Sustav za automatiziranu ishranu životinja

Horvat, Adrian

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni preddiplomski studij računarstva

SUSTAV ZA AUTOMATIZIRANU ISHRANU ŽIVOTINJA

Završni rad

Adrian Horvat

Osijek, 2024.

SADRŽAJ

1. UVOD

U modernom vremenu, razna industrijska područja se susreću s brojnim izazovima poput potrebe za povećanjem produktivnosti, rastućim troškovima rada, potrebom za smanjenje ljudskog faktora radi efikasnosti ili sigurnosti te brojnim drugim razlozima. Kao jedan od glavnih faktora za rješavanje navedenih izazova koristi se automatizacija. Automatizacija u dosta slučajeva izbacuje dio ljudskog faktora te povećava efikasnost i sigurnost nudeći optimizaciju izvršavanja određenih zadataka, konkretno, obavlja prijenos rada čovjeka na strojeve kroz različite računalne instrukcije i algoritme.

Automatizirana ishrana životinja ima široku rasprostranjenost. Koriste se razne metode i tehnologije kako bi se realizirala automatizitana ishrana životinja. Ključno je imati konzinstentu i preciznu isporuku hrane te zaštiti hranu u spremnicima od vanjskih utjecaja. Pri projektiranju hranilica bitno je definirati tip životinja za koje je hranilica predviđena, vanjske uvjete i specifične potrebe poput veličine farme pri izradi hranilica za domaće životinje te financijskih resursa. Definiranje vrste životinja pokriva drugačija ponašanja životinja i potrebe za konzumiranjem hrane, krupnije domaće životinje konzumiraju veće količine hrane, brzo i agresivno, stoga hranilica mora biti robusno dizajnirana. Suprotno tome, manje životinje imaju manje potrebe za konzumacijom hrane te nema pretjerane potrebe za robusnom izradom kao u prošlo navedenom primjeru. Drugi faktor je obratiti pozornost na lokaciju gdje se hranilica planira postaviti, odnosno njena zaštita od vanjskih utjecaja. Važno je uzeti u obzir temperaturu i vlažnost područja. Hranilice moraju biti otporne na kišu i vlagu kako bi se sprječilo kvašenje hrane, materijali trebaju biti otporni na niske temperature kako bi se sprječilo smrzavanje hrane. Najjednostavniji oblik hranilice za životinje je gravitacijska hranilica koja koristi gravitaciju kako bi isporučila hranu, jednostavne su za održavanje i korištenje. Složeniji oblici hranilica koriste senzore ili kameru za detekciju prisutnosti životinja što osigurava da hrana bude isporučena pri prisutnosti životinje, hranilice s vremenskom odgodom izbacivanja hrane osiguravaju vremenski kontroliranu isporuku hrane te jedne od najsloženijih hranilica su robotske hranilice koje same vrše isporuku hrane na različitim lokacijama područja definiranog za ishranu životinja.

Motiv završnog rada za izradu automatizirane hranilice je osiguranje pravilne prehrane za kućne ljubimce poput mačaka i pasa na području gdje nije prisutan ljudski faktor. U nekim slučajevima vlasnici nisu u mogućnosti hraniti životinje radi odsutnosti s područja na kojem se životinje nalaze, također hranilice mogu rezultirati uštedom vremena.

Cilj završnog rada je napraviti pouzdanu automatizitanu hranilicu za automatsko hranjenje životinja koje radi neovisno o ljudskom faktoru.

1.1. Zadatak i struktura završnog rada

Zadatak završnog rada je izraditi automatiziranu hranilicu za životinje koja će na temelju vrste i veličine životinje ispravno dozirati hranu te pomoću IoT sustava omogućiti nadzor i uvid u količinu potrošene hrane, uvid u detektirane živovtinje te njihovo prebrojavanje. Uvod opisuje postojeća rješenja vezana za ovaj projekt i detalje o njegovim različitim rješenjima. Drugo poglavlje opisuje teorijsku podlogu i prijedlog programskog rješenja za izradu automatizirane hranilice za životinje. Treće poglavlje opisuje realizaciju konstrukcijskog i sklopovnog dijela, korištene komponente u izradi te programsko rješenje koje je vezano za detekciju životinja i načinom doziranja. Četvrto poglavlje opisuje metodologiju testiranja koja će se izvršavati s pravim životinjama ili zbog njihove moguće odsutnosti s isprintanim slikama životinja te rezultate testiranja.

2. SUSTAV ZA AUTOMATIZIRANU ISHRANU ŽIVOTINJA NA PRINCIPU RAČUNALNOG VIDA

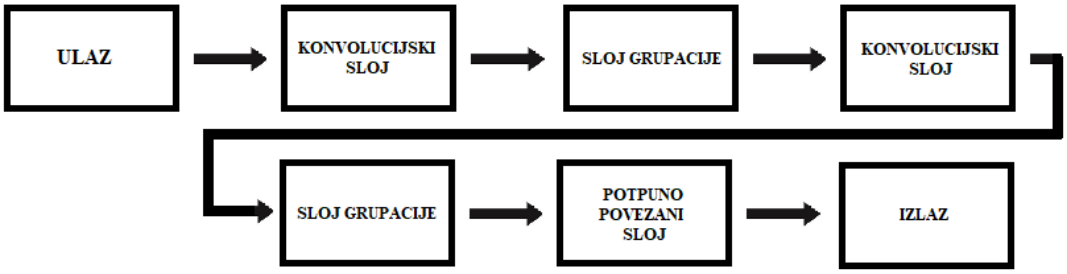
2.1. Teorijski osvrt na rješavanje zadatka

Postoje razna rješenja za realizaciju automatizirane hranilice za životinje. Jedno od rješenja temelji se na korištenju senzora koji pri detekciji pokreta izbace hranu van kao što je primjer u [1]. Također, popularno rješenje su hranilice s vremenskom odgodom izbacivanja hrane koje osiguravaju vremenski kontroliranu isporuku hrane kao što je navedeno u [2]. Realizirana automatizirana hranilica za životinje se temelji na kombinaciji automatizacije mehaničkog sustava i principima iz područja računalnog vida. Hrana se nalazi u spremniku čije izbacivanje hrane regulira impeler s elastičnim lopaticama računalnim instrukcijama nakon dobivenih informacija s kamere. U ovom poglavlju će se obraditi osnovne teorijske principe funkcioniranja navedenoga.

Računalni vid je znanstveno računalno područje koje omogućava računalu da interpretira vizualne informacije. Stavke računalnog vida su snimanje slike, obrada iste te prepoznavanje objekata. Kamera postavljena na hranilicu kontinuirano snima slike, zatim algoritmi dubokog učenja identificiraju vrstu životinje te se šalju informacije na računalo koje daje daljnje instrukcije sklopovlju za doziranje hrane. Duboko učenje je područje strojnog učenja koja koristi umjetne neuronske mreže koje se sastoje od slojeva za obradu podataka. Algoritmi prepoznavanja objekata se temelje na principima linearne algebre i optimizacije poput konvolucije. Konvolucijska neuronska mreža je vrsta dubokog modela neuronske mreže koji se koristi za analizu vizualnih podataka.

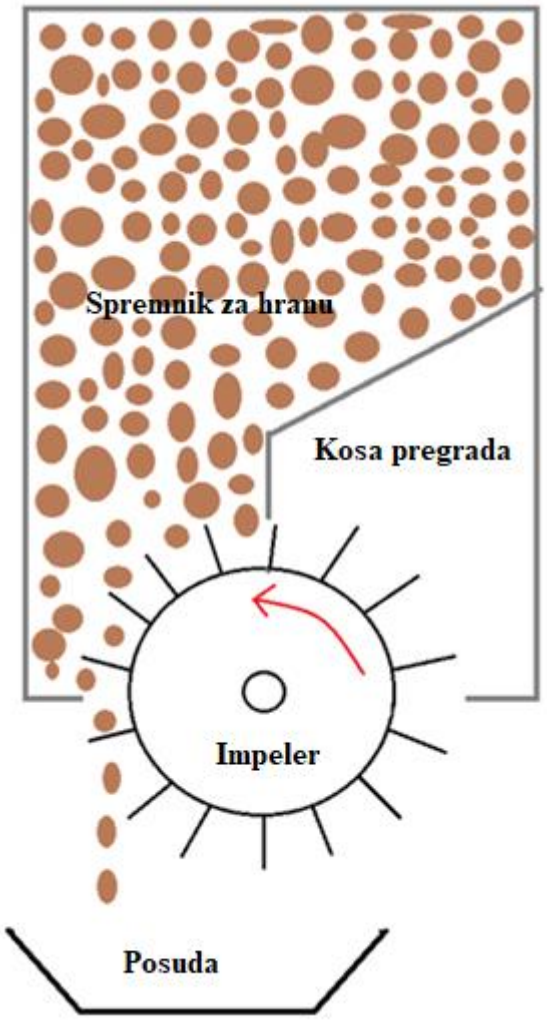
Slojevi konvolucijske neuronske mreže (prikazano na slici 2.1):

- konvolucijski sloj
- sloj za grupiranje
- potpuno povezani sloj



Slika 2.1. *Struktura konvolucijske neuronske mreže.*

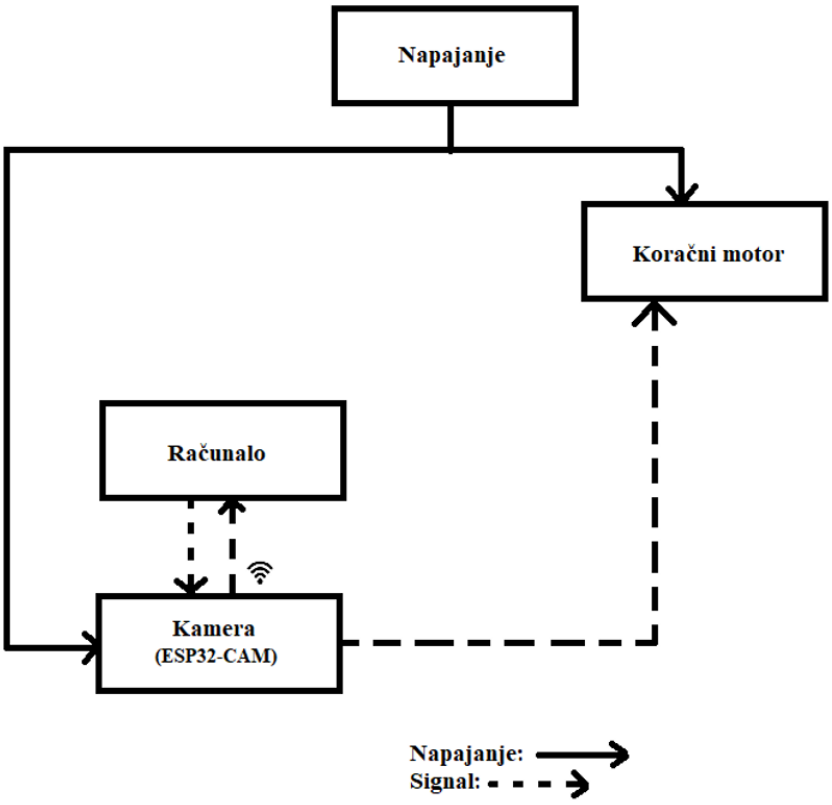
Mehanički sustav za isporuku hrane temelji se na rotirajućem okomito postavljenom impeleru s elastičnim perima. Mehanički princip rotacije impelera omogućuje ravnomjerno zahvaćanje hrane i njeno dozirano izbacivanje. Zatim pod utjecajem gravitacijske sile hrana pada u posudu ispod hranilice. Iznad polovice impelera koja pri rotaciji ima smjer kretanja prema gore, postavljena je kosa pregrada kako bi se spriječila sila pritiska hrane na taj dio impelera i smanjilo opterećenje motora koji bi morao tom polovicom podizati svu hranu u spremniku, a drugom spuštati hranu van kao što se vidi na slici 2.2. Time je povećana učinkovitost motora i smanjenje potrošnje energije.



Slika 2.2. *Prikaz smanjenja sile pritiska hrane na dio* impelera.

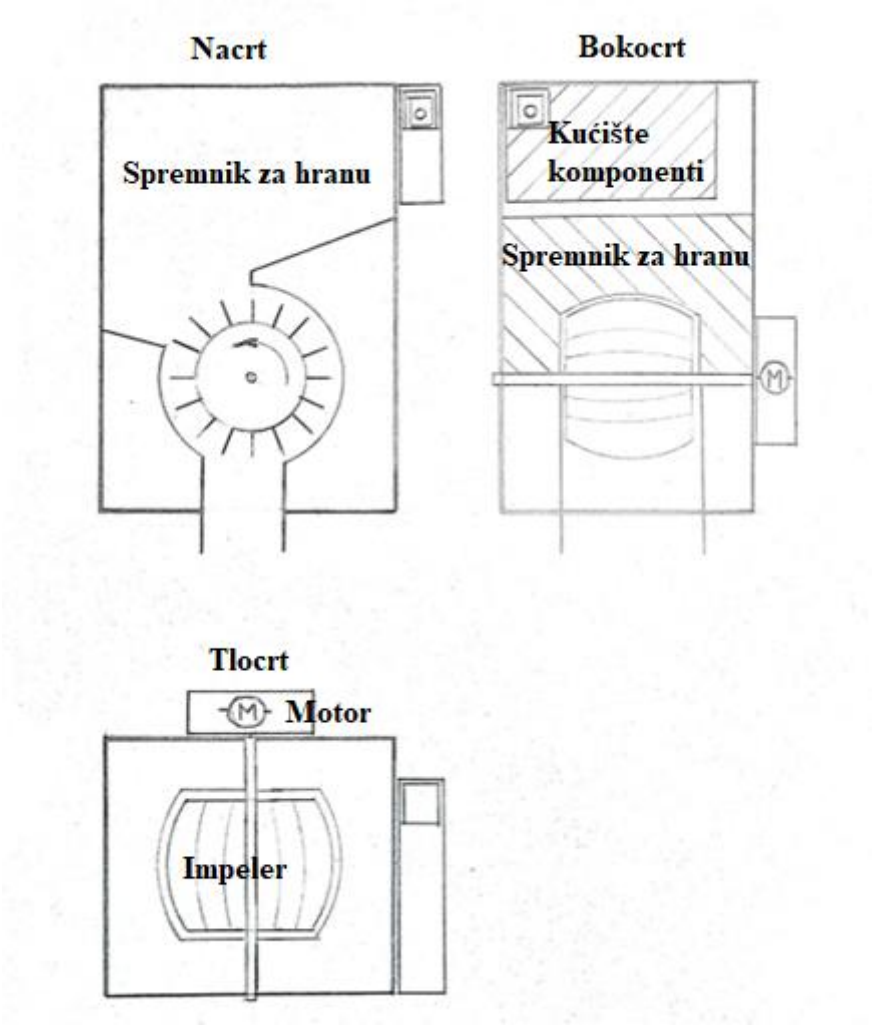
2.2. Prijedlog sklopovskog rješenja

Ovo poglavlje opisuje prijedlog i ideju sklopovskog rješenja automatiziranje hranilice za životinje. Pomoću blokovskog funkcionalnog blok dijagrama biti će prikazana zamišljena komunikacija između komponenti, te pomoću crteža zamišljena izvedba konačnoga sklopovlja.



Slika 2.3. *Blokovski prikaz zamišljenog sklopovskog rješenja.*

Prema slici 2.3. za napajanje je zamišljena prijenosna baterija koja napaja koračni motor te kameru. Kamera snima slike koje računalo preuzima na obradu te na temelju obrade slike i prepoznate životinje šalje signal nazad kameri koja ima u sebi ugrađen mikroupravljač. Kamera komunicira putem bežične internetske veze s računalom, također koristi bateriju kao izvor napajanja. Na temelju obrađene slike odnosno prepoznate životinje mikroupravljač integriran na kameri šalje zadani signal koračnom motoru za dozu hrane predviđenu za tu životinju.



Slika 2.4. *Nacrt, bokocrt i tlocrt prijedloga sklopovskog rješenja.*

Na slici 2.4. prikazan je prijedlog sklopovskog rješenja automatizirane hranilice za životinje. Kućište hranilice izrađeno je od nehrđajućeg čelika. Prednji dio izrađen je od akrilnog stakla radi bolje preglednosti spremnika. Za prijenos hrane iz kućišta prema van koristi se impeler s elastičnim lopaticama koje pogoni koračni motor koji se nalazi na stražnjoj strani istog. Motor pogoni impeler preko vratila. Preko desne polovice impelera postavljena je pregrada jer na toj strani impeler ima tendenciju gibanja prema gore. Time se izbjegava nepotrebni otpor impelera pritisnut težinom granula hrane i smanjuje opterećenje samog motora kao što je objašnjeno u ranijem poglavlju. Plan je postaviti kameru na bočnu stranu kućišta kako bi mogla imati jasan pregled na životinju koja se nalazi ispred hranilice. Također, na bočnoj strani kućišta zamišljeno je manje kućište u kojem se nalaze ostale komponente zajedno s kamerom potrebne za rad sustava zaštićene od vanjskih uvjeta.

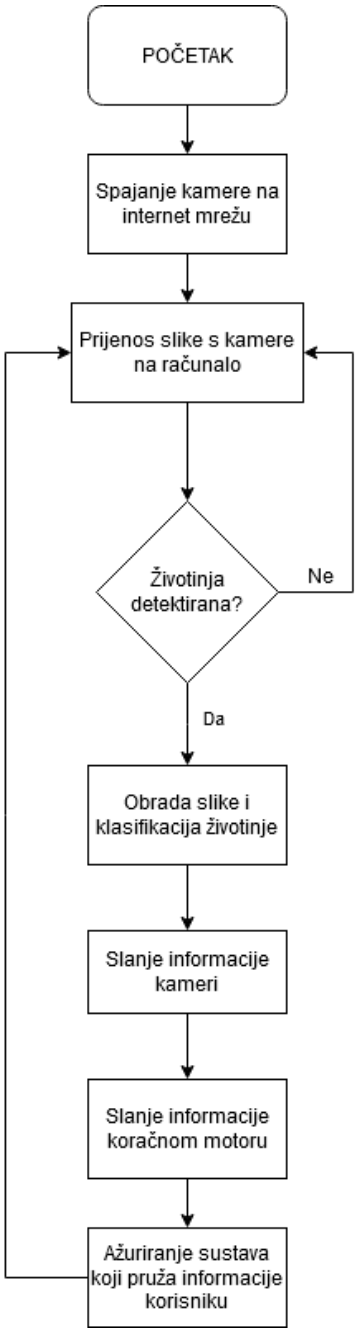
2.3. Prijedlog programskog rješenja

U ovom programskom rješenju, sustav za automatsko hranjenje životinja koristi integraciju kamere, računala, mikroupravljača i koračnog motora za učinkovitu detekciju i isporuku hrane. Glavni elementi sustava uključuju kameru za snimanje slika, računalo za obradu podataka, mikroupravljač kao most u programiranju i opcionalnom napajanju kamere, koračni motor za doziranje hrane i web aplikaciju za korisničko praćenje i upravljanje. Uloga mikroupravljača je prvotno bila za programirananje kamere, zatim radi jednostavnosti u izradi ovoga rada se nastavio koristiti kao izvor napajanja za kameru.

Kamera snima slike u stvarnom vremenu koje se prenose putem bežične mreže do računala. Računalo zatim preuzima slike i koristi algoritme za obradu slike i strojno učenje za prepoznavanje životinja. Ovaj korak uključuje analizu slike kako bi se utvrdilo prisustvo i vrsta životinje, što omogućuje sustavu da identificira specifične potrebe za hranom. Nakon što računalo prepozna životinju i odredi količinu hrane koja je potrebna, šalje signal mikroupravljaču na kameri. Kamera, koja je fizički povezana s koračnim motorom, prima ove naredbe i upravlja motorom kako bi izbacivao hranu prema specifičnim parametrima definiranim od strane računala. Ova operacija uključuje rotaciju motora koja omogućuje izbacivanje točne količine hrane.

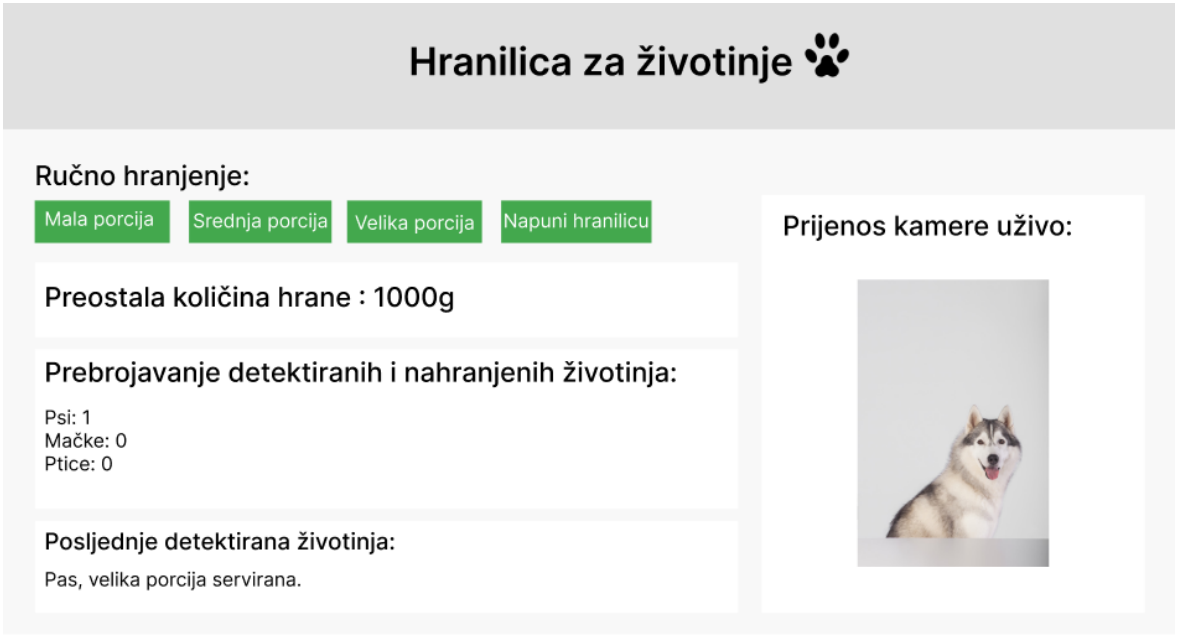
Računalo također *„hosta“* web aplikaciju koja omogućuje korisnicima praćenje i upravljanje sustavom. Kroz ovu web aplikaciju, korisnici mogu vidjeti informacije poput preostale količine hrane i broj životinja koje su bile detektirane. Ova aplikacija uz implementiranu logiku za računanje preostale hrane i prebrojavanje životinja pruža korisničko sučelje za upravljanje hranilicom, omogućujući praćenje stanja u stvarnom vremenu. Time se osigurava da korisnici uvijek imaju točan uvid u stanje hranilice.

Ovo rješenje prikazano je na slici 2.5. koje omogućuje automatsko hranjenje životinja uz minimalnu ljudsku intervenciju, koristeći sofisticirane tehnologije za obradu slike i upravljanje hranjenjem te pruža korisnicima jednostavan način za praćenje i upravljanje cijelim sustavom putem web aplikacije.



Slika 2.5. *Zamišljeni blok dijagram programskog rješenja.*

Na slici 2.6. je prikaz zamišljenog korisničkog web sučelja hranilice. Korisnici imaju uvid u preostalu količinu hrane, broj i vrstu naranjenih životinja, posljednju nahranjnu životinju i prijenos slika odnosno videa s kamere uživo.



Slika 2.6. *Zamišljen prikaz korisničkog sučelja web aplikacije.*

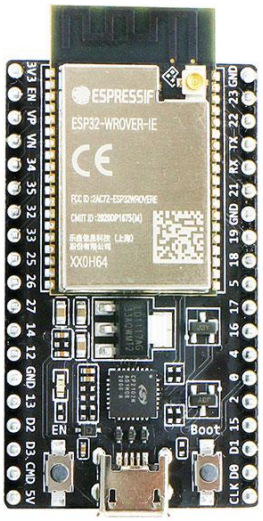
3. REALIZACIJA AUTOMATIZIRANE HRANILICE ZA ŽIVOTINJE

3.1. Korištene komponente, alati i programska okruženja

Korištene komponente:

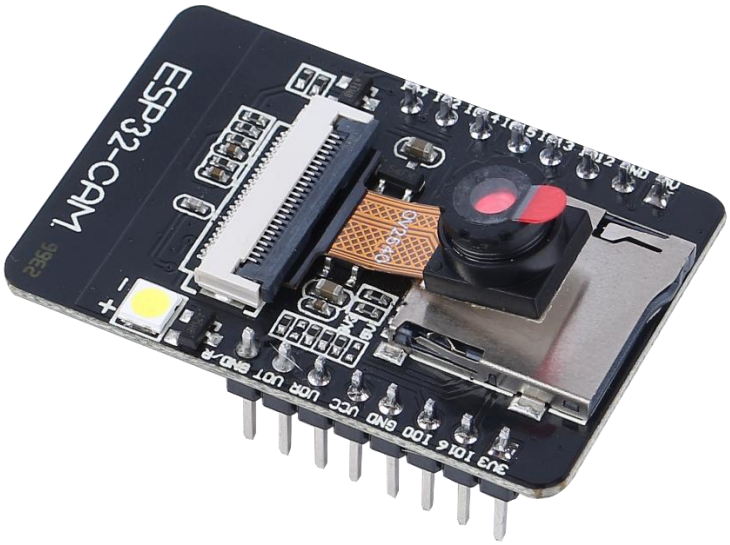
- ESP32
- ESP32-CAM
- Koračni motor 28BYJ-48 s *„driverom“*

ESP32 [3] je mikrokontroler (prikazan na slici 3.1.) s integriranim *„Wi-Fi“* i *„Bluetooth“* modulima što ga čini idealnim za IoT projekte i bežičnu komunikaciju. Zbog dvostrukih jezgri omogućava istovremeno izvršavanje više zadataka, što je posebno korisno za projekte s višestrukim senzorskim ulazima i zahtjevima za brzim odgovorom. Uz to, podržava digitalne i analogne ulaze/izlaze, što omogućava lako povezivanje raznih komponenti. Također, podržava protokole poput MQTT i HTTP, što ga čini pogodnim za projekte koji zahtijevaju udaljeni nadzor i upravljanje. Uloga ESP32 u ovome radu je most za programiranje i napajanje ESP32-CAM komponente. ESP32 omogućuje komunikaciju i konfiguraciju ESP32-CAM modula, osiguravajući stabilnu vezu tijekom programiranja. Osim toga, ESP32 je zadržan u projektu kako bi neprekidno napajao kameru.



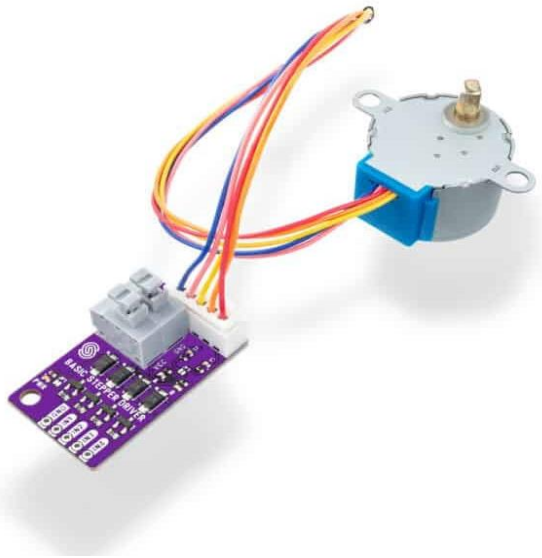
Slika 3.1. *ESP32 mikrokontroler, [4].*

ESP32-CAM [5] je kompaktni razvojni modul koji kombinira ESP32 mikrokontroler s integriranom kamerom i mogućnostima bežične komunikacije (*„Wi-Fi“* i *„Bluetooth“)* prikazan na slici 3.2. Ovaj modul nudi visokokvalitetnu sliku i video putem OV2640 kamere od 2 MP, koja omogućava snimanje i prijenos slika i videa u stvarnom vremenu. Kamera ima glavnu ulogu u izradi automatizirane hranilice za životinje. Kamera je isprogramirana za hvatanje slika u stvarnome vremenu, koje se šalju na Flask server gdje se radi obrada slike te ukoliko je životinja prepoznata, web aplikacija šalje informaciju kameri koja je također povezana s *„driverom“* koračnog motora te mu daje upute za izbacivanje potrebne količine hrane iz hranilice.



Slika 3.2. *ESP32-CAM, [6].*

Koračni motor 28BYJ-48 [7] je popularan i široko korišten motor u malim robotskim projektima i automatizaciji zbog svoje preciznosti i jednostavnosti upravljanja prikazan na slici 3.3. Radi na principu rotacije kroz fiksne korake, što omogućava precizno pozicioniranje. Obično se koristi s *„driverom“* ULN2003 ili njegovom modificiranom kopijom što je slučaj u ovom radu, koji služi za upravljanje strujom kroz namotaje motora, omogućujući kontrolu smjera i brzine. Ovaj motor ima 5 žica, gdje su četiri namijenjene upravljanju namotajima, a jedna žica služi za napajanje (5V). Također, ima reduktor koji povećava obrtni moment, ali smanjuje brzinu rotacije. Rotacija motora preko pogonskog vratila pokreće impeler koji svojim lopaticama zagrabi određenu količinu hrane te ju ispusti van hranilice.

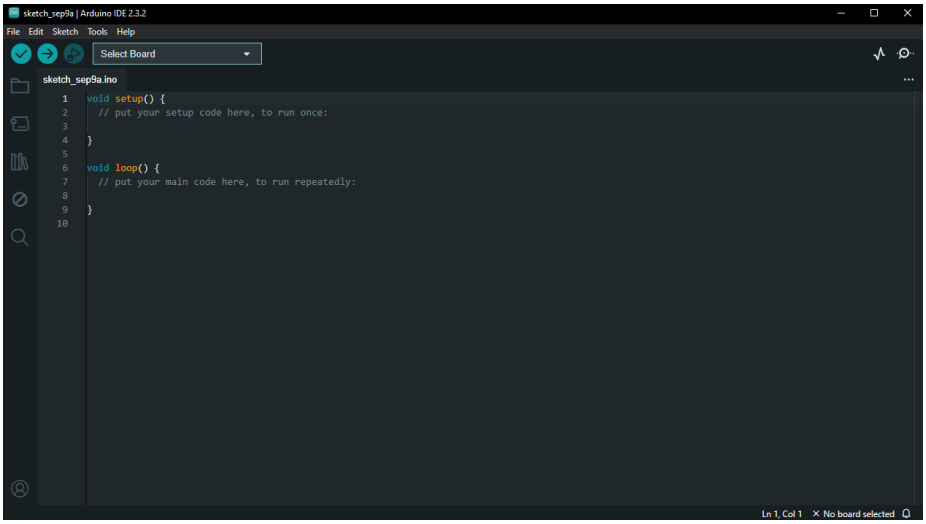


Slika 3.3. *Koračni motor 28BYJ-48 s „driverom“, [8].*

Korišteni alati i programska okruženja:

- Arduino IDE
- Visual Studio Code

Arduino IDE [9] je otvoreno razvojno okruženje koje omogućava programiranje mikrokontrolera kao što su Arduino ploče. Nudi korisničko sučelje za pisanje, učitavanje i koda na hardver koristeći prilagođeni C++ ili C jezik (prikaz na slici 3.4.). S brojnim ugrađenim bibliotekama i podrškom za različite platforme, Arduino IDE pruža jednostavan način za programiranje mikrokontrolera. U ovom slučaju, koristi se za programiranje i učitavanje koda na ESP32-CAM. Kod upravlja kamerom, web serverom iste te koračnim motorom.



Slika 3.4. *Primjer prikaza programa Arduino IDE.*

Visual Studio Code [10] je razvojno okruženje koje se koristi kao uređivač koda za različite vrste programiranja i razvoj raznih vrsta *„softwerskih“* aplikacija. Podržava širok raspon tehnologija i jezika kao što su Python, HTML, JavaScript i mnogi drugi. Idealan je za izradu web aplikacija i skripti. Također, Visual Studio Code omogućuje jednostavnu integraciju s alatima za kontrolu verzija i izgradnju, poput Git-a. Zahvaljujući velikom broju ekstenzija, može se prilagoditi specifičnim potrebama i preferencijama korisnika, što ga čini idealnim za razvoj na različitim platformama. U ovom radu se koristi za razvoj Flask web aplikacije koja obrađuje slike s kamere pomoću OpenCV biblioteke u Pythonu omogućujući detekciju životinja i upravljanje hranilicom. Visual Studio Code korišten je za uređivanje HTML i CSS datoteka koje služe za prikaz korisničkog sučelja za automatiziranu hranilicu, uključujući informacije o preostaloj hrani i prebrojavanje nahranjenih životinja. Aplikacija upravalja logikom koja razmjenjuje podatke odnosno prima slike s kamere putem IP adrese njenog web servera te računa preostalu količinu hrane na temelju zadanih parametara.

3.2. Realizacija konstrukcijskog i sklopovskog rješenja

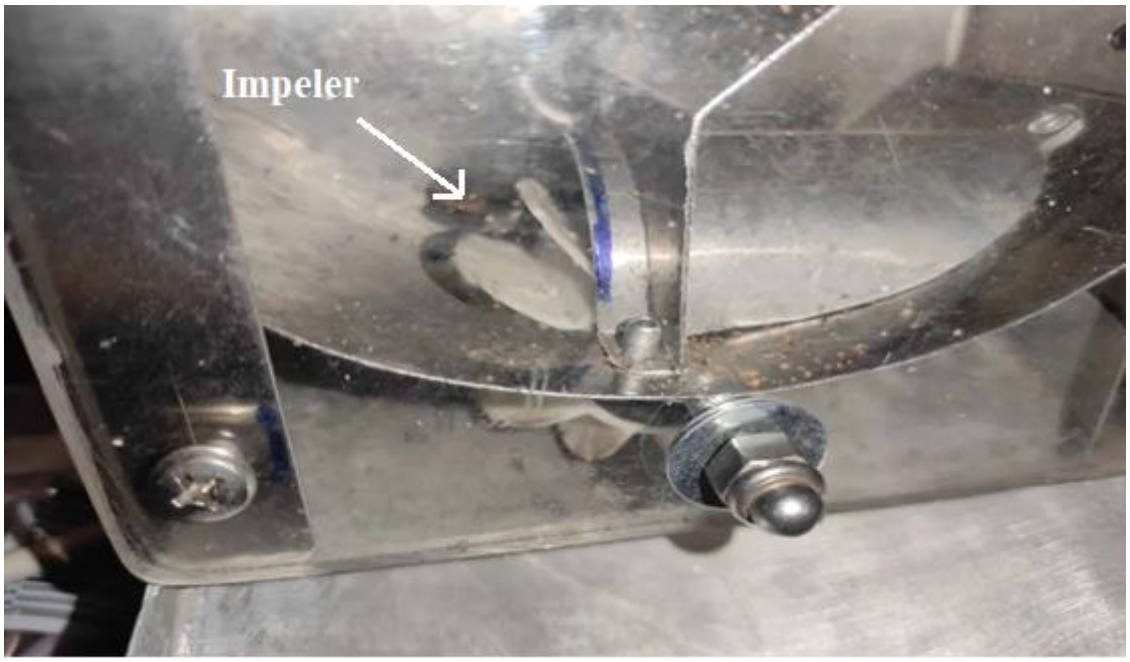
Konstrukcijsko rješenje hranilice temelji se na čvrstom i otpornom kućištu od nehrđajućeg čelika, koje osigurava dugotrajnost i otpornost na vanjske utjecaje. Prednja strana hranilice izrađena je od prozirnog akrilnog stakla, što omogućuje jasan pregled unutrašnjosti i razinu hrane u svakom trenutku. Kombinacija nehrđajućeg čelika i akrilnog stakla osigurava estetski privlačan i funkcionalan dizajn, dok su materijali odabrani zbog svojih kvalitetnih mehaničkih i kemijskih svojstava, uključujući otpornost na koroziju i lakoću održavanja. Takva konstrukcija također omogućava jednostavno čišćenje i dugotrajnu upotrebu u raznim uvjetima.

Slika 3.5. *Prva verzija kućišta hranilice.*

Prva verzija kućišta hranilice je trebala koristiti pužni spiralni prijenos hrane van hranilice kao što je prikazano na slikama 3.5. i 3.6. Međutim, to se pokazalo neefikasnim jer su granule bile previše tvrde, što je uzrokovalo probleme pri prijenosu. Zbog krutosti pužne spirale, granule su često zapinjale, što je ometalo protok hrane. Kao rezultat toga, ovaj sustav prijenosa nije bio pouzdan za ovu primjenu te je zahtijevao daljnju prilagodbu odnosno zamjenu.

Slika 3.6. *Pužni spiralni prijenos (prva verzija).*

Kao efikasno rješenje, umjesto pužne spirale, implementiran je impeler s elastičnim lopaticama na kojima su dodatno isječeni prorezi kao što se vidi na slici 3.7. Ovaj dizajn omogućuje fleksibilnije i pouzdanije rukovanje hranom, jer elastične lopatice mogu prilagoditi svoj oblik prilikom rotacije, što smanjuje mogućnost zapinjanja tvrdih granula. Impeler se okreće i svojim lopaticama zahvaća hranu, ravnomjerno je transportira i izbacuje van hranilice. Zahvaljujući elastičnosti lopatica i prorezima na njima, one se mogu prilagoditi različitim veličinama i oblicima granula, osiguravajući konstantan protok bez blokada. Ovakav sustav pruža veću efikasnost i pouzdanost u usporedbi s pužnom spiralom, posebno kada se radi o hranjenju s tvrđim ili nepravilno oblikovanim granulama. Također, izrađena je spojnica između vratila motora i vratila impelera kako bi se osigurao prijenos gibanja s motora na impeler. Motor, smješten na stražnjoj strani kućišta, preko ove spojnice prenosi rotacijsko gibanje na vratilo impelera, omogućujući njegovo okretanje i učinkovito transportiranje hrane prema izlazu. U kućištu je postavljena kosa pregrada kako bi se spriječila sila pritiska hrane na taj dio impelera i smanjilo opterećenje motora koji bi morao tom polovicom podizati svu hranu u spremniku, a drugom spuštati hranu van.



Slika 3.7. *Postavljeni impeler s elastičnim lopaticama (konačna verzija).*

Ugrađeno je kućište za prijenosnu bateriju koje napaja ESP32 putem USB-B (engl. Universal serial bus, hrv. Univerzalna serijska sabirnica tip B) priključka, dok je driver koračnog motora napajan putem USB (engl. Universal serial bus, hrv Univerzalna serijska sabirnica, u

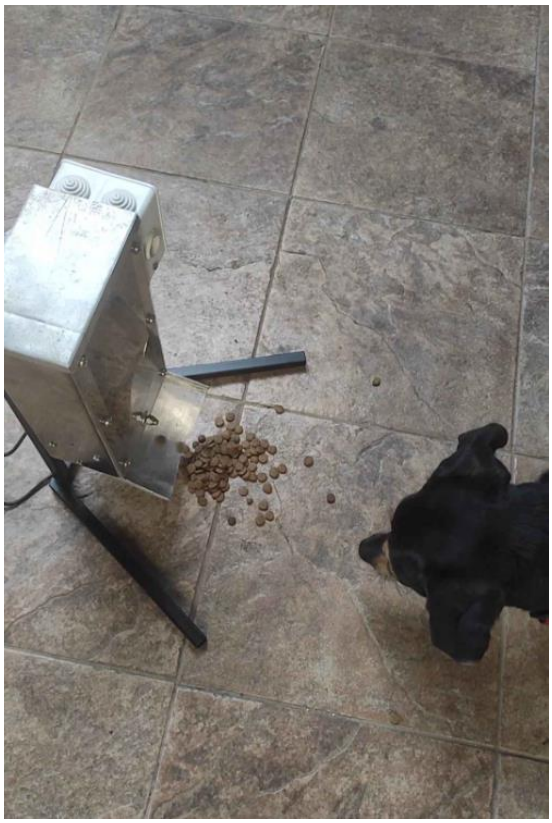
daljnjem tekstu korišten kao 'USB') kabela iz kojeg su izvučene VCC i GND žice. Sve navedene komponente smještene su u razvodnu kutiju koja se nalazi na bočnoj strani hranilice kao što je prikazano na slici 3.8. Na ovoj kutiji nalazi se otvor za kameru, što omogućava nadzor prostora ispred hranilice za detekciju prisutne životinje.

Slika 3.8. *Komponente ugrađene na hranilicu.*

Na kraju procesa izrade, dodan je poklopac koji osigurava zaštitu hrane u hranilici od vanjskih uvjeta. Nakon završetka sastavljanja, hranilica je postavljena na stalak, što omogućava stabilnu instalaciju i lakše korištenje u svakodnevnoj primjeni. Na slici 3.9. prikazana je konačna verzija hranilice. Na slikama 3.10. i 3.11. može se vidjeti hranilica dok obavlja svoju radnju, odnosno detektira životinju te prema njenoj vrsti izbacuje zadanu dozu hrane.

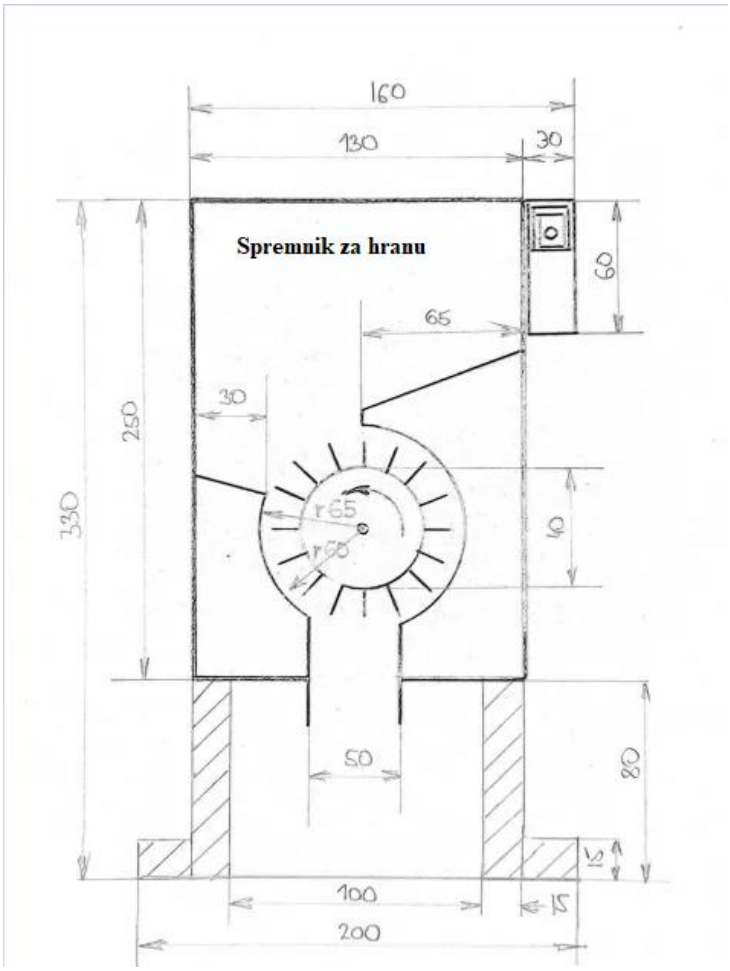
Slika 3.9. *Konačna verzija konstrukcijskog rješenja hranilice.*

Slika 3.10. *Detektiran pas koji se nalazi ispred kamere hranilice.*



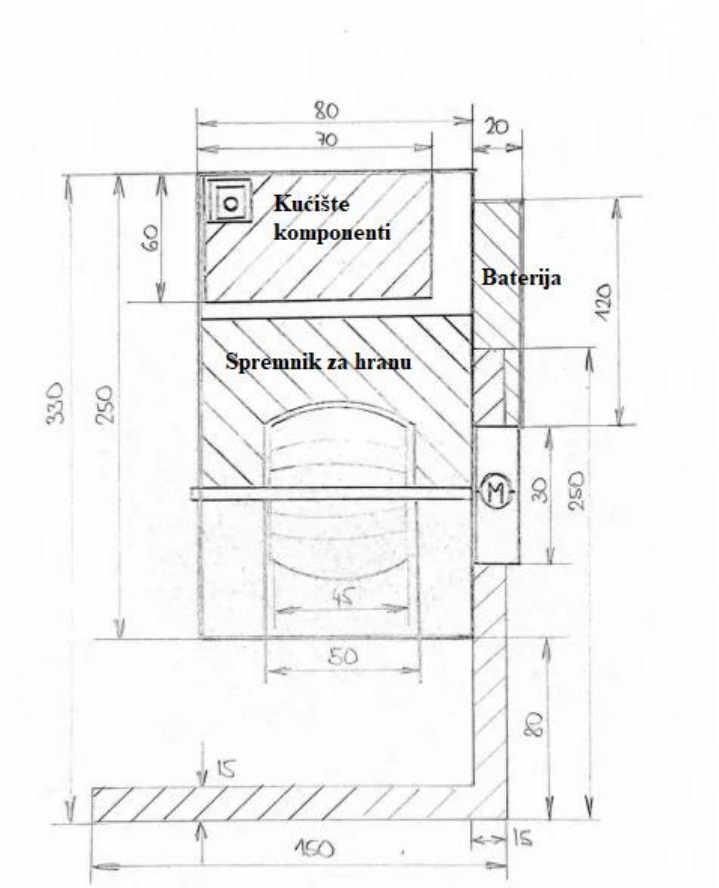
Slika 3.11. *Izbacivanje potrebne količine hrane za psa.*

Također, napravljen je detaljan tehnički nacrt konstrukcijskog rješenja hranilice sa životinjama uz dimenzije svakog njenog dijela. Na slici 3.12 prikazan je nacrt hranilice za životinje.



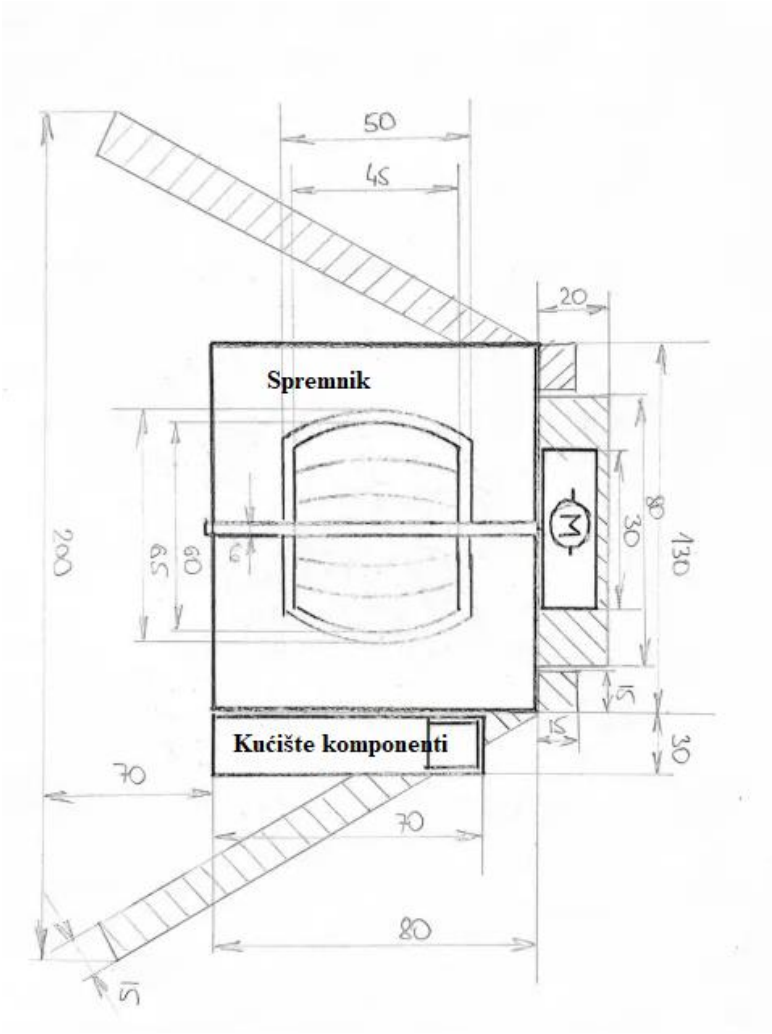
Slika 3.12. *Nacrt hranilice za životinje.*

Na slici 3.13. prikazan je bokocrt hranilice za životinje s označenim dijelovima hranilice iz tog prikaza.



Slika 3.13. *Bokocrt hranilice za životinje.*

Na slici 3.14. prikazan je tlocrt hranilice za životinje s označenim dijelovima hranilice iz tog prikaza.

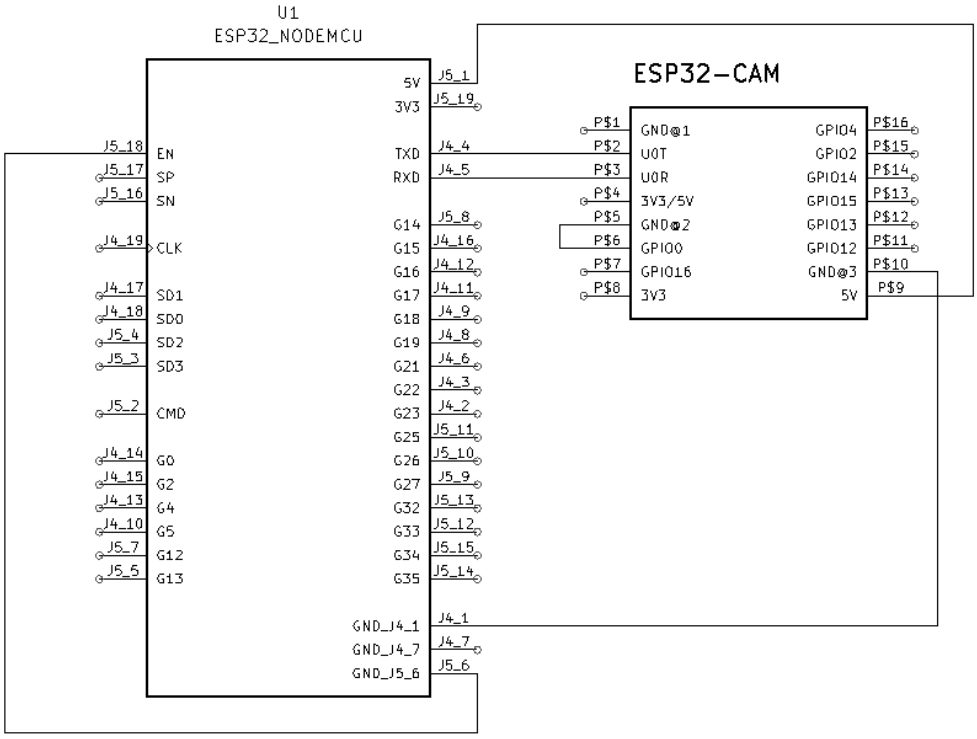


Slika 3.14. *Tlocrt hranilice za životinje.*

ESP32-CAM može se programirati na nekoliko načina. Jedan od najčešćih načina je pomoću FTDI adaptera (USB na serijski), koji se povezuju sa serijskim linijama za prijenos podataka TX (engl. *„transmit“*, hrv. prijenos), RX (engl. *„receive“*, hrv. primanje), 5V i GND pinovima ESP32-CAM modula, omogućavajući prijenos koda iz Arduino IDE [11]. Drugi način je programiranje pomoću ESP32-CAM-MB USB programera, što je jednostavniji način jer se radi o specijaliziranom modulu s ugrađenim USB šučeljem, koji se direktno povezuje s ESP32-CAM bez potrebe za vanjskim žicama [12]. Programiranje je moguće i pomoću ESP32 modula ili čak Arduino Nano i drugih Arduino pločica kao USB to serial konvertera, iako je taj postupak složeniji.

U ovom radu ESP32-CAM je programirana pomoću ESP32 modula, na slici 3.15. prikazana je električna shema spajanja ESP32-CAM na ESP32 kako bi se programirala. Potrebno je spojiti TX pin ESP32 modula na TX pin ESP32-CAM modula, RX pin ESP32 modula na RX

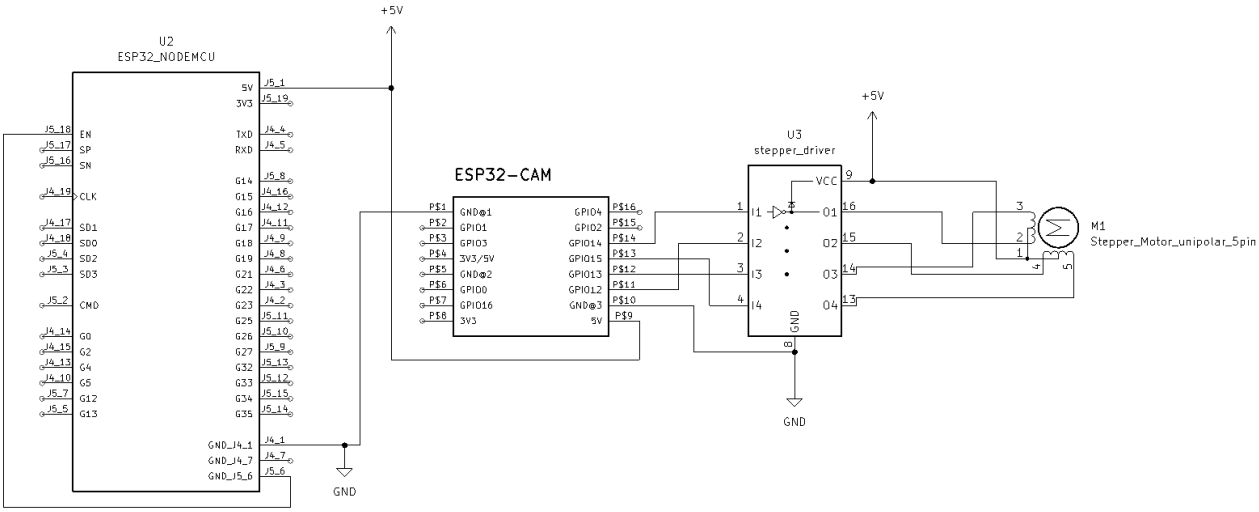
pin ESP32-CAM modula, 5V pin ESP32 modula na 5V pin ESP32-CAM modula te GND pinove međusobno povezati. Bitno je imati spojen EN (engl. *„enable“*, hrv. omogući) pin na GND pin ESP32 modula kako bi se izbjeglo ometanje serijske komunikacije između računala i ESP32-CAM, tada je ESP32 u stanju resetiranja odnosno isključenja. U osnovi, ESP32 (koji koristi serijski port) bi se mogao sukobljavati s ESP32-CAM na linijama za prijenos podataka (TX/RX linije). Dakle, stavljanjem EN pina na GND, zapravo se isključuje ESP32, što osigurava da ESP32-CAM može primati podatke bez konflikta. Također, bitno je svaki put prilikom uploadanja koda držati GPIO0 pin povezan na GND, čime se ESP32-CAM prebacuje u mod za programiranje. Nakon prijenosa koda s računala na ESP32-CAM, potrebno je odspojiti GPIO0 pin s GND pina.



Slika 3.15. *Električna shema pri programiranju kamere.*

Na slici 3.16. može se vidjeti konačna električna shema sustava. ESP32 modul korišten je kao izvor napajanja za ESP32-CAM (sama baterija se također može spojiti na ESP32-CAM kako bi ga napajala) jer je prethodno bio potreban za programiranje kamere, pa je ostao u spoju kako bi osiguravao stabilno napajanje. ESP32 je napajan putem USB kabela iz prijenosne baterije. Nakon programiranja, pinovi ESP32-CAM modula povezani su s odgovarajućim pinovima *„drivera“* koračnog motora. *„Driver“* upravlja koračnim motorom, a uz to ima i dodatno vanjsko napajanje

putem (USB kabl iz kojeg su iskorištene VCC i GND žice) prijenosne baterije, koje je neophodno za rad motora, s obzirom na to da sam ESP32-CAM ne može pružiti dovoljno energije za motor. Ovakva konfiguracija omogućava da ESP32-CAM kontrolira rad koračnog motora preko *„drivera“*, dok isti osigurava pravilno upravljanje i napajanje motora.



Slika 3.16. *Električna shema sustava.*

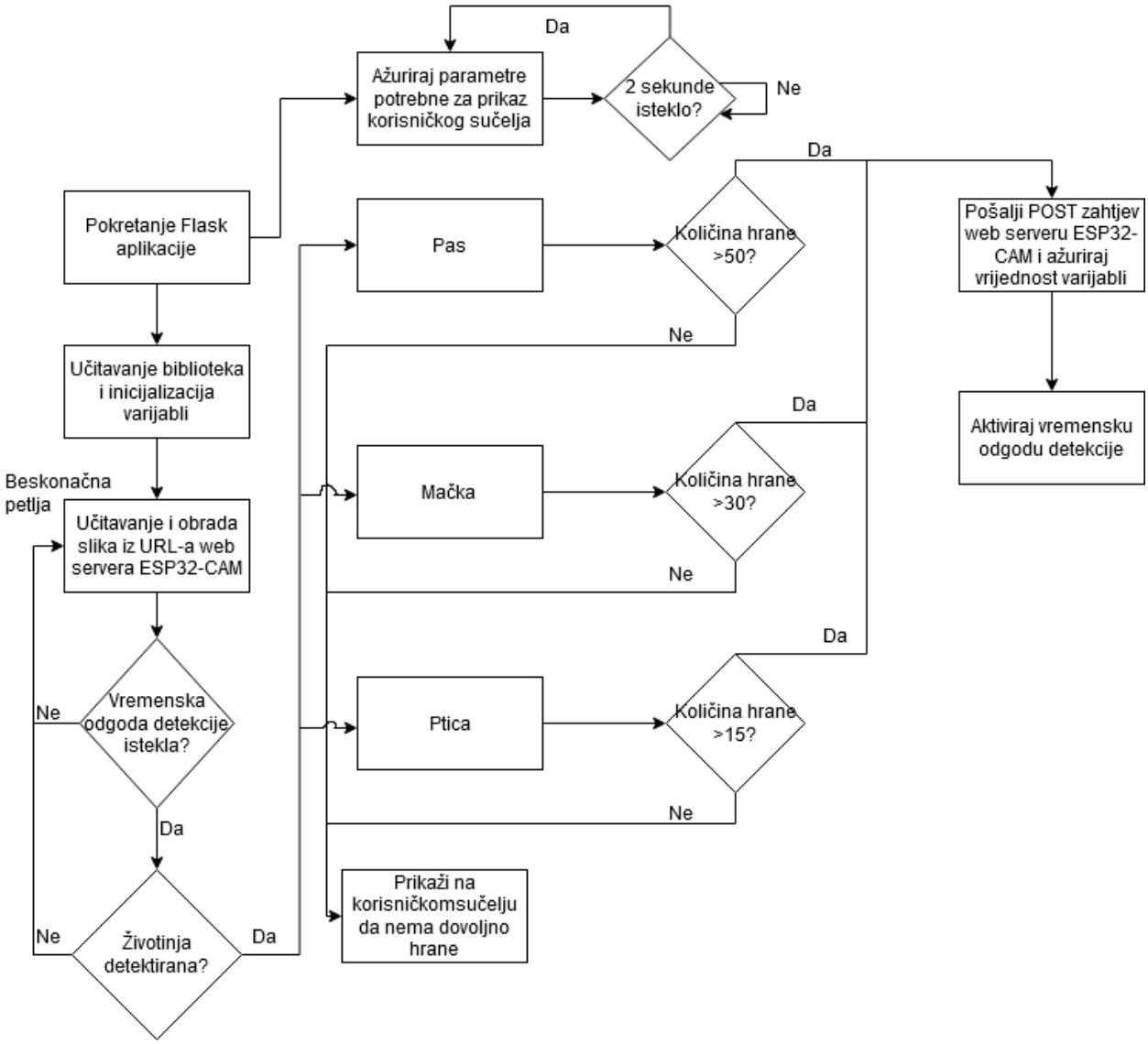
3.3.Realizacija programskog rješenja

Ovaj program implementira sustav automatske hranilice za životinje s detekcijom putem ESP32-CAM i kontrolom količine hrane. Sustav je razvijen pomoću web aplikacije koja koristi Flask okvir u Pythonu na serveru i ESP32-CAM za praćenje i upravljanje hardverom. Sastoji se od frontend i backend komponenti, gdje korisnici mogu ručno hraniti životinje ili pratiti automatsku distribuciju hrane temeljenu na prepoznavanju životinja.

HTML stranica prikazuje grafičko sučelje koje omogućuje korisnicima kontrolu nad hranilicom kao što je opisano u prilogu P20. Koristeći HTML, JavaScript i jQuery, korisnici mogu ručno odabrati različite porcije hrane pritiskom na gumb – mala porcija od 15 grama, srednja od 30 grama i velika od 50 grama. Tu je također opcija za ponovno punjenje hranilice (kada ju korisnik napuni), što vraća količinu hrane na maksimalnu vrijednost 500 grama kao što je opisano u prilogu P16. Stranica također prikazuje preostalu količinu hrane, broj detektiranih pasa, mačaka i ptica te posljednju detektiranu životinju putem kamere uživo. Ključna funkcionalnost na *„frontend“* strani uključuje slanje zahtjeva serveru putem AJAX POST i GET zahtjeva. Kad korisnik pritisne gumb za hranjenje, *„frontend“* šalje zahtjev serveru, koji odgovara s trenutnom

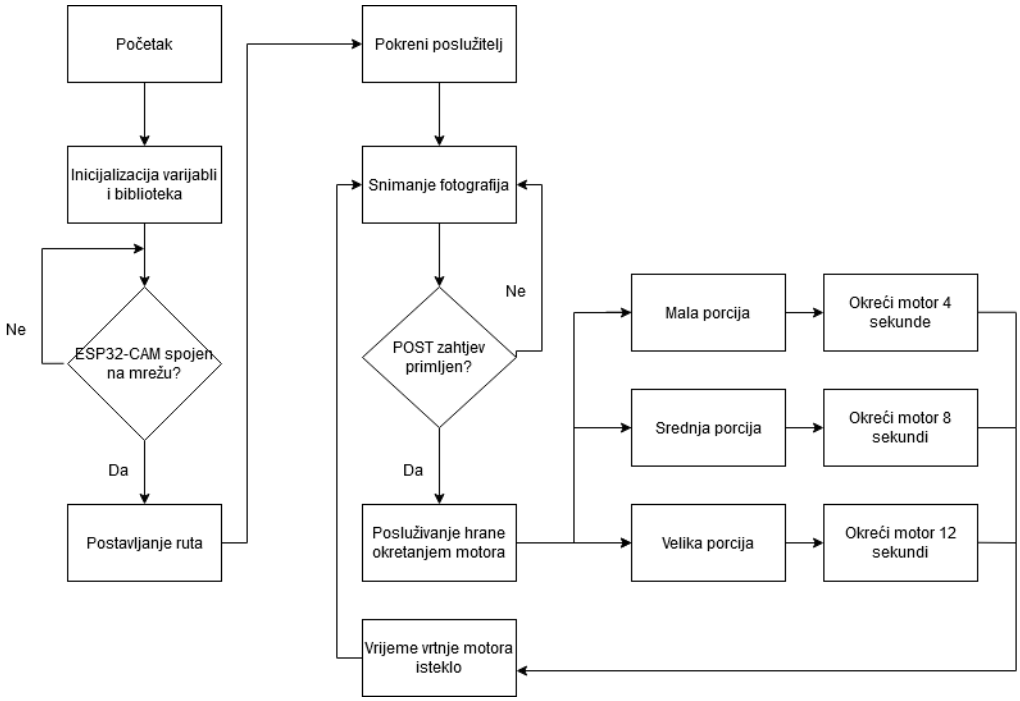
količinom preostale hrane, koja se dinamički ažurira na stranici bez ponovnog učitavanja. Program koristi intervale od 2 sekunde za automatsko ažuriranje podataka o posljednjoj detektiranoj životinji i broju detektiranih pasa, mačaka i ptica. To se postiže korištenjem JavaScript funkcije, koja u pozadini periodično šalje GET zahtjeve serveru (prilog P21). Na taj način, korisničko sučelje je uvijek sinkronizirano s realnim stanjem sustava. ESP32-CAM prenosi slike na web stranicu koristeći Flask aplikaciju, koja periodički dohvaća slike s kamere kao što je objašnjeno u prilogu P14. Ova funkcionalnost omogućava korisnicima da prate situaciju oko hranilice.

„Backend“ dio aplikacije pokreće se pomoću Flask okvira, koji se koristi za upravljanje HTTP zahtjevima i obradom podataka s kamere. Kada korisnik zatraži hranjenje, *„backend“* prima POST zahtjev s odabranom veličinom porcije i kontrolira količinu preostale hrane. Ako je dostupno dovoljno hrane, aplikacija smanjuje količinu na osnovu odabrane porcije i vraća ažurirane podatke na *„frontend“*. Ako hrane nema dovoljno, korisnik dobiva odgovarajuću poruku. Flask aplikacija koristi OpenCV model za detekciju objekata, prepoznajući pse, mačke i ptice, što omogućava automatsko hranjenje. Kada sustav detektira životinju, donosi odluku o količini hrane koju treba izdati, ovisno o vrsti životinje. Na primjer, psi dobivaju najveću porciju, mačke srednju, a ptice najmanju(prilog P14). Ako je dostupno dovoljno hrane, sustav smanjuje preostalu količinu, bilježi detekciju i šalje naredbu za hranjenje ESP32-CAM kao što je objašnjeno u prilogu P17, koja zatim pokreće izdavanje hrane. Nakon slanja informacije kameri za izdavanje hrane, aktivira se vremenska odgoda detekcije u trajanju od 3 minute (definirano u prilogu P12) kako bi se spriječilo konstantno izbacivanje hrane životinji koja se nalazi ispred hranilice. Na slici 3.17. prikazan je dijagram toka web aplikacije.



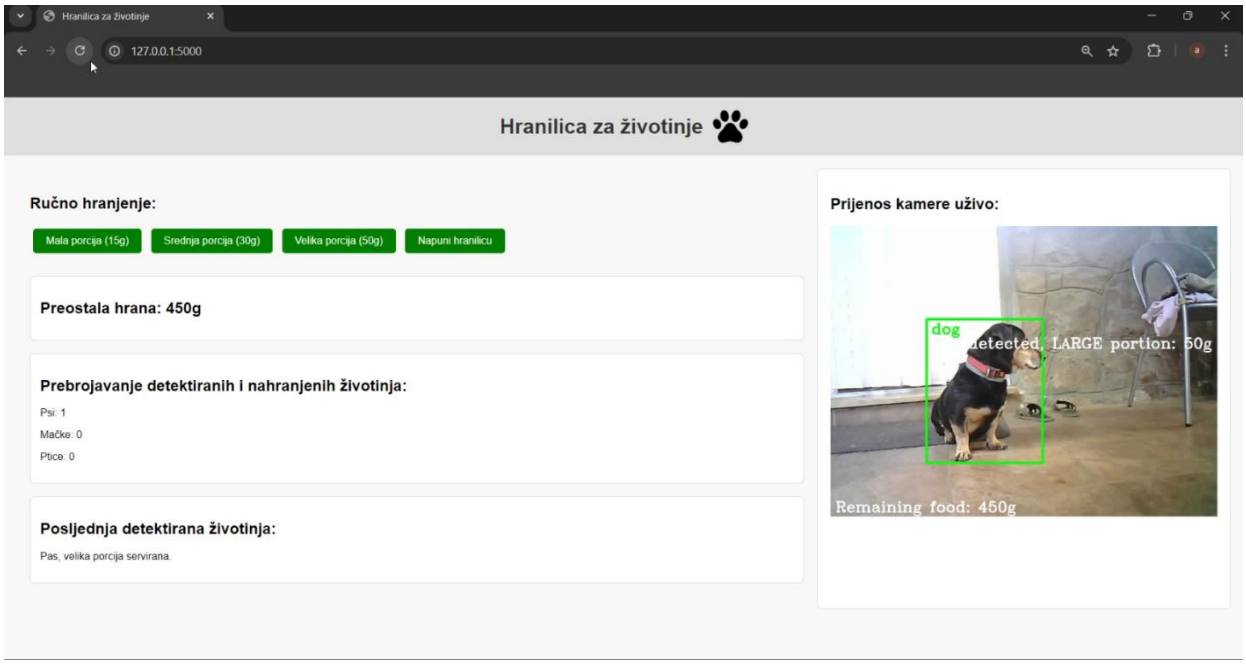
Slika 3.17. *Dijagram toka rada Flask aplikacije.*

ESP32-CAM ne služi samo za slanje slika putem web servera nego i za slanje informacija *„driveru“* koračnog motora. Ova kamera može snimati slike različitih rezolucija. Kada *„backend“* dio aplikacije šalje naredbu za hranjenje, ESP32-CAM pokreće motor, koji rotira i izdaje određenu količinu hrane ovisno o odabranoj porciji kao što je objašnjeno u prilogu P11. Rotacija motora kontrolira količinu hrane izdanu iz hranilice. Različite naredbe (mala, srednja, velika porcija) pokreću motor na određeni interval vrtnje. Komunikacija između Flask aplikacije i ESP32-CAM omogućena je putem HTTP POST zahtjeva. Objašnjen tok rada ESP32-CAM može se vidjeti na slici 3.18.



Slika 3.18. *Dijagram toka rada programa ESP32-CAM.*

Na slici 3.19. prikazan je konačan izgled korisničkog web sučelja za automatiziranu hranilicu za životinje u trenutku detekcije psa.



Slika 3.19. *Prikaz korisničkog web sučelja.*

4. TESTIRANJE I REZULTATI

4.1.Metodologija testiranja

Testiranje hranilice obuhvaća dva glavna aspekta:

1. Točnost detekcije životinja
2. Masa izbačene hrane

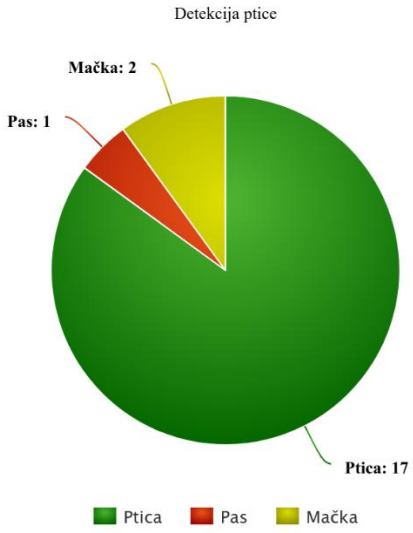
Testirat će se sposobnost modela da uspješno prepozna vrstu životinje i pouzdanost modela za različite vrste životinja, bilo da se radi o psu, mački ili ptici. Ovaj test će biti izvršen pomoću različitih isprintanih fotografija navedenih životinja zbog ograničenosti testiranja s pravim životinjama.

Također, testirat će se i odrediti srednja vrijednost mase količine hrane u granulama koju hranilica izbacuje u različitim intervalima okretanja motora, ovisno o vrsti životinje koja je detektirana. Motor ima 3 isprogramirana vremenska intervala vrtnje 4, 8, 12 sekundi odnosno mala, srednja i velika porcija. Za psa se izbacuje velika, za mačku srednja a za pticu mala porcija.

Rezultati testiranja će se koristiti za procjenu vjerodostojnosti i preciznosti sustava, kao i za utvrđivanje uvjeta pod kojima se testiranje može uspješno provesti i interpretirati.

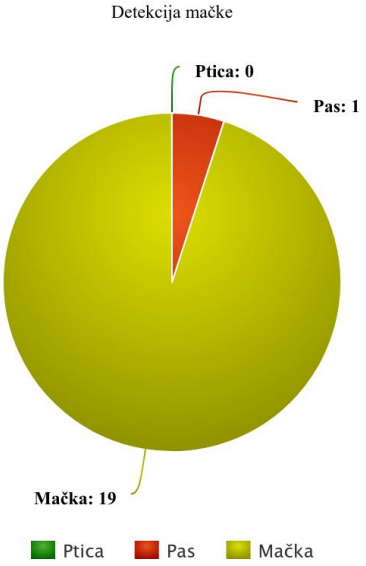
4.2. Rezultati testiranja

U analizi performansi modela za prepoznavanje životinja implementiranog u web aplikaciji Flask, uočene su značajne varijacije u točnosti detekcije ovisno o vrsti životinje. Model je postigao najbolje rezultate u prepoznavanju mačaka, što se može pripisati kvaliteti i dosljednosti slika korištenih za obuku. Nasuprot tome, detekcija pasa i ptica pokazala je manje preciznosti, što može biti rezultat različitih kvaliteta fotografija i varijacija u izgledu životinja. Na primjer, ako je korištena slika manjeg, čupavog psa, model bi ga prepoznao kao mačku. Na slikama 4.1., 4.2., 4.3. može se vidjeti uspješnost modela da točno detektira prikazanu životinju. Na slici 4.1. prikazan je dijagram točnosti prepoznavanja ptice, fotografije ptice prikazane su 20 puta, model je detektirao pticu 17 puta, mačku 2 puta te psa 1 put.



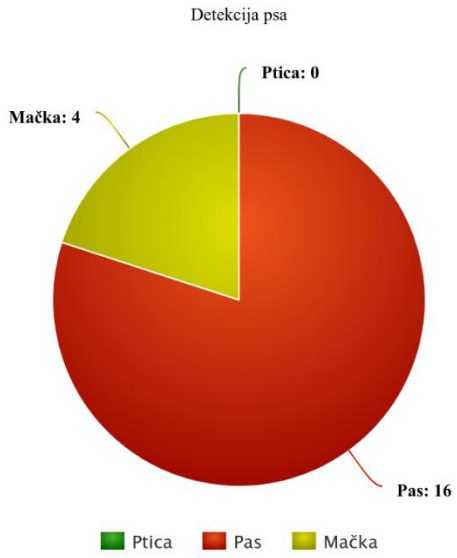
Slika 4.1. *Dijagram točnosti prepoznavanja ptice.*

Na slici 4.2. prikazan je dijagram točnosti prepoznavanja mačke, fotografije mačke prikazane su 20 puta, model je detektirao mačku 19 puta, psa 1 put te pticu niti jednom.



Slika 4.2. *Dijagram točnosti prepoznavanja mačke.*

Na slici 4.3. prikazan je dijagram točnosti prepoznavanja psa, fotografije psa prikazane su 20 puta, model je detektirao psa 16 puta, mačku 4 puta te pticu niti jednom.



Slika 4.3. *Dijagram točnosti prepoznavanja psa.*

Na sljedeće 2 fotografije mačke koje se nalaze na slici 4.4. proveden je test pouzdanosti modela da prepozna mačku. Imale su sličan rezultat, model je tvrdio da je test pouzdanosti za lijevu mačku 71%, dok je za desnu mačku 68%.

Slika 4.4. *Fotografije mačaka korištene za testiranje modela.*

Na sljedeće 2 fotografije psa koje se nalaze na slici 4.5. proveden je test pouzdanosti modela da prepozna psa. Za psa lijevo na fotografiji model je izbacio rezultat pouzdanosti 70%, dok je za psa desno(koji ne nalikuje baš tipičnom izgledu psa) izbacio rezultat pouzdanosti 58%.

Slika 4.5. *Fotografije pasa korištene za testiranje modela.*

Na sljedeće 2 fotografije ptica koje se nalaze na slici 4.6. proveden je test pouzdanosti modela da prepozna pticu. Za pticu lijevo na fotografiji model je izbacio rezultat pouzdanosti 72%, dok je za pticu desno izbacio rezultat od 67%.

Slika 4.6. *Fotografije ptica korištene za testiranje modela.*

Rezultat sljedećeg testiranja korišten je kao temelj za razvoj sustava koji automatski izračunava preostalu količinu hrane u hranilici. Tijekom testiranja mjerene su mase količine hrane u granulama koju hranilica izbaci pri određenim vremenskim intervalima vrtnje motora odnosno impelera. Na ovaj način, implementirana je jednostavna logika računanja preostale hrane u hranilici. Izračunat će se aritmetička srednja vrijednost 20 mjerenja za svaku porciju hrane. Ta vrijednost se izračunava prema formuli na slici 4.7.

$$\bar{x} = \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right) = \frac{x_1 + x_2 + \cdots + x_n}{n}$$

Slika 4.7. *Formula za aritmetičku srednju vrijednost.*

Prema tablici 4.1. može se vidjeti mala odstupanja u količini izbačene hrane u gramima za malu porciju odnosno kada koračni motor okreće impeler 4 sekunde.

Tablica 4.1. *Mjerenja mase izbačene hrane za malu porciju.*

1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
14g	15g	15g	14g	16g	15g	14g	15g	16g	15g
11.	12.	13.	14.	15.	16.	17.	18.	19.	20.
15g	14g	15g	14g	15g	14g	16g	15g	15g	14g

Za aritmetičku srednju vrijednost izbacivanja male porcije hrane, odnosno mase izbačene količine hrane za vremenski interval vrtnje motora od 4 sekunde dobije se vrijednost 14,8g što je zaokruženo na 15g radi preglednosti na korisničkom web sučelju.

Prema tablici 4.2. može se vidjeti mala odstupanja u količini izbačene hrane u gramima za srednju porciju odnosno kada koračni motor okreće impeler 8 sekundi.

Tablica 4.2. *Mjerenja mase izbačene hrane za srednju porciju.*

1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
30g	29g	29g	31g	28g	30g	30g	31g	29g	30g
11.	12.	13.	14.	15.	16.	17.	18.	19.	20.
30g	30g	31g	29g	30g	31g	30g	30g	29g	30g

Za aritmetičku srednju vrijednost izbacivanja srednje porcije hrane, odnosno mase izbačene količine hrane za vremenski interval vrtnje motora od 8 sekundi dobije se vrijednost 29.85g što je zaokruženo na 30g radi preglednosti na korisničkom web sučelju.

Prema tablici 4.3. može se vidjeti mala odstupanja u količini izbačene hrane u gramima za veliku porciju odnosno kada koračni motor okreće impeler 12 sekundi.

Tablica 4.3. *Mjerenja mase izbačene hrane za veliku porciju.*

1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
50g	51g	51g	47g	50g	52g	49g	51g	52g	50g
11.	12.	13.	14.	15.	16.	17.	18.	19.	20.
50g	51g	53g	48g	49g	50g	51g	50g	49g	50g

Za aritmetičku srednju vrijednost izbacivanja velike porcije hrane, odnosno mase izbačene količine hrane za vremenski interval vrtnje motora od 12 sekundi dobije se vrijednost 50.2g što je zaokruženo na 50g radi preglednosti na korisničkom web sučelju.

5. ZAKLJUČAK

U ovom radu razvijen je sustav automatizirane hranilice za životinje koji koristi kameru za detekciju životinja, koračni motor za upravljanje isporukom hrane i Flask web aplikaciju za obradu podataka i kontrolu hranjenja. Sustav je pokazao potencijal za primjenu u realnom svijetu, no tijekom izrade susret s brojnim izazovima je bio neizbježan, koji je rješavan kroz iterativni proces razvoja i prilagodbe sklopovlja i programske podrške komponenti.

Jedna od glavnih prednosti ovog sustava je njegova sposobnost automatizacije, daljinskog upravljanja i nadzora. Hranilica može raditi neovisno o ljudskom faktoru, također korisnik može imati uvid u stanje hrane u hranilici te on sam može izvršiti isporuku hrane (na primjer u svhru nekakvog testiranja sustava). Flask aplikacija poslužila kao učinkovito sučelje za obradu podataka i komunikaciju s korisnicima.

Tijekom izrade sklopovskog i programskog dijela, pojavilo se par izazova. Glavni izazov sklopovskog rješenja je bio način na koji će hrana biti distribuirana van hranilice. Kao prvo rješenje je bio testiran pužni spiralni prijenos, ali zbog krutosti samog matrijala istog i krutosti hrane u granulama, granule su uvijek zapinjale pri distribuciji. Bolje rješenje je bilo iskoristiti nešto elastičnijeg materijala poput impelera s elastičnim lopaticama, na kraju se ispostavilo da je to puno bolje rješenje. Pri izradi programskog rješenja, nedostatak uobičajenih komponenti za programiranje kamere je predstavio nešto manji izazov koji je rješen korištenjem ESP32 kao most između računala i kamere.

U sustavu postoji nekoliko mana i potencijalnih poboljšanja. Prva mana je ovisnost o internetskoj vezi koja služi za razmjenu podataka između kamere i web aplikacije. U području s nestabilnom mrežnom vezom performanse mogu biti ugrožene. Također, model za prepoznavanje životinja bi mogao biti poboljšan kako bi se umanjila kriva detekcija životinja. Sustav se može poboljšati dodavanjem više spremnika za hranu što bi zahtjevalo i više motora kako bi se mogao primjeniti na širem spekru životinja i njihovoj različitoj prehrani. Uz to, sama hranilica bi mogla koristiti i jači motor kako bi šanse za zaglavljivanje bile što manje pri još većem spremniku hrane.

Sustav automatizirane hranilice za životinje razvijen kroz ovaj rad pokazuje potencijal za primjenu u svakodnevnom životu. Automatizirano hranjenje uz mogućnost praćena životinja u stvarnom vremenu smanjuje korisnicima potrebu za stalnim nadzorom.

LITERATURA

[1] *Automatizirana hranilica za životinje sa senzorom*, https://www.circuito.io/blog/automatic-pet-feeder/, Pristup: 18.9.2024.

[2] *Automatizirana hranilica za životinje s vremenskom odgodom hranjenja*, https://www.norwegiancreations.com/2018/07/automatic-pet-feeder/, Pristup: 18.9.2024.

[3] *Uvod u ESP32*, https://www.tutorialspoint.com/esp32_for_iot/esp32_for_iot_introduction.htm, Pristup: 8.9.2024.

[4] *ESP32-DEVKITC-VIE*, https://www.digikey.lt/en/products/detail/espressif-systems/ESP32-DEVKITC-VIE/12091811, Pristup: 8.9.2024.

[5] *Sve o ESP32-CAM modulu*, https://robocraze.com/blogs/post/all-about-esp32-camera-module, Pristup: 8.9.2024.

[6] *ESP32 CAM modul*, https://docs.sunfounder.com/projects/galaxyrvr/en/latest/hardware/cpn_esp_32_cam.html, Pristup: 9.9.2024.

[7] *Sve o 28BYJ-48 koračnom motoru i "driveru"*, https://lastminuteengineers.com/28byj48-stepper-motor-arduino-tutorial/, Pristup: 9.9.2024.

[8] *Koračni motor 28BYJ-48*, https://soldered.com/product/stepper-motor-with-driver/, Pristup: 9.9.2024.

[9] *Program Arduino IDE*, https://www.javatpoint.com/arduino-ide, Pristup: 9.9.2024.

[10] *Program Visual Studio Code*, https://code.visualstudio.com/docs/editor/whyvscode, Pristup: 9.9.2024.

[11] *Programiranje ESP32-CAM korištenjem FTDI*, https://www.instructables.com/Fidi-ESP826632-Wiring/, Pristup: 10.9.2024.

[12] *Programiranje ESP32-CAM korištenjem ESP32-CAM-MB USB*, https://randomnerdtutorials.com/upload-code-esp32-cam-mb-usb/, Pristup: 10.9.2024.

SAŽETAK

Zadatak završnog rada je bio razviti, izgraditi i testirati sustav za automatiziranu ishranu životinja bez prisutnosti ljudskog faktora pomoću IoT sustava. Bilo je potrebno omogućiti daljinsko upravljanje i nadzor, prebrojavanje nahranjenih životinja te uvid u preostalu količinu hrane. Rad se sastoji od ESP32-CAM, koračnog motora s *„driverom“* , ESP32 komponenti napajanih iz prijenosne baterije i kućišta hranilice. Opisano je programsko rješenje sustava uz pomoć web aplikacije i detekcije objekata modelom.

Ključne riječi: ESP32-CAM, hranilica za životinje, IoT, koračni motor, web aplikacija

SUMMARY

AUTOMATED ANIMAL FEEDER

The task of the final paper was to develop, build and test a system for automated animal feeding without the presence of the human factor using the IoT system. It was necessary to enable remote management and monitoring, counting of fed animals and insight into the remaining amount of food. The work consists of an ESP32-CAM, a stepper motor with a driver, ESP32 components powered by a portable battery and a feeder housing. The program solution of the system is described with the help of a web application and the detection of objects by the model.

Keywords: animal feeder, ESP32-CAM, IoT, stepper motor, web application

ŽIVOTOPIS

Adrian Horvat rođen je 17.07.2002. godine u Vinkovcima. Pohađa Osnovnu školu Vladimira Nazora u Vinkovcima od 2009. godine. Obrazovanje nastavlja upisujući Gimnaziju Matije Antuna Reljkovića u Vinkovicma 2017. godine. Nakon završetka srednje škole upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija, prijediplomski studij programsko inženjerstvo 2021. godine.

PRILOZI I DODACI

U daljnjem tekstu, biti će objašnjen kod koji je korišten za programiranje ESP32-CAM modula koji je programiran u Arduino IDE.

P1, potrebne biblioteke i varijable.

```
#include <Arduino.h>
#include <WebServer.h>
#include <WiFi.h>
#include <esp32cam.h>
#include <WiFiManager.h>
#include <Stepper.h>


const int stepsPerRevolution = 2048;
const int motorPin1 = 14;
const int motorPin2 = 12;
const int motorPin3 = 13;
const int motorPin4 = 15;
const int ledPin = 4;
```

Slika 7.1. *Potrebne biblioteke i inicijalizacija varijabli.*

Prema slici 7.1. možemo vidjeti potrebne biblioteke korištene za programiranje kamere. „Arduino.h“ je potrebna biblioteka za za rad s mikrokontrolerima, „WebServer.h“ omogućava kreiranje web servera kamere. „WiFi.h“ i „WiFiManager.h“ služi za spajanje na mrežu i upravljanje Wi-Fi konekcijama. „esp32cam.h“ upravlja kamerom na ESP32-CAM modulu te „Stepper.h“ kontrolira koračni motor. U varijablama su definirani pinovi koračnog motora, broj koraka po revoluciji i pin za ledicu na ESP32-CAM. Na slici 7.2 prikazano je kreiranje potrebnih objekata.

P2, kreiranje objekta koračnog motora i web servera.

```
Stepper myStepper(stepsPerRevolution, motorPin1, motorPin3, motorPin2, motorPin4);
WebServer server(8080);
```

Slika 7.2. *Kreiranje objekta koračnog motora i web servera.*

Također, potrebno je definirati 3 različite rezolucije za kameru, visoku, srednju i nisku

prikazano na slici 7.3.

P3, rezolucije kamere.

```
static auto loRes = esp32cam::Resolution::find(320, 240);
static auto midRes = esp32cam::Resolution::find(350, 530);
static auto hiRes = esp32cam::Resolution::find(800, 600);
```

Slika 7.3. *Definiranje rezolucija kamere.*

Na slici 7.4. može se vidjeti funkcija koja služi za snimanje slike kamerom, ukoliko snimanje ne uspije vraća se HTTP status 503 (servis nedostupan), ako uspije šalje dimenzije slike na serijski monitor i šalje slike klijentu putem HTTP odgovora.

P4, funkcija za služenje slike u JPG formatu.

```
void serveJpg()
{
  auto frame = esp32cam::capture();
  if (frame == nullptr) {
    Serial.println("CAPTURE FAIL");
    server.send(503, "", "");
    return;
  }
  Serial.printf("CAPTURE OK %dx%d %db\n", frame->getWidth(), frame->getHeight(),
                static_cast<int>(frame->size()));

  server.setContentLength(frame->size());
  server.send(200, "image/jpeg");
  WiFiClient client = server.client();
  frame->writeTo(client);
}
```

Slika 7.4. *Funkcija koja služi sliku u JPG formatu.*

Potrebno je napisati funkcije za postavljanje samih rezolucija slika kamere prikazano na slici 7.5.

P5, funkcija za podešavanje rezolucija slike.

```
void handleJpgLo()
{
  if (!esp32cam::Camera.changeResolution(loRes)) {
    Serial.println("SET-LO-RES FAIL");
  }
  serveJpg();
}

void handleJpgHi()
{
  if (!esp32cam::Camera.changeResolution(hiRes)) {
    Serial.println("SET-HI-RES FAIL");
  }
  serveJpg();
}

void handleJpgMid()
{
  if (!esp32cam::Camera.changeResolution(midRes)) {
    Serial.println("SET-MID-RES FAIL");
  }
  serveJpg();
}
```

Slika 7.5. *Funkcije za podešavanje različitih rezolucija slike.*

Setup funkcija na slici 7.6. služi za inicijalizaciju serijske komuikacije na 115200 baud, Konfiguira kameru te provjerava je li uspješno pokrenuta.

P6, setup funkcija.

```
void setup(){
  Serial.begin(115200);
  Serial.println();
  {
    using namespace esp32cam;
    Config cfg;
    cfg.setPins(pins::AiThinker);
    cfg.setResolution(hiRes);
    cfg.setBufferCount(2);
    cfg.setJpeg(80);

    bool ok = Camera.begin(cfg);
    Serial.println(ok ? "CAMERA OK" : "CAMERA FAIL");
  }
```

Slika 7.6. *Setup funkcija.*

Kod na slici 7.7 postavlja brzinu koračnog motora na 10 koraka po sekundi. Zatim koristi

„WiFiManager“ biblioteku za automatsko povezivanje na WiFi mrežu pod nazivom „PetFeederCam“. Ako povezivanje ne uspije, ispisuje prikladnu poruku i provjerava status veze u petlji sve dok ne bude uspješna.

P7, postavljanje brzine okretanja motora i Wi-Fi-a .

```
myStepper.setSpeed(10);

WiFi.mode(WIFI_STA);

WiFiManager wm;
bool res;
res = wm.autoConnect("PetFeederCam");
if(!res) {
  Serial.println("Failed to connect to Pet Cam");
}
while (WiFi.status() != WL_CONNECTED) {
  Serial.print(".");
  delay(500);
}
```

Slika 7.7. *Postavljanje brzine motora i Wi-Fi moda.*

Na slici 7.8. mogu se vidjeti naredbe za ispis IP adrese web servera i putanje do slika u različitim rezolucijama, također može se vidjeti odgovarajući pozivi funkcija kada se pristupi određenoj putanji.

P8, ispis URL-ova za hvatanje slika s kamere na serijski monitor.

```
Serial.print("http://");
Serial.println(WiFi.localIP());
Serial.println("  /cam-lo.jpg");
Serial.println("  /cam-hi.jpg");
Serial.println("  /cam-mid.jpg");

server.on("/cam-lo.jpg", handleJpgLo);
server.on("/cam-hi.jpg", handleJpgHi);
server.on("/cam-mid.jpg", handleJpgMid);
```

Slika 7.8. *Ispis na serijski monitor i povezivanje funkcija s URL putanjama.*

Slika 7.9. prikazuje započinjanje rada web servera, postavljanje ledice kamere u isključeno stanje te loop funkcija koja obrađuje HTTP zahtjeve.

P9, funkcije za web server, ledicu i obradu HTTP zahtjeva.

```
  server.on("/", handleRoot);
  server.begin();
  Serial.println("HTTP server started");

  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, LOW);
}

void loop()
{
  server.handleClient();
}
```

Slika 7.9. *Funkcije za web server, ledicu i obradu HTTP zahtjeva.*

Funkcija za rotaciju koračnog motora je napravljena na jednostavan način, rotira koračni motor za određeno vrijeme u sekundama, dokle god vrijeme nije isteklo motor se pomiče za jedan korak u lijevo. Ovaj kod je prikazan na slici 7.10.

P10, funkcija za rotaciju koračnog motora.

```
void rotateStepper(int durationInSeconds) {

  unsigned long startTime = millis();

  while (millis() - startTime < durationInSeconds * 1000) {
    myStepper.step(-1);
  }
}
```

Slika 7.10. *Funkcija za rotaciju koračnog motora.*

Na slici 7.11. prikazana je handleRoot() funkcija. Ona služi za obradu HTTP zahtjeva. Kada stigne POST zahtjev, funkcija provjerava je li tip zahtjeva POST. Nakon toga preuzima sadržaj parametra „message“ koji se koristi da odredi koju radnju treba izvršiti. Na osnovu te poruke, funkcija kontrolira vrijeme rotacije koračnog motora.

P11, funkcija za obradu HTTP zahtjeva.

```
void handleRoot() {

  if (server.method() == HTTP_POST) {
    String message = server.arg("message");
    Serial.print("Received message: ");
    Serial.println(message);

    if (message.equals("SMALL")) {
      Serial.println("Rotating stepper for small portion");
      server.send(200, "text/plain", "ESP32: Feeding small portion");
      rotateStepper(4);

    } else if (message.equals("MEDIUM")) {
      Serial.println("Rotating stepper for medium portion");
      server.send(200, "text/plain", "ESP32: Feeding medium portion");
      rotateStepper(8);

    } else if (message.equals("LARGE")) {
      Serial.println("Rotating stepper for large portion");
      server.send(200, "text/plain", "ESP32: Feeding large portion");
      rotateStepper(12);

    } else {
      Serial.println("Invalid message received");
    }

    server.send(200, "text/plain", "Message received. ESP32 CAM Connected");

  } else {
    server.send(200, "text/plain", "Hello from ESP32 CAM!");
  }
}
```

Slika 7.11. *Funkcija handleRoot().*

U daljnjem tekstu, biti će objašnjen kod koji je korišten za programiranje Flask web aplikacije koja je implementirana koristeći Python, HTML i CSS u Visual Sudio Code okruženju.

Kod na slici 7.12. postavlja varijable za praćenje količine hrane i brojanja pojave životinja (pas, mačka, ptica), dok koristi IP adresu web servera ESP32-CAM za preuzimanje slika. Također, uključuje vremensku odgodu od 3 minute između detekcija životinja.

P12, biblioteke i varijable web aplikacije.

```python
from flask import Flask, render_template, Response, request, jsonify
import cv2
import urllib.request
import numpy as np
import time

app = Flask(__name__)

# Početna količina hrane (grami)
initial_food_quantity = 500
current_food_quantity = initial_food_quantity

# Brojači za životinje
animal_counts = {"dog": 0, "cat": 0, "bird": 0}

# Postavljanje IP adrese kamere
cam_ip = '192.168.233.176'
url = f'http://{cam_ip}:8080/cam-hi.jpg'

last_detection_time = 0
detection_delay = 180  # Pauza od 3 minute
```

Slika 7.12. *Korištene biblioteke i inicijalizirane varijable.*

Na slici 7.13. učitana je datoteka „coco.names“ iz koje se čitaju nazivi klasa objekata odnosno životinja koje se detektiraju, također navedene su putanje do konfiguracijske datoteke modela i do treniranih mreža modela. Daljnje linije u kodu konfiguriraju ulazne parametre za model podešavajući veličinu ulazne slike, skaliranje piksela i slično.

P13, učitan model, podešavanje parametara.

```python
classNames = []
classFile = 'coco.names'
with open(classFile, 'rt') as f:
    classNames = f.read().rstrip('\n').split('\n')

configPath = 'ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt'
weightsPath = 'frozen_inference_graph.pb'

net = cv2.dnn_DetectionModel(weightsPath, configPath)
net.setInputSize(320, 320)
net.setInputScale(1.0 / 127.5)
net.setInputMean((127.5, 127.5, 127.5))
net.setInputSwapRB(True)
```

Slika 7.13. *Učitavanje OpenCV modela za detekciju.*

Funkcija na slici 7.14. koristi se za obradu slika u stvarnom vremenu s ciljem detekcije životinja i automatskog doziranja hrane. Prvo, funkcija preuzima sliku putem *„URL“*, pretvara je u oblik koji OpenCV može obraditi, a zatim koristi unaprijed treniranu mrežu za detekciju objekata

(životinja) na slici. Kada se detektira pas, mačka ili ptica, slika se ažurira pravokutnikom i oznakom koja označava detektiranu životinju. Na osnovu detektirane životinje, određuje se količina hrane koja će biti izdvojena (mala, srednja, velika porcija), a trenutna količina hrane se umanjuje. Ako ima dovoljno hrane, funkcija šalje naredbu za hranjenje i ažurira status na slici, a ako nema, prikazuje poruku da nema dovoljno hrane. Funkcija kontinuirano vraća obrađene okvire kao JPG slike za prikazivanje putem video prijenosa.

P14, funkcija za detekciju životinja.

```
def gen_frames():
    global current_food_quantity, last_detected_animal, last_portion_type, animal_counts, last_detection_time
    while True:
        imgResponse = urllib.request.urlopen(url)
        imgNp = np.array(bytearray(imgResponse.read()), dtype=np.uint8)
        img = cv2.imdecode(imgNp, -1)

        classIds, confs, bbox = net.detect(img, confThreshold=0.5)

        current_time = time.time()
        if len(classIds) != 0 and (current_time - last_detection_time) >= detection_delay:
            for classId, confidence, box in zip(classIds.flatten(), confs.flatten(), bbox):
                label = classNames[classId - 1]

                if label in ["dog", "cat", "bird"]:
                    cv2.rectangle(img, box, color=(0, 255, 0), thickness=3)
                    cv2.putText(img, label, (box[0] + 10, box[1] + 30), cv2.FONT_HERSHEY_COMPLEX, 1, (0, 255, 0), 2)

                    if label == "dog":
                        food_amount = 50
                        portion_type = "LARGE"
                    elif label == "cat":
                        food_amount = 30
                        portion_type = "MEDIUM"
                    elif label == "bird":
                        food_amount = 15
                        portion_type = "SMALL"

                    if current_food_quantity >= food_amount:
                        current_food_quantity -= food_amount
                        last_detected_animal = label
                        last_portion_type = portion_type
                        animal_counts[label] += 1

                        send_feed_command(portion_type)

                        cv2.putText(img, f"{label} detected, {portion_type} portion: {food_amount}g",
                                    (box[0] + 10, box[1] + 60), cv2.FONT_HERSHEY_COMPLEX, 1, (255, 255, 255), 2)
                        cv2.putText(img, f"Remaining food: {current_food_quantity}g",
                                    (10, img.shape[0] - 10), cv2.FONT_HERSHEY_COMPLEX, 1, (255, 255, 255), 2)
                        else:
                            cv2.putText(img, f"Not enough food for {label}!",
                                        (box[0] + 10, box[1] + 60), cv2.FONT_HERSHEY_COMPLEX, 1, (0, 0, 255), 2)

        ret, buffer = cv2.imencode('.jpg', img)
        frame = buffer.tobytes()

        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')
```

Slika 7.14. *Funkcija za detekciju životinja.*

Vraćanje podataka u JSON formatu s posljednjom detektiranom životinjom, vrstom porcije, preostalom hranom i brojem detekcija svake životinje prikazano je na slici 7.15.

P15, vraćanje podataka za posljednje detektiranu životinju i njihovo prebrojavanje.

```
@app.route('/last_detected')
def last_detected():
    return jsonify({
        "animal": last_detected_animal,
        "portion_type": last_portion_type,
        "remaining_food": current_food_quantity
    })

@app.route('/animal_counts')
def animal_counts_route():
    return jsonify(animal_counts)
```

Slika 7.15. *Vraćanje podataka u JSON formatu.*

Funkcija za punjenje hranilice koja trenutnu vrijednost hrane postavlja na početnu zadanu vrijednost prikazana je na slici 7.16.

P16, funkcija za punjenje hranilice.

```
@app.route('/refillFeeder', methods=['POST'])
def refillFeeder():
    global current_food_quantity
    current_food_quantity = initial_food_quantity
    return jsonify({"remaining_food": current_food_quantity, "message": "Hranilica je napunjena."})
```

Slika 7.16. *Funkcija za punjenje hranilice.*

Funkcija na slici 7.17. šalje HTTP POST zahtjev ESP32-CAM modulu kako bi pokrenula doziranje hrane na osnovu tipa porcije (mala, srednja, velika). Na osnovu vrijednosti „portion_type“, određuje se odgovarajuća naredba, koja se zatim šalje putem HTTP zahtjeva na određenu IP adresu ESP32-CAM. Ako dođe do greške prilikom slanja zahtjeva, greška se hvata i ispisuje.

P17, funkcija za slanje naredbi ESP32-CAM.

```python
def send_feed_command(portion_type):
    if portion_type == "LARGE":
        command = "LARGE"
    elif portion_type == "MEDIUM":
        command = "MEDIUM"
    elif portion_type == "SMALL":
        command = "SMALL"

    try:
        url = f'http://{cam_ip}:8080/'
        data = {'message': command}
        response = urllib.request.urlopen(url, data=urllib.parse.urlencode(data).encode())
        print(f'Sent {command} command to ESP32. Response: {response.read().decode()}')
    except Exception as e:
        print(f'Error sending {command} command: {e}')
```

Slika 7.17. *Funkcija za slanje naredbi ESP32-CAM.*

Funkcija za obradu HTTP POST zahtjeva (prikazana na slici 7.18.), posluživanje određene količine hrane na osnovu vrijednosti predanih varijablom „portion_type“ te sadrži ostalu logiku za izračun i provjeru preostale količine hrane koju vraća u JSON formatu prikazana je na slici 7.18.

P18, funkcija za obradu HTTP POST zahtjeva.

```python
@app.route('/feed', methods=['POST'])
def feed():
    global current_food_quantity
    portion_type = request.form.get('portion_type')

    if portion_type == "LARGE" and current_food_quantity >= 50:
        current_food_quantity -= 50
        send_feed_command("LARGE")
        message = "Velika porcija poslužena."
    elif portion_type == "MEDIUM" and current_food_quantity >= 30:
        current_food_quantity -= 30
        send_feed_command("MEDIUM")
        message = "Srednja porcija poslužena."
    elif portion_type == "SMALL" and current_food_quantity >= 15:
        current_food_quantity -= 15
        send_feed_command("SMALL")
        message = "Mala porcija poslužena."
    else:
        message = "Nema dovoljno hrane."

    return jsonify({"remaining_food": current_food_quantity, "message": message})
```

Slika 7.18. *Funkcija za izračun preostale hrane, provjeru i ispis.*

Na slici 7.19. prikazana je funkcija za vraćanje prikaza HTML stranice, kreiranje HTTP video prijenosa te pokretanje aplikacije.

P19, funkcije za vraćanje prikaza HTML stranice, kreiranje HTTP video prijenosa te pokretanje aplikacije.

```
@app.route('/')
def index():
    return render_template('index.html', food_quantity=current_food_quantity)

@app.route('/video_feed')
def video_feed():
    return Response(gen_frames(), mimetype='multipart/x-mixed-replace; boundary=frame')

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=5000, debug=True)
```

Slika 7.19. *Funkcije za vraćanje prikaza HTML stranice, kreiranje HTTP video prijenosa te pokretanje aplikacije.*

HTML dokument na slici 7.20. prikazuje korisničko web sučelje za hranilicu za životinje, koja također omogućuje ručno hranjenje životinja. Postoji niz gumba za odabir veličine porcija hrane te prikaz preostale količine hrane. Stranica također prikazuje brojač detektiranih životinja (psi, mačke, ptice) te posljednju detektiranu životinju. Na desnoj strani nalazi se video prijenos uživo kamere postavljene na hranilici.

P20, HTML kod za korisničko web sučelje.

```html
<!DOCTYPE html>
<html lang="hr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Hranilica za životinje</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
</head>
<body>
    <header>
        <h1>Hranilica za životinje <img src="{{ url_for('static', filename='paw.png') }}" alt="Paw" class="header-image"></h1>
    </header>

    <div class="main-content">
        <div class="info-content">
            <div class="feeding-controls">
                <h2>Ručno hranjenje:</h2>
                <button onclick="feed('SMALL')">Mala porcija (15g)</button>
                <button onclick="feed('MEDIUM')">Srednja porcija (30g)</button>
                <button onclick="feed('LARGE')">Velika porcija (50g)</button>
                <button onclick="refillFeeder()">Napuni hranilicu</button>
            </div>

            <div class="info-section">
                <h2 id="remaining-food">Preostala količina hrane: {{ food_quantity }}g</h2>
            </div>

            <div id="animal-counts" class="info-section">
                <h2>Prebrojavanje detektiranih i nahranjenih životinja:</h2>
                <p id="dog-count">Psi: 0</p>
                <p id="cat-count">Mačke: 0</p>
                <p id="bird-count">Ptice: 0</p>
            </div>

            <div id="last-detected" class="info-section">
                <h2>Posljednja detektirana životinja:</h2>
                <p id="animal-info">Niti jedna životinja nije detektirana.</p>
            </div>
        </div>

        <div class="video-content">
            <h2>Prijenos kamere uživo:</h2>
            <div class="video-wrapper">
                <img src="{{ url_for('video_feed') }}" width="640" height="480">
            </div>
        </div>
    </div>
```

Slika 7.20. *HTML kod za prikaz web sučelja hranilice za životinje.*

Sljedeći JavaScript kod na slici 7.21. upravlja interakcijom sa serverom i ažurira web sučelje u realnom vremenu. Funkcija „feed(portion_type)“ šalje zahtjev za hranjenje sa specifičnom porcijom i ažurira prikaz preostale hrane i informacije o posljednjoj detektiranoj životinji. Funkcije „updateLastDetected()“ i „updateAnimalCounts()“ redovno dobivaju podatke sa servera i ažuriraju prikaz broja detektiranih životinja i preostale hrane na stranici.

P21, JavaScript kod za korisničko web sučelje.

```
<script>
    function updateDisplay(data) {
        $("#remaining-food").text("Preostala hrana: " + data.remaining_food + "g");
        if (data.animal) {
            const animalText = {
                'dog': 'Pas',
                'cat': 'Mačka',
                'bird': 'Ptica'
            }[data.animal.toLowerCase()] || 'Nepoznata životinja';

            const portionText = {
                'SMALL': 'mala porcija',
                'MEDIUM': 'srednja porcija',
                'LARGE': 'velika porcija'
            }[data.portion_type.toUpperCase()] || 'nepoznata porcija';

            $("#animal-info").text(`${animalText}, ${portionText} servirana.`);
        }
    }
    function feed(portion_type) {
        $.post("/feed", { portion_type }, updateDisplay);
    }
    function updateLastDetected() {
        $.get("/last_detected", updateDisplay);
    }
    function updateAnimalCounts() {
        $.get("/animal_counts", function(data) {
            $("#dog-count").text("Psi: " + data.dog);
            $("#cat-count").text("Mačke: " + data.cat);
            $("#bird-count").text("Ptice: " + data.bird);
        });
    }
    function refillFeeder() {
        $.post("/refillFeeder", function(data) {
            $("#remaining-food").text("Preostala hrana: " + data.remaining_food + "g");
        });
    }
    setInterval(updateLastDetected, 2000);
    setInterval(updateAnimalCounts, 2000);
</script>
```

Slika 7.21. *JavaScript kod za korisničko web sučelje.*

Na slici 7.22. prikazan je CSS kod za HTML dio koda, odnosno kod koji služi za dizajn korisničkog web sučelja stranice za hranilicu životinja.

P22, CSS kod za korisničko web sučelje.

```css
body {
    font-family: Arial, sans-serif;
    background-color: #f8f8f8;
    margin: 0;
    padding: 0;
}

header {
    background-color: #e0e0e0;
    padding: 20px;
    text-align: center;
}

h1 {
    font-size: 2em;
    color: #333;
    display: flex;
    align-items: center;
    justify-content: center;
    margin: 0;
}

.header-image {
    margin-left: 15px;
    height: 50px;
    vertical-align: middle;
}

.main-content {
    display: flex;
    justify-content: space-between;
    padding: 20px;
}

.info-content {
    flex: 2;
    padding: 20px;
}

.video-content {
    flex: 1;
    padding: 20px;
    background-color: #fff;
    border: 1px solid #ddd;
    border-radius: 8px;
}

.feeding-controls {
    margin-bottom: 30px;
}

.info-section {
    background-color: #fff;
    border: 1px solid #ddd;
    border-radius: 8px;
    padding: 15px;
    margin-bottom: 20px;
}

button {
    padding: 10px 20px;
    font-size: 16px;
    margin: 5px;
    cursor: pointer;
    border: none;
    border-radius: 5px;
    background-color: green;
    color: white;
}

button:hover {
    background-color: #015817;
}

.video-wrapper {
    text-align: center;
}

img {
    max-width: 100%;
    height: auto;
    display: block;
}
```

Slika 7.22. *CSS kod za korisničko web sučelje.*