

SVEUČILIŠTE U SPLITU
FAKULTET ELEKTROTEHNIKE, STROJARSTVA I
BRODOGRADNJE

ZAVRŠNI RAD

**PAMETNA HRANILICA ZA KUĆNE
LJUBIMCE**

Ante Doko

Split, rujan 2021.



Preddiplomski sveučilišni studij: **Elektrotehnika i informacijska tehnologija**

Smjer/Usmjerenje: **Elektronika i računalno inženjerstvo**

Oznaka programa: 112

Akademска godina: 2020./2021.

Ime i prezime: **ANTE DOKO**

Broj indeksa: 89-2018

ZADATAK ZAVRŠNOG RADA

Naslov: **Pametna hranilica za kućne ljubimce**

Zadatak: Proučiti problematiku pametnih uređaja namijenjenih za kućne ljubimce i pomoći oko kućnih ljubimaca, s posebnim osvrtom na hranilice. Zatim osmislati i konstruirati (mehanički, elektronički i programske) hranilicu koja će moći raditi na dva osnovna načina: automatski (u određenim vremenskim intervalima dozirati određenu količinu hrane) i ručno (korisnik se spoji s uređajem s udaljene lokacije i ručno dozira količinu hrane za ljubimca). U sustav ugraditi i kameru koja korisniku omogućava nadzor nad hranilicom i kućnim ljubimcem s udaljene lokacije. Testirati razvijenu sustav i prezentirati ostvarene rezultate. Predložiti moguća poboljšanja i daljnje pravce razvoja.

Datum obrane: 17.09.2021.

Mentor:

izv. prof. dr. sc. Josip Musić

IZJAVA

Ovom izjavom potvrđujem da sam završni rad s naslovom **PAMETNA HRANILICA ZA KUĆNE LJUBIMCE** pod mentorstvom izv. prof. dr. sc. Josipa Musića pisao samostalno, primjenivši znanja i vještine stečene tijekom studiranja na Fakultetu elektrotehnike, strojarstva i brodogradnje, kao i metodologiju znanstveno-istraživačkog rada, te uz korištenje literature koja je navedena u radu. Spoznaje, stavove, zaključke, teorije i zakonitosti drugih autora koje sam izravno ili parafrazirajući naveo u završnom radu citirao sam i povezao s korištenim bibliografskim jedinicama.

Student



Ante Doko

Sadržaj

1.	UVOD	1
1.1	Pametni uređaji namijenjeni za kućne ljubimce	1
2.	RASPBERRY Pi	4
2.1	Raspberry Pi 4 Model B	6
2.2	GPIO pinovi.....	7
2.3	Raspberry Pi Operating System.....	8
2.3.1	Instalacija Raspberry Pi OS-a	10
3.	APACHE HTTP WEB POSLUŽITELJ.....	11
3.1	Postavljanje Apache web poslužitelja	12
4.	RAZVOJ WEB APLIKACIJE	13
4.1	Stvaranje virtualnog okruženja.....	13
4.2	Flask.....	15
4.2.1	Kreiranje Flask aplikacije.....	15
4.3	HTML i CSS.....	17
4.3.1	HTML.....	17
4.3.2	CSS.....	18
4.4	HTTP metode GET i POST.....	19
4.5	WSGI.....	19
4.5.1	Konfiguracija Apache-a	20
4.6	PORT FORWARDING	21
4.6.1	DDNS – DYNAMIC DNS	23
4.7	Prikaz sučelja web aplikacije.....	25
5.	SERVO MOTOR	28
5.1	Princip rada servo motora	28
5.2	TowerPro MG995 servo motor s kontinuiranom rotacijom	29
5.3	Povezivanje servo motora s Raspberry Pi-em	30

5.4	Programski kôd za upravljanje servo motorom	31
6.	RASPBERRY Pi KAMERA MODUL	33
6.1	Spajanje kamere na Raspberry Pi	33
6.2	Instaliranje <i>Motion</i> programa	34
7.	KÔD ZA AUTOMATSKO HRANJENJE.....	36
8.	IZRADA SKLOPOVSKOG DIJELA HRANILICE	37
9.	ZAKLJUČAK	42
9.1	Budućnost	42
	LITERATURA.....	44
	POPIS OZNAKA I KRATICA	47
	SAŽETAK.....	49
	ABSTRACT	50
	PRILOZI.....	51
	PRILOG A.....	51
	PRILOG B.....	52
	PRILOG C.....	53
	PRILOG D.....	55
	PRILOG E	56
	PRILOG F	57
	PRILOG G	60
	PRILOG H.....	62

1. UVOD

Zadatak završnog rada je realizacija pametne hranilice za kućne ljubimce koja ima dva načina rada: automatsko hranjenje i hranjenje s udaljene lokacije putem web aplikacije. Automatsko hranjenje podrazumijeva doziranje određene količine hrane u zadano vrijeme. Hranjenje s udaljene lokacije moguće je jednostavnim pritiskom na tipku „Nahrani sad“ na web stranici aplikacije čija je detaljna izrada opisana u radu. Osim mogućnosti hranjenja vlasnik putem web aplikacije ima uvid o kućnom ljubimcu te stanju hranilice.

U uvodu rada spomenuti su pametni uređaji za kućne ljubimce s posebnim osvrtom na pametne hranilice. U drugom poglavlju detaljnije je opisano računalo Raspberry Pi, koje je „srce“ hranilice. Navedene su neke osnovne karakteristike samog računala, instalacija OS-a i postavljanje osnovnih postavki. U trećem poglavlju, korak po korak opisan je postupak postavljanja Apache HTTP web poslužitelja. U četvrtom poglavlju objašnjen je postupak izrade web aplikacije te osnovni alati korišteni pri realizaciji iste. U petom i šestom poglavlju detaljnije su predstavljene karakteristike, način spajanja s Raspberry-em i programiranje servo motora i modula kamere. U sedmom poglavlju prikazan je kôd za automatsko hranjenje. U osmom poglavlju, opisana je i slikovno prikazana izrada sklopovskog dijela hranilice. Na kraju su navedeni nedostatci razvijenog sustava te su predložena moguća poboljšanja.

1.1 Pametni uređaji namijenjeni za kućne ljubimce

U posljednjih par desetljeća došlo je do brzog napretka tehnologije. Znanstvenici i ljudi iz svijeta tehnologije nastoje različitim inovacijama što više olakšati čovjekovu svakodnevnicu. Napretkom tehnologije došlo je do pojave pametnih uređaja tzv. „Internet stvari“ (IoT). Ti uređaji opisuju tehnologiju povezivanja fizičkih uređaja, vozila i drugih stvari koje prikupljaju, dijele i razmjenjuju podatke putem interneta. Spajanje uređaja može biti bežično ili žično, a omogućuje interakciju između ljudi i različitih sustava. Neki primjeri ovakvih uređaja su pametni hladnjaci, perilice, tepisi, itd. Hladnjak će korisniku putem aplikacije dojaviti kojeg proizvoda je nestalo, perilica će započeti pranje da bude gotovo kada se korisnik vrati doma, tepih će, ako vlasnik nije u stanu, dojaviti kretanje itd. Osim olakšavanja vlastite svakodnevnice čovjek putem IoT tehnologije nastoji olakšati život svojih najboljih prijatelja, kućnih ljubimaca. Danas u trgovinama i na internetu mogu se pronaći

različiti pametni uređaji namijenjeni životinjama. Neki od najzanimljivijih i jako korisnih uređaja su npr. pametni sanduk koji je osmišljen kako bi se ljubimac osjećao sigurno i smireno. Sanduk ima ugrađeno prigušivanje vibracija i zvuka za smanjenje tjeskobe tijekom oluja i vatrometa, ventilaciju te popis glazbe za reprodukciju za smanjenje tjeskobe koji se aktivira pokretom. Pametni WC za kućne ljubimce je IoT uređaj kod kojeg se motorizirane grablje aktiviraju 20 minuta nakon što ljubimac iziđe iz kutije skupljajući i odlažući otpad u zasebnu posudu kako bi se izbjegao neugodan miris. Pametna vrata koriste radio-frekveničku tehnologiju. Ljubimac na ogrlici nosi SmartKey koji vrata čitaju kada im se ljubimac približi te se otvaraju na baterijski pogon.

Jedan od najkorisnijih pametnih uređaja namijenjenih kućnim ljubimcima je hranilica. Hranilice za kućne ljubimce dizajnirane su tako da omogućuju elektroničko upravljanje vremenom hranjenja ljubimca. Pružaju olakšanje kućnim ljubimcima koji su ostavljeni sami kod kuće, kao i njihovim vlasnicima. Pametne hranilice održavaju ritam hranjenja ljubimaca. Upravo zato što su psi i mačke najčešći izbor za kućnog ljubimca, većina hranilica je prilagođena njima. Postoje modeli koji služe isključivo doziranju hrane, ali i modeli koji osim hranjenja imaju podržan i video nadzor ili pak video komunikaciju s ljubimcem. Moguće je ručno hranjenje, hranjenje zadano u određenim vremenskim intervalima, i najvažnije, hranjenje s udaljene lokacije. Kvaliteta hranilice ovisi o kvaliteti komponenti korištenih za izradu iste. Cijena je očekivano veća kod hranilica koje imaju više mogućnosti. Prilikom odabira hranilice važno je uzeti u obzir veličinu kućnog ljubimca, koliko kućnih ljubimaca je potrebno hraniti, koju vrstu hrane koristi, cijenu pametne hranilice, jednostavnost korištenja itd. [1].

Na *slici 1.1.* je prikazana pametna hranilica za pse koja ima mogućnost namještanja vremena hranjenja, mogućnost korištenja aplikacije pomoću koje je moguće nahraniti ljubimca s udaljene lokacije, video kameru pomoću koje vlasnik ima nadzor nad kućnim ljubimce, mogućnost komunikacije s ljubimcem te mogućnost kontroliranja količine hrane koja se dozira [2]. Na *slici 1.2.* je prikazana hranilica za ribe koja se pričvrsti za akvarij [3]. Ona za razliku od prikazane hranilice za pse ili mačke nema mogućnost video nadzora, niti mogućnost hranjenja s udaljene lokacije. Na *slici 1.3.* prikazan je pametni uređaj za hranjenje konja [4]. Ova pametna hranilica automatski dozira hranu konjima u točno određeno vrijeme.



Slika 1.1 Pametna hranilica za pse



Slika 1.2 Pametna hranilica za ribe



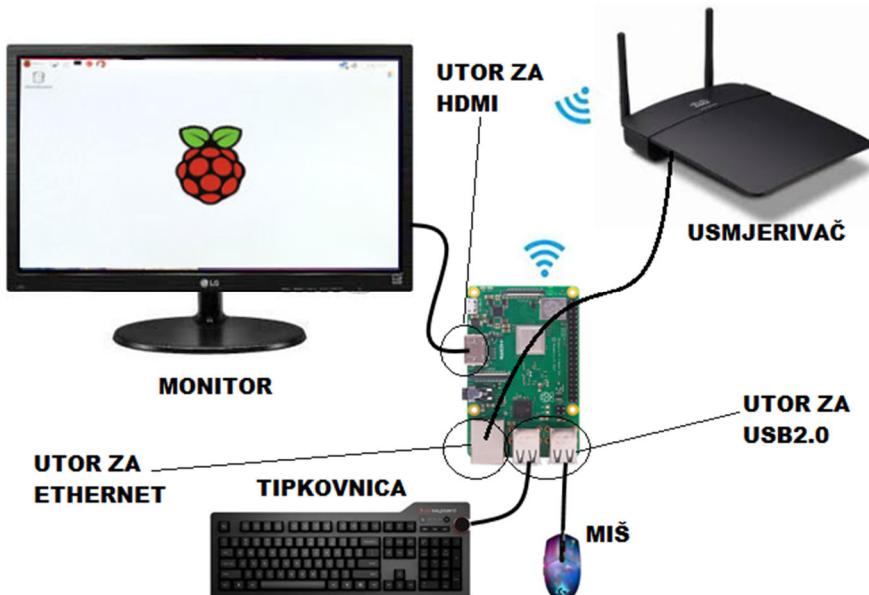
Slika 1.3 Pametna hranilica za konje

Uspoređujući ove tri slike, primjećuje se razlika u veličini hranilice, odnosno veličina hranilice je prilagođena životinji kojoj je namijenjena. Osim veličine uočava se razlika u kvaliteti izrade. Hranilica za konje je izrađena od čvršćeg materijala nego druge dvije hranilice jer bi konj plastičnu hranilicu slomio.

2. RASPBERRY Pi

Raspberry Pi je računalo nešto veće od kreditne kartice u potpunosti smješteno na jednoj matičnoj ploči. Namijenjeno je da bude jeftino i lagano za upotrebu ljudima svih dobnih skupina. Razlog zašto je cijena Raspberry Pi uređaja niska je taj što neke komponente nisu na ploči tj. potrebno ih je dodatno nabaviti. Na primjer Raspberry Pi ne dolazi sa kućištem koje bi ga štitilo već dolazi kao gola ploča. Ne dolazi niti s izvorom napajanja pa je i njega potrebno dodatno nabaviti. Omogućuje istraživanje računalstva i učenje programiranja na jezicima Pythona i Scratcha. S njim je moguće obaviti sve one funkcije koje se mogu obaviti sa standardnim stolnim računalom, od reprodukcije videa visoke kvalitete, pregledavanja interneta, izrade proračunskih tablica, do obrade teksta i igranja igrica. Jednostavno se povezuje sa standardnom tipkovnicom i mišom te putem HDMI-a (engl. *High-Definition Multimedia Interface*) sa uređajima za prikaz slike na način prikazan na *slici 2.1*. Moguće ga je povezati na Internet putem WiFi- a (engl. *Wireless Fidelity*) ili putem Etherneta.

Razvila ga je britanska tvrtka *Raspberry Pi Foundation* u suradnji s tvrtkom *Broadcom* 2012. godine. Projekt je izvorno težio promicanju poučavanja osnovnih računalnih znanosti u školama i zemljama u razvoju. Izvorni model je postao popularniji nego što se očekivalo, prodajući se izvan ciljanog tržišta za namjene poput robotike. Danas se koristi u mnogim područjima, primjerice za praćenje vremena, za razvoj IoT uređaja itd. Široku primjenu je postigao upravo zbog niske cijene, otvorenog dizajna i modularnosti [5].



Slika 2.1 Povezivanje Raspberry Pi-a s tipkovnicom, mišem, monitorom i usmjerivačem

U tablici 2.1 navedeno je nekoliko modela Raspberry Pi-a te su navedene neke od osnovnih karakteristika za svaki od spomenutih modela.

Tablica 2.1 Osnovne karakteristike Raspberry Pi modela

Model Raspberry Pi-a	Vrste napajanja	Preporučeni iznos napajanja	Video kvaliteta	Ethernet	Bluetooth	WiFi	Vanjska memorija
Raspberry Pi 1 Model A	Micro USB	5.1V, 700mA	1080p30	-	-	-	SD/MMC
Raspberry Pi 1 Model B+	Micro USB	5.1V, 1.2A	1080p30	10/100 Mbit/s	-	-	Micro SD
Raspberry Pi 2 Model B	Micro USB	5.1V, 1.8A	1080p30	10/100 Mbit/s	-	-	Micro SD
Raspberry Pi Zero Wireless	Micro USB	5.1V, 1.2A	1080p30	-	Bluetooth 4.1	2.4GHz	Micro SD
Raspberry Pi 3 Model B+	Micro USB	5.1V, 2.5A	1080p60	10/100 Mbit/s	Bluetooth 4.2/BLE	2.4GHz, 5 GHz	Micro SD
Raspberry Pi 4 Model B	USB-C	5.1V, 3A	4kp60	Gigabit Ethernet	Bluetooth 5.0	2.4GHz, 5 GHz	Micro SD

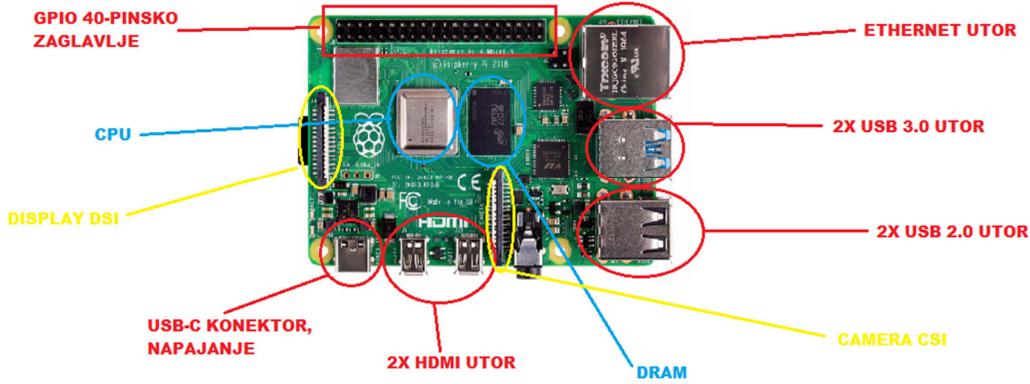
Prilikom realizacije projekta *Pametna hranilica za kućne ljubimce* korišten je Raspberry Pi 4 Model B koji je detaljnije opisan u narednom poglavlju.

2.1 Raspberry Pi 4 Model B

Raspberry Pi 4 Model B najnoviji je proizvod u Raspberry Pi assortimanu računala. Nudi povećanje brzine procesora, multimedijskih performansi, povezivanja i memorije u usporedbi s prethodnom generacijom Raspberry Pi 3 modela B+, zadržavajući sličnu potrošnju energije i kompatibilnost uz nešto veću cijenu. Za krajnjeg korisnika, Raspberry Pi 4 Model B pruža performanse radne površine usporedive s početnim x86 računalnim sustavima. Ključne značajke ovog proizvoda uključuju 64-bitni četverojezgreni procesor visokih performansi, dvopojasni 2.4/5.0 GHz bežični LAN, Bluetooth 5.0, Gigabitni Ethernet, USB (engl. *Universal Serial Bus*) 3.0, podršku za dva zaslona pri rezolucijama do 4K putem para mikro-HDMI portova i PoE (engl. *Power over Ethernet*) [6]. Osnovne karakteristike prikazane su u *tablici 2.2*, a dijelovi pločice Raspberry Pi 4 Model B računala prikazani su na *slici 2.2*.

Tablica 2.2 Osnovne karakteristike Raspberry Pi 4 Modela B[6]

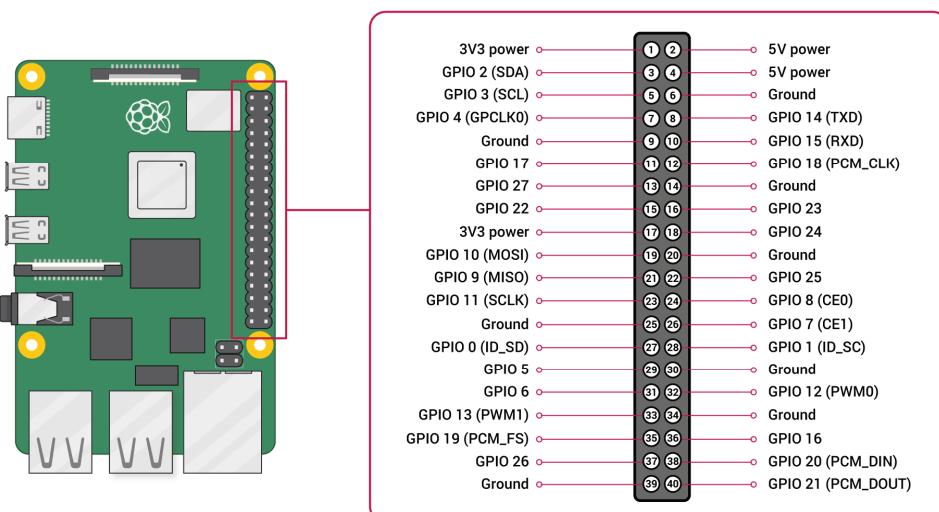
Procesor	Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
Memorija	1GB, 2GB ili 4GB LPDDR4 (ovisno o modelu)
Povezivanje	2.4 GHz i 5.0 GHz IEEE 802.11b/g/n/ac bežični LAN, Bluetooth 5.0/ BLE, Gigabit Ethernet, 2×USB 3.0 utor, 2×USB 2.0 utor
GPIO	Standardno 40-pinsko GPIO zaglavlje (potpuno kompatibilan sa prethodnim pločama)
Video i zvuk	2 × micro HDMI utora (podržava 4Kp60) MIPI DSI display port, MIPI CSI camera port, četveropolni stereo audio i composite video port
SD Card podrška	Utor za mikro SD karticu za učitavanje operacijskog sustava i pohranu podataka
Napajanje	5V DC preko USB-C konektora (minimalno3A1) 5V DC putem GPIO zaglavlja (minimalno3A1) Omogućeno napajanje preko Etherneta (PoE)



Slika 2.2 Dijelovi Raspberry Pi 4 Model B pločice

2.2 GPIO pinovi

Raspberry Pi 4 Model B čine 28 BCM2711 GPIO-a (engl. *General-Purpose Input/Output*) koji su dostupni putem 40-pinskog zaglavlja. Zaglavljje ovog modela kompatibilno je sa svim prethodnim Raspberry Pi pločicama koje imaju 40-pinsko zaglavljje. Pinovi predstavljaju fizičko sučelje između računala i vanjskog svijeta. Na najjednostavnijoj razini, mogu se zamisliti kao prekidači koje korisnik može uključiti ili isključiti (ulaz) ili koje Pi može uključiti ili isključiti (izlaz). Pi može kontrolirati LED diode, uključivati ih ili isključivati, pokretati motore i mnoge druge stvari. Također može otkriti je li prekidač pritisnut i detektirati iznos temperature. Ovo se naziva fizičkim računarstvom [7]. GPIO pinovi Raspberry Pi 4 Model B računala prikazani su na slici 2.3.



Slika 2.3 GPIO pinovi [8]

Prilikom realizacije rada *Pametna hranilica za kućne ljubimce* na GPIO pinove Raspberry Pi-a spojen je servo motor. Na pin broj 2, odnosno 5V napajanje, spojena je plus žica servo motora. Na *ground* pin je spojena minus žica servo motora, a kao kontrolni pin korišten je GPIO 17 tj. pin broj 11 na koji je spojena upravljačka žica servo motora. Primjer jednostavnog Python kôda kojim se putem Raspberry Pi-a osovina servo motora okreće za 180 stupnjeva prikazan je na *slici 2.4. Raspberry-gpio-python* ili *RPi.GPIO* je Python modul za kontrolu GPIO sučelja na Raspberry Pi-u. Modul *RPi.GPIO* je prema zadanim postavkama instaliran na novije verzije Raspbian Linuxa. Da bi se modul mogao koristiti u Python programu potrebno ga je uvesti naredbom *import RPi.GPIO as GPIO*. Naredba *GPIO.setmode(GPIO.BOARD)* koristi se kako bi se omogućilo korištenje brojeva pinova; npr. umjesto GPIO17 koristi se pin broj 11. Po završetku korištenja GPIO knjižice dobra je praksa vratiti korištene pinove na zadanu vrijednost što se čini naredbom *GPIO.cleanup()*.

```
import RPi.GPIO as GPIO #uvoz potrebnog modula

GPIO.setmode(GPIO.BOARD) #omogućuje korištenje brojeva pinova

GPIO.setup(11,GPIO.OUT) #pin 11 postavljamo kao izlazni

servo=GPIO.PWM(11,50) #na pin 11 šaljemo PWM signal frekvencije 50 Hz
servo.start(0) #uključuje servo motor, nema pulsa
servo.ChangeDutyCycle(12) #rotacija za 180 stupnjeva
servo.stop() #isključuje se servo motor

GPIO.cleanup() #vraćanje na zadanu vrijednost
```

Slika 2.4 Python kôd kojim se osovina servo motora okreće za 180 stupnjeva

2.3 Raspberry Pi Operating System

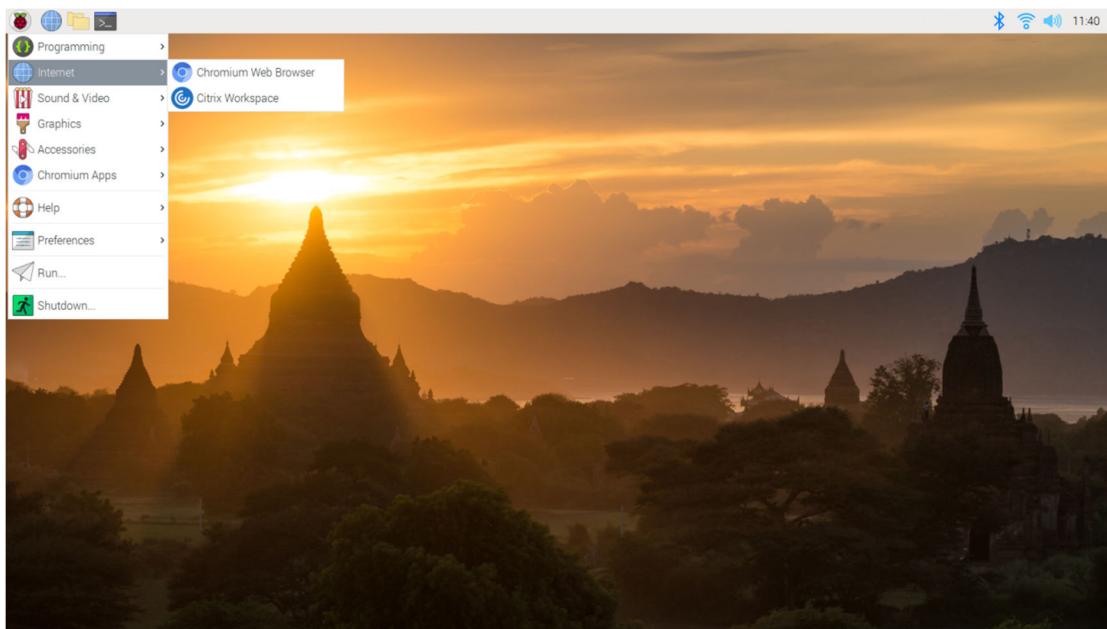
Raspberry Pi Operating System (ranije Raspbian OS) je operacijski sustav zasnovan na Debianu za Raspberry Pi. Radi na svim Raspberry Pi izuzev Pico mikrokontrolera. Raspberry Pi OS koristi izmijenjeni LXDE (engl. *Lightweight X11 Desktop Environment*) kao desktop okruženje. Isporučuje se s kopijom programa algebre Wolfram Mathematica i verzijom Minecrafta pod nazivom *Minecraft: Pi Edition*, kao i laganom verzijom web preglednika Chromium. Na *slici 2.5* može se primijetiti da Raspberry Pi OS izgleda slično mnogim uobičajenim stolnim računalima, kao što su macOS, Microsoft Windows, a najsličniji je LXDE-u. Traka izbornika nalazi se pri vrhu i sadrži izbornik aplikacija i prečace

do Terminala, Chroma i Upravitelja datoteka. S desne strane nalazi se Bluetooth izbornik, Wi-Fi izbornik, kontrola glasnoće i digitalni sat što je također vidljivo na *slici 2.5*.

Moguće je instalirati Raspberry Pi OS kao:

- Raspberry Pi OS Lite,
- Raspberry Pi OS,
- Raspberry Pi OS Full.

Raspberry Pi OS Lite je najmanja verzija i ne uključuje desktop okruženje. Raspberry Pi OS uključuje Pixel Desktop Environment, a Raspberry Pi OS Full dolazi unaprijed instaliran s dodatnim softverom za produktivnost [9].



Slika 2.5 Izgled radne površine Raspberry Pi OS-a

Osim Raspberry Pi OS-a na Raspberry Pi moguće je instalirati i operacijske sustave kao što su Ubuntu Core, OpenELEC, OSMC, Arch Linux ARM i mnoge druge. Ubuntu core dizajniran je za IoT aplikacije, OSMC se koristi ako Raspberry Pi služi za upravljanje medijskim sadržajem itd. [10]. Prilikom realizacije projekta *Pametna hranilica za kućne ljubimce* korišten je Raspberry Pi Os Full operacijski sustav, a njegova instalacija na Raspberry Pi je objašnjena u narednom poglavlju.

2.3.1 Instalacija Raspberry Pi OS-a

Da bi se započela instalacija operacijskog sustava na računalo Raspberry Pi potrebne su sljedeće komponente: monitor računala ili TV, mikro HDMI kabel, tipkovnica, miš, napajanje, USB-C kabel i memorijska micro SD kartica (preporučen kapacitet ne manji od 8 GB). Micro SD kartica jedna je od najvažnijih komponenti jer su na njoj pohranjeni svi podatci. Ono što je lokalni disk kod laptopa ili stolnog računala to je micro SD kartica kod Raspberry Pi-a. Na službenoj stranici moguće je na Windows, Mac OS, Linux i Chrome OS preuzeti *Raspberry Pi Imager* aplikaciju pomoću koje je na brz i jednostavan način moguće instalirati OS na SD karticu. SD kartica se umetne u računalo te se pokrene aplikacija *Raspberry Pi Imager* čime se otvara prozor prikazan na *slici 2.6*. Unutar polja *Choose OS* odabere se Raspberry Pi OS, a unutar polja *Choose storage* odabere se umetnuta SD kartica. Klikom na *write* pokreće se instalacija sustava na SD karticu. Nakon instalacije OS-a na Raspberry Pi te povezivanjem prethodno navedenih komponenti moguće je započeti postavljanje Raspberry Pi-a.



Slika 2.6 Raspberry Pi Imager [11]

3. APACHE HTTP WEB POSLUŽITELJ

Jednostavnim rječnikom rečeno web poslužitelj je poput konobara u nekom restoranu. Kada gost stigne u restoran konobar ga pozdravlja, zatim provjeri njegove podatke o rezervaciji i vodi gosta do njegova stola. Na sličan način funkcionira web poslužitelj. On provjerava ima li web stranicu koju korisnik zatraži i dohvaća je kako bi korisnik pregledavao stranicu. Obrađuje komunikaciju s web stranicom, obrađuje zahtjeve te se brine jesu li su svi moduli spremni poslužiti korisnika. Osim toga web poslužitelj obavlja i funkciju čišćenja memorije i predmemorije kako bi bio dostupan novim korisnicima. Zaključno web poslužitelj je program koji prima zahtjev korisnika za pristup web stranici, pokreće nekoliko sigurnosnih provjera korisnikova HTTP¹ (engl. *Hypertext Transfer Protocol*) zahtjeva i vodi korisnika na web stranicu [12].

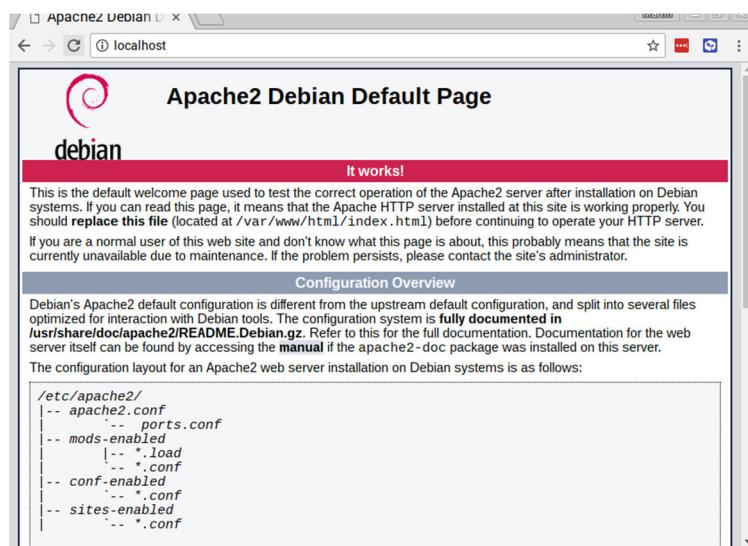
Apache HTTP web poslužitelj danas je daleko najpopularniji web poslužitelj na Internetu. Razvila ga je tvrtka *Apache Software Foundation*. Besplatan je, brz, pouzdan i siguran. Apache HTTP web poslužitelji može raditi na Linux, Unix i Windows platformi. Apache nudi i statički i dinamički sadržaj. Statički sadržaj spremi se u obične datoteke na poslužitelju kojima je lako upravljati. Dinamički sadržaj generira se za korisnika u trenutku kada zatraži stranicu. Dokument koji pregledava u tom trenutku postoji samo za njega; ako netko drugi gleda u isto vrijeme, može se dobiti drugačiji sadržaj.

Osim Apache web poslužitelja popularni su i Ngnix, IIS, te Google Web Server. Do prošle godine Apache je posluživao više od 50% aktivnih web stranica. Od travnja 2020. godine najpopularniji web server je Ngnix koji je poznat po visokim performansama, jednostavnoj konfiguraciji, niskoj potrošnji resursa i stabilnosti. Apache web poslužitelj je besplatan, sposoban je posluživati velike i male stranice, sadrži više od 60 službenih i mnogo neslužbenih modula koji omogućuju bogat skup značajki, otvorenog je kôda i jednostavno je konfigurirati poslužitelj [13]. Zbog tih značajki je korišten pri implementaciji rada *Pametna hranilica za kućne ljubimce*.

¹ HTTP je glavna i najčešća metoda prijenosa informacija na Web-u. Osnovna namjena ovog protokola je omogućavanje objavljivanja i prezentacije HTML dokumenata, tj. web stranica. HTTP je samo jedan od protokola aplikacijske razine koji postoje na Internetu.

3.1 Postavljanje Apache web poslužitelja

Raspberry Pi idealno je rješenje ako se želi „ugostiti“ neka web stranica prije nego li se podijeli sa ostatkom svijeta. Prije nego se kreće s postavljanjem Apache web poslužitelja na Raspberry Pi potrebno je obaviti prethodno navedeni korak postavljanja Raspberry Pi-a. Nakon toga potrebno je povezati Raspberry Pi na lokalnu mrežu. Ponovnim pokretanjem Raspberry-a te otvaranjem prozora terminala pokrene se naredba *hostname -I* pomoću koje se saznaje IP (engl. *Internet Protocol*) adresa web poslužitelja. Prilikom realizacije projekta *Pametna hraničica za kućne ljubimce* Pi je nakon svakog pokretanja imao istu IP adresu, no ukoliko to nije slučaj potrebno je statičkom DHCP (engl. *Dynamic Host Configuration Protocol*) rezervacijom osigurati statičku IP adresu Raspberry-a. Prije same instalacije Apache-a potrebno je provjeriti da li je Raspberry Pi OS ažuriran, a to se čini upisivanjem naredbi *sudo apt-get update* i *sudo apt-get upgrade*. Instalacija Apache Web poslužitelja obavlja se naredbom *sudo apt-get install apache2 -y*. Dodavanjem *-y* na kraj naredbi program automatski odgovara *YES* na bilo koje pitanje, umjesto da čeka da korisnik unese *y* (ili *n*). Apache postavlja testnu HTML (engl. *HyperText Markup Language*) datoteku u web mapu koja se može pregledavati sa Pi-a ili sa nekog drugog računala na lokalnoj mreži. Ta HTML datoteka nalazi se u direktoriju */var/www/html* i nazvana je *indeks.html*. Otvaranjem Chromium Web preglednika na Raspberry-u ili bilo kojeg web preglednika na nekom drugom računalu unutar lokalne mreže te upisivanjem IP adrese Raspberry-a ili „*localhost*“ otvara se zadana web stranica Apache web poslužitelja.



Slika 3.1 Zadana stranica Apache web poslužitelja [14]

4. RAZVOJ WEB APLIKACIJE

Web aplikacija predstavlja posrednika između hranilice i korisnika. Cilj je da korisnik prilikom pristupa određenoj web stranici ima mogućnost nahraniti kućnog ljubimca pritiskom na tipku „*Nahrani sada*“ te da pritiskom na tipku „*Live video*“ ima mogućnost uvida gdje se nalazi ljubimac, uvid u stanje hranilice i doma. U ovom dijelu rada opisani su neki od alata koji su korišteni pri izradi web aplikacije te postupak razvoja web aplikacije. To su:

- Stvaranje virtualnog okruženja,
- Flask,
- HTML i CSS,
- HTTP metode GET i POST,
- WSGI (engl. *Web Server Gateway Interface*),
- Port Forwarding,
- DDNS (engl. *Dynamic Domain Name System*).

4.1 Stvaranje virtualnog okruženja

Nakon uspješnog postavljanja Apache web poslužitelja potrebno je instalirati dva Python paketa. *Raspberry Pi Foundation* je prilikom razvoja Raspberry Pi računala odabrala Python kao glavni programski jezik zbog njegove jednostavnosti uporabe, snage i svestranosti. Python dolazi unaprijed instaliran na Raspberry Pi OS. Trenutno s Raspberry Pi OS-om dolazi verzija *Python 3.7*. Instalacija dvaju potrebnih paketa se pokreće naredbama *sudo apt-get install python3-dev* i *sudo apt-get install python3-venv*. Prvi paket sadrži datoteke zaglavlj, razvojni alat za izgradnju Python modula, te alat za ugradnju Python-a u aplikacije, a drugi paket služi za stvaranje izoliranog Python okruženja. Nakon ovog koraka prelazi se na stvaranje virtualnog okruženja. Stvaranjem virtualnog okruženja omogućuje se odvajanje različitih Python projekata u zasebne cjeline. Python aplikacije često koriste pakete i module koji nisu dio standardne biblioteke. Neka će aplikacija zahtijevati zastarjelu verziju biblioteke kako bi se neka greška ispravila. To znači da jedna Python instalacija neće moći zadovoljiti zahtjeve svake aplikacije. Ako je za aplikaciju A potrebna verzija 1.0 određenog modula, a aplikaciji B verzija 2.0, tada su ti zahtjevi u sukobu, a instaliranje verzije 1.0 ili 2.0 će jednu od aplikacija učiniti nesposobnom za rad. Rješenje ovog problema je stvaranje

virtualnog okruženja. Različite aplikacije mogu koristiti različita virtualna okruženja. Tako će aplikacija A imati vlastito virtualno okruženje s instaliranom verzijom 1.0, dok će aplikacija B imati virtualno okruženje s instaliranom verzijom 2.0. Ukoliko aplikacija B zatraži nadogradnju verzije na 3.0, ta nadogradnja neće utjecati na aplikaciju A [15].

Instalacija virtualnog okruženja vrši se unosom naredbe `sudo apt-get install virtualenv` u terminal. Nakon uspješne instalacije naredbom `sudo cd /var/www/` premješta se u direktorij koji se stvara instalacijom Apache-a. Unutar tog direktorija stvoreno je virtualno okruženje nazvano **feeder** unosom naredbe `sudo virtualenv -p /usr/bin/python3 feeder`. Prije samog dodavanja modula i paketa u virtualno okruženje potrebno je prilagoditi dopuštenja za direktorij kako bi direktorij bio siguran. Pristup za čitanje/pisanje omogućuje se sljedećim naredbama:

- `sudo chown -R pi:www-data /var/www/feeder/` - naredba `chown` mijenja korisnika i/ili grupu koja posjeduje datoteku. Datoteka je u vlasništvu `root` korisnika pa je za pravilno uređivanje datoteke potrebno koristit naredbu `sudo`. Ovom naredbom promijeni se `pi` u vlasnika datoteke, a `root` u grupu. Za svaku datoteku postoje tri dopuštenja. Jedno se odnosi na vlasnika datoteke, drugo na grupu, a treće za sve ostale. Tako možemo učiniti da vlasnik može imati sva dopuštenja, grupa samo čitati i izvršavati, a ostali samo izvršavati.
- `sudo chmod 750 -R /var/www/feeder/` - naredbom `chmod` kontroliramo tko može pristupiti datotekama, pretraživati direktorije i pokretati skripte. Dopuštenjem `750` vlasniku je omogućeno čitanje, pisanje i izvršavanje datoteke, a grupi čitanje i izvršavanje.

Nakon što su dopuštenja prilagođena, naredbom `cd /var/www/feeder/` premješta se u kreirani direktorij te naredbom `source bin/activate` aktivira se kreirano virtualno okruženje. Aktivacijom virtualnog okruženja trebao bi se pojaviti prefiks `feeder` ispred konzole.

Paketi i moduli unutar virtualnog okruženja mogu se instalirati, nadograditi i obrisati pomoću programa **pip**. Paketi koji su potrebni instaliraju se ovim naredbama:

- `pip3 install flask` - Python web okvir.
- `pip3 install RPi.GPIO` - instalacija paketa kojim se omogućava jednostavnija komunikacija s GPIO pinovima.

- *pip3 install mod_wsgi* - modul Apache HTTP poslužitelja kojim se omogućuje dijeljenje web aplikacija zasnovanih na Pythonu.

4.2 Flask

Web okvir je arhitektura koja sadrži knjižnice i alate prikladne za izgradnju i održavanje masivnih web projekata. Dizajnirani su za pojednostavljinjanje programa i promicanje ponovne upotrebe kôda. Python Flask Framework je mikrookvir temeljen na Werkzeug, Jinja2. Flask nije *full-stack framework* poput TurboGearsa ili Djanga, ali je zbog tzv. Flask ekstenzija lako proširiv i jednako moćan kao dva navedena. Razumljiv je i jednostavan za korištenje. Ekstenzije omogućuju da se aplikacija proširi baš onim što je korisniku potrebno pri realizaciji aplikacije pa ne dolazi do nepotrebnog gomilanja kôda. Flask aplikacija stvara se unutar izoliranog okruženja pa je zbog toga kreirano virtualno okruženje *feeder*. Instalacija Flask paketa na Raspberry-u obavlja se naredbom *pip install flask*. [16]

4.2.1 Kreiranje Flask aplikacije

Svaka Flask aplikacija zahtjeva uvoz objekta klase Flask, definiranje same aplikacije, definiranje *view* funkcije s pripadnim dekoratorom *@app.route*. te pokretanje poslužitelja.

Prilikom realizacije projekta *Pametna hranilica za kućne ljubimce* kôd Flask aplikacije napisan je unutar dokumenta *app.py*. Cjeloviti kôd nalazi se u prilogu C. Kako bi aplikacija ispravno funkcionalala potrebno je uvesti sve potrebne module. Moduli se uvoze *import* naredbom. Modul je datoteka koja se sastoji od Python kôda i može definirati funkcije, klase i variable. Svaka datoteka koja sadrži Python kôd može se smatrati modulom. Naredbom *from* uvoze se samo određeni moduli iz neke sekcije. *Slika 4.1* prikazuje dio kôda *app.py* datoteke kojim se uvoze potrebni moduli za ispravnu funkcionalnost aplikacije. *__name__* je standardan argument Flask konstruktora [17].

```
from flask import Flask, redirect, render_template, request, url_for, flash
import subprocess
import commonTasks
import os
import configparser
import os, sys, time

app = Flask(__name__)
```

Slika 4.1 Uvoz potrebnih modula i definiranje standardnog argumenta Flask konstruktora

Svaki se zahtjev od strane Web preglednika usmjerava na funkciju koja poslani zahtjev (engl. *request*) obradi i odgovori željenom web stranicom (engl. *response*). Da bi se to omogućilo, funkciji se iznad deklaracije dodaje `@app.route` dekorator čiji prvi argument predstavlja rutu koju je klijent zatražio. Na *slici 4.2* prikazan je dio kôda *app.py* dokumenta koji sadrži `@app.route` dekorator `/feedbuttonclick`. Zahtjev korisnika usmjerava se na funkciju `feedbuttonclick()` koja poziva funkciju `spin_hopper()` definiranu unutar dokumenta *commonTasks.py* čiji se cijeloviti kôdnalazi u prilogu D. Ako je funkcija uspješno pozvana i izvršena, korisniku se šalje poruka „*Hranjenje uspješno*“. U suprotnom se korisniku javlja greška.

```
@app.route('/feedbuttonclick', methods=['GET', 'POST'])
def feedbuttonclick():
    try:
        spin = commonTasks.spin_hopper(hopperGPIO, hopperTime)

        if spin != 'ok':
            flash('Error! No feed activated! Error Message: ' + str(spin), 'error')
            return redirect(url_for('home_page'))

        flash('Hranjenje uspješno! :)')
        return redirect(url_for('home_page'))
    except Exception as e:
        return render_template('error.html', resultsSET=e)
```

Slika 4.2 Kreiranje rute „/feedbuttonclick“ unutar app.py dokumenta

Na samom kraju Flask aplikacije potrebno je pokrenuti poslužitelja aplikacije, a to činimo dijelom kôda *app.py* dokumenta prikazanim na *slici 4.3*. Flask prema zadanim postavkama pokreće poslužitelj na 127.0.0.1 (localhost) i portu 5000.

```
if __name__ == '__main__':
    app.debug = False
    app.run(host='0.0.0.0', threaded=True)
```

Slika 4.3 Pokretanje poslužitelja Flask aplikacije

4.3 HTML i CSS

4.3.1 HTML

HTML (engl. *HyperText Markup Language*) [18] je prezentacijski jezik za izradu web stranica. Jednostavan je za uporabu i lako se uči pa je zato jako popularan i prihvaćen. Raširen je upravo zbog svoje jednostavnosti i zbog činjenice da je besplatan i dostupan svima od samog početka. Razvio ga je 1993. Tim Berners-Lee. U to vrijeme u HTML dokument nije bilo moguće priložiti čak ni sliku. Danas je u uporabi HTML5 koji donosi razne mogućnosti kao npr. reprodukcija videa bez korištenja Adobe Flash-a. Web preglednik prikazuje HTML dokument. Temeljna zadaća HTML jezika jest uputiti web preglednik kako će prikazati HTML dokument. Nastoji se da taj dokument izgleda isto bez obzira o kojem računalu, operacijskom sustavu ili web pregledniku je riječ. HTML nije programski jezik. Njime se ne može izvršiti čak ni najjednostavnija operacija zbrajanja ili oduzimanja. Ekstenzija HTML datoteke je *.htm* ili *.html*. Svaki HTML dokument sastoj se od osnovnih građevnih blokova tj. HTML elemenata. Svaki HTML element sastoji se od para HTML oznaka. *<html>* elementom se označava početak HTML dokumenta. Unutar *<html>* dokumenta nalaze se *<head>* i *<body>* elementi. *<head>* element predstavlja zaglavje HTML dokumenta, a u *<body>* elementu se kreira sadržaj HTML dokumenta. Svaka HTML oznaka počinje znakom *<* (manje od), a završava znakom *>* (više od). Zatvarajuća HTML oznaka kreira se isto kao i otvarajuća, samo se prije završnog znaka *>* dodaje kosa crta / (engl. *slash*). HTML oznake u paru kreiraju HTML element.

Web aplikacije koriste HTML dokumente za prikaz informacija posjetitelju. HTML datoteke korištene su za prikaz informacija i prilikom realizacije ovog projekta, a nalaze se u prilogu F.

Korištenjem funkcije *render_template()* unutar Python dokumenta *app.py* (Flask aplikacije) vraća se (engl. *return*) HTML datoteka spremljena unutar direktorija *templates*. Jako je važno da se HTML datoteka koju Python skripta Flask aplikacije vraća nalazi u direktoriju *templates* i da se taj direktorij nalazi unutar istog virtualnog okruženja (*feeder*) u kojem je kreirana Python skripta *app.py* koja pokreće web stranicu.

4.3.2 CSS

CSS (engl. *Cascading Style Sheet*) [19] je jezik koji se koristi za oblikovanje web stranica. Uz HTML, CSS je osnovna tehnologija na kojoj se temelji današnji web. Style Sheet pojam se često upotrebljava za datoteku koja sadrži CSS kôd. Style Sheet datoteka definira izgled web stranice. Prva verzija CSS-a definirana je 1996. godine, a danas je u uporabi CSS3. CSS3 verzija još nije finalizirana i neprestano se razvija i nadograđuje. Glavna ideja CSS-a je odvajanje prezentacijskog kôda u zasebne datoteke i njegovo definiranje pomoću jednostavnih pravila koja se mogu odnositi na više elemenata odjednom. CSS pravilo može se napisati tako da bude primijenjeno na sve elemente ili samo na neke elemente.

CSS kôd se obično piše odvojeno od HTML kôda tj. u zasebnoj datoteci. Stoga je potrebno HTML dokument povezati s CSS datotekom. To se čini korištenjem HTML *linka*: `<link href="path" rel="stylesheet" type="text/css">`. Povezivanje *layout.html* datoteke s CSS datotekama korištenim pri realizaciji ovog projekta prikazano je na *slici 4.4*. Sve CSS datoteke korištene pri realizaciji rada *Pametna hranilica za kućne ljubimce* nalaze se u prilogu G.

Kada se HTML *link* koristi za uključivanje CSS datoteka, atribut *rel* mora imati vrijednost *"stylesheet"*, a atribut *type* vrijednost *"text/css"*. Atribut *href* postavlja se na putanju do CSS datoteke koja se želi uključiti.

```
<!DOCTYPE html>
<html lang="en">
<head>
    {% block head %}
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <meta name="description" content="">
        <meta name="author" content="">

        <title>Pametna hranilica</title>

        <link rel="icon" href="/static/images/titleIcon.png">

        <link href="static/css/bootstrap.min.css" rel="stylesheet">
        <link href="static/css/googleFonts.css" rel="stylesheet" type="text/css">
        <link href="static/css/feedButton.css" rel="stylesheet">
        <link href="static/css/camButton.css" rel="stylesheet">
        <link href="static/css/navBar.css" rel="stylesheet">

    {% endblock %}
```

Slika 4.4 Povezivanje HTML i CSS datoteka

Direktorij u kojem su smještene CSS datoteke naziva se *CSS* i nalazi se unutar direktorija *static* koji se kao i direktorij *templates* mora nalaziti unutar istog virtualnog okruženja kao i dokument *app.py*. Osim *CSS* direktorija unutar *static* direktorija smješteni je i direktorij *images* u kojem se nalaze fotografije koje su postavljene na web stranici.

4.4 HTTP metode GET i POST

Flask koristi različite dekoratore za rukovanje HTTP zahtjevima. Kao što je već spomenuto ranije, HTTP protokol osnova je za podatkovnu komunikaciju na World Wide Web-u. HTTP *GET* i *POST* metode su načini slanja podataka serveru ili klijentu. Podaci se između različitih stranica web aplikacije prenose preko URL-a (engl. *Uniform Resource Locator*). Prema zadanim postavkama koristi se *GET* metoda. Podaci koji se šalju *GET* metodom nisu zaštićeni pa nije preporučljivo koristiti ovu metodu ako se npr. unose lozinke ili neki važni osobni podaci. Da bi bilo moguće koristiti *POST* metodu potrebno je unutar dekoratora `@app.route` dodati argument *methods* [20].

```
<div class="row">
    [% if cameraStatus == '1' %]
        <div class="col-md-6 text-center">
            <form role="form" name="feedbuttonclick" method="post" action="/feedbuttonclick">
                <button class="btn btn-lg feedButton" type="submit">Nahrani sada</button>
            </form>
        </div>
```

Slika 4.5 Dio *kôda* HTML dokumenta *home.html*

Unutar HTML dokumenta *home.html* koji prikazuje početnu stranicu web aplikacije korištena je metoda *POST* kojom se prenose podaci između HTML stranice i Flask aplikacije. Korisnik pritiskom na tipku „*Nahrani sada*“ šalje zahtjev na URL „*/feedbuttonclick*“. Korisnik tim zahtjevom poziva funkciju *feedbuttonclick()* koja je objašnjena u poglavljju 4.2.1.

4.5 WSGI

WSGI (engl. *Web Server Gateway Interface*) [21] je sučelje između web poslužitelja i web aplikacije. Kako bi se Flask aplikacija uspješno objavila na Apache web poslužitelju potrebno je instalirati modul *mod_wsgi*. Instalacija ovoga modula obavlja se naredbom *sudo apt-get install libapache2-mod-wsgi-py3 python-dev*. Za pokretanje aplikacije potrebno je kreirati *imeaplikacije.wsgi* datoteku unutar virtualnog okruženja. Za većinu aplikacija

dovoljno je unutar `.wsgi` datoteke upisati samo `from app import app as application` no ukoliko se koristi virtualno okruženje onda je potrebno na vrh datoteke dodati sljedeće linije kôda:

```
activate_this = '/path/to/env/bin/activate_this.py'  
with open(activate_this) as file_:  
    exec(file_.read(), dict(__file__=activate_this))
```

`feeder.wsgi` datoteka korištena pri implementaciji ovog rada postavljena je u prilogu E.

4.5.1 Konfiguracija Apache-a

Pokretanjem naredbe `sudo nano /etc/apache2/sites-available/000-default.conf` te izmjenom sadržaja pokrenute datoteke sadržajem sa *slike 4.6* kreirana aplikacija uspješno je postavljena na Apache web poslužitelj [21].

```
<VirtualHost *:80>  
    WSGIDaemonProcess feeder user=www-data group=www-data threads=5  
    WSGIScriptAlias / /var/www/feeder/feeder/feeder.wsgi  
  
    <Directory /var/www/feeder/feeder>  
        WSGIProcessGroup feeder  
        Order allow,deny  
        Allow from all  
        Require all granted  
    </Directory>  
  
    <Files feeder.wsgi>  
        Order allow,deny  
        Allow from all  
    </Files>  
  
    ErrorLog /var/www/feeder/feeder/logs/apacheError.log  
    CustomLog /var/www/feeder/feeder/logs/apacheAccess.log combined  
  
</VirtualHost>
```

Slika 4.6 Konfiguracija Apache web poslužitelja

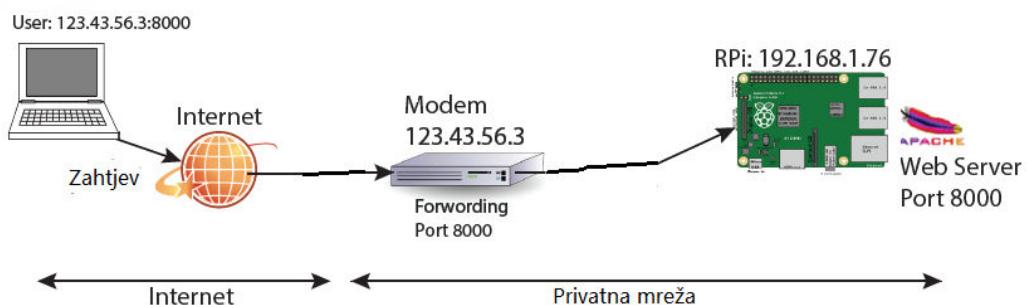
`<VirtualHost *:80>` osigurava da Apache sluša port 80, a zvjezdica `*` odgovara svakoj adresi. Kao primarna stranica poslužitelja postavljena je aplikacija kreirana unutar virtualnog okruženja *feeder*. `WSGIDaemonProcess` omogućava izvođenje procesa u pozadini, a kada je omogućen, jedino što procesi rade je pokretanje WSGI aplikacija dodijeljenih toj grupi procesa. Naredbom `threads` definira se broj niti (zadataka koji se istovremeno izvršavaju u okviru jednog procesa) koje je potrebno stvoriti za obradu zahtjeva svakog procesa koji se pokreće u pozadini. `WSGIScriptAlias` mapira URL na lokaciju datotečnog sustava i označava cilj kao WSGI skriptu. `Order allow, deny` govori web poslužitelju da su pravila `allow`

(dopusti) obrađena prije pravila *deny* (odbij). Ako se klijent ne slaže s pravilom *allow* tj. ako se slaže s pravilom *deny*, tada će klijentu biti zabranjen pristup [22].

4.6 PORT FORWARDING

Port forwarding omogućuje računalima na Internetu spajanje s određenim računalom ili uslugom unutar privatne mreže. Neko npr. računalo na Internetu želi pristupiti računalu (npr. Raspberry Pi) koje je spojeno na privatnu mrežu. Računalo na Internetu šalje zahtjev na javnu IP adresu računala s određenim brojem porta. Port je logička veza koju programi i usluge koriste za razmjenu informacija. Zahtjev koji računalo šalje proći će preko Interneta do usmjerivača i kada stigne usmjerivač mora znati kamo će proslijediti zahtjev za taj port. Bez port forwardinga računalo koje šalje zahtjev neće se moći povezati s drugim računalom jer usmjerivač ne zna što treba učiniti s tim zahtjevom. Port forwardingom usmjerivaču govorimo da sve zahtjeve s tim portom proslijedi na točno određeno računalo [23].

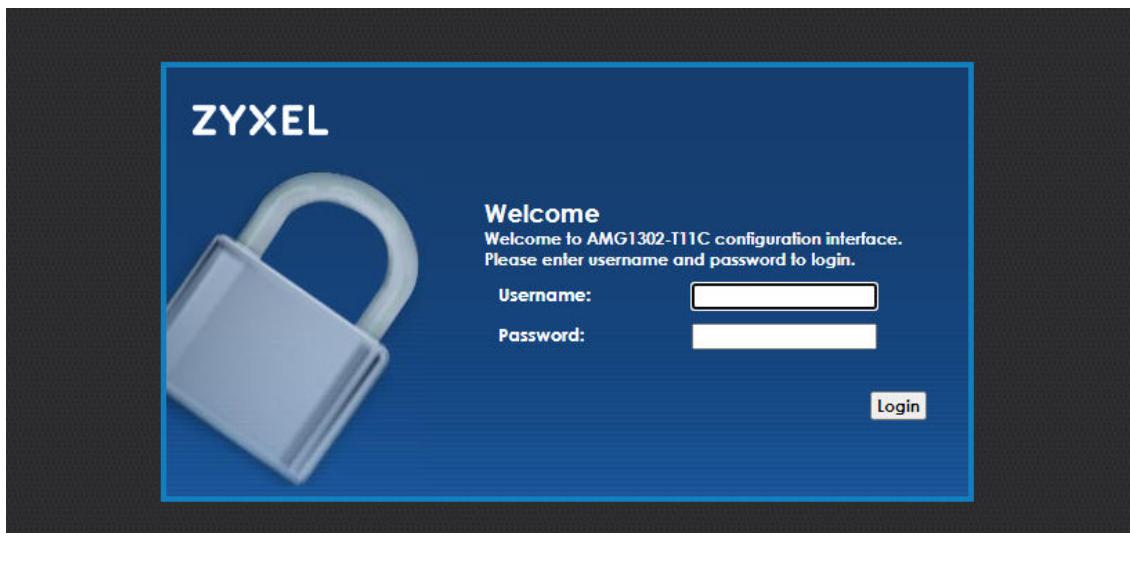
Na *slici 4.7* prikazan je proces komunikacije između pošiljatelja zahtjeva, usmjerivača i primatelja zahtjeva. Korisnik putem Interneta šalje zahtjev na javnu IP adresu modema/usmjerivača s određenim brojem porta, u ovom slučaju 8000. Raspberry Pi spojen je na privatnu mrežu ili putem Ethernet-a ili putem Wi-Fi-a. Usmjerivač prima korisnikov zahtjev i zna da svaki zahtjev koji pristigne s portom 8000 treba proslijediti Raspberry Pi-u jer je tako određeno port forwardingom. Raspberry Pi je postavljen kao web poslužitelj te je korisniku omogućeno pregledavanje web stranice koju poslužitelj nudi.



Slika 4.6 Komunikacija pošiljatelja zahtjeva, usmjerivača i primatelja zahtjeva

Port forwarding obavlja se na stranici za konfiguraciju usmjerivača. Stranica usmjerivača otvara se tako da se u web preglednik unese IP adresa usmjerivača. Nakon što se stranica učita potrebno je unijeti sigurnosne podatke za prijavu koje je moguće pronaći u uputama usmjerivača. Prilikom realizacije projekta *Pametna hranilica za kućne ljubimce* port

forwarding je obavljen preko usmjerivača Zyxel AMG1302-T11C. Učitavanjem stranice za konfiguraciju ovog modema prikazuje se prozor kao na *slici 4.8*.



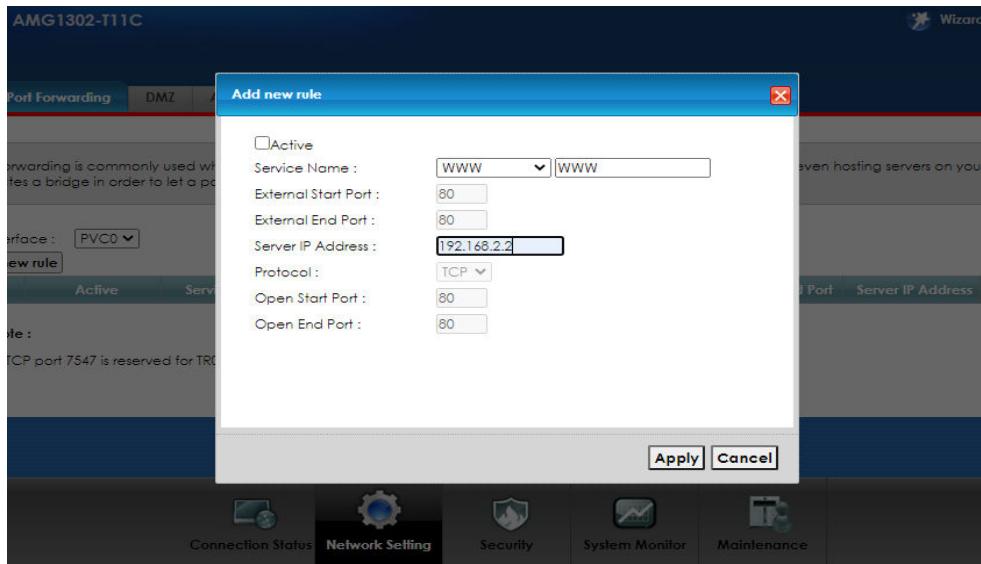
Slika 4.7 Početna stranica za konfiguraciju usmjerivača Zyxel AMG1302-T11C

Nakon unosa korisničkog imena i lozinke te pronađaska port forwarding odjeljka na stranici otvara se prozor prikazan na *slici 4.9*.

A screenshot of the "Port Forwarding" tab in the Zyxel AMG1302-T11C configuration interface. The top navigation bar shows "Zyxel AMG1302-T11C" and includes links for "Wizard", "Logout", and other tabs like "General", "Port Forwarding" (which is selected), "DMZ", and "ALG". The main content area has a sub-header "NAT". It contains a note about port forwarding: "Port Forwarding is commonly used when you want to do some Internet activities, such as online gaming, P2P file sharing, or even hosting servers on your network. It creates a bridge in order to let a party from the Internet contact a specific LAN client on your network correctly." Below this is a table with columns: "WAN Interface" (set to "PVC0"), "Add new rule", "#", "Active", "Service Name", "External Start Port", "External End Port", "Internal Start Port", "Internal End Port", "Server IP Address", and "Modify". A note at the bottom states: "The TCP port 7547 is reserved for TR069 connection request port." At the bottom of the page is a navigation bar with icons for "Connection Status", "Network Setting" (which is highlighted in black), "Security", "System Monitor", and "Maintenance".

Slika 4.9 Port forwarding odjeljak

Klikom na *Add new rule* otvara se prozor s više polja prikazan na *slici 4.10*. Unutar polja „Service Name“ odabire se vanjski port. Prilikom realizacije ovog projekta odabran je port 80. Unutar polja server IP address potrebno je unijeti statičku IP adresu Raspberry Pi-a. Klikom na „*apply*“ uspješno je obavljen port forwarding.



Slika 4.10 Proces port forwardinga Raspberry Pi-a

Svi zahtjevi koji sada budu dolazili s Interneta s portom 80 biti će usmjereni na Raspberry Pi. Tako će korisnik koji želi pristupiti Web aplikaciji za hranjenje kućnog ljubimca morati unijeti čitavu javnu IP adresu (npr. 95.156.161.226:80). Taj zahtjev koji korisnik podnese doći će do usmjerivača i usmjerivač će taj zahtjev proslijediti Raspberry-u koji će korisniku „poslužiti“ web stranicu za hranjenje kućnog ljubimca. Jako je nezgodno svaki put kada se želi pristupiti stranici unositi sve ove brojive javne IP adrese pa je potrebno naći rješenje da se to izbjegne. Javna IP adresa također se često mijenja što dodatno otežava pristup aplikaciji. Moguće rješenje ovog problema objašnjeno je u idućem poglavlju.

4.6.1 DDNS – DYNAMIC DNS

DDNS označava dinamički sustav domene. To je usluga koja mapira imena internetskih domena na IP adresu. Omogućuje pristup kućnom računalu s bilo kojeg mesta na svijetu. Servisi dinamičkog DNS-a pomažu na način da prilikom svake promjene IP adrese povežu uvijek s istim imenom domene tj. jednostavnim nazivom umjesto brojeva u IP adresi.

Dinamički DNS servisi održavanje nekoliko adresa nude besplatno, uz neko od besplatnih imena domena po izboru. Princip rada je relativno jednostavan – usmjerivač tj. modem ili računalo će periodično ili prilikom svake promjene statusa veze na internet odabranom DNS servisu javiti novu IP adresu – i s tom vrijednošću će DNS servis odgovarati na upite za određenu domenu [24].

Otvaranjem odjeljka Dynamic DNS na stranici za konfiguraciju usmjerivača otvara se prozor prikazan na *slici 4.11*.

Dynamic DNS Configuration

Dynamic DNS Service Provider : Host Name : Username : Password :	<input type="radio"/> Enable <input checked="" type="radio"/> Disable <input type="text" value="www.no-ip.com"/> <input type="text"/> <input type="text"/> <input type="text"/>
--	---

Slika 4.11 Odjeljak za konfiguraciju DDNS-a

Otvaranjem polja „Service Provider“ nude se tri servisa. Pri realizaciji ovog projekta odabran je *noip.com* DNS servis zbog jednostavne izrade besplatnih domena. Slične alternative su *Dyndns*, *Duckdns*, itd. Prije popunjavanja polja na stranici za konfiguraciju usmjerivača potrebno je na web stranici www.noip.com izraditi korisnički račun te kreirati ime domene pod kojim će se pretraživati web aplikacija za hranjenje. Na stranici za konfiguraciju usmjerivača moguće je pronaći dinamičku IP adresu na koju korisnik šalje zahtjev i koja će biti zamijenjena odabranim nazivom domene. Na *slici 4.12* prikazan je način kreiranja imena domene za stranicu koja se objavljuje na Internetu.

Create Hostname		Search... <input type="button" value="x"/> <input type="button" value=""/>	
Hostname ▾	Last Update	IP / Target	Type
pametnahranilica.ddns.net	Aug 27, 2021 22:29 PDT	95.156.161.226	A <input type="button" value=""/> <input type="button" value="Modify"/> <input type="button" value="x"/>

Slika 4.12 Kreiranje naziva stranice koju objavljujemo na internetu

Nakon što je kreirana domena, na stranici za konfiguraciju usmjerivača popune se polja *Host Name*, *Username* i *Password* kao što je prikazano na *slici 4.13*. U polje *Username*

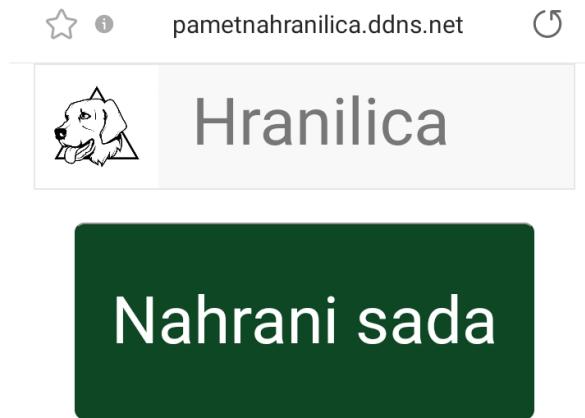
i *Password* potrebno je unijeti korisničko ime i lozinku koje je upotrijebljeno pri izradi računa na www.noip.com web stranici. Omogućavanjem (engl. *Enable*) DDNS-a uspješno je na Internet postavljena aplikacija pod nazivom www.pametnahranilica.ddns.net. Na *slici 4.13* je također vidljivo da se može pročitati i status DNS-a. Usmjerivač u određenim intervalima provjerava IP adresu poslužitelja te obavještava DNS servis o svakoj promjeni te adrese, tako da bez obzira na promjenu korisnik bude uslužen stranicom www.pametnahranilica.ddns.net

Dynamic DNS Configuration	
Dynamic DNS	<input checked="" type="radio"/> Enable <input type="radio"/> Disable
Service Provider :	www.no-ip.com ▾
Host Name :	pametnahranilica.ddns.net
Username :	adoko00
Password :	*****
 Dynamic DNS Status	
User Authentication Result:	Accepted
Last Updated Time:	Sat Aug 28 07:29:06 2021
Current Dynamic IP:	95.156.161.226

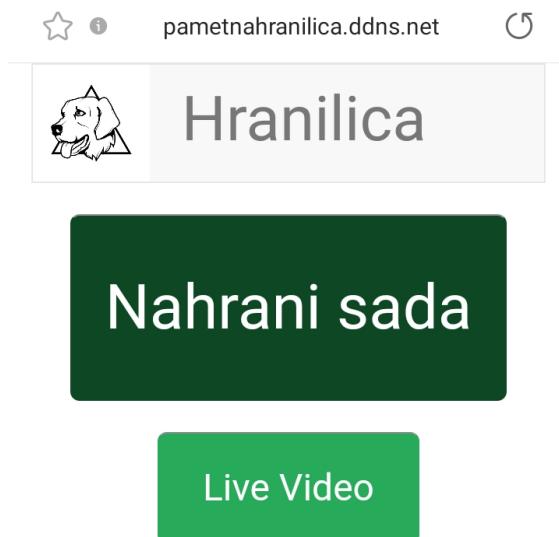
Slika 4.13 Konfiguirani DDNS

4.7 Prikaz sučelja web aplikacije

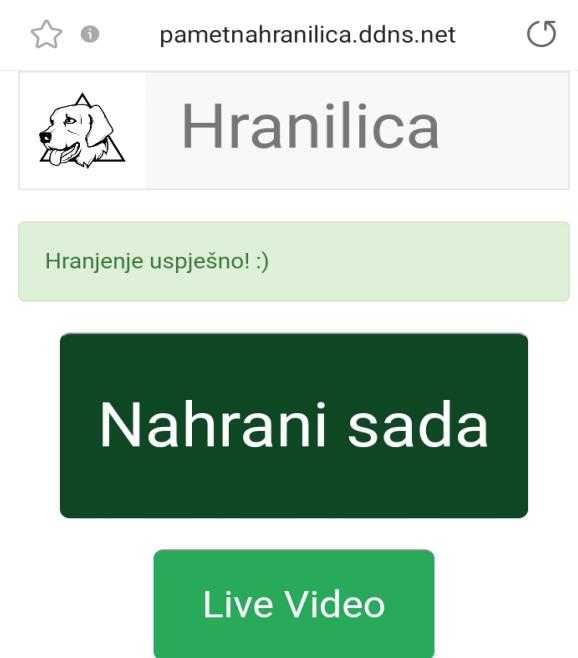
Na *slici 4.14* prikazano je sučelje kada je status *camera = 0*, odnosno kada kamera nije priključena. Na *slici 4.15.* je prikazano sučelje web aplikacije kada je kamera uključena, odnosno kada je status *camera = 1*. Pritiskom na tipku „*Nahrani sada*“ servo motor bi se trebao pokrenuti i izbaciti određenu količinu hrane, a ako je hranjenje obavljenno uspješno ispisat će se poruka „*Hranjenje uspješno! :)*“ kao što je prikazano na *slici 4.16*. Klikom na tipku „*Live video*“ otvorit će se prozor prikazan na *slici 4.17*.



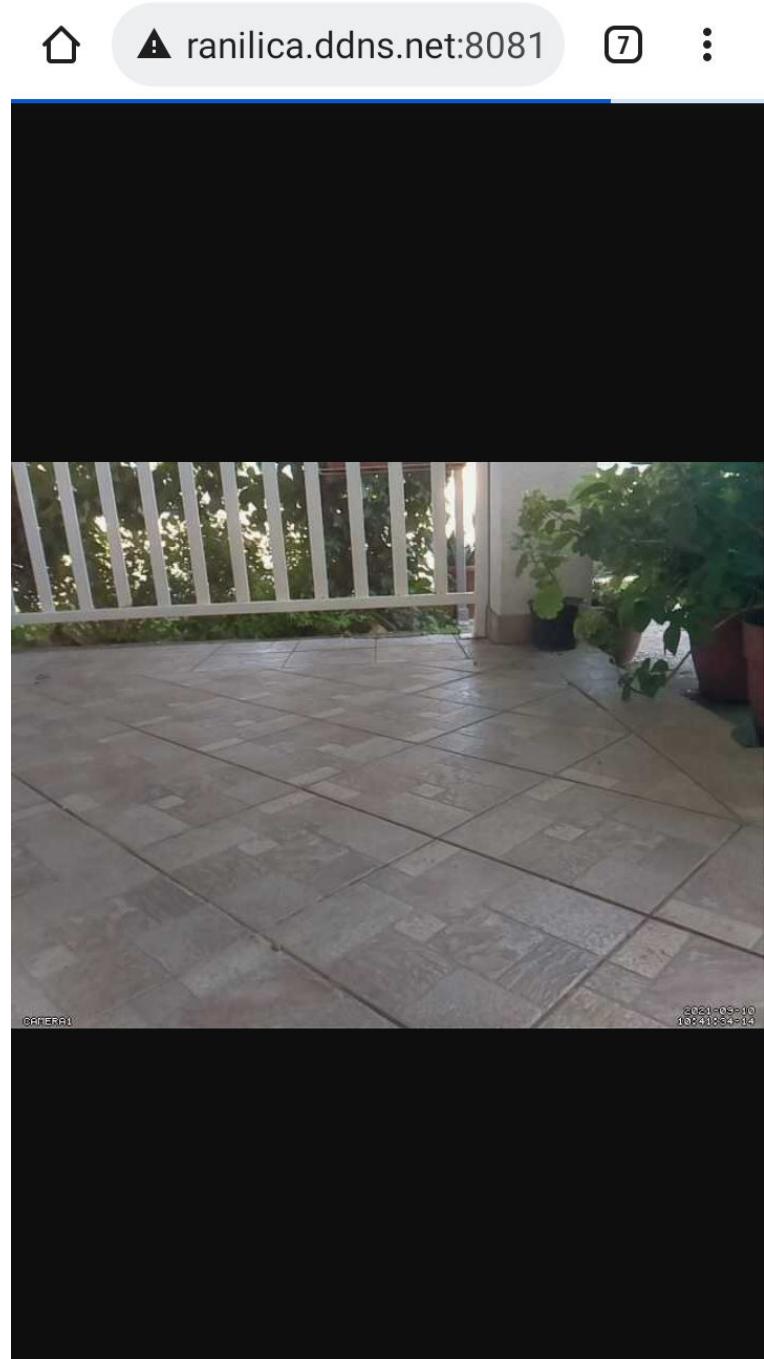
Slika 4.14 Sučelje web aplikacije kada kamera nije priključena



Slika 4.15 Sučelje web aplikacije kada je kamera priključena



Slika 4.16 Sučelje web aplikacije nakon uspješnog hranjenja



Slika 4.17 Video prijenos uživo

5. SERVO MOTOR

Servo motori su električni uređaji s preciznom kontrolom položaja i velikom energetskom učinkovitosti. Kao i ostali motori, rade na principima elektromehaničke pretvorbe energije. Servo motor koristi običan motor i povezuje ga sa senzorom za povratnu informaciju o položaju. Tu informaciju koristi za kontrolu brzine i položaja. Servo motor se od ostalih motora razlikuje zbog sljedećih karakteristika:

- Izvrsna kontrola položaja i brzine osovine koju običan motor nema.
- Smjer vrtnje mogu promijeniti jako brzo zbog niskog momenta inercije.
- Ne pregrijavaju se.
- Mogu brzo ubrzati ili usporiti.
- Sposobni su održavati statički položaj.

Servo motori klasificirani su na temelju električnih i mehaničkih aspekata. Na temelju električnih aspekata servo motori se mogu podijeliti ovisno o tome jesu li napajani istosmjernom (DC motori) ili izmjeničnom (AC motori) strujom. Na temelju mehaničkih aspekata mogu se podijeliti na polukružne servo motore (rotacija ograničena na 180 stupnjeva) i servo motore s kontinuiranom rotacijom (okreće se za 360 stupnjeva). Dizajn, konstrukcija, način rada i područje primjene su različiti. Koriste se za upravljanje automobilima, robotima, za radio upravljanje, u farmaciji, vojsci itd. [25].

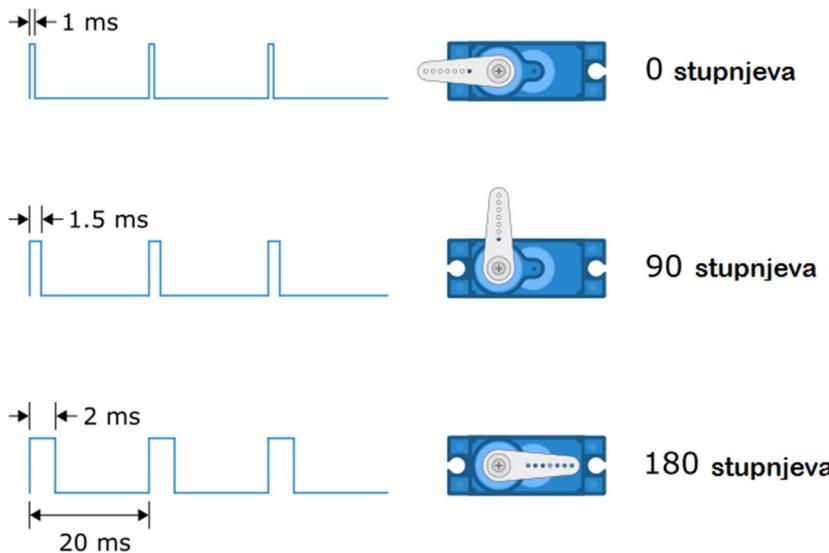
5.1 Princip rada servo motora

Da bi se razumjelo kako servo motor radi potrebno je poznavati njegove osnovne dijelove, a to su: **istosmjerni motor**, **potenciometar**, **upravljački krug** i **zupčanici**. Svi navedeni dijelovi smješteni su unutar kućišta, a jedini vidljivi dio je osovina s kojim je motor povezan preko zupčanika. Prilikom okretanja motora mijenja se otpor potenciometra, pa upravljački krug može precizno regulirati položaj i brzinu motora. Potenciometar može „osjetiti“ mehanički položaj osovine.

Servo motori prilikom rada koriste mehanizam povratne veze. Trenutni položaj osovine se pomoću potenciometra pretvara u električni signal i uspoređuje se sa željenim signalom. Kad osovina dosegne željeni položaj, prekida se dotok energije, a motor ostaje u stanju mirovanja držeći dosegnuti položaj. Da bi se servo motor usmjerio, preko signalne linije mu

se šalju elektronički pulsovi. Kako je stvarni ulaz u motor razlika između povratnog signala (trenutnog položaja) i potrebnog signala, tada je brzina motora proporcionalna razlici između trenutnog položaja i potrebnog položaja tj. ukoliko se motor nalazi blizu željenog položaja okretat će se sporije, a ukoliko je udaljeniji od željenog položaja rotirat će se brže. Količina energije koja je potrebna motoru proporcionalna je udaljenosti koju treba prijeći. Ovakav način rada servo motora naziva se proporcionalno upravljanje (engl. *Proportional Control*).

Servo motor sadrži 3 žice: plus, minus i upravljačka žica. Servo motorom se upravlja slanjem signala s moduliranom širinom impulsa, PWM (engl. *Pulse-Width Modulation*) signala, kroz upravljačku žicu. Puls se šalje svakih 20 ms (frekvencija 50 Hz). Širina impulsa određuje položaj vratila. Primjer upravljanja servo motorom putem PWM signala prikazan je na *slici 5.1*. Ukoliko servo motor primi signal kraći od 1.5 ms, motor se okreće u smjeru suprotnom od kazaljke na satu prema 0° , a ako je trajanje signala duže od 1.5 ms vrtnja se odvija od 0° prema 180° . [26]



Slika 5.1 Kontrola servo motora PWM signalom

5.2 TowerPro MG995 servo motor s kontinuiranom rotacijom

Prilikom realizacije projekta *Pametna hranilica za kućne ljubimce* korišten je TowerPro MG995 servo motor s metalnim zupčanikom i kontinuiranom rotacijom (360 stupnjeva) prikazan na *Slici 5.2*. Radi se o digitalnom servo motoru koji brzo prima i obrađuje PWM signale, pruža jako dobar okretni moment i snagu zadržavanja položaja. Rad ovog servo motora razlikuje se od rada ostalih servo motora jer će ovaj motor prilikom slanja impulsa širine 1.5 ms biti statičan, dok će se pri širim signalima rotirati prema naprijed, a pri užim

signalima prema natrag. Pakiran je u čvrsto plastično kućište što ga čini otpornim na vodu i prašinu. Opremljen je kao i svi ostali servo motori sa 3 žice. Crvena žica predstavlja plus žicu, smeđa žica minus žicu, a narančasta upravljačku žicu. Osnovne karakteristike prikazane su u Tablici 5.1. [27].

Tablica 5.1 Karakteristike TowerPro MG995 servo motora

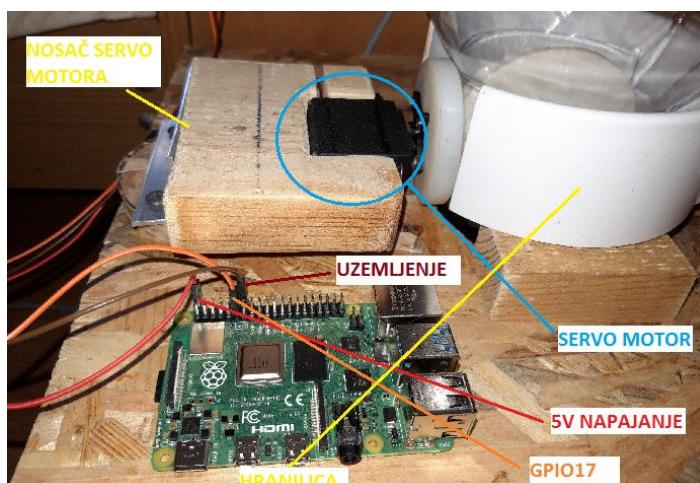
Radna temperatura	-30 ° ~ +60 °
Kut rotacije	360 °
Okretni moment	4.8 V: 130,4 oz-in (9.40 kg-cm), 6.0 V: 152,8 oz-in (11.00 kg-cm)
Brzina	4.8 V: 0.20 s/60 °, 6.0 V: 0.16 s/60 °
Tip servo motora	analogni servo
Radni napon	4.8 - 7.2V



Slika 5.2 TowerPro MG995 servo motor [27]

5.3 Povezivanje servo motora s Raspberry Pi-em

Kako bi se omogućilo povezivanje servo motora na GPIO pinove potrebne su 3 muško-ženske žice. Kao što je prikazano na slici 5.3, crvena žica servo motora spaja se na pin broj 2 ili 4, odnosno na napajanje +5V. Smeđa žica spojena je na jedan od pinova za uzemljenje (pri realizaciji ovog projekta korišten je pin broj 14). Narančasta tj. upravljačka žica spaja se na GPIO17 odnosno pin broj 11.



Slika 5.3 Servo motor povezan na GPIO pinove

5.4 Programski kôd za upravljanje servo motorom

Kôd prikazan na *slici 5.4* je pohranjen unutar datoteke *commonTasks.py*. Unutar datoteke definirana je funkcija *spin_hopper()* koja se poziva unutar *app.py* flask aplikacije kada korisnik na web stranici za hranjenje pošalje zahtjev na URL „/feedbuttonclick“. Naredbom *import* potrebno je uvesti već spomenuti modul *RPi.GPIO* kojim se Python kôdu omogućuje upravljanje GPIO pinovima. Servo motor spojen je na GPIO pinove Raspberry Pi-a na način opisan u prethodnom potpoglavlju.

```
#uvoz potrebnih biblioteka
import configparser
import time
import os
import RPi.GPIO as GPIO

#Pronalazi konfiguracijsku datoteku
dir = os.path.dirname(__file__) # os.getcwd()
configFilePath = os.path.abspath(os.path.join(dir, "app.cfg"))
configParser = configparser.RawConfigParser()
configParser.read(configFilePath)

# Čita varijable iz konfiguracijske datoteke
hopperGPIO = str(configParser.get('feederConfig', 'Hopper_GPIO_Pin'))
hopperTime = str(configParser.get('feederConfig', 'Hopper_Spin_Time'))

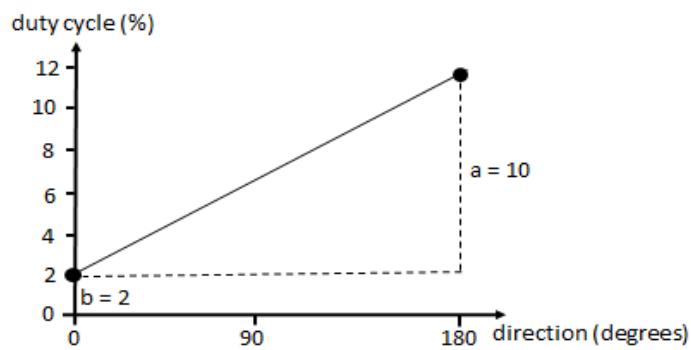
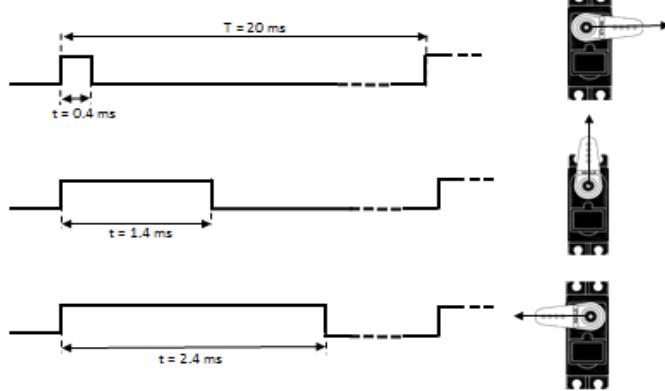
def spin_hopper(pin, duration): #funkcija za okretanje servo motora
    try:
        pin=int(pin) #kontrolni pin (11)
        duration=float(duration)
        GPIO.setwarnings(False)
        GPIO.cleanup(pin)
        GPIO.setmode(GPIO.BCM) #omogućuje korištenje brojeva pinova
        GPIO.setup(pin, GPIO.OUT) #postavlja pin 11 kao izlazni, onaj na koji se šalje PWM signal
        pwm=GPIO.PWM(pin,50) #50 je frekvencija PWM signala
        pwm.start(50) #šalje se PWM signal
        pwm.ChangeDutyCycle(12) #rotacija za 180 stupnjeva
        pwm.ChangeDutyCycle(12)
        time.sleep(duration) #čeka neko vrijeme
        pwm.start(0) #prestaje se slati PWM signal, zaustavlja se rotacija
        GPIO.cleanup(pin)

        return 'ok'
    except Exception as e:
        return 'ok' # e
```

Slika 5.4 commonTasks.py datoteka

Za definiranje kuta okreta korištena je funkcija *ChangeDutyCicle()* koja je dio knjižice *RPi.GPIO*. Većina servo motora koristi frekvenciju PWM signala jednaku 50 Hz što odgovara periodu od 20 ms. Na *slici 5.5* prikazana je ovisnost kuta okreta servo motora o vrijednosti duty cicle-a [28]. Postavljanjem vrijednosti duty cycle-a na 12% servo motor se okreće za 180 stupnjeva. U gornjem primjeru servo motor je rotiran za 360 stupnjeva jer količina hrane koja ispadne rotacijom servo motora za taj kut odgovara željenoj količini. *Start()* funkcijom uključuje se servo motor i započinje slanje PWM signala preko kontrolne žice, a *stop()* funkcijom zaustavlja se slanje PWM signala.

t	Duty Cycle	Direction
0.4 ms	0.4/20 = 2%	0 degs
1.4 ms	1.4/20 = 7%	90 degs
2.4 ms	2.4/20 = 12%	180 degs



Slika 5.5 Ovisnost kuta okreta o vrijednosti duty cycle-a [28]

6. RASPBERRY Pi KAMERA MODUL

Prilikom realizacije projekta *Pametna hranilica za kućne ljubimce* korišten je Raspberry Pi kamera modul kako bi vlasnik u svakom trenutku imao nadzor nad kućnim ljubimcem, ali i nad stanjem hranilice ili svoga doma. Kako bi se taj cilj ispunio, tj. kako bi vlasnik imao što bolji pregled, kamera je pozicionirana na hranilicu na način da obuhvaća što veći dio prostorije u kojoj se nalazi. Na *slici 6.1* prikazan je položaj kamere na hranilici.

Korisnik pristupom na web stranicu hranilice (www.pametnahranilica.ddns.net) te klikom na tipku „*Live video*“ ima pregled nad većim dijelom prostorije u kojoj se hranilica nalazi. U idućem poglavlju objašnjen je način spajanja kamere te instaliranje *motion* programa na Raspberry Pi.

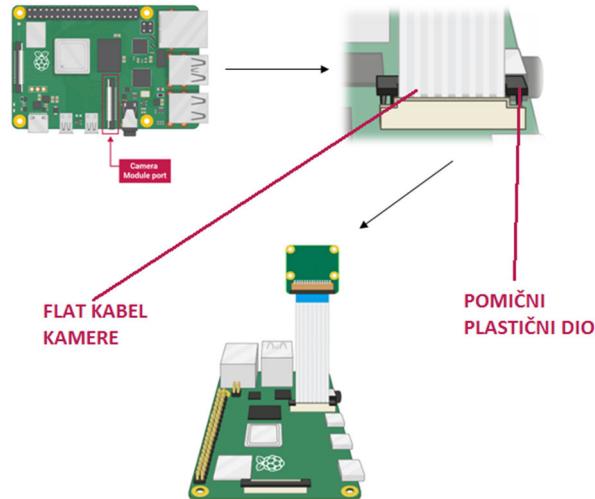


Slika 6.1 Pozicija kamere na hranilici

6.1 Spajanje kamere na Raspberry Pi

Prije spajanja kamere i njezinog korištenja potrebno je u postavkama Raspberry Pi omogućiti (engl. *enable*) sučelje kamere. Otvaranjem *Raspberry Pi Configuration* odjeljka u postavkama Raspberry Pi-a te klikom na *Interfaces* moguće je omogućiti sučelje kamere.

Postupak spajanja kamere na Raspberry Pi je prikazan na *slici 6.2*. Kamera modul se spaja na za nju predviđen dio na Raspberry Pi pločici. Podizanjem pomičnog plastičnog dijela umetne se flat kabel kamere te ponovnim spuštanjem plastičnog dijela kamera je uspješno spojena. Pri spajaju je potrebno pažljivo rukovati sa svime jer su dijelovi Raspberry Pi pločice jako osjetljivi.



Slika 6.2 Spajanje kamere na Raspberry Pi

6.2 Instaliranje *Motion* programa

Motion je program koji omogućava prikaz video signala s više vrsta kamera. Koristi se za razne zahtjeve kao što su npr. sigurnosne kamere, promatranje ptica, spremanje fotografija i videozapisa u baze podataka itd. Prije nego se kreće na instaliranje *Motion* programa otvori se terminal te naredbama *sudo apt-get update* i *sudo apt-get upgrade* provjeri se jesu li instalirana najnovija ažuriranja na OS. Tek nakon što su instalirana najnovija ažuriranja za sustav može se krenuti na instaliranje *Motion* programa.

Motion program nudi više različitih verzija, a pri realizaciji rada *Pametna hranilica za kućne ljubimce*, *motion* program preuzet je s popularne stranice [www.github.com](https://github.com) jer osim live stream-a nudi dodatne informacije kao npr. trenutni datum i vrijeme. Unosom naredbe *wget https://github.com/Motion-Project/motion/releases/download/release-4.2.2/pi_buster_motion_4.2.2-1_armhf.deb* [30] pokreće se instalacija *motion* programa. Kada je instalacija uspješno obavljena potrebno je modificirati *motion.cfg* datoteku. Izvršene su promjene prikazane ispod:

width 320 > width 640 – proširuje se slika.

height 240 > height 480 – povisuje se slika.

framerate 2 > framerate 20 – povećava se brzina prikaza slike.

stream_quality 50 > stream_quality 80 – povećava se kvaliteta slike.

stream_maxrate 1 > stream_maxrate 15 – povećava se broj sličica u sekundi.

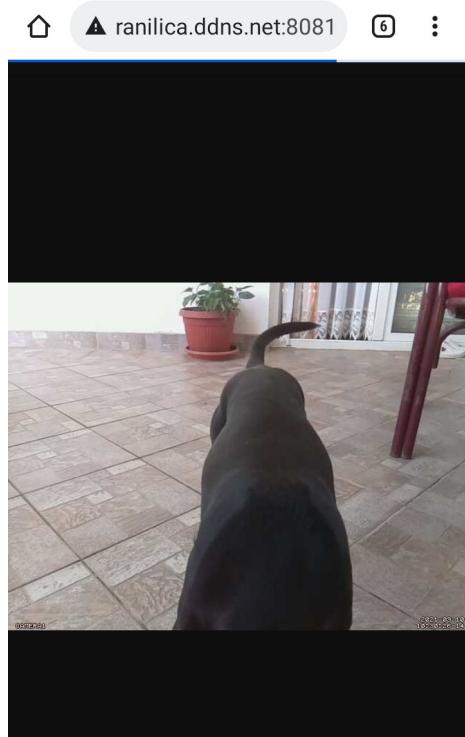
stream_localhost on > stream_localhost off – ukoliko je postavljen na *on*, samo localhost ima pristup web kamери.

daemon off > daemon on – omogućeno pokretanje u pozadini.

Pokretanjem naredbe *sudo nano /etc/modules* otvara se datoteka u koju je potrebno dodati na dno *bcm2835-v4l2* kako bi kamera modul bio prepoznat. Sada je samo potrebno aktivirati pokretanje instaliranog programa u pozadini. To se čini otvaranjem datoteke naredbom *sudo nano /etc/gefault/motion* i zamjenom linije *start_motion_daemon=no* s linijom *start_motion_daemon=yes*. Na kraju se servis pokreće naredbom *sudo service motion start*.

Kako bi se testiralo radi li livostream potrebno je unijeti IP adresu Raspberry Pi-a s pripadajućim portom 8081. Pošto je poslužitelj objavljen na Internetu pod domenom www.pametnahranilica.ddns.net unutar konfiguracijske datoteke *app.cfg* varijabli *Motion_Camera_Site_Address* potrebno je umjesto vrijednosti *localhost:8081* pridijeliti vrijednost <http://pametnahranilica.ddns.net:8081> kako bi livostream bilo moguće prikazati izvan privatne mreže. Datoteka *app.cfg* priložena je na dnu u prilogu B, a kreirana je unutar Python datoteke *createFiles.py* koja se nalazi u prilogu A.

Pokretanjem web aplikacije te klikom na tipku „*Live video*“ otvara se stranica kao na *slici 6.3*.



Slika 6.3 Livestream tijekom hranjenja pasa

7. KÔD ZA AUTOMATSKO HRANJENJE

Kôd za automatsko hranjenje napisan je u Python datoteci *automaticfeeding.py* koja se nalazi u prilogu H, a prikazana je i na *slici 7.1.*

```
# uvoz potrebnih modula
import RPi.GPIO as GPIO
import time
from datetime import datetime

def spin_hopper():
    GPIO.setmode(GPIO.BOARD) #omogućuje korištenje brojeva pinova
    GPIO.setwarnings(False) #onemogućujemo upozorenja
    GPIO.setup(11,GPIO.OUT) #pin 11 postavljamo kao izlazni
    servo=GPIO.PWM(11,50) #na pin 11 šaljemo PWM signal frekvencije 50 Hz
    servo.start(0) #uključuje servo motor, nema pulsa
    servo.ChangeDutyCycle(12) #rotacija za 180 stupnjeva
    servo.ChangeDutyCycle(12)
    time.sleep(2)

while True: #beskonačna petlja
    now=datetime.now() #saznaje trenutno vrijeme
    current_time=now.strftime("%H:%M:%S") #trenutno vrijeme u formatu sati,minute,sekunde
    vrijemehranjenja1='10:00:00'
    vrijemehranjenja2='14:00:00'
    if vrijemehranjenja1==current_time: #uspoređuje 1. vrijeme hranjenja s trenutnim vremenom
        spin_hopper()
    elif vrijemehranjenja2==current_time: #uspoređuje 2. vrijeme hranjenja s trenutnim vremenom
        spin_hopper()
```

Slika 7.1 Kôd za automatsko hranjenje

Kôd za automatsko hranjenje jako je sličan programskom kôdu za upravljanje servo motorom koji je objašnjen u potpoglavlju 5.4, samo je razlika što se ovaj kôd pokreće u pozadini pri pokretanju Raspberry Pi-a i okreće servo motor u točno određeno vrijeme, a prethodno spomenuti kôd se pokreće kada korisnik putem interneta pošalje zahtjev na URL „/feedbuttonclick“. Kao i kod kôda datoteke *commonTasks.py*, priložene u prilogu D, i ovdje je korištena *ChangeDutyCycle()* funkcija koja okreće osovinu motora za željeni kut.

Pokretanje kôda u pozadini potrebno je namjestiti na sljedeći način. Unutar */home/pi/.config* direktorija kreira se novi direktorij koji se nazove *autostart*. Unutar njega kreira se datoteka proizvoljnog naziva s ekstenzijom *.desktop*. Unutar te datoteke potrebno je dodati sljedeće dvije linije kôda:

[Desktop Entry]- definira konfiguracijsku datoteku koja opisuje kako se pokreće određeni program.

Exec = python3 /home/pi/Desktop/automaticfeeding.py – pokreće *automaticfeeding.py* skriptu.

8. IZRADA SKLOPOVSKOG DIJELA HRANILICE

Prilikom izrade sklo povskog dijela pametne hranilice korišteni su većinom dijelovi i alati koji su pronađeni u garaži. Od kupljenih dijelova tu je samo dozator za žitarice koji je naručen preko Interneta, prikazan na *slici 8.1*. Radi se o dozatoru marke *Vanora*, obujma 3,5 L.



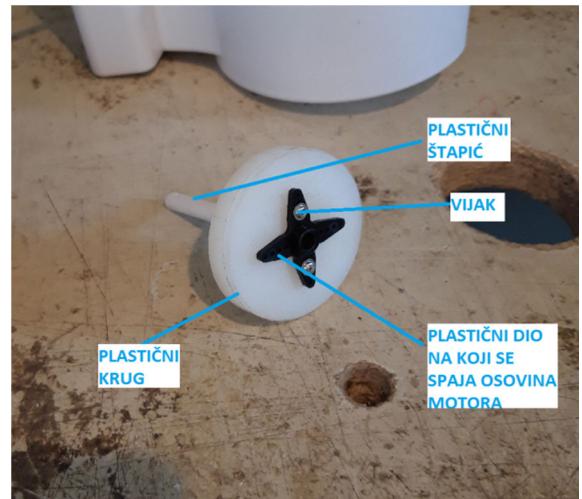
Slika 8.1 Dozator za žitarice *Vanora*

Dozator se, kao što se može vidjeti na *slici 8.1* sastoji od plastičnog bubenja u koji se stavlja hrana, plastičnog držača bubenja, plastičnog postolja za posudicu, gumenog kotača unutar bubenja koji je spojen s ručkom pomoću plastičnog štapića. Okretanjem tog gumenog kotača hrana ispada iz bubenja.

Proces izrade sklo povskog dijela hranilice započet je otklanjanjem ručke s ciljem da se plastični štapić spoji na plastični dio na koji će se pričvrstiti osovina servo motora. Ručka je od štapića odvojena pomoću ručne pile. Kako bi se omogućilo da se rotacijom servo motora okreće i gumeni dio korišten je deblji komad plastike u obliku kruga promjera 6 cm i debljine 1,5 cm prikazan na *slici 8.2*. Taj dio izrađen je u tvornici plastičnih i aluminijskih dijelova Zec d.o.o. Pomoću bušilice u središtu kruga izbušena je rupa u koju je postavljen plastični štapić te je lemilicom otopljen dio štapića i plastike kako bi štapić bio nepomičan unutar kruga. Zatim je na drugu stranu kruga pomoću sitnih vijaka pričvršćen plastični dio koji ide na osovinu servo motora. Konačni proizvod je prikazan na *slici 8.3*.

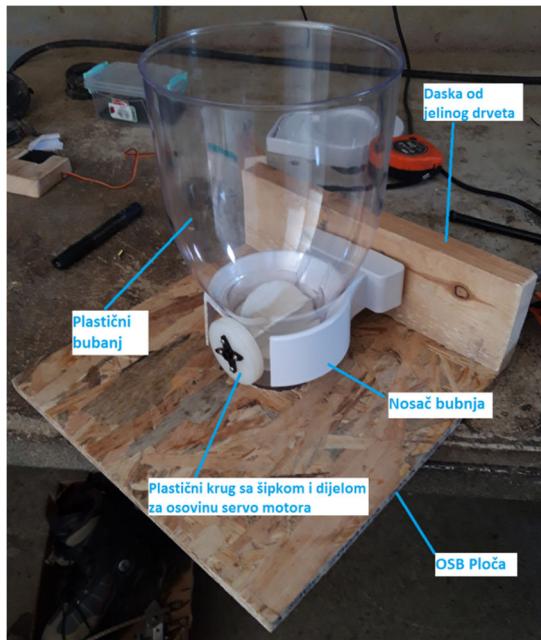


Slika 8.2 Plastični krug



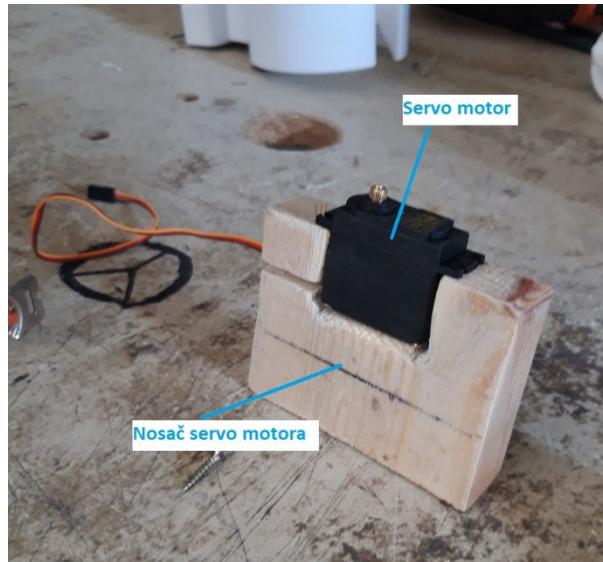
Slika 8.3 Plastični krug sa šipkom i dijelom za vratilo servo motora

Nakon toga započeta je izrada postolja za bubanj, električne komponente i servo motor. Na komadu OSB (engl. *Oriented Strand Board*) ploče pomoću električne pile za drvo izbušena je rupa koja odgovara rupi na dnu bubenja dozatora. Zatim je na OSB ploču pričvršćen komad daske od jelinog drveta te na njega pomoću matica i vijaka pričvršćen je nosač bubenja. Dio nosača bilo je potrebno skratiti jer bi inače plastični krug prilikom vrtnje zapinjao za nosač.



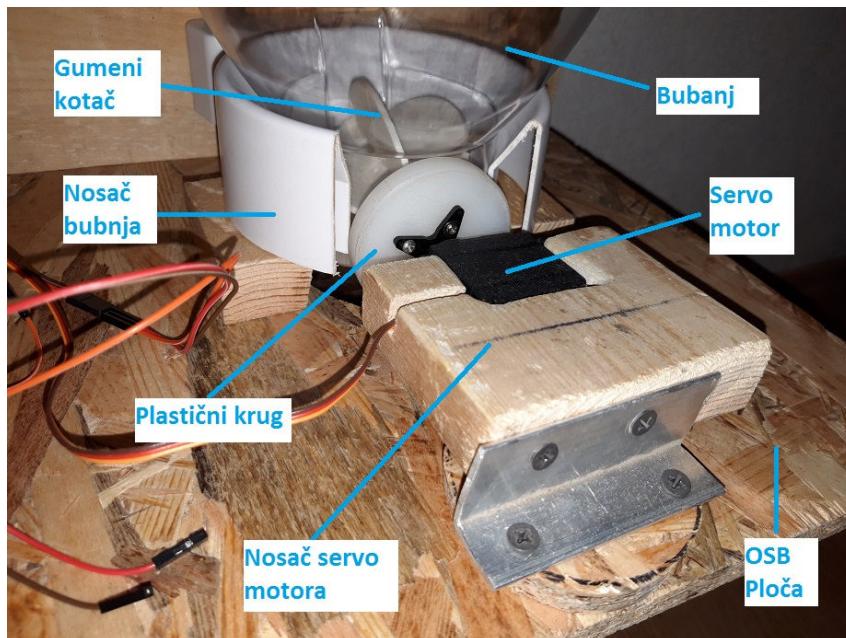
Slika 8.4 Postolje za dozator

Od jelinog drveta izrađen je nosač servo motora. Pomoću već spomenute električne pile za drvo izrezan je dio daske prema dimenzijama servo motora. Zatim je u taj izrezani dio stavljen servo motor te je pomoću vijaka pričvršćen za dasku. Sa strane je zarezan dio daske kako bi se kroz taj dio provukle žice servo motora. Servo motor pričvršćen na nosač je prikazan na slici 8.5.



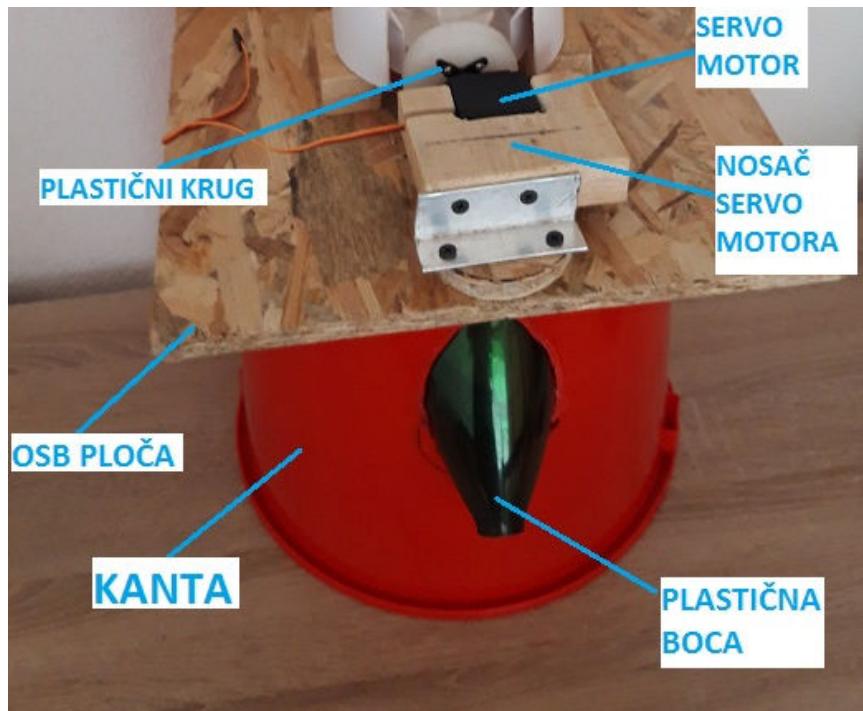
Slika 8.5 Nosač servo motora sa servo motorom

Nosač servo motora zajedno sa servo motorom postavljen je na OSB ploču i spojen sa plastičnim krugom kako je prikazano na *slici 8.6.*



Slika 8.6 Nosač servo motora pričvršćen za OSB ploču i spojen s plastičnim krugom

OSB ploča pričvršćena je na plastičnu kantu kojoj je prethodno izrezano dno. Na jednom dijelu kante izbušena je rupa kroz koju će ispadati hrana. Korištene su dvije plastične boce koje su izrezane tako da je moguće povezati točku A tj. otvor bubnja dozatora kroz koji ispada hrana i točku B tj. otvor na kanti kroz koji ispada hrana u posudu za hranjenje kućnog ljubimca. Prva boca klamericom je pričvršćena oko otvora na OSB ploči te je za nju pričvršćena druga boca čiji kraj izlazi kroz otvor kante kao što je vidljivo na *slici 8.7.*



Slika 8.7 OSB ploča pričvršćena na kantu

Na OSB ploču je pomoću vijaka pričvršćena plastična posudica u koju je smješteno računalo Raspberry Pi. Sa strane posudice napravljen je otvor kako bi se Raspberry mogao priključiti na napajanje. Posudica sadrži poklopac koji je moguće odvojiti od same posude. Na poklopcu su napravljene dvije rupe kroz koje su provučene žice servo motora i spojene na GPIO pinove, a kroz drugu rupu je provučen flat kabel kamere koja je pričvršćena na bubenj kamere. Konačni izgled hranilice prikazan je na *slici 8.8*.



Slika 8.8 Konačni izgled hranilice



Slika 8.9 Hranjenje psa

9. ZAKLJUČAK

Cilj rada bila je realizacija pametne hranilice koja ima dva načina rada: automatsko i hranjenje s udaljene lokacije. Automatsko hranjenje podrazumijeva hranjenje ljubimca u određeni trenutak svaki dan. Hranjenje s udaljene lokacije omogućeno je putem web aplikacije koja je razvijena korištenjem Python-a, Flask-a, HTML-a, CSS-a, HTTP metoda i Apache HTTP web poslužitelja. Osim hranjenja na web aplikaciji je moguće putem livestream-a dobiti uvid o kućnom ljubimcu i stanju hranilice.

Realizacijom rada stečena su osnovna znanja o Raspberry Pi računalu, načinu rada i načinu spajanja komponenti na GPIO pinove Raspberry Pi-a. Osim toga svrha rada je bila bolje se upoznati s razvojem web aplikacije te shvatiti način na koji funkcioniра postavljanje razvijenih aplikacija na Internet tako da bi bile dostupne izvan privatnih mreža. Potrebno je bilo upoznati se s funkcionalnostima servo motora i Raspberry Pi kamera modula kako bi uspješno obavljali potrebnu zadaću. Stečeno je i bolje znanje o pisanju programskog kôda kao i razumijevanje programerske logike te znanja i vještine izrade mehaničkog dijela hranilice.

9.1 Budućnost

Kao daljnje poboljšanje sustava predlaže se korištenje 3D printera za izradu sklopovskog dijela hranilice. Time bi hranilica dobila na izgledu/dizajnu te bi se otvorila mogućnost masovne proizvodnje, a komponente bile manje izložene mogućim oštećenjima. Također, pošto je pri realizaciji ovog projekta korišteno snažno računalo, bila bi stvorena neka baza podataka u koju bi se spremali podaci o hranjenju ili na primjer videozapisi kada bi kamera detektirala pokret tako da vlasnik ima potpuni nadzor nad svojim ljubimcem. Kao još jedno poboljšanje može se predložiti hranjenje ljubimca korištenjem aplikacije Google Assistant. Time bi se ljudima koji rade poslove na kojima ne smiju ili nemaju vremena pretraživati po mobitelu omogućilo hranjenje ljubimca jednostavno izgovorom dviju riječi, a da pri tome ni ne dotaknu mobitel.

Kao osnovni nedostatak projekta Pametna hranilica za kućne ljubimce može se navesti cijena čitavog sklopa. Za rad je ukupno potrošeno oko 1.000 HRK, a danas je pametne hranilice na internetu moguće uzeti za manje od 500 HRK. Razlog visoke cijene ovog projekta je Raspberry Pi 4 Model B računalo koje bi moglo biti zamijenjeno nekim jeftinijim mikrokontrolerom poput Arduina ili ESP8266 mikrokontrolera. Kao nedostatak može se

navesti i izostanak ventilatora. Hranilica treba konstantno biti upaljena, a to bi dovelo do pregrijavanja Raspberry Pi-a. Ubacivanjem ventilatora u sustav, rizik od pregrijavanja bio bi uvelike smanjen. Također, mogao se koristiti i snažniji servo motor ili koračajni motor jer korištenjem TowerPro MG955 servo motora ograničena je količina hrane koja se stavlja u spremnik. Ako se stavi previše hrane, motor neće imati snage zakrenuti kotač te može doći do uništenja.

LITERATURA

- [1] „The 20 best Smart Home Pet Products (and 4 Apps) for a Better Life With Your Pet“, s Interneta, <https://www.rover.com/blog/best-pet-tech-devices-dogs/>, preuzeto 16. kolovoza 2021.
- [2] S interneta, <https://www.ubuy.co.id/en/product/FDODWAG-leekooluu-q01-hd-1080p-camera-wifi-smart-feeder-6l-automatic-cat-feeder-automatic-dog-feeder-timer-p>, preuzeto 8. rujna 2021.
- [3] S interneta, <https://www.mytrendyphone.ie/shop/automatic-fish-feeder-dispenser-270673p.html>, preuzeto 8. rujna 2021.
- [4] S Interneta, <http://www.feed-smart.com/>, preuzeto 8. rujna 2021.
- [5] Simon Monk: „Programming the Raspberry Pi™ Getting started with Python“, McGraw- Hill, Sjedinjene Američke Države, 2013.
- [6] „Raspberry Pi 4 Model B Datasheet“, s Interneta, <https://datasheets.raspberrypi.org/rpi4/raspberry-pi-4-datasheet.pdf>, preuzeto 17.kolovoza 2021.
- [7] „Simple Guide to the Raspberry Pi GPIO Header“, s Interneta, <https://www.raspberrypi-spy.co.uk/2012/06/simple-guide-to-the-rpi-gpio-header-and-pins/>, preuzeto 17. kolovoza 2021.
- [8] S Interneta, <https://www.the-diy-life.com/gpio-pinout-diagram-2/>, preuzeto 17.kolovoza 2021.
- [9] „Raspbian – Raspberry Pi OS Guide“, s Interneta, <https://www.pcwld.com/raspbian-raspberry-pi-os-guide>, preuzeto 17. kolovoza 2021.
- [10] „20 Best Operating Systems You Can Run on Raspberry Pi in 2021“, s Interneta, <https://www.fossmint.com/operating-systems-for-raspberry-pi/>, preuzeto 8.rujna 2021.
- [11] S Interneta, <https://www.raspberrypi.org/blog/raspberry-pi-imager-imaging-utility/>, prevezeto 18. kolovoza 2021.
- [12] „What is Apache? What is web server?“, s Interneta, <https://www.wpbeginner.com/glossary/apache/>, preuzeto 19. kolovoza 2021.
- [13] „What is the most popular web server?“, s Interneta, <https://digitalintheround.com/what-is-the-most-popular-web-server/>, preuzeto 8. rujna 2021.

- [14]S Interneta, <https://ubiq.co/tech-blog/wp-content/uploads/2020/04/Apache-Default-Page-on-Debian-Ubuntu.png>, preuzeto 19. kolovoza 2021.
- [15],“Virtualenv and venv: Python virtual environments explained”, s Interneta, <https://www.infoworld.com/article/3239675/virtualenv-and-venv-python-virtual-environments-explained.html>, preuzeto 20. kolovoza 2021.
- [16],“Što je Flask?”, s Interneta, <https://medium.com/@nikovrdoljak/uvod-u-flask-3859458237f7>, preuzeto 20. kolovoza 2021.
- [17],“Flask Documentation”, s Interneta, <https://flask.palletsprojects.com/en/1.1.x/api/>, preuzeto 20. kolovoza 2021.
- [18]S Interneta, <https://hr.wikipedia.org/wiki/HTML>, preuzeto 20. kolovoza 2021.
- [19]S Interneta, <https://hr.wikipedia.org/wiki/CSS>, preuzeto 20.kolovoza 2021.
- [20],“Flask HTTP methods, handle GET & POST requests”, s Interneta, <https://pythonbasics.org/flask-http-methods/>, preuzeto 21. kolovoza 2021.
- [21]S Interneta, <https://pypi.org/project/mod-wsgi/>, preuzeto 22. kolovoza 2021.
- [22]S Interneta, <https://modwsgi.readthedocs.io/en/master/user-guides/quick-configuration-guide.html>, preuzeto 10. rujna 2021.
- [23],“Što je port forwarding?”, s Interneta, <https://bs.eyewated.com/sto-je-port-forwarding-kako-da-postavim-svoje/>, preuzeto 23. kolovoza 2021.
- [24],“Što je DDNS i kako funkcionira?”, s Interneta, <https://hr.eyewated.com/sto-znaci-dinamicki-dns/>, preuzeto 23. kolovoza 2021.
- [25],“What is servo motor and how it works?”, s Interneta, <https://realpars.com/servo-motor/>, preuzeto 26. kolovoza 2021.
- [26],“How Servo motor works?”, s Interneta, <https://www.jameco.com/Jameco/workshop/Howitworks/how-servo-motors-work.html>, preuzeto 26. kolovoza 2021.
- [27]S Interneta, <https://www.electronicscomp.com/mg995-metal-gear-servo-motor-360-degree-rotation>, preuzeto 27. kolovoza 2021.
- [28]S Interneta, http://www.python-exemplary.com/drucken.php?inhalt_mitte=raspi/en/servomotors.inc.php, preuzeto 9. rujna 2021.
- [29]S Interneta, https://motion-project.github.io/4.3.2/motion_config.html#basic_setup_picam, preuzeto 28. kolovoza 2021.

[30] S Interneta, <https://github.com/Motion-Project/motion/issues/979>, preuzeto 28. kolovoza 2021.

POPIS OZNAKA I KRATICA

WEB - World Wide Web

Wi-Fi - Wireless Fidelity

OS - Operating System

HTTP - Hypertext Transfer Protocol

IoT - Internet of Things

SD - Secure Digital

USB - Universal Serial Bus

HDMI - High-Definition Multimedia Interface

LAN - Local Area Network

PoE - Power over Ethernet

ARM - Advanced RISC Machines

SoC - System on a Chip

IEEE - Institute of Electrical and Electronics Engineers

GPIO - General-Purpose Input/Output

DSI - Display Serial Interface

CSI - Camera Serial Interface

V – volt

URL - Uniform Resource Locator

A - amper

DC - Direct Current

AC - Alternating Current

LED - Light Emitting Diode

LXDE - Lightweight X11 Desktop Environment

TV - TeleVision

UNIX - UNiplexed Information Computing System

DHCP - Dynamic Host Configuration Protocol

HTML - HyperText Markup Language

CSS - Cascading Style Sheets

WSGI - Web Server Gateway Interface

DNS - Domain Name System

DDNS - Dynamic Domain Name System

WWW - World Wide Web

IP - Internet Protocol

ms - milisekunda

kg - kilogram

cm - centimetar

s - sekunda

3D - Three Dimensional

MMC - Multimedia Card

OSB - Oriented Strand Board

SAŽETAK

Cilj rada je realizacija pametne hranilice za kućne ljubimce. Projekt je odrđen tako da ima dva načina hranjenja: automatsko hranjenje koje u točno određeno vrijeme svaki dan dozira određenu količinu hrane i daljinsko hranjenje, tj. korisnik pristupi web stranici i klikom na tipku dozira hranu. Osim hranjenja vlasniku hranilice omogućen je nadzor nad kućnim ljubimcem, stanjem hranilice i svoga doma pritiskom na odgovarajuću tipku na istoj web stranici.

Razvijeni sustav je funkcionalan i obavlja sve predviđene radnje. Unatoč tome, postoji potreba za unaprjeđenjem tj. hranilica bi izgledom mogla biti robusnija i vizualno ljepša, a cjenovno bi mogla biti pristupačnija.

KLJUČNE RIJEČI

Pametni uređaji za kućne ljubimce, Pametna hranilica za kućne ljubimce, Raspberry Pi, servo motor, kamera, web aplikacija

SMART PET FEEDER

ABSTRACT

The goal of this project is the realization of the smart pet feeder. The project was done in a way that it has two types of feeding: automatic feeding which releases a certain amount of food at the same exact time every day and remote feeding, i.e. user accesses the web page and by clicking on the button doses the food. Except feeding, surveillance of the pets, state of feeder, and his home are allowed for the owner by clicking on the appropriate button on the same website.

The developed system performs all the intended actions and the result is functional, but there is a need for improvement. Namely, a feeder could look more robust and visually more appealing, and price-wise it could be more economic.

KEY WORDS

Smart pet devices, Smart pet feeder, Raspberry Pi, servo motor, camera, web application

PRILOZI

PRILOG A

createFiles.py

```
#!/var/www/feeder/bin/python
import sys

sys.path.extend(['/var/www/feeder/feeder'])
import os


try:
    appCFGPath = '/var/www/feeder/feeder/app.cfg'

    if os.path.isfile(appCFGPath):
        print('app.cfg already exists. To create again first delete current co
py')
    else:
        print('Creating app.cfg. Please wait.')
        f = open(appCFGPath, "w+")

        f.write("""[feederConfig]
Hopper_GPIO_Pin=11
Hopper_Spin_Time=0.6
Motion_Camera_Site_Address=http://yourRemoteAddress.duckdns.org:8081
Seconds_Delay_After_Button_Push=3
Secretkey=SUPER_SECRET_KEY
""")

        f.close()
        # os.chmod(appCFGPath, 0o777)
        print('app.cfg created')

except Exception as e:
    print('Error: ' + str(e))
```

PRILOG B

app.cfg

```
[feederConfig]
Hopper_GPIO_Pin=11
Hopper_Spin_Time=0.6
Motion_Camera_Site_Address=http://pametnahranilica.ddns.net:8081
Seconds_Delay_After_Button_Push=3
Secretkey=SUPER_SECRET_KEY
```

PRILOG C

app.py

```
from flask import Flask, redirect, render_template, request, url_for, flash
import subprocess
import commonTasks
import os
import configparser
import os, sys, time

app = Flask(__name__)

# Find config file
# dir = os.path.dirname(__file__) # os.getcwd()
# configFilePath = os.path.abspath(os.path.join(dir, "app.cfg"))
configParser = configparser.RawConfigParser()
configParser.read('/var/www/feede
r/feede
r/app.cfg')

# Read in config variables
SECRETKEY = str(configParser.get('feederConfig', 'Secretkey'))
hopperGPIO = str(configParser.get('feederConfig', 'Hopper_GPIO_Pin'))
hopperTime = str(configParser.get('feederConfig', 'Hopper_Spin_Time'))
motionCameraSiteAddress = str(configParser.get('feederConfig', 'Motion_Camera_Site_Address'))

@app.route('/', methods=['GET', 'POST'])
def home_page():
    try:
        cameraStatusOutput = DetectCamera()

        # cameraStatusOutput = 'supported=0 detected=1'
        if "detected=1" in str(cameraStatusOutput):
            cameraStatus = '1'
        else:
            cameraStatus = '0'

        # Return page
        return render_template('home.html', cameraSiteAddress=motionCameraSiteAddress, cameraStatus=cameraStatus)

    except Exception as e:
        return render_template('error.html', resultsSET=e)

@app.route('/feedbuttonclick', methods=['GET', 'POST'])
def feedbuttonclick():
    try:
```

```

spin = commonTasks.spin_hopper(hopperGPIO, hopperTime)

if spin != 'ok':
    flash('Error! No feed activated! Error Message: ' + str(spin), 'error')
    return redirect(url_for('home_page'))

    flash('Hranjenje uspješno! :)')
    return redirect(url_for('home_page'))
except Exception as e:
    return render_template('error.html', resultsSET=e)

@app.route('/video', methods=['GET', 'POST'])

def DetectCamera():
    try:

        process = subprocess.Popen(["vcgencmd", "get_camera"],
                                  stdout=subprocess.PIPE,
                                  stderr=subprocess.STDOUT)
        return process.stdout.read()
    except Exception as e:
        return 'status=0'

app.secret_key = SECRETKEY

if __name__ == '__main__':
    app.debug = False
    app.run(host='0.0.0.0', threaded=True)

```

PRILOG D

commonTasks.py

```
#uvoz potrebnih biblioteka
import configparser
import time
import os
import RPi.GPIO as GPIO

#Pronalazi konfiguracijsku datoteku
dir = os.path.dirname(__file__) # os.getcwd()
configFilePath = os.path.abspath(os.path.join(dir, "app.cfg"))
configParser = configparser.RawConfigParser()
configParser.read(configFilePath)

# Čita varijable iz konfiguracijske datoteke
hopperGPIO = str(configParser.get('feederConfig', 'Hopper_GPIO_Pin'))
hopperTime = str(configParser.get('feederConfig', 'Hopper_Spin_Time'))

def spin_hopper(pin, duration): #funkcija za okretanje servo motora
    try:
        pin=int(pin) #kontrolni pin (11)
        duration=float(duration)
        GPIO.setwarnings(False)
        GPIO.cleanup(pin)
        GPIO.setmode(GPIO.BOTH) #omogućuje korištenje brojeva pinova
        GPIO.setup(pin, GPIO.OUT) #postavlja pin 11 kao izlazni, onaj na koji
        se šalje PWM signal
        pwm=GPIO.PWM(pin,50) #50 je frekvencija PWM signala
        pwm.start(50) #šalje se PWM signal
        pwm.ChangeDutyCycle(12) #rotacija za 180 stupnjeva
        pwm.ChangeDutyCycle(12)
        time.sleep(duration) #čeka neko vrijeme
        pwm.start(0) #prestaje se slati PWM signal, zaustavlja se rotacija
        GPIO.cleanup(pin)

    return 'ok'
    except Exception as e:
        return 'ok' # e
```

PRILOG E

feeder.wsgi

```
import os, sys

activate_this = '/var/www/feeder/bin/activate_this.py'
#execfile(activate_this, dict(__file__=activate_this))
exec(compile(open(activate_this, "rb").read(), activate_this, 'exec'))

sys.path.append('/var/www/feeder/feeder')

from app import app as application

home='/var/www/feeder/feeder'
```

PRILOG F

error.html

```
{% extends "layout.html" %}

{% block head %}
{{ super() }}
{% endblock %}

{% block main %}
<h1 class="page-header">ERROR</h1>
<br>
<h3>{{resultsSET}}</h3>
<br>
<button class="btn btn-primary btn-lg" onclick="goBack()">&lquo; Go Back</button>
<script>
function goBack() {
window.history.back();
}
</script>
{% endblock %}
```

home.html

```
{% extends "layout.html" %}

{% block head %}
{{ super() }}
{% endblock %}

{% block main %}

<div class="row">
    {% if cameraStatus == '1' %}
        <div class="col-md-6 text-center">
            <form role="form" name="feedbuttonclick" method="post" action="/feedbuttonclick">
                <button class="btn btn-lg feedButton" type="submit">Nahrani sada</button>
            </form>
        </div>

        <div class="col-md-6 text-center">
            <br>
            <form role="form" action="{{cameraSiteAddress}}">
                <button class="btn btn-lg camButton" type="submit">Live Video</button>
            </form>
        </div>
    {% else %}
        <div class="col-md-12 text-center">
            <form role="form" name="feedbuttonclick" method="post" action="/feedbuttonclick">
                <button class="btn btn-lg feedButton" type="submit">Nahrani sada</button>
            </form>
        </div>
    {% endif %}
</div>
{% endblock %}
```

```

<!DOCTYPE html>
<html lang="en">
<head>
    {% block head %}
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="description" content="">
    <meta name="author" content="">

    <title>Pametna hranilica</title>

    <link rel="icon" href="/static/images/titleIcon.png">

    <link href="static/css/bootstrap.min.css" rel="stylesheet">
    <link href="static/css/googleFonts.css" rel="stylesheet" type="text/css">
    <link href="static/css/feedButton.css" rel="stylesheet">
    <link href="static/css/camButton.css" rel="stylesheet">
    <link href="static/css/navBar.css" rel="stylesheet">

    {% endblock %}
</head>

<body>
<div class="container">

    <nav class="navbar navbar-default">
        <div class="navbar-header">
            <p class="navbar-brand1 navbar-
brand navBarTitle">&nbsp;&nbsp;Hranilica </p>
            
        </div>
    </nav>

    <div>
        {% with messages = get_flashed_messages(with_categories=true) %}
        {% if messages %}
            {% for category, message in messages %}
                {% if category == 'warning' %}
                    <div class="alert alert-warning"> {{ message }} </div>
                {% elif category == 'error' %}
                    <div class="alert alert-danger"> {{ message }} </div>
                {% else %}
                    <div class="alert alert-success"> {{ message }} </div>
                {% endif %}
            {% endfor %}
        {% endif %}
    </div>

```

```
    {% endif %}  
    {% endwith %}  
    </div>  
  
    {% block main %}  
    {% endblock %}  
  
    {% block footer %}  
    {% endblock %}  
</div>  
  
</body>  
</html>
```

PRILOG G

feedButton.css

```
.feedButton {  
width:7em; height:3em;  
border-top: 1px solid #919191;  
background: #0e4724;  
color: #ffffff;  
font-size: 40px;  
}  
.feedButton:hover {  
border-top: 1px solid #919191;  
background: #0e4724;  
color: #ffffff;  
}
```

navBar.css

```
.navbar-brand1  
{  
position: absolute;  
width: 100%;  
left: 0;  
text-align: center;  
margin: 0 auto;  
  
}  
  
.navBarTitle {  
font-size: 40px;  
padding: 25px;  
  
}
```

camButton.css

```
.camButton {  
width:7em; height:3em;  
border-top: 1px solid #919191;  
background: #27ab5a;  
color: #ffffff;  
font-size: 24px;  
text-align: center;
```

```
    }
.camButton:hover {
border-top: 1px solid #919191;
background: #27ab5a;
color: #ffffff;
}
```

PRILOG H

automaticfeeding.py

```
# uvoz potrebnih modula
import RPi.GPIO as GPIO
import time
from datetime import datetime

def spin_hopper():
    GPIO.setmode(GPIO.BOARD) #omogućuje korištenje brojeva pinova
    GPIO.setwarnings(False) #onemogućujemo upozorenja
    GPIO.setup(11,GPIO.OUT) #pin 11 postavljamo kao izlazni
    servo=GPIO.PWM(11,50) #na pin 11 šaljemo PWM signal frekvencije 50 Hz
    servo.start(0) #uključuje servo motor, nema pulsa
    servo.ChangeDutyCycle(12) #rotacija za 180 stupnjeva
    servo.ChangeDutyCycle(12)
    time.sleep(2)

while True: #beskonačna petlja
    now=datetime.now() #saznaje trenutno vrijeme
    current_time=now.strftime("%H:%M:%S") #trenutno vrijeme u formatu sati,minuti,sekunde
    vrijemehranjenja1='10:00:00'
    vrijemehranjenja2='14:00:00'
    if vrijemehranjenja1==current_time: #uspoređuje 1. vrijeme hranjenja s trenutnim vremenom
        spin_hopper()
    elif vrijemehranjenja2==current_time: #uspoređuje 2. vrijeme hranjenjas s trenutnim vremenom
        spin_hopper()
```