Izjavljujem da sam ovaj rad izradio samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem se prof. dr. sc. Mariu Essertu na prihvaćanju mentorstva, pruženoj podršci i korisnim savjetima. Također se zahvaljujem asistentu Juraju Beniću na pruženoj podršci i korisnim savjetima. Zahvaljujem se i svim prijateljima te roditeljima na pruženoj podršci tijekom školovanja.

Mario Breški

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za završne ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa:	
Ur.broj:	

ZAVRŠNI ZADATAK

Student: **Mario Breški** Mat. br.:0035196323

Naslov rada na hrvatskom jeziku: **Mrežno upravljanje pokretnim laboratorijskim objektima**

Naslov rada na engleskom jeziku: **Network control of mobile laboratory objects**

Opis zadatka:

Pojavom jeftinih mikro-kompjutorskih (μC) komponenti (*NodeCMU, WEMOS D1-mini* i sl.) s ugrađenom WI-FI komunikacijom, omogućena je bežična veza pokretnih objekata i njihovo upravljanje na daljinu. S druge strane, jeftini mikro-kompjutorski sustavi (*Raspberry pi 3, Arduino* i sl.) također se mogu spajati u bežičnu mrežu i na taj način upravljati μC komponentama ugrađenima u pokretne objekte. S obzirom da se radi o jeftinim, ali moćnim računalnim sustavima, moguće je na središnje mjesto upravljanja instalirati mrežni server (tipa WEB2PY) kako bi se komunikacija s komponentama mogla provoditi i putem *http* protokola.

U radu je potrebno načiniti:

1. Objasniti *WEMOS D1-mini* μC komponentu i njene funkcije za bežično spajanje i ulazno-izlazne operacije (pogon istosmjernog motora, signalne lampice i sl.).
2. Objasniti *Raspberry pi 3* mikro-kompjutorski sustav i njegove mogućnosti s obzirom na WEB2PY tehnologiju.
3. Projektirati i izvesti bežičnu komunikaciju *Raspberry pi 3* sustava s jednim pokretnim objektom koji je upravljan *WEMOS D1-mini* μC komponentom.
4. Projektirati i izvesti *Raspberry pi 3* sustava mrežni server temeljen na WEB2PY tehnologiji
5. Osmisliti mrežu pokretnih objekata i budući razvoj ovakvih laboratorijskih modela.

Zadatak zadan:
30. studenog 2017.

Rok predaje rada:
1. rok: 23. veljače 2018.
2. rok (izvanredni): 28. lipnja 2018.
3. rok: 21. rujna 2018.

Predviđeni datumi obrane:
1. rok: 26.2. - 2.3. 2018.
2. rok (izvanredni): 2.7. 2018.
3. rok: 24.9. - 28.9. 2018.

Zadatak zadao:

M. Essert

Prof.dr.sc. Mario Essert

Predsjednik Povjerenstva:

Izv. prof. dr. sc. Branko Bauer

SADRŽAJ

POPIS SLIKA

POPIS KRATICA

Oznaka	Jedinica	Opis
µC		Mikro-kontroler
GHz	GHz	Jedinica mjere za frekvenciju u Međunarodnom sustavu
GB	GB	Jedinica mjere količine podataka u računalstvu
TCP		Transmission Control Protocol
IPv4		Internet Protocol version 4

SAŽETAK

Kroz ovaj rad objašnjen je princip rada mikro-računala. Objašnjene su funkcije za bežično spajanje i ulazno izlazne operacije mikro-računalne komponente s ugrađenom WI-FI komunikacijom te računalni sustav Raspberry Pi. Objašnjena je mogućnosti implementacije web2py tehnologije u Raspberry Pi računalo. Također je objašnjena primjena navedenih komponenti i navedene tehnologije u svrhu upravljanja pokretnim laboratorijskim objektima (pogon istosmjernog motora, signalne lampice, i sl.). Projektirana je i izvedena bežična komunikacija Raspberry pi uz mrežni server temeljen na web2py tehnologiji u svrhu sustava koji će upravljati mikro-računalnom komponentom WEMOS D1-mini µC za pokretanja jednog laboratorijskog objekta preko interaktivnih elemenata aplikacije.

Ključne riječi: WI-FI komunikacija, mikro-računalne, Raspberry pi, web2py, WEMOS D1-mini µC, pokretni objekt, mrežni server

SUMMARY

In this thesis it is explained how micro-computer with WI-FI module works together with a Raspberry Pi computer. Also the possibility of a Web2py implementation into a Raspberry Pi computer with a purpose of controlling the lab devices (DC motor, signaling LED lights and more) is explained. Web based network for device controlling is designed using web2py technology. Micro-computer Demos D1-mini is used to control the devices depending on the data which is sent by the network caused by the user using interactive elements of the application.

Key words: WI-FI communication, Raspberry pi , web2py, micro-computer, WEMOS D1-mini μC, mobile device, network server

1. UVOD

Raspberry Pi i Wemos D1 dio su moderne tehnologije kojoj je cilj sačuvati funkcionalnost i performanse, a smanjiti dimenzije proizvoda. Raspberry Pi je verzija mikro-računala koje sadrži sve komponente običnog stolnog računala ali u znatno manjem obliku. Predviđeno je i proizvedeno u svrhu promoviranja računalnih znanosti u školstvu, a ubrzo nakon proizvodnje primijetio se mnogo širi potencijal i mogućnosti koje Raspberry može ponuditi [3].

Wemos D1 predstavlja mikro-kontroler čija je primjena kontroliranje i upravljanje uređajima kojima se ne može direktno(žičano) pristupiti, a ta mogućnost osigurana je zahvaljujući integriranom WIFI modulom na pločici. D1 mini nudi mogućnost programiranja koristeći arduino programski jezik uz njihovo sučelje što olakšava njegovu primjenu i korištenje.

Obje komponente su odabrane za primjenu u ovom završnom radu zbog prednosti navedenih gore te velike pristupačnosti u nabavi komponenti i njihove niske cijene.

Slika 1. Raspberry Pi 3 (lijevo) [1] - Wemos D1 mini (desno) [2]

2. WEMOS D1 MINI

Mala cijena i dimenzija te mogućnost programiranja u Arduino sučelju, jedan su od glavnih čimbenika odabira kod daljinskog upravljanja. U sljedećim poglavljima objašnjena je WEMOS D1 komponenta, princip rada, te njena funkcija za bežično spajanje i ulazno izlazne operacije.

2.1. Wemos D1 općenito

Mikrokontroler je malo računalo ili drugim riječima integrirani elektronički krug koji se sastoji od niza malih elektroničkih komponenti povezanih u jednu cjelinu. Mikrokontroler sadrži sve što je računalu potrebno da bih funkcioniralo, memoriju, procesorsku jedinicu, te ulazne i izlazne priključke. Navedene komponente zajedno čine spoj koji primljene podatke sa ulaza obrađuje i zatim šalje na izlaz (slika 2).

Koriste se za automatizirano upravljanje ostalih uređaja u:

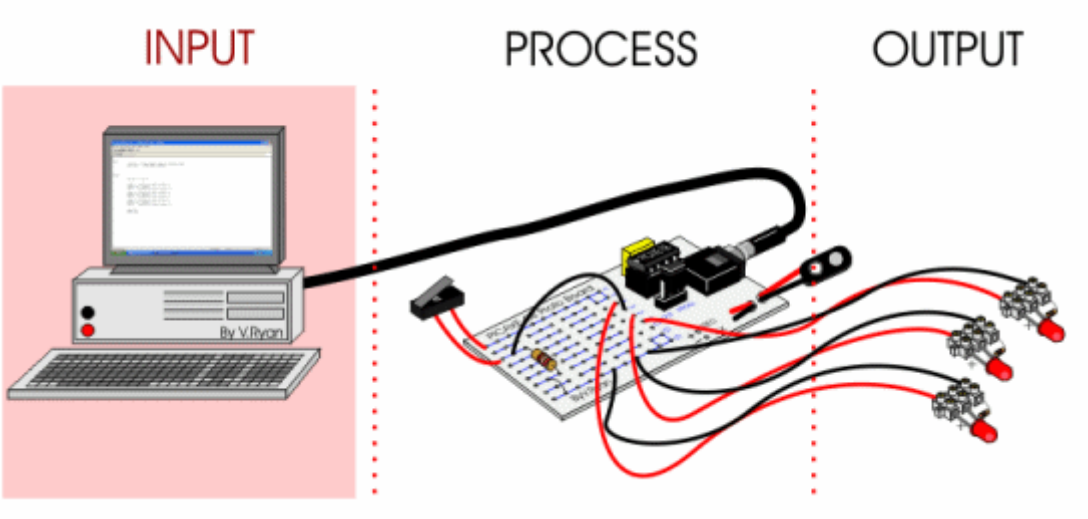
- automobilskoj industriji
- medicinskoj industriji
- zrakoplovnoj industriji
- kućanskim uređajima
- uredskim uređajima
- električnim alatima.[4]

Kao što se je vidljivo sa slike 1, Wemos D1 mini se sastoji od šesnaest pinova od koji se jedanaest odnosi na digitalne ulaze i izlaze preko kojih šalje i prima podatke. Osim digitalnih ulaza i izlaza nalaze se pinovi za uzemljenje, napajanje od 5 i 3.3V te reset i analogni ulaz koji radi na 3.3V.

2.2. Bežično spajanje i ulazno izlazne operacije

Funkcioniranja ulaza i izlaza objasniti će se na primjeru računala i istosmjernog motora. Računalo je spojeno na mikrokontroler koji je spojen sa motorom. Na slici 2 prikazano je računalo kao ulaz a motor izlaz. Na primjeru paljenja motora radnja bi se odvijala na sljedeći način. Unosom podatka za paljenje motora (koji je prethodno programiran i programski kod spremljen na memoriju mikrokontrolera) računalo šalje podatak na digitalni ulaz

mikrokontrolera gdje zatim kontroler prima i obrađuje taj podatak te prepoznaje koju zadaću mora odraditi. Prepoznavanjem podatka kontroler šalje dobiveno rješenje u obliku digitalnog signala preko digitalnog izlaza do motora koji se zatim pokreće. Navedeni primjer se dodatno može poboljšati korištenjem Wemos kontrolera jer time više nije potrebna žičana veza između računa i kontrolera već se kontroler može nalaziti direktno na motoru i preko WIFI modula postavljenog na pločicu primati signal.



Slika 2. Grafički prikaz izlaza i ulaza podataka [5]

2.3. Prednosti Wemos D1 kontrolera

Unatoč širokom izboru mikrokontrolera na tržištu potrebna za korištenjem bežičnog prijenosa signala skraćuje se mogući izbor. Ako se usporede performanse D1 kontrolera sa svojim alternativama ne vidi se značajni pomak u performansama, ali se primjećuje razlika u cijeni. Najpopularniji izbor prema trenutnim pretragama bio bi Adafruit Feather HUZZAH ESP8266 koji ima sve potrebno za ovaj projekt, ali nasuprot tome cijena mu je četverostruko veća dostupnost mu je ograničena. Uzimajući u obzir slične performanse i veliku razliku u cijeni odluka je jednostavna. Druge manje popularne alternative dolaze s višim cijenama, a s pronalaskom cjenovne prihvatljive verzije obično dolazi zamka da modul nije ujedno i mikrokontroler već samostalan WIFI modul koji je potrebno spojiti na neki drugi kontroler primjerice Arduino i slične koji samo dodaju dodani posao prilikom spajanja.

3. RASPBERRY PI 3

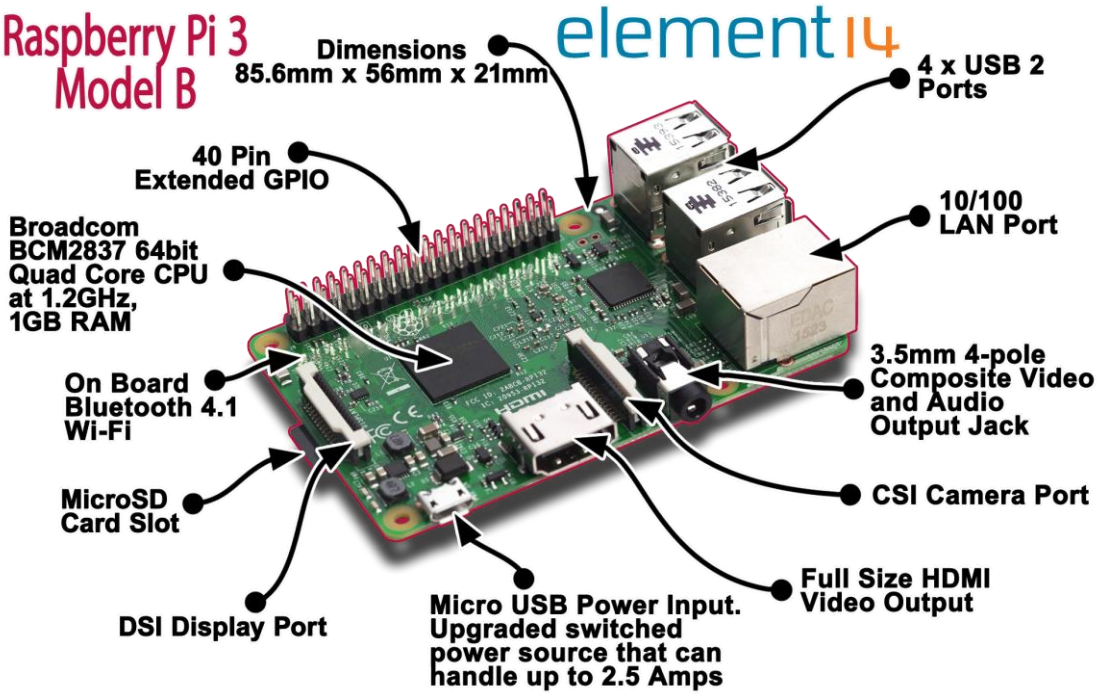
Raspberry Pi(slika 3.) je do 2016. godine prodan u više od jedanaest milijun primjeraka. Taj broj nam daje uvid u njegovu rasprostranjenost, podršku i očito zadovoljstvo korisnika s njegovim performansama i učinkom.[3]

3.1. Raspberry Pi kompjuterski sustav

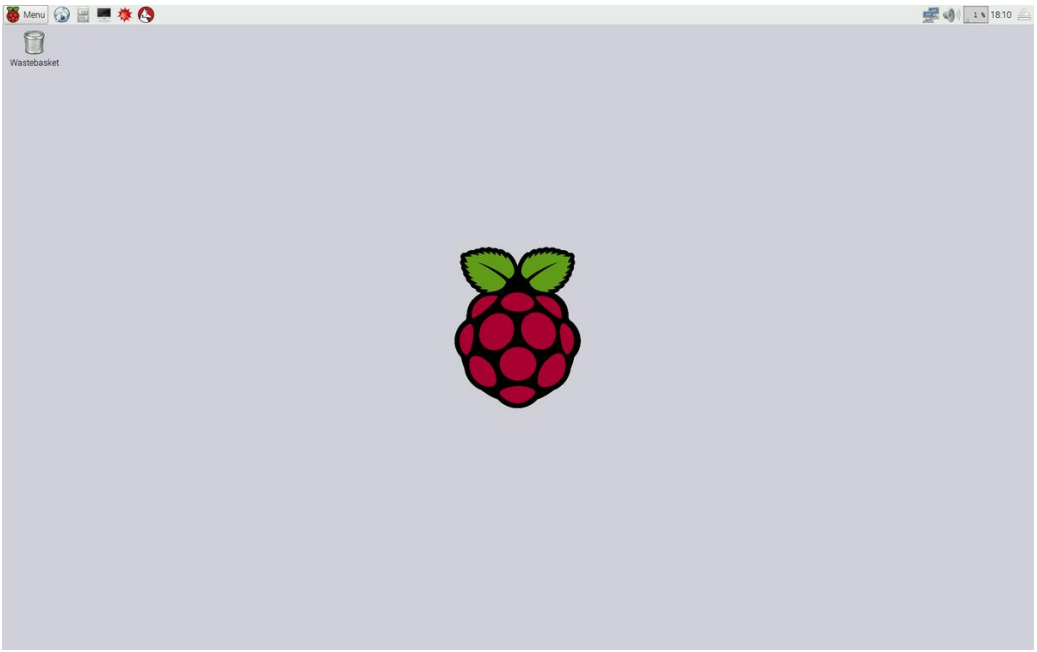
Kao i druga računala Raspberry se sastoji od osnovnih dijelova koji su potrebni da bi računalo funkcioniralo, ali i nekih dodataka. Sastoji se od:

- Radne memorije
- Procesorske jedinice
- Stalne memorije (SD kartica)
- Mrežna jedinica (WIFI i LAN)
- USB port-ovi te priključak za kameru i HDMI

Sve verzije Raspberry Pi-a koriste Broadcom procesorske jedince, a zadnja verzija ujedno i korištena u ovom završnom zadatku koristi Broadcom-ove četverojezgrene procesor sa radnim taktom od 1.2GHz . Navedeni takt zadovoljava sve potrebe za koje je raspberry zamišljen, a zajedno sa 1GB radne memorije čini ga pravim izborom za široki opseg projekata. Kao stalnu memoriju nema ugrađen tvrdi disk niti drugačiji oblik stalne memorije pa je kao rješenje tog nedostatka ugrađen utor za SD karticu. Kartica se koristi za pohranu podataka i samog sustava s jedinim ograničenjem maksimalnog prostora do 64GB. WIFI modul već ugrađen na pločicu čini raspberry pravim odabirom za ovaj zadatak jer performansama i mogućnostima može s lakoćom izvršavati bežičnu komunikaciju koja je potrebna za ovaj projekt. Spajanjem računala na napajanje podižemo besplatni sustav raspbian(slika 4) koji je baziran na linux-u(operativni sustav otvorenog koda) i osmišljen upravo za Raspberry za ispunjenje njegovog punog potencijala.[3]



Slika 3. Raspberry Pi 3B integrirani dijelovi [8]



Slika 4. Raspbian sučelje [9]

3.2. Odabir Raspberry pi-a

Kod odabira računala bitne stavke su performanse, cijena i podrška zajednice. Prema navedenim stavkama odabiremo Raspberry Pi. Najpoznatije alternative Raspberry Pi-u su Asus Tinker Bord, Banana Pi i Orange Pi. Usporedbom Raspberry Pi-a sa Asus-ovom

verzijom vide se snažnije performanse Asus-a, ali i duplo veća cijena nabave. Slika 5 prikazuje usporedbu performansi dva navedena računala. Jači procesor i više radne memorije za ovaj zadatak nisu potrebni iz razloga što Web2py predstavlja hardver-ski lagan i ne zahtjevni oblik tehnologije, pa pokretanje aplikacija i na slabijim performansama ne usporava rad aplikacije. Ovom usporedbom prednost dobiva Raspberry Pi sa svojim jeftinijom cijenom.

	WIN ASUS Tinker Board	Raspberry Pi 3 Model B
Core Processor (SoC)	Rockchip RK3288 Quad-Core 1.8Ghz	Broadcom BCM2837 Quad-Core 1.2Ghz
Benchmark Score (tested by GeekBench)	3925 Almost 2X faster!!	2092
RAM	2GB	1 GB RAM
Display	HDMI with H.264 4K decode capability	HDMI with HD resolution
NIC	Gb LAN	100M LAN
Audio	Supports up to 192K/24bit sample rate	48K/16bit
Wi-Fi	Yes, Wi-Fi 802.11 b/g/n + swappable antenna	Yes, Wi-Fi 802.11 b/g/n
Bluetooth	Yes, 4.0 + EDR	Yes, 4.1 + LE
SDIO	SDIO 3.0	SDIO 2.0
Official Supported OS	Linux – Debian / KODI	Linux – Debian

Slika 5. Raspberry Pi – Asus Tinker [11]

Banana Pi(slika 6) verzija računala daje identične performanse kao i Raspberry Pi, a jedina vidljiva razlika je u 1GB radne memorije više. Već spomenuta razlika radne memorije ne predstavlja bitni značaj, pa uz višu cijenu Banana Pi-a izbire se Raspberry Pi.

Zadnja alternativa vrijedna spomena je Orange Pi, kineska verzija Raspberry Pi-a koja performansama vijerno predstavlja dobru konkurenciju. Svojom manjom cijenom nadoknađuje upola manju radnu memoriju koja bi teoretski bila prikladna za mrežno upravljanje preko Web2py. Uzimajući u obzir činjenicu da se uz računalno upravljanje pojavljuju nepredviđeni problemi i u ovoj usporedi prevladao je Raspberry Pi. Razlog odabira je veća radna memorija te viša opća zastupljenost i podrška za Raspberry Pi. Podrška kao jedna od najbitnijih čimbenika kod odabira računala omogućava brzo i efikasno rješavanje nepredviđenih poteškoća i problema..

	Banana pi BPI-M64	Raspberry Pi 3
Photo		
Processor	Allw inner A64 64bit quad core Cortex A53 processor @ 1.2 GHz	Broadcom BCM2837 quad core Cortex A53 processor @ 1.2 GHz(4x ~2760 DMIPS)
GPU	ARM Mali-400MP2	VideoCore IV @ 300/400 MHz
Video Decoding	H.265/HEVC @ up to 4K @ 30 fps, H.264, VP8, AVS/AVS+ & MPEG1/2/2 @ 1080p60 , VC1 and MJPEG up to 1080p @ 30 fps	1080p30 for H.264, MPEG2* and VC1* 1080p video encoding (H.264) * Extra licenses required
Video Encoding	H.264 up to 1080p@60fps	Full HD H.264 video encoding
RAM	2GB DDR3	1GB LPDDR2
Storage	micro SD card slot & eMMC 8GB	micro SD card slot, non EMMC 8GB
Ethernet	Gigabit Ethernet	10/100M Ethernet via USB bridge
Wireless Connectivity	WiFi 802.11 b/g/n (2.4GHz) and BT 4.0 LE	WiFi 802.11 b/g/n (2.4GHz) and BT 4.1 LE
USB	2x USB 2.0 host ports 1x micro USB OTG port	4x USB 2.0 host ports 1x micro USB port
Disply	MIPI DSI	MIPI DSI
Camera	MIPI CSI	MIPI CSI
Video	HDMI 1.4 w ith CEC and 3.5mm composite	HDMI 1.4 w ith CEC and 3.5mm composite
Audio	HDMI and 3.5mm audio jack	HDMI and 3.5 mm audio jack (Shared w ith composite video)
GPIO	40-PIN: PWM,GPIO,UART,I²C bus,I²S bus,SPI bus,+3.3v,+5v,ground.	40-pin header w ith 26 –GPIOs, 1x UART (debugging), 1x SPI, 2x I2C, PCM/I2S, 2x PWM
Button	Reset button, Power button, Uboot button	Non Reset, Pow er and Uboot button
LED	User define LED (red/power, blue, green)	LED (red/power & green)
Dimensions	90 x 62 mm	85 x 56 mm
Linux Support	Official: Ubuntu 16.04 64-bit w ith Kernel 3.10 (No sure about GPU and VPU support)	Official: Raspbian w ith recent Linux 4.x kernel. 32-bit user space only (currently)
Android Support	Android 5.1	No
Windows 10 IoT	No	Yes

Slika 6. Raspberry Pi – Banana Pi [12]

3.3. Raspberry Pi i Web2py

Web2py je sustav otvorenog koda za brzu i jednostavnu izradu web aplikacija baziran na programskom jeziku Python[6]. Opremljen sa svim potrebnim paketima i knjižnicama podataka potrebnih korisniku prilikom izrade aplikacije. Rješavanjem danog problema Raspberry se ponaša kao isporučitelj bežične lokalne mreže na koju je spojen Wemos D1 kontroler. Drugim riječima Raspberry će biti ulaz. Ujedno se na mikro-računalu uspostavlja web aplikacija izrađena Web2py tehnologijom. Aplikacija sadržava sve potrebne dijelove koji omogućavaju kontroliranje pokretnih laboratorijskih objekta. Podatci primljeni preko interaktivnog dijela aplikacije se preko bežične mreže šalju između računala i kontrolera.

4. RASPBERRY PI MREŽNI SUSTAV TEMELJEN NA WEB2PY TEHNOLOGIJI

Pokretanjem Raspberry Pi-a podiže se sustav Raspbian te nudi mogućnost spajanja na ostale bežične mreže. Spajanje na širokopojasnu internet vezu se koristi za skidanje svih potrebnih nadogradnji Raspbian sustava kako bi pravilno funkcionirao, a nakon nadogradnje sustava obrnuo se proces rada WIFI modula, odnosno postavlja se u način rada *router* što omogućava spajanje drugih uređaja na sam Raspberry.

4.1. Raspberry Pi kao WIFI odašiljač [7]

Broadcom BCM43438 je WIFI čip korišten u Raspberry-u i njegov programibilni kod je otvoren za uređivanje pa će se napraviti izmjene modula da se prilagodi na drugačiji način rada. Prije početka izmjene načina rada potrebno je spojiti računalo na širokopojasnu internetsku vezu i izvršiti nadogradnju raspbian sustava te preuzimanje svih potrebnih paketa.

- sudo apt-get update
- sudo apt-get dist-upgrade
- sudo apt-get install dnsmasq hostapd

Prve dvije linije nadograđuju sustav Raspbian sa najnovijim verzijama paketa i što je još bitnije zadnjim važećim knjižnicama podataka. Treća linija koda instalira dva paketa *dnsmasq* i hostapd potrebna za izmjenu načina rada WIFI modula. *Hostapd* je paket koji omogućava pretvaranje WIFI modula u izvor mreže, a *dnsmasq* omogućuje primjenu važećeg mrežnog protokol *DHCP* i važećih DNS postavki. *DHCP* (*Dynamic Host Configuration Protocol*) je mrežni protokol koji dodjeljuje IP adrese i ostale mrežne postavke te kontrolira pridijeljene IP adrese da ne bi došlo do podudaranja što dalje omogućava nesmetano spajanje uređaja na mrežu. DNS ili Domenski sustav imena je osnovni element funkcionalnosti interneta čija je najosnovnija ideja uzimanje naziva web stranice i pretvaranje naziva odnosno domene u važeću IP adresu. Upravljanje Raspbian bežičnog sučelja je moguće sa dvije verzije upravljačkih paketa, *wlan0* i *dhcpd*. Server koristi *dhcpd* koji je ujedno postao osnovno korišteno sučelje u zadnjim verzijama sustava a predstavlja statičku vrstu mreže. Korištenje željenog sučelja za ovaj projekt je potrebno strogo ograničiti na *dhcpd* pa se blokira upotreba alternative wlan0.

```
➢ sudo nano /etc/dhcpcd.conf
➢ denyinterfaces wlan0
```

U prvom koraku otvaramo konfiguracijski dokument za upravljanje bežičnim sučeljima a tada se na sam početak te datoteke doda druga linija koda koja zabranjuje korištenje wlan0 sučelja. Zabranjivanje korištenja wlan0 sučelja pretvara dinamičku mrežu u mrežu sa statičkom IP adresom. Statičku mrežu treba definirat pomoću sljedećih naredbi.

```
➢ sudo nano /etc/network/interfaces

➢ allow-hotplug wlan0
➢ iface wlan0 inet static
➢     address 172.24.1.1
➢     netmask 255.255.255.0
➢     network 172.24.1.0
➢     broadcast 172.24.1.255
➢ #    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf

➢ sudo service dhcpcd restart
➢ sudo ifdown wlan0; sudo ifup wlan0
```

Prva linija koda otvara konfiguracijsku datoteku za definiranje statičke IP adrese. Nakon otvaranja datoteke izmjenjuje se dio koda sa željenim adresama servera. *Address* definira statičku adresu servera odnosno računala, a *netmask* definira mogući broj spojenih uređaja. Nakon što su se postavile željene postavke mreže potrebno je resetirati bežično sučelje sa zadnje dvije navedene naredbe. Postavke mreže su spremljene i pripremljene pa je sada potrebno modificirati i pokrenuti hostapd paket kako bi računalo započelo sa odašiljanjem mreže. *Hostapd* je konfiguriran preko datoteke *hostapd.conf* koja se pokreće sljedećom naredbom.

```
➢ sudo nano /etc/hostapd/hostapd.conf
```

U konfiguracijskoj datoteci je potrebno napraviti izmjene koda, a zbog duljine samog koda primjer sa uputama koje podatke je potrebno izmijeniti se nalazi u prilogu. Nakon što su izmjene unesene i spremljene, te iste izmjene treba pokrenuti, a pokretanjem navedenog konfiguracijskog paketa mreža započinje s radom, odnosno WIFI mreža željenog imena u ovom slučaju Pi3-AP je vidljiva drugim uređajima koji se sada mogu spojiti. Poželjno je da se mreža upali prilikom svakog pokretanja Raspberry Pi računala s čime se sprječava duži vremenski prekid rada. Automatsko pokretanje se ostvaruje uređivanjem paketa *hostapd*.

- sudo nano /etc/default/hostapd
- DAEMON_CONF="/etc/hostapd/hostapd.conf"

Prvom naredbom se otvara uređivanje paketa, gdje se zatim uređuje redak sa DAEMON CONF kodom u kojem definiramo koju mrežnu konfiguracijsku datoteku sustav uzima pri paljenju sustava. Nakon konfiguriranje rada WIFI modula i same mreže potrebno je još konfigurirat DNS server. DNS se konfigurira otvaranjem *dnsmasq.conf* datoteke.

```
sudo nano /etc/dnsmasq.conf
```

U ovoj konfiguracijskog datoteci se brišu sve postojeće linije koda i zamjenjuju sa novima navedenima ispod.

- interface=wlan0 # Koristi se sučelje wlan0
- listen-address=172.24.1.1 # Eksplicitno praćenje definirane statičke adrese na koju se spajaju uređaji
- bind-interfaces # Zbrinjavanje da se podatci šalju samo sa odabranog sučelja
- server=8.8.8.8 # Korištenje javnih Google DNS servera
- domain-needed # Potrebna je domena prilikom spajanja, adrese moraju biti ispravne
- bogus-priv # Ako nema spojene adrese ne šalji podatke u prazno.
- dhcp-range=172.24.1.50,172.24.1.150 # Novi uređaju se spajaju na IP adrese u intervalu od 172.24.1.50 do 172.24.1.150

Mreža i DNS serveri su spremni za rad pa je potrebno dopustit slanje paketa između uređaja i servera. Dopuštanje slanja paketa se izvodi uređivanjem sysctl.conf konfiguracijske datoteke.

- sudo nano /etc/sysctl.conf
- #net.ipv4.ip_forward=1

Nakon otvaranja konfiguracijske datoteke potrebno je pronaći naredbu navedenu iznad te maknuti simbol ljestvi ispred naredbe nakon čega preostaje samo spremiti datoteku i napraviti resetiranje računala. Ako je sve pravilno konfigurirano zadnje dvije naredbe dati će potpuno spreman i funkcionalan server na koji se uređaji spajaju.

- sudo service hostapd start
- sudo service dnsmasq start

4.2. Primjena Web2py tehnologije na Raspberry Pi računalu

Za pokretanje Web2py sučelja na Raspbian sustavu potreban nam je python koji dolazi već pred instaliran sa sustavom. Python se koristi za pokretanje skripte Web2py koja se nalazi u *„source code“* verziji Web2py-a koja se preuzima sa službene web stranice. Nakon preuzimanja, datoteka se raspakira te se preko terminala pokreće Web2py sučelja.

```
➢ python web2py.py
```

Nakon što se skripta pokrene otvara se prozor sa slike 7.

Slika 7. Web2py sučelje za pokretanje aplikacije

U prikazanom prozoru potrebno je odabrati *Public* server IP adresu što omogućava pristup aplikaciji, server port te lozinku koja omoguće administratorske ovlasti nad aplikacijama postavljenim u web2py. Pritiskom gumba start server pokrenuti će se server Web2py aplikacije, i tada se dobiva pristup željenoj aplikaciji u ovom slučaju aplikaciji za upravljanje vozilom. Link koji vodi do aplikacije uz primjer sa slike izgledao bi: https://192.168.1.2:8000/zavrsni gdje nam „zavrsni“ predstavlja ime aplikacije kojoj pristupamo.

5. PROJEKTIRANJE SUSTAVA S POKRETNIM OBJEKTOM

Sustav za upravljanje uređaja se sastoji od Raspberry Pi računala koje predstavlja server odnosno nositelja Web2py aplikacije te uređaja sa kontrolerom. Sustav funkcionira spajanjem uređaja na bežičnu mrežu od računala gdje zatim pristupa aplikaciji stvorenoj u Web2py sučelju. Aplikacija sadrži dio koda smještenog u kontroler a dio u indeksnu stranicu. Uređaj također na svom kontroleru ima spremljen kod koji upravlja radom uređaja u ovisnosti o varijablama poslanih od strane aplikacije sa servera.

5.1. Programiranje kontrolera na upravljanom uređaju

Upravljani uređaj je vozilo koje se sastoji od dva motora sa dva smjera vrtnje. Motori omogućuju vozilu da ide naprijed, nazad, lijevo i desno ovisno o želji korisnika. Kao i svaki uređaj jedna od sastavnih dijelova koda su definirani izlazi za pinove, pa tako svaki motor ima definiran pin digitalno ulaza. Cijeli kod se nalazi u prilogu, a u sljedećim odlomcima objašnjeni su najbitniji dijelovi za funkcioniranje vozila.

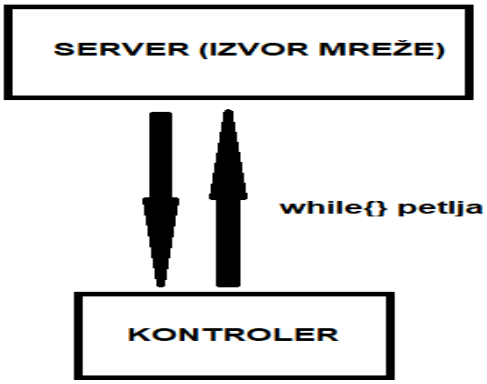
```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <WiFiServer.h>

WiFiClient client;
WiFiServer server(PORT);

const char *ssid="Pi3-AP";
const char *pass="123456789";
```

Prve tri naredbe nam definiraju uvoz knjižnice podataka potrebnih za rad kontrolera, odnosno prva knjižnica je knjižnica za sam rad kontrolera, a druge dvije se odnose na mogućnost korištenja WIFI modula koju kontroler nudi. Sve uvezene knjižnice podataka ograničene su na kontroler sa WIFI modulom, u ovom slučaju na D1 mini i nisu primjenjive na drugim vrstama kontrolera. Četvrta naredba pravi klijenta koji ima mogućnost povezivanja na druge mreže, dok WIFIServer definira port kroz koji će se komunikacija vršit što je u ovom slučaju port 90. Zadnje dvije naredbe definiraju naziv i lozinku mreže na koju se kontroler spaja, a naziv mreže i lozinka su definirani prilikom konfiguracije Raspberry Pi-a za rad kao server.

```
WiFi.config(ip, gateway, subnet);

 WiFi.begin(ssid,pass);
 while(WiFi.status()!=WL_CONNECTED){
  Serial.print(".");
  delay(500);
 }
```

WiFi.config() naredbom se postiže konfiguracija mrežnog identiteta vozila, a u naredbu ulaze tri argumenta, odnosno tri adrese, ip, gateway i subnet koje su prethodno definirane. Kada je uređaja spojen na mrežu postoji mogućnost prepoznavanja uređaja prema njegovoj definiranoj IP adresi, a upravo to se koristiti za upravljanje vozilom. U aplikaciji se bira vozilo s obzirom na njegovu IP adresu i na tu adresu se šalju podatci za upravljanje. Naredba u drugom redu započinje povezivanje kontrolera na bežičnu mrežu, u ovom slučaju je to bežična mreža odašiljana od strane Raspberry Pi računala. Nakon prethodne naredbe dolazi petlja koja služi da se programski kod ne nastavlja sve dokle se veza nije uspostavila.

Dio koda zaslužan za funkcioniranje odnosno upravljanje vozilo se nalazi na sljedećoj stranici. Ovaj dio koda se nalazi u petlji koja se izvršava sve do trenutka kada se mreža prekine. Aplikacija pokretana u Web2py sučelju šalje podatke u formatu broj,broj,broj,broj, a podatci se šalju redom, pa tako prvo stigne broj pa zarez, pa broj i tako u krug. Petlja provjera podatak po podatak redom kojim dolaze. Kada stigne broj provjerava se koji je to broj po redu s obzirom na zareze. Ako petlja stigne do prvog zareza bilježi se taj podatak te prepoznaj da nakon zareza stiže novi broj koji je u tom trenutku definiran kao drugi broj po redu. U ovom primjer je zamišljeno da se prvi broj odnosi na prvi motor, drugi broj na drugi motor, a treći i četvrti broj nam definiraj smjer vrtnje pojedinog motora, odnosno na koju stranu će voziti i u koju stranu će skretati.



Slika 8. Shema rada

```
while(client.connected()){

   if (client.available()){

     int inChar = client.read();

     if (isDigit(inChar)) {
       inString += (char)inChar;
     }

     // ako je stiago zarez
     else if (inChar == ','){
       if (i==0){
         analogWrite(M1,inString.toInt());
       }
       else if (i==1){
         analogWrite(M2,inString.toInt());
       }
       else if (i==2){
         digitalWrite(DIR1,inString.toInt());
       }
       i++;
       inString = "";
     }

     // novi red
     else if (inChar == '\n') {
       digitalWrite(DIR2,inString.toInt());
       i=0;
       inString = "";
     }
   }
 }
```

5.2. Programiranje Web2py aplikacije

Web2py aplikacija zaslužna je za upravljanje vozilom odnosno odabir brzine i smjera te slanjem tih podataka. U teoriji Web2py aplikacija predstavlja web stranicu koja se sastoji od dva glavna dijela, *controller* i *view* te ostalih pratećih dijelova kako bi stranica bila potpuna. Pozivanjem stranice odnosno aplikacije prvo se odlazi u *controller* gdje nas preusmjerava na *view* odnosno index.html. Index sadrži interaktivne izbornike osmišljene za upravljanje vozilom. Ovaj primjer se sastoji od padajućeg izbornika, gumba i interaktivnog virtualnog upravljača(*joystick*). Index.html započinje s naredbama kojima uvodimo knjižnice podataka

bootstrap-a i jquery-a prikazanima kodom ispod. Bootstrap je niz alata koji olakšavaju korištenje raznih komponenti kod izrada web stranica kao sto su dodavanje gumba, klizača, formi, upozorenja, kartica i mnogih drugi interaktivnih elemenata. Jquery predstavlja knjižnicu podataka za javascript(programski jezik html-a odnosno web dizajna). Bootstrap je u zadatku korišten za programiranje padajućeg izbornika i gumba, a jquery odnosno javascript će se koristiti kod virtualno upravljača i kod slanja podataka vezanih uz navedene komponente.

```
        <link rel="stylesheet"
href="{{=URL('static','css/bootstrap.min.css')}}"/>
        <link rel="stylesheet" href="{{=URL('static','css/bootstrap-
slider.min.css')}}"/>

        <!-- java script -->
        <script src="{{=URL('static','js/jquery.js')}}"></script>
        <script src="{{=URL('static','js/bootstrap.min.js')}}"></script>
        <script src="{{=URL('static','js/bootstrap-slider.min.js')}}"></script>
```

Stranica se po pravilu sastoji od glave i tijela, u glavu(head) se piše kod koji se neće vizualno prikazati na stranici, a u tijelo(body) ide dio koda koji se vizualno prikazuje. Uz knjižnice podataka, stilove odnosno uređenja elemenata u područje head idu i skripte. Skripte su elementi html koji vrše zadanu radnju na stranici odnosno pomažu stranici da bude interaktivna. U aplikaciji se unutar glave nalaze skripte koje šalju podatke u kontroler tek kada su aktivirane, a aktiviraju se interakcijom s elementima na stranici. Unutar tijela aplikacije uz svaki element definiran je njegov posebno programiran prostor i svaki element ima svoj identifikacijsku oznaku „id“ koja se koristi za čitanje podataka s toga elementa.

```
<div class="form-group">
    <label for="ip">Odaberi auto:</label>
    <select class="form-control" name="ip" id="ip">
        <option value="192.168.172.101">Auto 1</option>
        <option value="192.168.172.102">Auto 2</option>
        <option value="192.168.172.103">Auto 3</option>
    </select>
</div>
```

Slika 9. Padajući izbornik

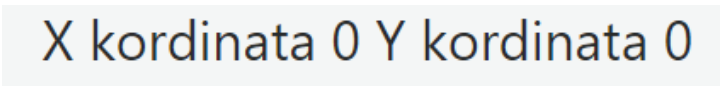
Kod iznad pokazuje dio html-a zaslužnog za vizualizaciju padajućeg izbornika. Izbornik se sastoji od tri opcije odabira prikazane samo kao primjer u slučaju potrebe upravljanja većeg broja vozila. Potrebno je upravljati samo jednim vozilom pa će u ovom slučaju biti dovoljna opcija samo za Auto 1. Može se primijetiti da Auto 1 imam zadanu vrijednost(*value*) koja predstavlja IP adresu vozila koje se spaja. *Class* naredba unutar elementa definira izgleda funkcionalnost padajućeg izbornika a poziva se na knjižnicu podataka bootstrapa koju smo uvezli na početku koda.

```
<div class="form-group">

        <br>
        <button type="button" id="iskljuci" value="iskljuci" class="btn btn-
danger">Isključći motore</button>

</div>
```

Slika 10. Gumb za isključivanje

Naredbe za prikaz gumba prikazane su iznad. Gumb je uključen u aplikaciju iz sigurnosnih razloga. Ako dođe do nepredviđenih smetnji prilikom upravljanja i lošeg reagiranja vozila, uvijek se može pritisnuti gumb koji mijenja digitalni izlaz prema motoru u stanje nula, odnosno gasi motore.

```
<span>X kordinata  </span>
<span id="x"></span>
<span>      </span>
<span>Y kordinata  </span>
<span id="y"></span>
```



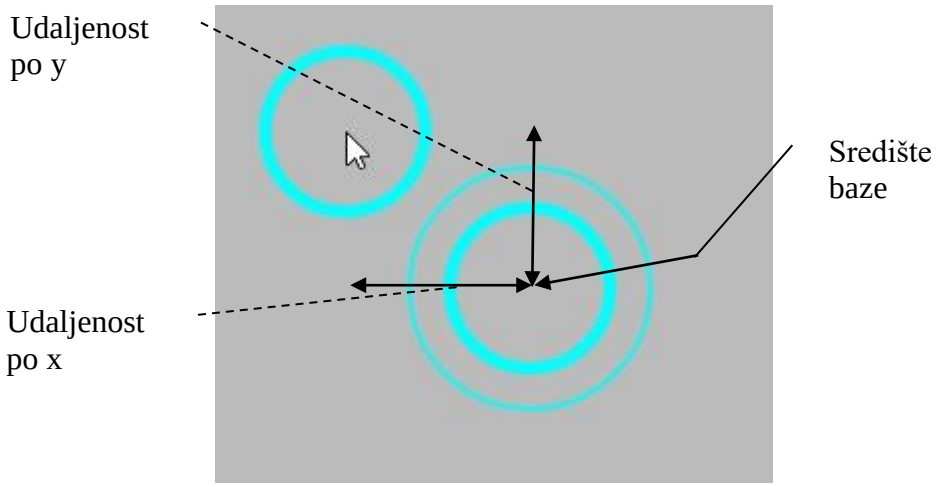
Slika 11. Pozicija upravljača s obzirom na x,y

Iz informativnih razloga kako bi pozicija upravljača bila poznata postavljen je element koji pokazuje poziciju upravljača s obzirom na svoju bazu. Pozicija upravljača u odnosu na bazu

definirana je udaljenošću prema koordinatama x i y. Osim informativnog značaja provjere smjera kretanja vozila ovaj element služi za uzimanje tih koordinata koje se preko jqeury-a i python-a šalju prema vozilu.

Najsloženiji element u aplikaciji je virtualni upravljač [10] čiji prostor se nalazi ispod prethodno navedenih elemenata. Cijeli taj prostor je interaktivan što znači da se upravljač poziva klikom i držanjem miša bilo gdje unutar toga prostora.

```
<script src="{{=URL('static','js/virtualjoystick.js')}}" charset="utf-8"></script>
<script>
	console.log("touchscreen is", VirtualJoystick.touchScreenAvailable() ?
"available" : "not available");

	var joystick	= new VirtualJoystick({
		container	: document.getElementById('container'),
		mouseSupport: true,
		limitStickTravel: true,
		stickRadius: 100

	});
	joystick.addEventListener('touchStart', function(){
		console.log('down')
	})
	joystick.addEventListener('touchEnd', function(){
		console.log('up')
	})

	setInterval(function(){
		var outputEl	= document.getElementById('x');
		var outputE2	= document.getElementById('y');
		outputEl.innerHTML = joystick.deltaX()
		outputE2.innerHTML= joystick.deltaY()
			}, 100);
</script>
```



Slika 12. Virtualni upravljač

Slika 12 prikazuje virtualni upravljač s bazom(veliki krug) i aktivnim dijelom(izvučeni manji krug). Aktivni dio se može povući za udaljenost 200 po x i 200 po y koordinati. Pomicanje aktivnog dijela prema dole vozilo ide nazad, a pomicanjem prema gore vozilo se kreće naprijed, po istom načelu funkcionira i skretanje. Html kod vezan za upravljač se zasniva na Javascript programskom jeziku pa se tako nalazi unutar skripte. Kod započinje uvoženjem knjižnice podataka za upravljač. Upravljač se realizira naredbom *new VirtualJoystick({})* koja stvara upravljač sa željenim opcija. Korištene su četiri opcije, prva definira područje rada odnosno *container*, druga daje dopuštenje da se upravljačem može upravljati i klikom miša, treća i četvrta opcija ograničavaju aktivni dio na 200 jedinica u odnosu na bazu. Iduće dvije funkcije unutar skripte određuju kako se upravljač ponaša kada klik započne i kada klik prestane, odnosno kada je upravljač pušten vrijednosti se vraćaju u nulu, upravljanje prestaje. Za kraj je dodana opcija praćenje koordinata koju upisujemo u span element prethodno naveden i pokazan. Svakim pomakom upravljača koordinate se mijenjaju, a ta promjena se prikazuje i unutar span elementa.

Sada kada su kreirani svi interaktivni elementi i pripadajuće funkcije potrebno je očitati vrijednosti koje se mijenjaju. Funkcija koja odrađuje taj dio nalazi u glavi(head) html koda a izvedena je na sljedeći način.

```
$(document).ready(function(){
        var mouseStillDown = false;

        $('#container').mousedown(function() {
          vozi();
        }).mouseup(function() {
          clearInterval(mouseStillDown);
          mouseStillDown = false;
        });

        function vozi() {
                $.ajax({
                        url: "{{=URL('default' , 'posalji')}}",
                        type: 'POST',
                        async:false,
                        data: {
                                "ip": $("#ip").val(),
                                "m":$("#y").text(),
                                "dir":$("#x").text() },
                        dataType: 'json',
                         });

                if (!mouseStillDown) {
                        mouseStillDown = setInterval(do_something, 50);
                                        }
        }

        $("container").mouseup(function(){
                $.ajax({
                        url: "{{=URL('default' , 'iskljuci')}}",
                        type: 'POST',
                        data: {"ip": $("#ip").val()},
                        dataType: 'json',
                        });
        });

        $("#iskljuci").click(function(){
                $.ajax({
                        url: "{{=URL('default' , 'iskljuci')}}",
                        type: 'POST',
                        data: {"ip": $("#ip").val()},
                        dataType: 'json',
                        });
        });
});
```

Kod započinje sa simbolom dolara što predstavlja sintaksu jQuery-a, točnije prva linija koda govori da je funkcija spremna za izvršavanje tek ako je stranica u potpunosti učitana sa svim pripadajućim elementima. Unutar navedene funkcije imamo četiri funkcije. Prva funkcija se odnosi na element imena „container“ što predstavlja *id* elementa upravljača. Funkcija detektira klik miša, pa ako je otkriven klik poziva se funkcija „vozi“. Navedena funkcija koristi Ajax kako bi očitala i poslala podatke unutar *span* elemenata koji sadrže koordinate kontrolera. Očitane koordinate se šalju u *controller* Web2py aplikacije. Dodana je i petlja koja omogućava odvijanje odnosno ponovno pozivanje funkcije „vozi“ u slučaju ako se klik miša nalazi u pritisnutom stanju. Kada klikne prestane, taj događaj se otkriva pomoću funkcije *mouseup* te se u tom trenutku prekida petlja i šalju podatci za gašenje motora. Programski dio Ajax metode služi za slanje vrijednosti elemenata između različitih dijelova web aplikacije. Najbitniji dio Ajax metode za ovaj zadatak je dodavanje linije *async:false* koja šalje podatke po načelu jedan po jedan. Odnosno tek kada prvi dio podataka stigne na željenu adresu tek onda se počinje slati sljedeći niz podataka. Ova naredba je potrebna kako ne bih došlo do zagušenja mreže. Ujedno ova naredba je i najveća mana ove aplikacije jer njena brzina nije u skladu sa današnjim standardima te se upravljanje odvija sa blagim kašnjenjem.

Controller je jedan glavnih dijelova Web2py aplikacije koji je baziran na Python kodu, a zadužen je za upravljanje cijelom aplikaciju. Unutar *controller* dijela ove aplikacije programirane su funkcije koje uzimaju vrijednosti html elemenata i uz pravilno formatiranje ih šalju u mikrokontroler vozila. Na početku *controller*-a poziva se „*socket*“ koji predstavlja mrežno sučelje za komunikaciju, u ovom slučaju između servera i vozila.

```
global s
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

Socket se definira sa dva člana, AF_INET definira vrstu adresa koje se koriste prilikom komunikacije, u ovom slučaju AF_INET predstavlja IPv4 protokol. SOCK_STREAM definira da se koristi TCP protokol koji je jedan od najraširenijih internetski protokola i omogućuje brzu komunikaciju preko kreiranih virtualnih konekcija.

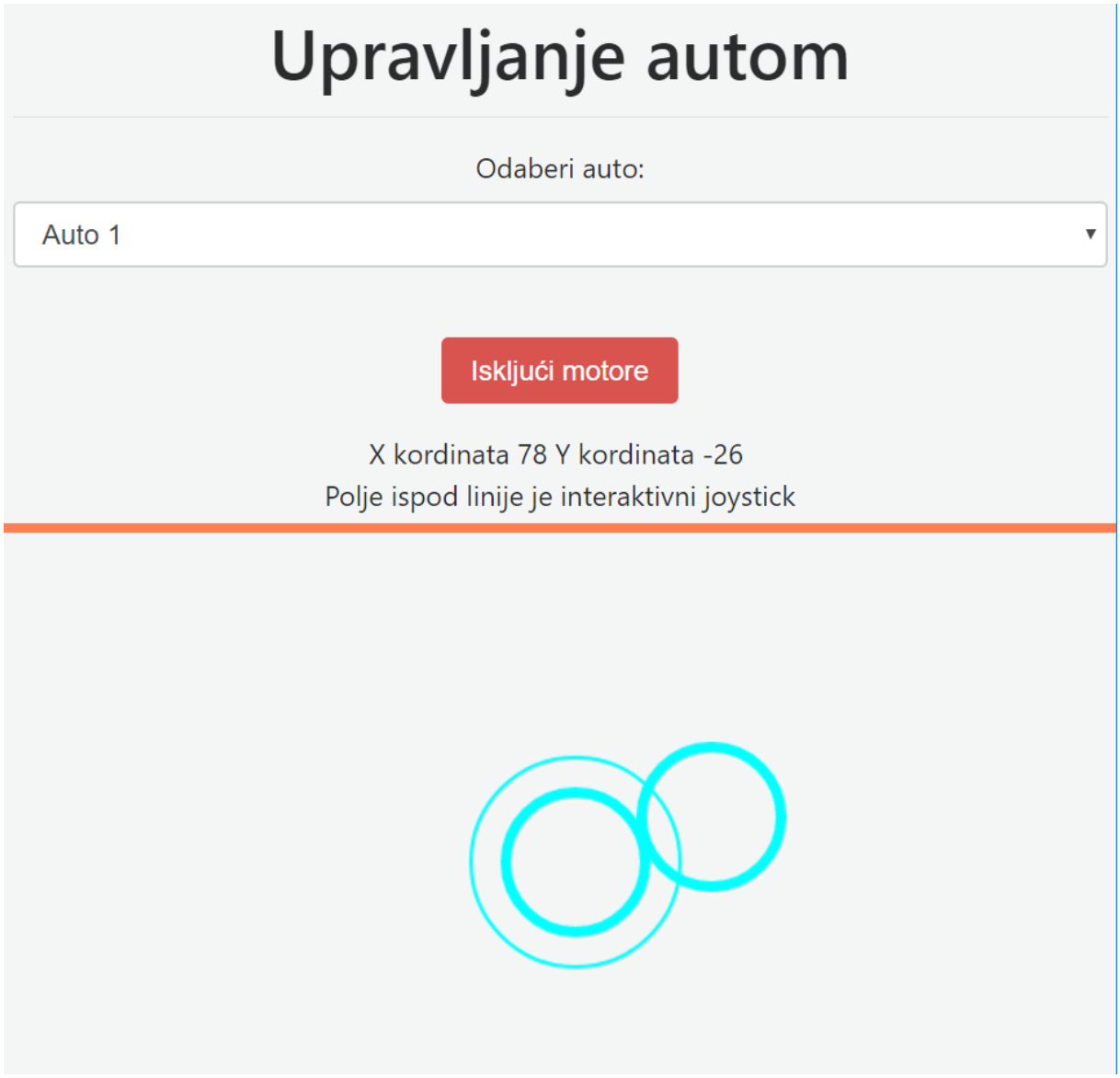
```
def iskljuci():
    global s
    s.connect((request.vars.ip, 90))
    s.sendall("0,0,1,0")
    s.close()
```

Funkcija „iskljuci“ je pozvana pritiskom gumba za isključivanje motora uz čiji se pritisak kao što je ranije navedeno šalje IP adresa vozila kojim se upravlja. Dana funkcija stvara konekciju s vozilom i šalje vrijednosti 0,0,1,0 što programskim jezikom pročitano znači da prvi i drugi motor dobivaju nulu na izlazu, smjer prvog motora se postavlja na naprijed, a drugog motora na desno. Sljedeća funkcija u *controller*-u je funkcija „posalji“ koja prima koordinate upravljača i vrijednosti tih koordinata prilagođava i šalje u mikro-kontroler vozila. Funkcija „posalji“ započinje dohvaćanjem IP adrese upravljanog vozila, a zatim nakon toga dohvaća vrijednosti koordinata x i y od upravljača. Pozitivna x os gleda desno, a pozitivna y os gleda prema dole. Uzimajući u obzir smjer pozitivnih osi provjerava se imaju li vrijednost x i y koordinata pozitivan i negativan predznak, u ovom slučaju ako upravljač gleda prema desno koordinata x osi je pozitivna i to nam govori da auto skreće prema desno. Pomicanjem upravljača prema gore koordinata y osi je ispisana sa minus predznakom pa tako funkcija postavlja smjer kretanja auta ravno. Uz provjeru smjera, provjerava se udaljenost upravljača od središta baze koja se nalazi u razmaku od 200 jedinica. Raspon od nula do dvjesto se mapira vrijednostima od 500 do 1023 koje predstavljaju PWM izlaz na kontroleru za motor. Početak izlaznih vrijednosti motora zadane su sa 500 jer manje vrijednosti nisu u mogućnosti izazvati pokretanje motora. Manje vrijednosti od 500 nemaju dovoljno dane struj kako bi svladali gubitke odnosno otpore uzrokovane lošom izradom motora. Koordinatna udaljenost od 0 predstavlja izlaz 500 a skladno tome koordinatna vrijednost od 200 predstavlja izlaz 1023. Vrijednosti između 500 i 1023 su raspoređene odnosno mapirane u pravilnom razmaku korištenjem interp1d funkcije. Dio funkcija „posalji“ prikazan je ispod, a u prilogu je dana cijela verzija koja sadrži dva dijela koja omogućavaju upravljanje dva različita vozila dostupnima za rješavanje ovog završnog zadatka. Razlika u vozilima je pozicijama motora, pa je skladno tome programski kod sličan u oba slučaja.

```
def posalji():
    global s
    ip = request.vars.ip
                    #-------------
        p1 = int(request.vars.m1)
        p2 = int(request.vars.dir)
        m = interp1d([0,200],[500,1023])

        if p1<0:
            dir1=dir2=1
            if p2>20:
                m2=math.sqrt((p1*p1)+(p2*p2))
                m1=abs(200-p2)
                m1 = m(m1)
                m2 = m(m2)
            elif p2<-20:
                m1=math.sqrt((p1*p1)+(p2*p2))
                m2=abs(200+p2)
                m1 = m(m1)
                m2 = m(m2)
            else:
                m1=m2=abs(p1)
                m1 = m(m1)
                m2 = m(m2)

        elif p1>0:
            dir1=dir2=0
            if p2>0:
                m2=math.sqrt((p1*p1)+(p2*p2))
                m1=abs(200-p2)
                m1 = m(m1)
                m2 = m(m2)
            elif p2<0:
                m1=math.sqrt((p1*p1)+(p2*p2))
                m2=abs(200+p2)
                m1 = m(m1)
                m2 = m(m2)
            else:
                m1=m2=p1
                m1 = m(m1)
                m2 = m(m2)
        else:
            m1=m2=0
            m1 = m(m1)
            m2 = m(m2)
```

Spajanjem dva funkcionalna dijela, aplikaciju i kontroler dobije se upravljiva cjelina. *View* i *controller* unutar aplikacije zasluženi su za povezivanje interakcije korisnika i slanje pripadajućih podataka na kontroler, a kontroler za daljnju obradu podataka i pokretanje

motora. Konačni izgleda aplikacije prikazan je slikom 13. Također sa slike 13 se može vidjeti primjer upravljača u položaju koji šalje podatke za kretanje vozila u smjeru ravno te uz blago skretanje prema desno.



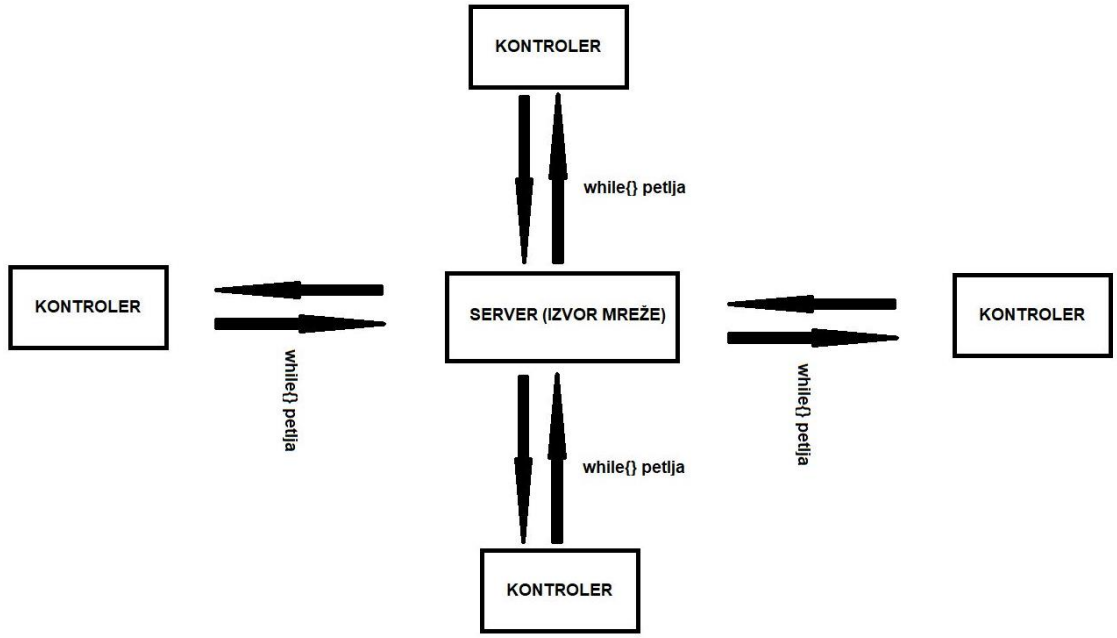
Slika 13. Web2py aplikacija za upravljanje uređajem

Kao glavni nedostatak ovog sustava mreže između servera i vozila izdvaja se već ranije spomenuta sporost Ajax metode. Ubrzavanje tog djela komunikacije nije moguće što je glavni nedostatak primjene Ajax-a. Kao alternativno rješenje se nameće jedino skroz drugih oblik način upravljanja ili pokušaj uspostavljanja *web socket* veze između servera i vozila. Navedena veza je još u razvojnoj fazi i nije namijenjena za primjere upravljanja prikazane u ovom zadatku.

6. MREŽE POKRETNIH OBJEKATA I BUDUĆI RAZVOJ OVAKVIH LABORATORIJSKIH MODELA

Mreža pokretnih objekata je zanimljiv koncept u kojem se ne bih upravljalo samo jednim uređajem kao što je prikazano u ovom zadatku, već bi postojala mogućnost povezivanja više uređaja u jednu aplikaciju preko koje bi korisnik mogao ručno upravljati ili postavljati automatsko upravljanje laboratorijskim modelima.

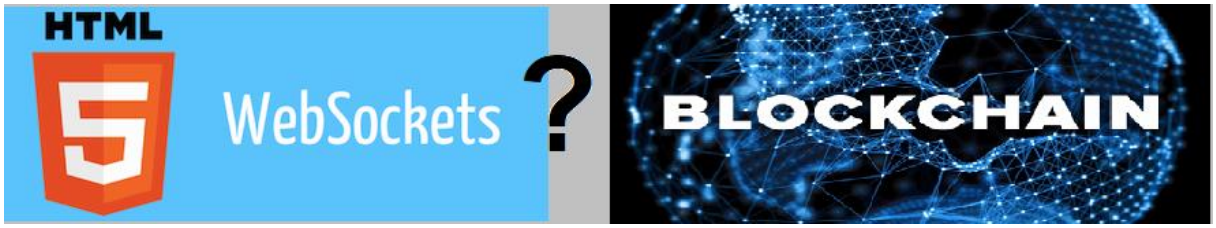
6.1. Mreža objekata i budući razvoj



Slika 14. Mreža objekata

Trenutno je aplikacija predviđena za rad samo s jednim objektom što je bilo predstavljeno kao problem zadatka koji treba riješiti. Iako je ovo rješenje funkcionalno bolja opcija bi bila sposobnost aplikacije podržavanja upravljanja cijele mreže uređaja. Zamisli se skladište puno vozila, robota i raznih modela koji se upravljaju preko jedne centralne aplikacije, a aplikacija stvorena u Web2py sučelju koje je dostupno svima bez potrebe licence koje obično dolaze sa visokim troškovima nabave. Cijela ideja leži na temelju jednog centralnog izvora bežične mreže koji može biti Raspberry Pi ili ako potrebe zahtijevaju drugi oblik jačeg računala koje uspostavlja konekciju sa ostalim uređajima unutar zadanog prostora. Svi uređaju bi imali

uspostavljenu stalnu konekciju sa serverom preko kojeg bi primali podatke o upravljanju i slali podatke sa senzora ako su potrebni. Moguća je i aplikacija s bazom podataka u koju se spremaju podatci senzora te se tako može mapirati određeni radni ciklus uređaja sa svim njegovima koracima koje se može iskoristiti za daljnje povećanje učinkovitosti automatiziranog sustava. Ovaj završni zadatak bi svoj budući razvojni smjer mogao pronaći u obliku upravljanja s tipkovnicom te mobilnim uređajem. Također nedostatak sporosti Ajax metode bi mogao zamijenit brži oblik mreže temeljen na novoj *web socket* tehnologiji, ili implementacije mrežnog upravljanja u nove tehnologije *blockchain* mreže. Sustav upravljanja više modela temeljen na *blockchain* tehnologiji bi riješio problem ograničenosti sustava na stabilnost jednog servera, jer ovdje cijela *blockchain* tehnologija se zasniva na velikom broju servera sa kojih se komunikacija uvijek odvija. Izvela bi se i sposobnost automatskog prepoznavanja uređaja koja se ne bih bazirala samo na IP adresi klijenta već po njegovom privatnom identitetu koji se ranije dodjeli za sve uređaje kojima se omogućava pristup mreži.

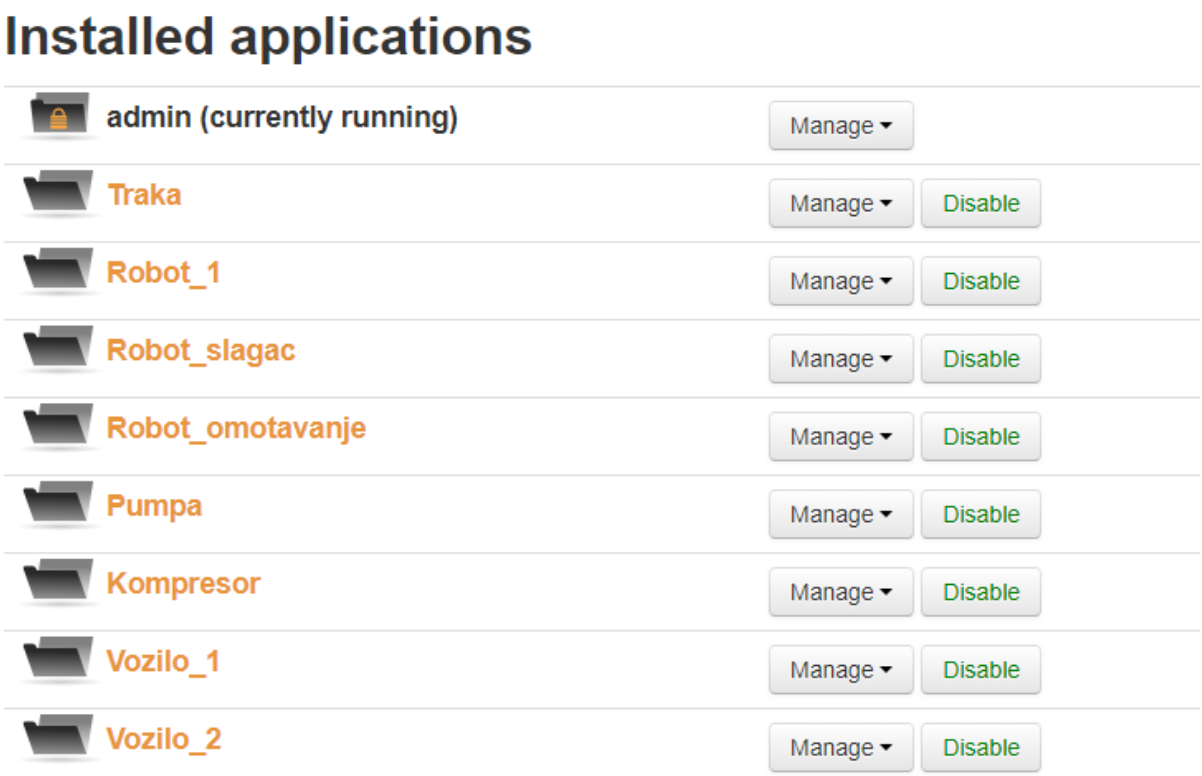


Slika 15. WebSocket i Blockchain [14,15]

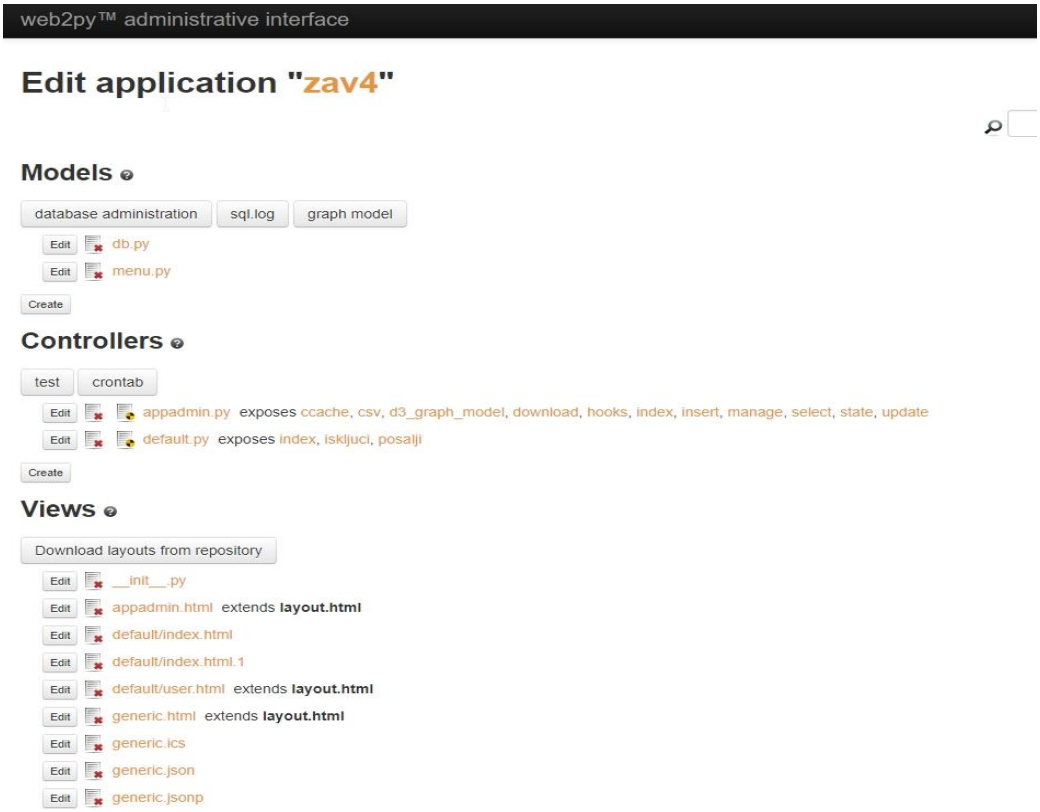
6.2. Prednosti mreže upravljanih objekata

Prednosti ovakvog sustava dolaze u tome što postoji jedna centralna jedinica lako promjenjiva i upravljiva. Sustav temeljen na Web2py tehnologiji je brz i praktičan, te svaka izmjena koda za upravljanje može brzo i jednostavno biti implementirana. Izmjena koda može biti vršena na samom serveru pa se tako i učitavanje vrši na licu mjesta, a Web2py omogućava da se izmjena vrši i na daljinu, iz druge sobe, grada ili države sve uz pristupnu adresu i lozinku. Uz pristupnu adresu i lozinku se pristupa administratorskom sučelju (prikazano Slikom 16.) na kojem se vrši uređivanja aplikacija, instaliranje novih aplikacija ili brisanje starih i neželjenih. Cijeli sustav upravljanja može biti baziran na jednoj aplikaciji a isto tako može biti baziran na više aplikacija potpuno neovisna jedna o drugoj, ali ovisne samo o stanju servera. Ako se jedna aplikacija odluči ugasiti sve ostale mogu nastaviti raditi bez obzira na stanje ugašene aplikacije. Također aplikacije se mogu povezati i napraviti ovisnima, ako jedna dođe u stanje greške i prekine s radom druga aplikacija tada može stopirati svoje radno stanje. Web2py

tehnologija se može povezati s drugim servisima pa tako možemo napraviti sustav u kojem korisnik laboratorija dobiva na email stanje sustava u pravom vremenu ili bude obaviješten o mogućim problemima s uređajima. Sve navedeno čini Web2py tehnologiju prikladnim odabirom za upravljanje i kontrolu mreže laboratorijskih objekata.



Slika 16. Web2py administratorsko sučelje sa izborom aplikacija



Slika 17. Web2py administratorsko sučelje i mogućnosti izmjene

6.3. Nedostatci mreže upravljanih objekata

Kao i svaki sustav tako i sustav sa Web2py tehnologijom upravljanja mreža ima svoje nedostatke. Web2py sustav se temelji na Python programibilnom jeziku s kojim se jezik pisanja aplikacija ograničavamo samo na Python i njegovu strogo sintaksu. U slučaju potrebe određenih programskih modula koji nisu po osnovi uključeni u Python jezik i sustav potrebno je naknadno samostalno implementirat pakete koji nedostaju. Instalacija potrebnih paketa ponekad može predstavljati problem zbog starih knjižnica podataka koje pripadaju različitim verzijama Python jezika i slično. Najčešća greška koja se pojavljuje korištenjem vanjskih paketa je ne kompatibilnost sa postojećom verzijom Python. Naime Web2py se temelji na Python 2.7 verziji što predstavlja stariju verziju jezika. Kako je dostupna novija verzija tako se i dodani moduli i paketi prilagođavaju novoj verziji Python-a pa se nailazi na poteškoće pronalaska paketa sa prikladnom verzijom Python-a. Jedna od većih mana je brzina Web2py aplikacije ako se napuni programibilnim kodom. Dugačak kod zajedno sa velikim grafičkim sučeljem može usporiti upotrebu aplikacije. Navedeni problem se može zaobići korištenjem većeg broja aplikacija, prilagođavanjem programskog koda na jednostavniji oblik ili upotreba servera sa jačim performansama.

ZAKLJUČAK

Kroz izradu ovog završnog rada dodatno sam se upoznao sa sučeljem i aplikacijom Web2py-a, izradom web aplikacija koristeći zajedno sve web jezike uz Python. U izradi završnog zadatka bilo je potrebno implementirati znanje naučeno kroz ranije predmete studija kao što su Web programiranje i Mikroprocersko upravljanje. Sustav upravljanja laboratorijskim objektom koristeći bežične mreže zahvaća probleme tri područja, izrada web aplikacija, internetski protokoli i programiranje kontrolera. Potrebno je svladati izradu aplikacije koja se ponaša kao obična web stranica, a u pozadini vrši komunikaciju sa drugim uređajima koristeći internetske protokole i na kraju izraditi programski kod kontrolera koji preko istog komunikacijskog protokola vrši dijalog sa aplikacijom odnosno serverom. Prilikom izrade nailazilo se na mnoge probleme kao što su brzina komunikacija ili implementacija raznih interaktivnih elemenata. Prvi problem koji su nastali bili su vezani uz brzinu komunikacije i upravljanja objekt koji je dobivao potrebne podatke s kašnjenje od par sekundi što je bila krivica lošeg izbora internetskog protokola. Sljedeći problemi su nastajali povezivanje interaktivnih elemenata sa već navedenim protokolom, što izgleda jednostavno ali izvući sve potrebno podatke i poslati ih u pravom smjeru je predstavljalo pravi izazov. Slobodno mogu reći da sam uživao u izradi ovog završnog rada i uz sve probleme doživio sam dio izazova koji se pojavljuje u praksi kod upravljanja cijelog jednog industrijskog postrojenja. Mogu reći da je izgrađen i temelj novog znanja koje definitivno planiram koristit u budućnosti.

LITERATURA

[1] https://cdn.shopify.com/s/files/1/0174/1800/products/Raspberry_Pi_3_1_of_4_711f1ffe-af5e-4923-aa7f-d80651396258_1024x1024.JPG?v=1456565788

[2] https://ae01.alicdn.com/kf/HTB1l_igRpXXXXXxXpXXq6xXFXXXX/WEMOS-D1-mini-V2-3-0-WIFI-Internet-of-Things-development-board-based-ESP8266-ESP-12S.jpg_640x640.jpg

[3] https://en.wikipedia.org/wiki/Raspberry_Pi

[4] https://en.wikipedia.org/wiki/Microcontroller

[5] http://www.technologystudent.com/images3/picax6.gif

[6] http://www.web2py.com/book/default/chapter/01

[7] https://frillip.com/using-your-raspberry-pi-3-as-a-wifi-access-point-with-hostapd/

[8] https://www.element14.com/community/servlet/JiveServlet/showImage/102-80899-15-252356/Pi3+Breakout+Feb+29+2016.png

[9] https://upload.wikimedia.org/wikipedia/commons/thumb/2/2e/Raspberry-pi-3-raspbian-100656556-orig.png/1200px-Raspberry-pi-3-raspbian-100656556-orig.png

[10] https://github.com/jeromeetienne/virtualjoystick.js

[11] https://www.cnx-software.com/wp-content/uploads/2017/01/Asus-Tinker-Board-vs-Raspberry-Pi-3.jpg

[12] http://forum.banana-pi.org/uploads/default/original/2X/1/1eb2282c4ae0a23c491acc1d000dff0083f9eb52.jpg

[13] http://www.web2py.com/init/default/download

[14] http://zbrad.github.io/images/wscore/html5websockets.png

[15] https://www.whitehatsec.com/wp-content/uploads/2016/12/Blockchain-Image-3.jpg

PRILOZI

I. Programski kod za postavljanje mreže *hostapd*

II. Programski kod aplikacije

III. Programski kod D1 mini kontrolera

IV. Shema D1 mini kontrolera

V. CD-R

PRILOG 1. Programski kod za postavljanje mreže *hostapd*

```
#Sucelje koje se definira za stalnu varijantu
interface=wlan0

# Koristenje nl80211 upravljackog uredaja WIFI modula
driver=nl80211

# Naziv bezicne mreze
ssid=Pi3-AP

# Vrsta signala
hw_mode=g

# Broj kanala 6
channel=6

# Ukljucivanje 802.11n standarda
ieee80211n=1

# Ukljucivanje WMM opcije, prioriteti podataka
wmm_enabled=1

# Ukljucivanje 40MHz kanala sa radom u 20ns
ht_capab=[HT40][SHORT-GI-20][DSSS_CCK-40]

# Dopustiti sve MAC adrese
macaddr_acl=0

# WPA sigurnost
auth_algs=1

# Klijent ne mora znati ssid kako bi se spojio
ignore_broadcast_ssid=0

# Dodatna WPA2 zastita
wpa=2

# Mogucnost spajanja sa unaprijed podijeljenim kljucem
wpa_key_mgmt=WPA-PSK

# lozinka mreze
wpa_passphrase=123456789

# Koristenje AES, umjesto TKIP enkriptivnog standarda
rsn_pairwise=CCMP
```

PRILOG 2. Programski kod aplikacije

Index.html sadrži:

```
<html>
	<head>
		<title>Upravljanje</title>
		<meta charset="utf-8">
		<meta name="viewport" content="width=device-width, user-scalable=no,
minimum-scale=1.0, maximum-scale=1.0">
		   <!-- include stylesheets -->
		<link rel="stylesheet" href="{{=URL('static','css/bootstrap.min.css')}}"/>
		<link rel="stylesheet" href="{{=URL('static','css/bootstrap-
slider.min.css')}}"/>

		<!-- java script -->
		<script src="{{=URL('static','js/jquery.js')}}"></script>
		<script src="{{=URL('static','js/bootstrap.min.js')}}"></script>
		<script src="{{=URL('static','js/bootstrap-slider.min.js')}}"></script>

		<style>
		body {
			overflow	: hidden;
			padding		: 0;
			margin		: 0;
			background-color: #F4F6F6;
		}
		#info {

	border-bottom: 5px solid;
	border-color: coral;
			position	: absolute;
			top		: 0px;
			width		: 100%;
			padding		: 5px;
			text-align	: center;
		}


		#container {

			width		: 100%;
			height		: 100%;
			overflow	: hidden;
			padding		: 0;
			margin		: 0;
			-webkit-user-select	: none;
			-moz-user-select	: none;
		}
		</style>
```

```
		<script>
		$(document).ready(function(){
	var mouseStillDown = false;

	$('#container').mousedown(function() {
	  do_something();
	}).mouseup(function() {
	  clearInterval(mouseStillDown);
	  mouseStillDown = false;
	});

	function do_something() {
	  $.ajax({
		url: "{{=URL('default' , 'posalji')}}",
			type: 'POST',
	  async:false,
			data: {"ip": $("#ip").val(), "m1":$("#y").text(),
"m2":$("#y").text(), "dir":$("#x").text() },
			dataType: 'json',
				});

	  if (!mouseStillDown) {
	    mouseStillDown = setInterval(do_something, 50);
	  }

	}
	$("div").mouseup(function(){
	  $.ajax({
					url: "{{=URL('default' , 'iskljuci')}}",
					type: 'POST',
					data: {"ip": $("#ip").val()},
					dataType: 'json',
				});
	});
					// iskljucivanje svih motora
			$("#iskljuci").click(function(){
				$.ajax({
					url: "{{=URL('default' , 'iskljuci')}}",
					type: 'POST',
					data: {"ip": $("#ip").val()},
					dataType: 'json',
				});
			});
		});
		</script>
	</head>
```

```
        <body>
    <div id="container"></div>
            <div id="info"><h1>Upravljanje autom</h1>
                    <hr>
                    <!-- IP adresa -->
                    <div class="form-group">
                            <label for="ip">Odaberi auto:</label>
                            <select class="form-control" name="ip"
id="ip">
                                    <option
value="192.168.137.209">Auto 1</option>
                                    <option
value="192.168.172.102">Auto 2</option>
                                    <option
value="192.168.172.103">Auto 3</option>
                                    </select>
                    </div>

        <!-- iskljuci motore -->
        <div class="form-group">
                        <br>
                        <button type="button" id="iskljuci" value="iskljuci"
class="btn btn-danger">Isključi motore</button>
        </div>

        <span>X koordinata  </span><span id="x"></span><span>
</span><span>Y koordinata  </span><span id="y"></span>
        <br>
         <span>Polje ispod linije je interaktivni upravljač</span>
    </div>

            <script src="{{=URL('static','js/virtualjoystick.js')}}" charset="utf-
8"></script>
            <script>
                    console.log("touchscreen is",
VirtualJoystick.touchScreenAvailable() ? "available" : "not available");

                    var joystick     = new VirtualJoystick({
                            container        :
document.getElementById('container'),
                            mouseSupport: true,
                    limitStickTravel: true,
                    stickRadius: 200

                    });
```

```
                joystick.addEventListener('touchStart', function(){
                        console.log('down')
                })
                joystick.addEventListener('touchEnd', function(){
                        console.log('up')
                })

                setInterval(function(){
                        var outputEl   = document.getElementById('x');
                        var outputE2   = document.getElementById('y');
                        outputEl.innerHTML = joystick.deltaX()
                        outputE2.innerHTML= joystick.deltaY()
                }, 1);
        </script>
    </body>
</html>
```

Default.py unutar *Controller*-a sadrži:

```
# -*- coding: utf-8 -*-
# this file is released under public domain and you can use without
limitations
import os
import socket
import math

from scipy.interpolate import interp1d

global s
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)


def index():
    return dict()



# ajax funkcija
def iskljuci():
    global s
    s.connect((request.vars.ip, 90))
    s.sendall("0,0,1,0")
    s.close()
def posalji():
    global s
    ip = request.vars.ip
    print "*"*20
#     print request.vars

    if ip.endswith("1"):
        # upravljacki kog za prvi autic
        m = interp1d([0,100],[500,1023])

        m1 = int(request.vars.m1)

        if m1<0:
            dir1=1
            m1=m(abs(m1))

        elif m1>0:
            dir1=0
            m1=m(abs(m1))

        else:
            dir1=0
            m1=0
```

```
        m2 = int(request.vars.m2)

        if m2<0:
            dir2 = 0
            m2 = m(abs(m2))

        elif m2>0:
            dir2=1
            m2 = m(abs(m2))

        else:
            dir2=0
            m2=0


    else:
        # upravljacki kod za drugi autic
        p1 = int(request.vars.m1)
        p2 = int(request.vars.dir)
        #dir1 = dir2 = request.vars.dir

        # interpolacijska funkcija
        m = interp1d([0,200],[500,1023])

        if p1<0:
            dir1=dir2=1
            if p2>20:
                m2=math.sqrt((p1*p1)+(p2*p2))
                m1=abs(200-p2)
                m1 = m(m1)
                m2 = m(m2)
            elif p2<-20:
                m1=math.sqrt((p1*p1)+(p2*p2))
                m2=abs(200+p2)
                m1 = m(m1)
                m2 = m(m2)
            else:
                m1=m2=abs(p1)
                m1 = m(m1)
                m2 = m(m2)

        elif p1>0:
            dir1=dir2=0
            if p2>0:
                m2=math.sqrt((p1*p1)+(p2*p2))
                m1=abs(200-p2)
                m1 = m(m1)
                m2 = m(m2)
```

```
            elif p2<0:
                m1=math.sqrt((p1*p1)+(p2*p2))
                m2=abs(200+p2)
                m1 = m(m1)
                m2 = m(m2)
             else:
                m1=m2=p1
                m1 = m(m1)
                m2 = m(m2)
          else:
             m1=m2=0
             m1 = m(m1)
             m2 = m(m2)



    msg = "{},{},{},{}\n".format(int(m1), int(m2), dir1, dir2)
    s.connect((ip, 90))
    s.sendall(msg)
    s.close()
#     print "poslano"

    return
```

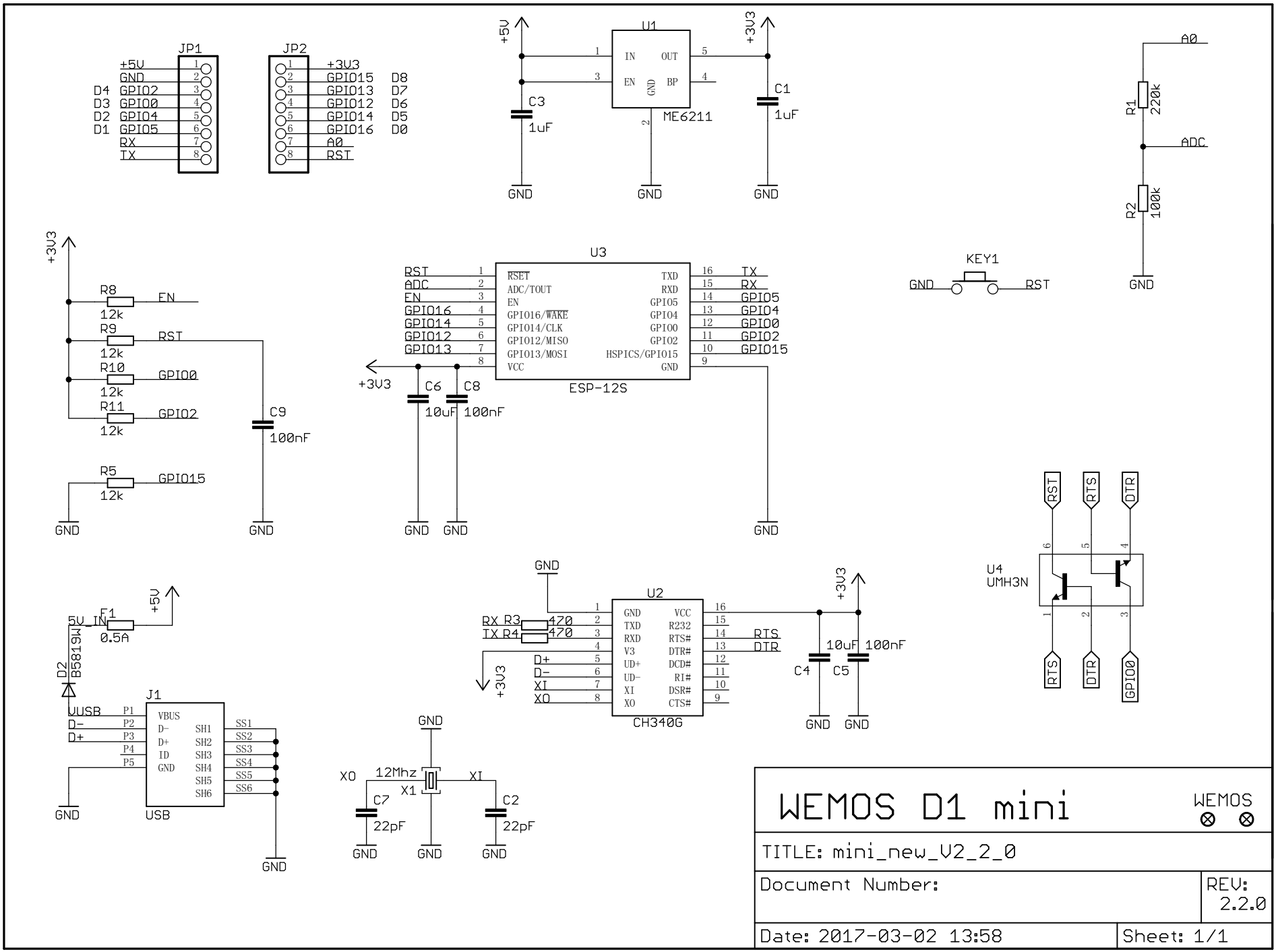
PRILOG 3. Programski kod D1 mini kontrolera

Autic.ino sadrži:

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <WiFiServer.h>

#include "defines.h"
#include "init.h"


WiFiClient client;
WiFiServer server(PORT);


String inString;
int i = 0;

const char *ssid="Pi3-AP";
const char *pass="123456789";


//IPAddress ip(192, 168, 1, 101);
//IPAddress gateway(192, 168, 1, 101);
//IPAddress subnet(255, 255, 255, 0);

void setup() {

  init_pins();

  Serial.begin(115200);
  Serial.println("Port: ");
  Serial.println(PORT);

  // konfiguracija wifi-a
//  WiFi.config(ip, gateway, subnet);

  WiFi.begin(ssid,pass);
  while(WiFi.status()!=WL_CONNECTED){
    Serial.print(".");
    delay(500);
  }
```

```
  Serial.println("Spojeno na : ");
  Serial.print(ssid);
  Serial.println("IP : ");
  Serial.println(WiFi.localIP());
  delay(200);
  server.begin();
}

void loop() {
  while(!client.connected()){
    client=server.available();
  }

  while(client.connected()){

    if (client.available()){

      int inChar = client.read();
      Serial.print(inChar);
      // ako je stigla brojka
      if (isDigit(inChar)) {
        inString += (char)inChar;
      }

      // ako je stiago zarez
      else if (inChar == ','){
        if (i==0){
          analogWrite(M1,inString.toInt());
        }
        else if (i==1){
          analogWrite(M2,inString.toInt());
        }
        else if (i==2){
          digitalWrite(DIR1,inString.toInt());
        }
        i++;
        inString = "";
      }

      // novi red
      else if (inChar == '\n') {
        digitalWrite(DIR2,inString.toInt());
        i=0;
        inString = "";
      }
    }
  }
}
```

PRILOG 4. Shema D1 mini kontrolera


JP1
+5V
GND
D4 GPIO2
D3 GPIO0
D2 GPIO4
D1 GPIO5
RX
TX
JP2
+3V3
GPIO15 D8
GPIO13 D7
GPIO12 D6
GPIO14 D5
GPIO16 D0
A0
RST
U1
IN
OUT
EN
GND
BP
ME6211
C3
1uF
C1
1uF
A0
R1
220k
ADC
R2
100k
KEY1
RST
U3
RSET
ADC/TOUT
EN
GPIO16/WAKE
GPIO14/CLK
GPIO12/MISO
GPIO13/MOSI
VCC
TXD
RXD
GPIO5
GPIO4
GPIO0
GPIO2
HSPICS/GPIO15
GND
ESP-12S
R8
12k
EN
R9
12k
RST
R10
12k
GPIO0
R11
12k
GPIO2
R5
12k
GPIO15
C9
100nF
C6
10uF
C8
100nF
U4
UMH3N
F1
0.5A
5V_IN
D2
B5819W
J1
VUSB
D-
D+
ID
VBUS
SH1
SH2
SH3
SH4
SH5
SH6
SS1
SS2
SS3
SS4
SS5
SS6
USB
U2
RX R3
TX R4
470
470
D+
D-
XI
XO
GND
TXD
RXD
V3
UD+
UD-
XI
XO
VCC
R232
RTS#
DTR#
DCD#
RI#
DSR#
CTS#
RTS
DTR
CH340G
C4
C5
10uF
100nF
12Mhz
X1
C7
22pF
C2
22pF
RST
RTS
DTR
GPIO0
WEMOS D1 mini
WEMOS
TITLE: mini_new_V2_2_0
Document Number:
REV: 2.2.0
Date: 2017-03-02 13:58
Sheet: 1/1