Razvoj sustava za praćenje divljih životinja primjenom Rasberry PI mikroračunala

Žužić, Luka

Istarsko veleučilište
Università Istriana
di scienze applicate

Luka Žužić

RAZVOJ SUSTAVA ZA PRAĆENJE DIVLJIH ŽIVOTINJA PRIMJENOM RASPBERRY PI MIKRORAČUNALA

Završni rad

Pula, rujan 2024.

Istarsko veleučilište
Università Istriana
di scienze applicate

Luka Žužić

RAZVOJ SUSTAVA ZA PRAĆENJE DIVLJIH ŽIVOTINJA PRIMJENOM RASPBERRY PI MIKRORAČUNALA

Završni rad

JMBAG: 0233009048, redoviti student

Studijski smjer: Prijediplomski stručni studij Mehatronike

Predmet: Projektiranje ugrađenih računalnih sustava

Mentor: Deni Vale, mag. phys., pred.

Pula, rujan 2024.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani Luka Žužić, kandidat za prvostupnika inženjera mehatronike ovime izjavljujem da je ovaj Završni rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

U Puli, _____ godine

Potpis: 

IZJAVA O KORIŠTENJU AUTORSKOG DJELA

Ja, Luka Žužić, dajem odobrenje Istarskom veleučilištu – Università Istriana di scienze applicate, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom „RAZVOJ SUSTAVA ZA PRAĆENJE DIVLJIH ŽIVOTINJA PRIMJENOM RASPBERRY PI MIKRORAČUNALA“ koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama. Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, _____ godine.

Potpis: 

SAŽETAK

U ovom radu predstavljen je sustav za praćenje divljih životinja koji uključuje kameru s automatskim okidanjem, poslužitelj s bazom podataka i web stranicom. Kamera je izrađena korištenjem Raspberry Pi 4B mikroračunala, Arduino Nano-a, PIR senzora, Raspberry Pi HQ modula kamere, baterije, pretvarača napona, infracrvene LED ploče za noćno snimanje te Qualcomm LTE 4G modula za povezivanje s poslužiteljem. Sustav omogućava neinvazivno praćenje divljih životinja u stvarnom vremenu, pružajući nadzor i tijekom noćnih sati. Fotografije i videozapisi se prenose na poslužitelj, a web stranica omogućava korisnicima upravljanje kamerom, te pregled, brisanje i preuzimanje sadržaja, čime se osigurava jednostavna interakcija između korisnika i dijelova nadzornog sustava. Arduino Nano upravlja napajanjem Raspberry Pi-a mikroračunala, omogućujući uključivanje i isključivanje uređaja prilikom detekcije kretanja životinje putem PIR senzora. Nakon detekcije Raspberry Pi pokreće fotografiranje ili snimanje videozapisa, a datoteke (fotografije ili videozapisi) se putem LTE modula prenose na poslužitelj. Web stranica je razvijena pomoću Apache2 poslužitelja, Flask razvojnog okvira i SQLite baze podataka za pohranu korisničkih informacija, podataka o kameri i odredišta fotografija. Evaluacija učinkovitosti sustava provedena je u stvarnim uvjetima kako bi se osigurala njegova primjenjivost u zaštiti prirodnih staništa. Zbog mogućnosti dodavanja većeg broja automatskih kamera, sustav omogućava praćenje životinjskih populacija u stvarnom vremenu, što pridonosi boljem razumijevanju njihovih migracijskih obrazaca i ponašanja. Također, razvojem predstavljenog nadzornog sustava stvoreni su preduvjeti za identificiranje i kontrolu invazivnih vrsta te sprječavanje prekomjernog rasta određenih populacija, čime bi ovakvi sustavi za nadzor mogli imati vrlo važnu ulogu za osiguravanje ravnoteže ekosustava.

Ključne riječi: nadzor divljih životinja, kamera s automatskim okidanjem, Raspberry Pi, Arduino, LTE komunikacija, zaštita okoliša

SUMMARY

In this thesis, a wildlife monitoring system is presented, which includes a trap camera, a server with a database, and a website. The camera is built using a Raspberry Pi 4B microcomputer, an Arduino Nano, a PIR sensor, a Raspberry Pi HQ camera module, a battery, a voltage converter, an infrared LED panel for night recording, and a Qualcomm LTE 4G module for server connectivity. The system enables non-invasive real-time wildlife monitoring, providing surveillance even during nighttime. Photos and videos are transmitted to the server, and the website allows users to manage the camera, as well as view, delete, and download content, ensuring seamless interaction between users and the system components. The Arduino Nano controls the power supply to the Raspberry Pi, enabling the device to turn on and off upon detecting animal movement via the PIR sensor. Once motion is detected, the Raspberry Pi initiates photo capturing or video recording, and the files (photos or videos) are transmitted to the server via the LTE module. The website is developed using the Apache2 server, Flask framework, and an SQLite database to store user information, camera data, and photo destinations. The system's efficiency was evaluated under real-world conditions to ensure its applicability in the conservation of natural habitats. Due to the possibility of adding multiple trap cameras, the system enables real-time monitoring of animal populations, contributing to a better understanding of their migration patterns and behavior. Furthermore, the development of this monitoring system lays the groundwork for identifying and controlling invasive species, as well as preventing the overpopulation of certain species, making such monitoring systems potentially crucial for maintaining ecosystem balance.

Keywords: Wildlife monitoring, trap camera, Raspberry Pi, Arduino, LTE communication, environmental protection

Zahvale

Zahvaljujem Patriku Radoloviću, koji je izradio odgovarajuće kućište za kameru različite vremenske uvjete i terene, osiguravajući tako da sustav može funkcionirati pouzdano u svim vanjskim uvjetima.

Sadržaj

Popis oznaka i kratica

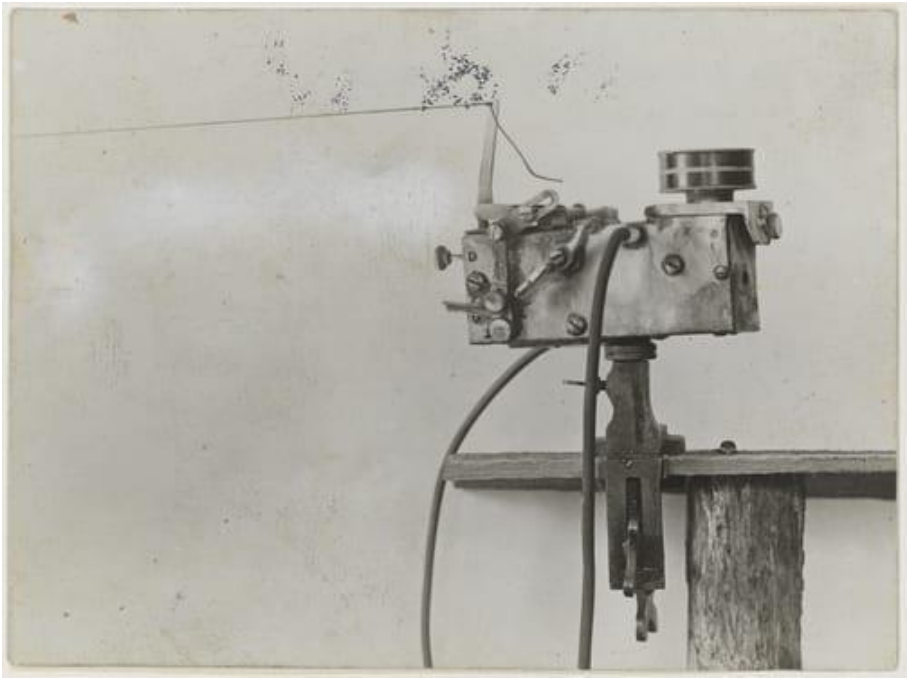
OZNAKA	OPIS	JEDINICA
U	napon	volt [V]
I	struja	amper [A]
VDC	volti istosmjerne struje	volt [V]
VAC	volti izmjenične struje	volt [V]

KRATICA	OPIS
USB	univerzalna serijska sabirnica (*engl. Universal Serial Bus*)
IR	infracrvena (*engl. infrared*)
LTE	Dugoročna evolucina (*engl. Long-term Evolution*)
VPS	virtual privatni poslužitelj (*engl. Virtual Private Server*)
IoT	internet stvari (*engl. Internet of Things*)
CSI	serijsko sučelje kamere (*engl. Camera Serial Interface*)
GPIO	ulaz/izlaz opće namjene (*engl. General Purpose Input/Output*)
PWM	modulacija širine impulse (*engl. Pulse Width Modulation*)
API	programsko sučelje (*engl. Application Programming Interface*)
HTML	prezentacijski jezik za izradu web stranica (engl. HyperText Markup Language)
CSS	kaskadni stilski listovi (*engl. Cascading Style Sheets)*
AJAX	asinkroni JavaScript i XML (*engl. Asynchronous JavaScript and XML*)
SSL	Protokol za prijenos zaštitno-kodiranih podataka (*engl. Secure Sockets Layer*)

PIR	pasivni infracrveni senzor (*engl. Passive Infrared Sensor*)
SQL	strukturirani upitni jezik (*engl. Structured Query Language*)
GPS	globalni položajni sustav (*engl. Global Positioning System*)
DNA	Deoksiribonukleinska kiselina (*engl. Deoxribonucleid acid*)
LED	svjetleća dioda (*engl. Light-emitting diode*)
IDE	integrirano razvojno okruženje (*engl. Integrated Development Environment*)
RTC	sat stvarnog vremena (*engl. Real-Time Clock*)
SDA	serijska podatkovna linija (*engl. Serial Data Line*)
SCL	serijski satna linija (*engl. Serial Clock Line*)
USB-C	univerzalna serijska sabirnica tip C (*engl. Universal Serial Bus type C*)
DC	istosmjerna struja (*engl. Direct Current*)
SSH	sigurna ljuska (*engl. Secure Shell*)
ADC	analogno u digitalno pretvarač (*engl. Analog-to-Digital Converter*)
BOD	detekcija niskog napona (*engl. Brown-out Detection*)
XML	jezik za označavanje podataka (*engl. Extensible Markup Language*)
PLA	polilaktična kiselina (*engl. Polylactic acid*)
ABS	akrilonitril butadien stiren (*engl. Acrylonitrile butadiene styrene*)
PETG	polietilen tereftalat glikol (*engl. Polyethylene terephthalate glycol*)

1. Uvod

Praćenje divljih životinja ključan je alat za lovce, istraživače i stručnjake u području ekologije i zaštite okoliša. Snimanje fotografija i video zapisa putem kamera s automatskim okidanjem predstavlja jednu od najraširenijih suvremenih metoda za neinvazivno prikupljanje podataka o divljim životinjama. Prva takva kamera izumljena je još davnih 1890-ih od strane George Shiras-a III i prikazana je na slici 1. Telemetrija, koja koristi radio uređaje ili globalni položajni sustav (*engl. Global Positioning System*, GPS) pričvršćen na životinje, omogućuje praćenje njihovih migracijskih obrazaca, ponašanja i korištenja staništa. Označavanje i praćenje pomoću prstenovanja ili oznaka pomaže u identifikaciji pojedinih životinja i analizi njihovog kretanja kroz prostor i vrijeme. Analiza uzoraka iz izmeta, posebno analiza deoksiribonukleinske kiseline (*engl. Deoxyribonucleic acid*, DNA), ova analiza omogućuje identifikaciju prisutnosti vrsta na određenim područjima i praćenje njihovih prehrambenih navika. Zvučna analiza, koja se temelji na snimanju i analizi zvukova koje životinje proizvode, pruža važne informacije o njihovim komunikacijskim obrascima i prisutnosti u određenom području (Žužić i Vale, 2024).



Slika 1: Prva kamera za fotografiranje divljih životinja s automatskim okidanjem. Izvor: https://naturespy.org/camera-traps-science/

Ovaj rad usredotočen je na opis vlastitog sustava koji koristi automatske kamere za snimanje divljih životinja u njihovom prirodnom okruženju, aktivirane detekcijom putem pasivnog infracrvenog senzora (*engl. Passive Infrared Sensor*, PIR). Snimljene fotografije se učitavaju na poslužitelj, čime postaju dostupne korisnicima putem web sučelja. U ovom radu koristi se Raspberry Pi 4B mikroračunalo i Arduino Nano mikroupravljač kao temeljne komponente za izradu automatske kamera za praćenje životinja.

Raspberry Pi 4B je moćno i svestrano mikroračunalo koje je postalo popularno za razvoj raznih automatiziranih sustava i internet stvari (*engl. Internet of Things*, IoT) projekta radi njegovih karakteristika, kao što su snažan procesor, mogućnost rada sa standardnim operativnim sustavima kao što je Linux, široka podrška za periferne uređaja putem univerzalne serijske sabirnice (*engl. Universal Serial Bus,* USB), ali i izrađenih perifernih uređaja kao što je Raspberry Pi HQ kamera modul koji se spaja putem utora za serijsko sučelje kamere (*engl. Camera Serial Interface*, CSI) na samom Raspberry Pi-u (Raspberry Pi Foundation, n.d.). Raspberry Pi platforma pruža izuzetnu fleksibilnost i funkcionalnost, omogućujući implementaciju kompleksnih algoritama za obradu podataka i slike, te podršku za mrežnu komunikaciju, što ga čini idealnim za razvoj ovakvog sustava. Arduino Nano mikroupravljač je jednostavan i vrlo učinkovit za rad sa senzorima i aktuatorima, zbog svoje male veličine i niske potrošnje energije koristi se u projektima gdje je prostor ograničen, a potrebna je pouzdana kontrola ulaznih i izlaznih signala.

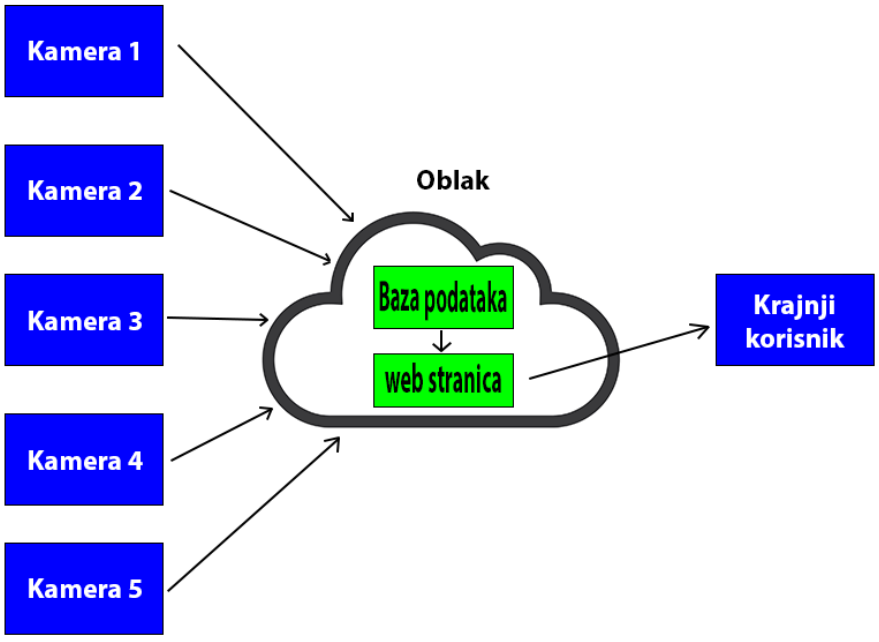
Softverski dio ovog projekta oslanja se na programski jezik Python i Bash skripte. Python je jednostavan programski jezik i ima široku primjenu u znanstvenim i inženjerskim projektima. Radi svoje bogate biblioteke modula i alata koji olakšavaju integraciju sa senzorskim i mrežnim komponentama, ali također omogućuju brzo i učinkovito pisanje koda za prikupljanje podataka, njihovu obradu i slanje na udaljene poslužitelje (Oliphant, 2007). Bash skripte koriste se za automatizaciju zadataka na operativnom sustavu Linux, što uključuje upravljanje datotekama, mrežnim postavkama i pokretanjem različitih kodova kao Python, C, C++ i drugi (Cai i Arney, 2018). Ukratko Bash skripte omogućuju jednostavnu i učinkovitu automatizaciju sustava čime se povećava učinkovitost i pouzdanost sustava. Inspiracija za ovaj projekt potječe od mog oca, lovca koji koristi slične sustave za nadzor divljih životinja.

Većina sustava na tržištu je bazirana na 2G mreži koja često nije pouzdana zbog niske brzine prijenosa podataka i slabije pokrivenosti signala, prijenos velikih količina podataka poput fotografija visoke rezolucije. Primjer moderne kamere s automatskim okidanjem prikazan je na slici 2. Cilj ovog rada je razviti napredni sustav koji koristi LTE (*engl. Long-term Evolution*) 4G tehnologiju, čime se osigurava bolja povezanost i pouzdanost. LTE 4G mreža omogućava brži prijenos podataka, veću pokrivenost i stabilniju vezu, što je ključno za praćenje i nadzor divljih životinja u stvarnom vremenu (Khan, Chowdhury i Nandy, 2023).

Slika 2: Primjer moderne kamere za fotografiranje divljih životinja s automatskim okidanjem. Izvor: https://naturespy.org/camera-traps-science/

2. Osnovne karakteristike sustava za nadzor divljih životinja

Sustav za nadzor divljih životinja razvijen u svrhu ovog završnog rada predstavlja naprednu platformu za automatsko prikupljanje i obradu podataka, s ciljem praćenja životinja u njihovom prirodnom okruženju, omogućujući lovcima i istraživačima pristup vrijednim informacijama u stvarnom vremenu, te pri minimalnom ometanju divljih životinja. Svaka kamera ovog sustava povezana je sa centralnim poslužiteljem putem LTE 4G mreže, te je opremljena sa PIR senzorom za detekciju pokreta, visoko kvalitetnim kamera modulom za snimanje fotografija i videozapisa, te pločom infracrvenih svjetlećih dioda (*engl. Light-Emitting Diode*, LED) koja omogućuje snimanje fotografija noću. Poslužitelj prima sve fotografije te ih automatski pohranjuje u bazu podataka. Baza podataka organizira sadržaj prema vremenu i lokaciji snimanja, što korisnicima omogućuje lakše pretraživanje i pristup relevantnim informacijama. Korištenje web sučelja omogućuje korisnicima pristup podacima u stvarnom vremenu s bilo koje lokacije, čime se značajno poboljšava učinkovitost praćenja i istraživanja. Arhitektura predstavljenog sustava prikazana je na slici 3.



Slika 3*: Arhitektura sustava.*

Osim osnovne funkcionalnosti praćenja, ovaj sustav omogućuje ovaj sustav daje lovcima i istraživačima mogućnost razlikovanja, prebrojavanja i praćenja kretanja jedinki divljih životinja kroz analizu snimljenih fotografija, mogu identificirati različite vrste i jedinke, što pomaže lovcima i istraživačima u bilježenju populacije. Lovcima i istraživačima to omogućuje praćenje migracijskih ruta i sezonskih kretanja životinja, pružajući precizne podatke o njihovim staništima i ponašanju. Ove informacije ključne su za očuvanje biološke raznolikosti. Implementacija tehnologije koja omogućuje prikupljanje podataka bez fizičkog kontakta sa životinjama dodatno smanjuje rizik od narušavanja njihovog prirodnog ponašanja, što pridonosi kvaliteti dobivenih podataka i točnosti u procjenama stanja populacije.

U sljedećim poglavljima ovog završnog rada detaljno će se obraditi ključni dijelovi sustava za praćenje divljih životinja. U 3. poglavlju bit će opisane dvije temeljne komponente sustava, mikroračunalo Raspberry Pi i mikroupravljač Arduino Nano, s posebnim naglaskom na njihove tehničke karakteristike te ulogu unutar sustava. 4. poglavlje posvećeno je hardverskim komponentama automatske kamere, gdje će biti obrađene sve korištene komponente sustava, način njihovog međusobnog povezivanja, kao i proces izrade kućišta korištenjem tehnologije 3D printanja. U 5. poglavlju bit će detaljno prikazana softverska implementacija sustava, uključujući programski kod za Arduino Nano, Raspberry Pi, te dijagram toka rada cijelog sustava. 6. poglavlje bavit će se poslužiteljem, njegovom strukturom, te izradom web stranice s objašnjenjem funkcionalnosti i implementiranog koda prezentacijskog jezika za izradu web stranica (*engl. HyperText Markup Language*, HTML). U 7. poglavlju obradit će se metode mjerenja, s posebnim naglaskom na prikupljanje i obradu podataka. 8. poglavlje prikazat će rezultate mjerenja vremena potrebnog za slanje fotografija i videa, kao i analizu fotografija snimljenih tijekom dana i noći. Nakon toga slijedi diskusija o postignutim rezultatima, nedostacima i prijedlozima za unapređenje, te na kraju zaključak.

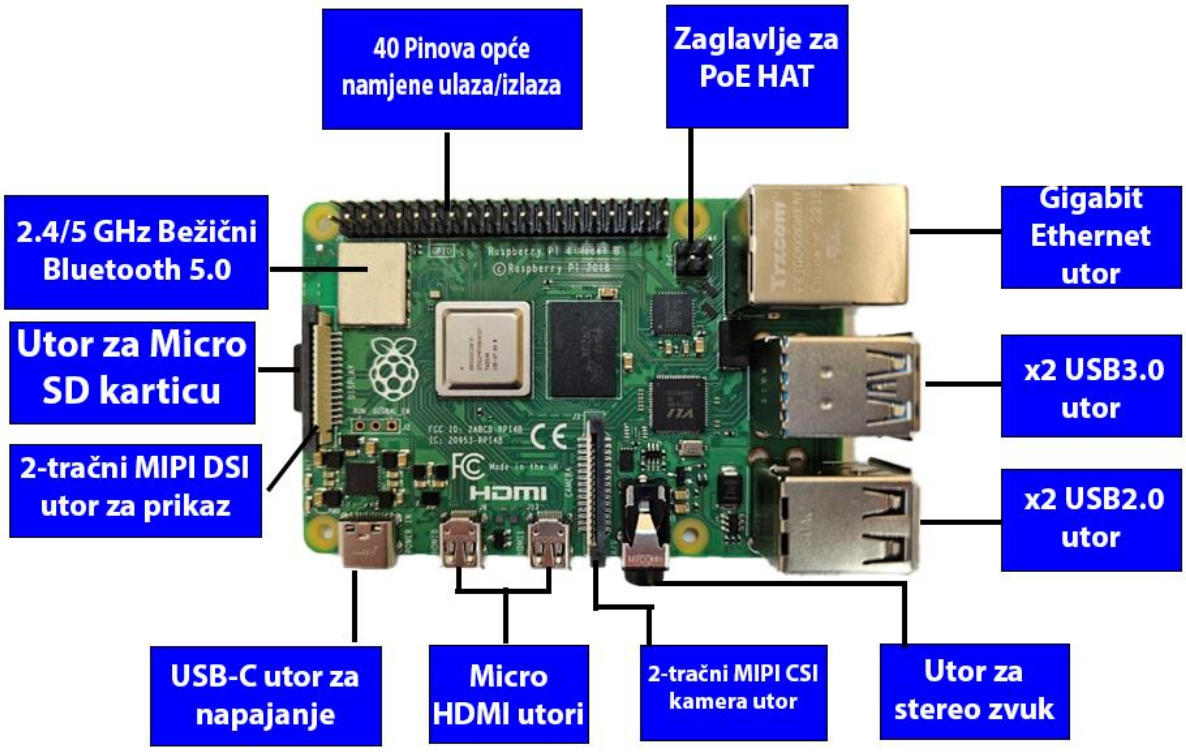
3. Mikroračunalo i mikroupravljač

3.1. Raspberry Pi mikroračunalo

Raspberry Pi je linija malih računala koja je razvila Raspberry Pi Foundation, neprofitna organizacija sa sjedištem u Velikoj Britaniji (Raspberry Pi Foundation, n.d.). Raspberry Pi postala je izuzetno popularna platforma zahvaljujući svojoj kombinaciji niske cijene, snažnih performansi i fleksibilnosti u primjeni. Prvi model, Raspberry Pi Model B, izašao je 2012. godine s ciljem da bude pristupačno mikroračunalo u svrhu edukacije u području računalnih znanost i elektronike (Ellis, Thompson i Haskell-Downland, 2019). Nakon prvog modela Raspberry Pi Foundation je nastavio sa razvojem novih modela mikroračunala, te predstavio idućih godina sljedeće modele: Raspberry Pi 2, 3, 4 i najnoviji model 5. Zahvaljujući svojoj pristupačnoj cijeni i svestranosti, Raspberry Pi pronašao je primjenu u raznim područjima, uključujući edukaciju, industriju, istraživanje i hobi projekte. U obrazovanju je Raspberry Pi postao neizostavan alat za učenje programiranja i elektronike, često korišten u školama i na fakultetima kao dio STEM (*engl. Science, Technology, Engineering, Mathematics*) kurikuluma (Balon i Simic, 2019). U industriji, koristi se za razvoj IoT rješenja, prototipiranje i automatizaciju. Inženjeri i hobi entuzijasti koriste Raspberry Pi za izradu pametnih kućnih uređaja, robota, malih poslužitelja i mrežnih uređaja, te za prikupljanje i analizu podataka u različitim istraživačkim projektima (Saari, Baharudin i Hyrynsalmi, 2017).

3.1.1. Specifikacije

U ovom radu koristi se Raspberry Pi 4B mikroračunalo koje ima četverojezgreni ARM Cortex-A72 procesor s radnim taktom od 1.5 GHz, 4GB LPDDR4-3200 SDRAM-a, USB-C utor za napajanje, CSI utor za kamera modul, dva micro-HDMI utora koji podržavaju dvostruki 4Kp60 video izlaz, dva USB 3.0 utora, dva USB 2.0 utora, gigabit Ethernet, dvopojasni 802.11ac Wi-Fi i Bluetooth 5.0 i četrdeset pinova opće namjene ulaza/izlaza (*engl. General Purpose Input/Output*, GPIO) za proširenje funkcionalnosti putem dodatnih modula i senzora, što ga čini idealnim za širok spektar aplikacija, od kućne automatizacije do industrijskih rješenja (Raspberry Pi Foundation, n.d.). Utori mikroračunala su također prikazani na slici 4.



Slika 4: Topologija Raspberry Pi 4B.

3.1.2. Programski jezici

Programiranje na Raspberry Pi-u moguće je na mnoštvu različitih programskih jezika. Python je jedan od najpopularnijih i najčešće korištenih programskih jezika na ovoj platformi radi svoje jednostavnosti, čitljivosti i široke primjene unutar zajednice. Bogata biblioteka modula je također jedna od velikih prednosti ovog programskog jezika, biblioteke poput GPIO Zero omogućavaju laku integraciju raznih senzora i aktuatora, omogućujući korisnicima da jednostavno upravljaju GPIO pinovima (Watkiss, 2016). Pored Pythona, i drugi programski jezici su podržani, poput C, C++, Java, Scratch, Ruby, Perl i Lua. C i C++ koriste se za aplikacije kojima su potrebne visoke performanse i gdje je potrebna nisko razinska kontrola nad hardverom. C je poznat po svojoj brzini i učinkovitosti, to ga čini odličnim za razvoj ugrađenih sustava i aplikacija koje rade u pravom vremenu, no na Raspberry Pi-u najčešće se koristi za pisanje upravljačkih programa i za rad sa hardverskim komponentama preko direktnog pristupa memoriji. C++ proširuje mogućnosti C jezika dodavanjem objektno orijentiranih značajki. C++ se najčešće koristi za razvoj aplikacija koje zahtijevaju složene podatkovne strukture i algoritme. Za pokretanje C/C++ koda potreban je alat poput GCC-a (*engl. GNU Compiler Collection*) za kompilaciju koda i GDB za

debuggiranje (Loukides i Oram, 1996).

3.1.2.1. Podržani operacijski sustavi

Što se tiče operacijskih sustava, jedan od najpopularnijih za Raspberry Pi je Raspberry Pi OS (prije poznat kao Raspbian), koji je specifično prilagođen za ARM arhitekturu procesora. Raspberry Pi OS je službeni operacijski sustav razvijen od strane Raspberry Pi Foundation i optimiziran je kako bi pružio dobro iskustvo rada na Raspberry Pi uređajima. Osim Raspberry Pi OS-a, moguće je instalirati i druge operacijske sustave poput Windows 10 IoT Core-a i različitih distribucija Linuxa. Windows 10 IoT Core je verzija Windows operacijskog sustava posebno dizajnirana za IoT uređaje, odnosno pruža podršku za osnovne funkcionalnosti potrebne za rad s IoT uređajima (Microsoft, 2021). Također kao što je prije spomenuto moguće je instalirati i različite distribucije Linuxa kao što su Ubuntu, Fedora, Arch, od kojih Ubuntu čak ima i prilagođenu verziju operacijskog sustava specifično za Raspberry Pi (Hart-Davis, 2017).

3.1.3. Povezivanje

Povezivanje na Raspberry Pi 4B moguće je na par načina, ali najjednostavniji načini su: 1) spajanje putem micro-HDMI kabela, i 2) spajanje putem sigurne ljuske (*engl. Secure Shell*, SSH). Prvi predstavlja jednostavan proces gdje je potreban micro-HDMI kabel koji se priključi na odgovarajući utor na Raspberry Pi-u, te je drugi kraj potreban biti prikladan uz utore koji su dostupni na monitoru gdje je željeno dobiti izlaz. Nakon dobivenog video izlaza potrebno je spojiti periferiju poput tipkovnice i miša, te je Raspberry Pi spreman za rad kao klasično računalo (naravno uz pretpostavku da je instaliran operacijski sustav). Drugi način omogućava daljinsko upravljanje Raspberry Pi-em putem komandne linije na Windows operacijskom sustavu, odnosno terminala na Linux operacijskom sustavu. Za spajanje putem SSH-a potrebno je računalo na kojem će se povezivati putem komandne linije, no povezivanje je moguće jedino ako su Raspberry Pi i računalo spojeni na istu internetsku mrežu, ili ako je IP adresa Raspberry Pi-a javna. Povezivanje se obavlja u komandnoj liniji koristeći komandu „ssh <korisničko-ime>@<IP-adresa>“, potrebno je zamijeniti „<korisničko-ime>“ sa pravim korisničkim imenom zadanim prilikom podešavanja sustava, te <IP-adresa> sa

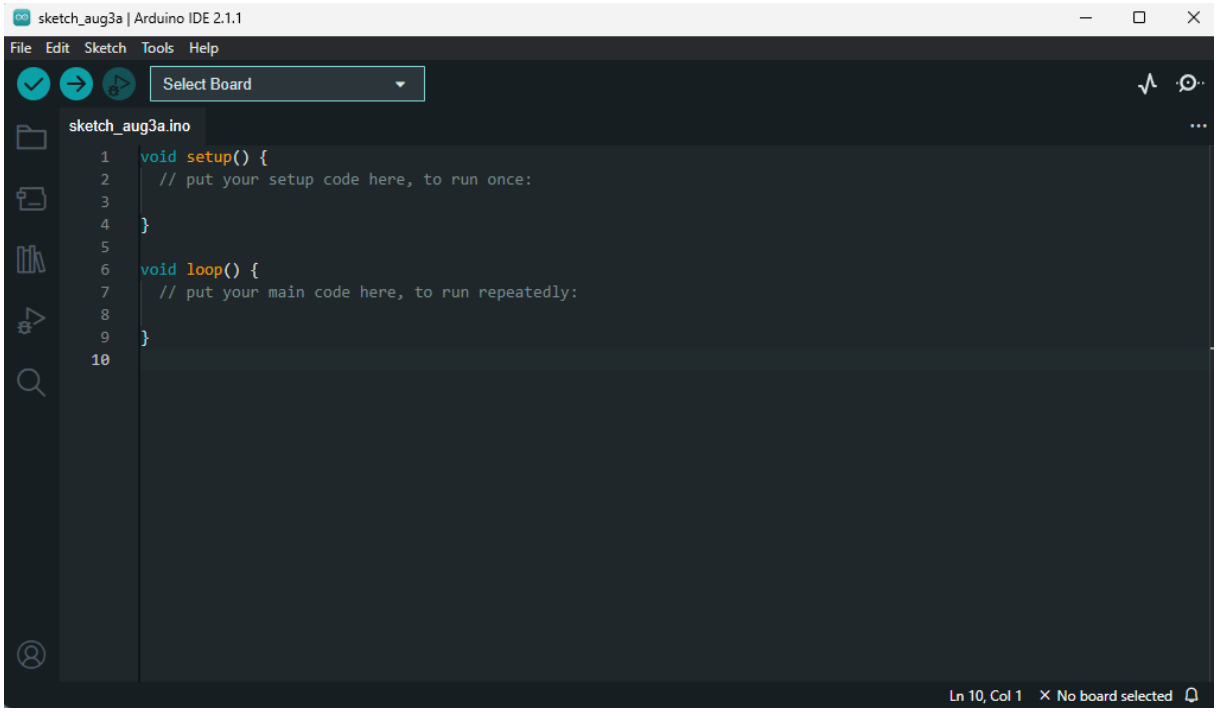
odgovarajućom javnom IP adresom ukoliko oba računala nisu na istoj mreži, ili ako jesu na istoj mreži potrebno je upisati lokalnu IP adresu koju je moguće pronaći putem raznih aplikacija ili putem rutera (SSH Communications Security, 2024).

3.2. Arduino mikroupravljač

Arduino je otvorena platforma za izradu prototipova i jednostavnih projekata, sastoji se od hardvera i softvera namijenjenih entuzijastima, inženjerima, studentima i učenicima za razvoj elektroničkih projekata (Galadima, 2014). Kombinacijom hardvera i softvera omogućava jednostavno programiranje mikroupravljača. Mikroupravljač je mali računalni čip koji upravlja ulazima i izlazima na ploči, omogućavajući korisnicima da komuniciraju s vanjskim komponentama kao što su senzori, motori, LED diode i mnoštvo drugih elektroničkih uređaja (Hodgson, 2021). Sve to je omogućeno korištenjem jednostavnog modela ulaza i izlaza (I/O) za interakciju s elektroničkim uređajima. Platforma je nastala 2005. godine nakon dugogodišnjeg razvoja na Ivrea Interaction Design institutu u Italiji, s ciljem olakšavanja stvaranja interaktivnih projekata i kao obrazovni alat (Pan i Zhu, 2018). Radi svoje pristupačnosti i jednostavnosti korištenja Arduino platforma je postala veoma popularna kod početnika, ali i kod profesionalaca koji žele brzo i jednostavno razvijati projekte iz područja robotike, automatizacije, interneta stvari, i mnogo drugih aplikacija.

3.2.1. Arduino IDE

Arduino integrirano razvojno okruženje (*engl. Integrated Development Environment*, IDE) prikazano na slici 5. je besplatni softver otvorenog koda razvijen od proizvođača Arduina, služi za jednostavno pisanje, kompiliranje i učitavanje koda na Arduino ploče. Dizajniran je s ciljem olakšavanja pisanja programskog koda za Arduino uređaje, te učitavanje istog na željeni Arduino uređaj. Program također dolazi integriran sa mnoštvom primjera kodova raznih namjena, te serijskim monitorom koji služi za komunikaciju sa Arduinom (Arduino, 2024).



Slika 5: Arduino IDE.

3.2.2. Programski jezici

Programiranje na Arduinu oslanja se na prilagođene programske jezike kako bi se olakšala interakcija s hardverom i kako bi se omogućila brza izrada prototipova za elektroničke projekte. Koristi se kombinacija C i C++ programskih jezika u obliku pojednostavljenog programskog jezika, Arduino C. Ovi jezici su moćni alati za razvijanje aplikacija, od jednostavnih amaterskih projekata do složenih sustava koji zahtijevaju veliki nivo učinkovitosti i kontrole nad hardverom (Badamasi, 2014).

3.2.2.1. C programski jezik

C je visoko-razinski programski jezik koji je razvijen početkom 1970-ih u Bell Labsu od strane Dennisa Ritchieja. Dizajniran je za razvoj sustavnog softvera i pruža efikasnu kontrolu nad hardverom, što ga čini idealnim za razvoj operativnih sustava, kompajlera, uređaja i drugih aplikacija koje zahtijevaju direktan pristup memoriji i hardverskim resursima. C programski jezik poznat je po svojoj jednostavnoj i moćnoj strukturi koda koja omogućuje modularno i proceduralno programiranje (Ritchie, 1993). Osnovna struktura C programa sastoji se od funkcija, s *main()* funkcijom kao ulaznom točkom. Korištenje blokova koda i funkcija omogućuje razvoj organiziranog i lako održivog

koda. C knjižnice (biblioteke) su integralni dio programiranja u C jeziku, pružajući gotove funkcije koje programerima omogućuju jednostavno obavljanje uobičajenih zadataka. Standardna knjižnica C jezika (*engl. C Standard Library*), uključuje funkcije za manipulaciju nizovima, upravljanje memorijom, ulazno-izlazne operacije, rad s datotekama, te matematičke operacije. Osim standardnih, postoje mnoge dodatne knjižnice za specifične svrhe kao što su mrežna komunikacija, obrada podataka i grafičko sučelje. Knjižnice omogućuju programerima da izbjegnu pisanje koda od nule, što ubrzava razvoj i smanjuje mogućnost grešaka. Također, C podržava korištenje vanjskih knjižnica, što dodatno proširuje njegove mogućnosti, omogućujući integraciju s različitim tehnologijama i platformama. Deklaracija i definicija varijabli, korištenje petlji i uvjetnih izraza, te upravljanje greškama su osnovne komponente C programiranja koje omogućuju kontrolu toka programa i izvršavanje kompleksnih operacija. Na Arduinu, C omogućava preciznu kontrolu nad mikroupravljačem, što ga čini idealnim za mnoštvo primjena koje zahtijevaju brzinu i učinkovitost. Radi svojeg principa rada na niskom nivou, C omogućuje maksimalnu kontrolu nad svim resursima hardvera kao što su memorija i procesorska iskorištenost što je veoma bitno kod projekata sa ograničenim resursima.

3.2.2.2. C++ programski jezik

C++ je objektno-orijentirani programski jezik razvijen kao nadogradnja C jezika početkom 1980-ih od strane Bjarnea Stroustrupa. Dizajniran je kako bi omogućio složenije strukture podataka i apstrakciju kroz klase i objekte, zadržavajući pritom efikasnost C jezika (Jordan, 1990). C++ kombinira proceduralno i objektno-orijentirano programiranje, što omogućuje stvaranje modularnog i lako održivog koda. Jezik nudi moćne alate poput enkapsulacije, nasljeđivanja i polimorfizma, što olakšava rad s kompleksnim sustavima. Standardna biblioteka C++ jezika (*engl. Standard Template Library,* STL) nudi širok raspon algoritama i struktura podataka, uključujući vektore, liste, skupove, mape i iteratore, čime se ubrzava razvoj i povećava efikasnost. Knjižnice su integralni dio C++ jezika, pružajući gotove komponente koje programerima omogućuju izbjegavanje pisanja koda od početka. Osim STL-a, C++ podržava razne druge knjižnice za specifične svrhe kao što su mrežna komunikacija, grafičko sučelje i znanstvene proračune. C++ je popularan u razvoju aplikacija koje zahtijevaju visoke performanse, kao što su igre, financijski sustavi, i ugrađeni sustavi.

Njegova podrška za generičko programiranje i upravljanje memorijom omogućuje programerima da razvijaju robusne i efikasne aplikacije s visokom razinom kontrole nad resursima. Na Arduinu, C++ je ključan za razvoj složenih projekata gdje se traže fleksibilnost i skalabilnost. Korištenjem C++, programeri mogu kreirati prilagođene knjižnice i klase koje mogu obraditi specifične zadatke, poput upravljanja raznim senzorima i modulima. Na primjer, klase omogućuju enkapsulaciju funkcionalnosti senzora, što pojednostavljuje korištenje istih senzora u različitim dijelovima programa ili u različitim projektima (Dmitrović, 2020).

3.2.2.3. Arduino C

Arduino C je pojednostavljena verzija C/C++ jezika, specifično dizajnirana za upotrebu na Arduino platformi. Arduino C pojednostavljuje proces programiranja omogućujući korisnicima da brzo nauče osnove mikroupravljačkog programiranja. Arduino skice se pišu u jednostavnom uređivaču koda unutar Arduino IDE-a, gdje programeri koriste unaprijed definirane funkcije za jednostavnu interakciju s hardverom, kao što su „digitalWrite()“, „analogRead()“, i „pinMode()“. Ovaj pristup omogućuje korisnicima svih razina iskustva da brzo razviju i testiraju svoje projekte. Arduino C dolazi s velikim brojem ugrađenih knjižnica koje olakšavaju integraciju i rad s perifernim uređajima, što omogućuje jednostavnu kontrolu senzora, aktuatora, i raznih komunikacijskih modula (Badamasi, 2014).

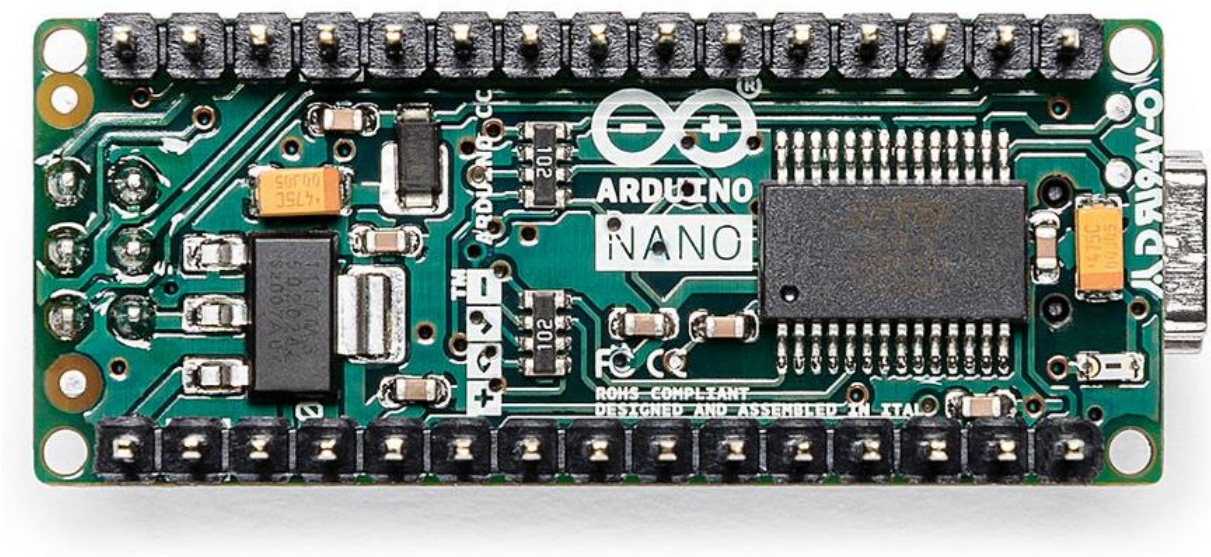
3.2.3. Odabir Arduino ploče

Odabir odgovarajuće Arduino ploče zahtijeva pažljivu analizu tehničkih zahtjeva koje ploča mora zadovoljiti kako bi projekt bio funkcionalan. U ovom završnom radu, ključni uvjeti uključuju mogućnost ploče da podržava stanje dubokog sna (*engl. Deep sleep*). Ovo stanje omogućava ploči prelazak u stanje vrlo niske potrošnje energije, što je posebno važno u ovom sustavu budući da baterija koja napaja sustav ima ograničen kapacitet bez mogućnosti solarnog punjenja. Ploča se mora moći automatski probuditi putem vanjskog signala i vratiti u normalan rad bez gubitka funkcionalnosti. Osim toga, ploča mora imati sposobnost upravljanja relejem putem svojih digitalnih priključaka, što je ključno za kontrolu perifernih uređaja u sklopu projekta. Ovi tehnički zahtjevi su postavljeni kako bi osigurali pouzdan i energetski učinkovit sustav sposoban za

kontinuirani rad u promjenjivim uvjetima. U obzir su uzete tri ploče, Arduino Nano, Arduino Mega i Arduino Uno. Sve ploče ispunjavaju zadane tehničke zahtjeve, ali sa tri različita cjenovna ranga.

3.2.3.1. Arduino Nano

Arduino Nano je kompaktna i funkcionalno bogata ploča koja se temelji na Atmega328P mikroupravljaču s radnim taktom od 16 MHz, slična Arduino Unu, ali u značajno manjem formatu. Sastoji se od 14 digitalnih ulaza/izlaza (od kojih šest ima mogućnost rada kao izlazi za modulator širine impulsa (*engl. Pulse Width Modulation*, PWM)), 8 analognih ulaza, 2 KB SRAM-a, 32 KB flash memorije i 1 KB EEPROM-a. Radi svojih malih dimenzija idealan je za integraciju u projektima koji imaju ograničen prostor. Ploča se može napajati putem USB kabela ili vanjskog izvora (Arduino, n.d.).



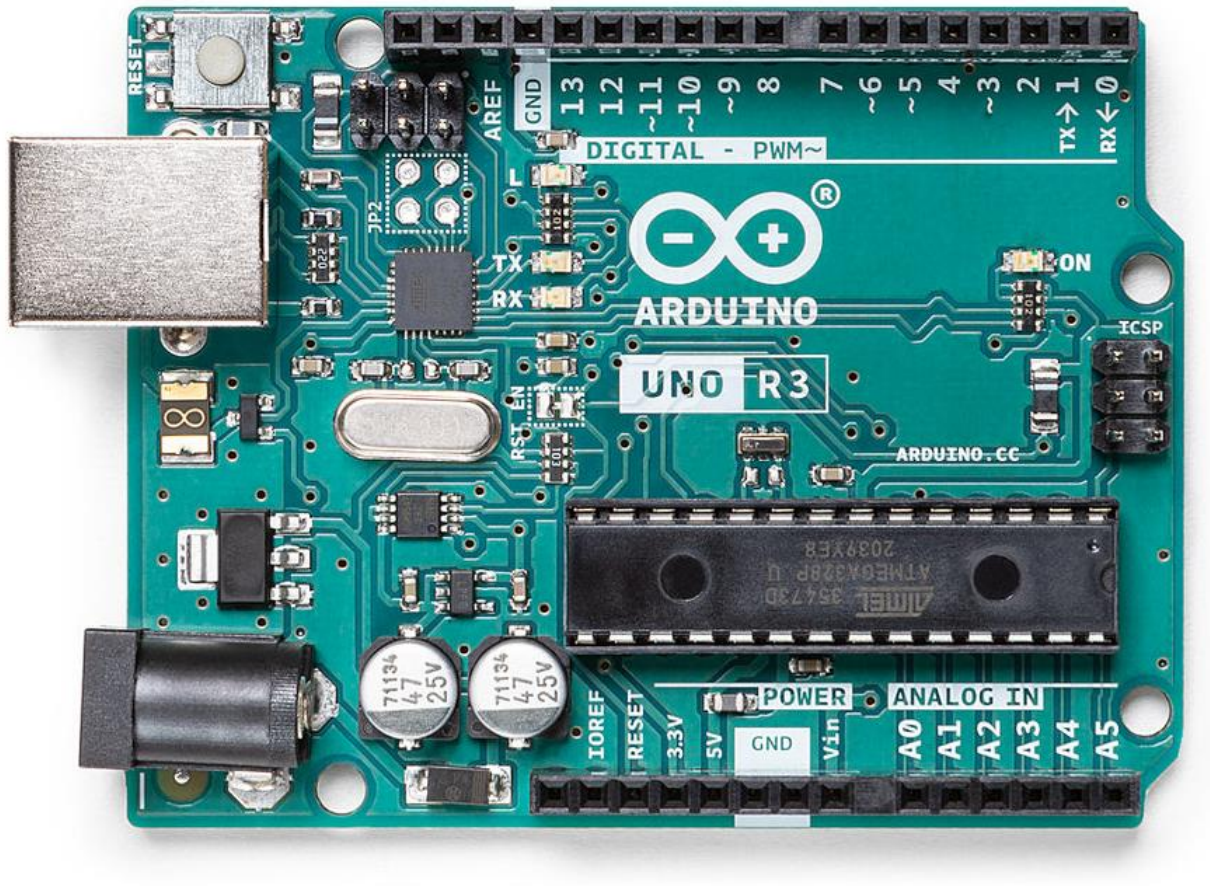
Slika 6: Prikaz Arduino Nano ploče.

Izvor: https://store.arduino.cc/products/arduino-nano

3.2.3.2. Arduino Uno

Arduino Uno je jedna od najčešće korištenih Arduino ploča, idealna je za početnike i jednostavne projekte. Temelji se na Atmega328P mikroupravljaču s radnim taktom od 16 MHz, 2 KB SRAM-a, 32 KB flash memorije i 1 KB EEPROM-a. Sastoji se od 14

digitalnih ulaza/izlaza (od kojih šest ima mogućnost rada kao PWM izlaza), te 6 analognih ulaza. Ploča se može napajati putem USB kabela ili vanjskog izvora napajanja (Arduino, n.d.).

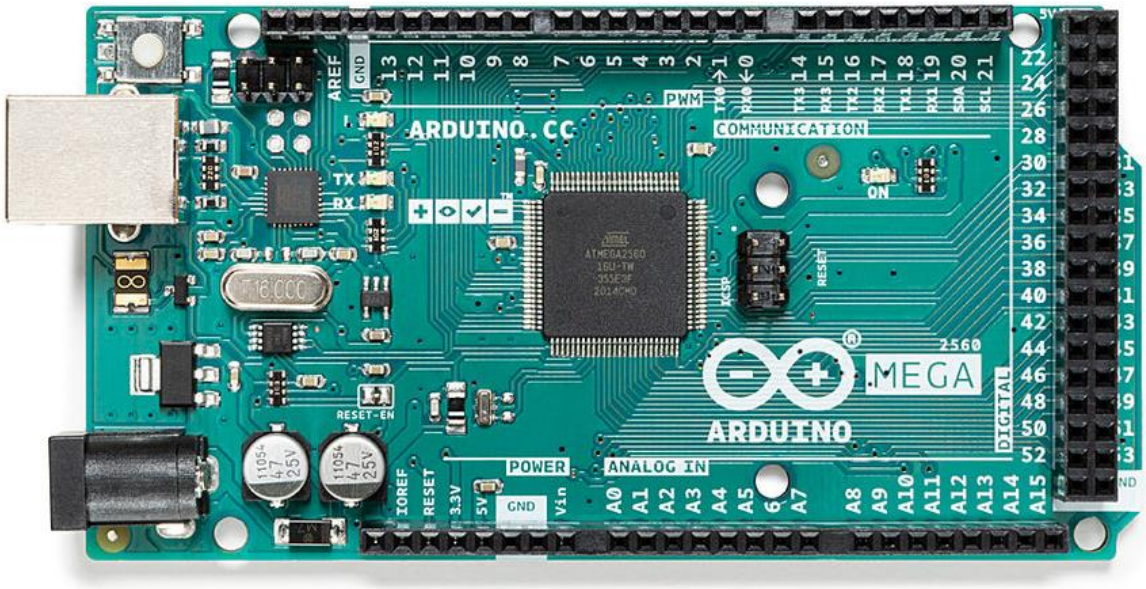


Slika 7: Prikaz Arduino Uno ploče.

Izvor: https://store.arduino.cc/products/arduino-uno-rev3

3.2.3.3. Arduino Mega

Arduino Mega je puno učinkovitija verzija Arduino Uno ploče, namijenjena je složenijim projektima koji zahtijevaju mnoštvo ulazno-izlaznih priključaka i memorije. Sustav je baziran na Atmega2560 mikroupravljaču s 54 digitalna ulaza/izlaza (od kojih 15 može raditi kao PWM izlazi), 16 analognih ulaza i 4 serijska porta. Ploča ima 256 KB flash memorije, 8 KB SRAM-a i 4 KB EEPROM-a, što je čini idealnom za projekte koji zahtijevaju rad s mnoštvom senzora, aktuatora ili kompleksnu obradu podataka (Arduino, n.d.).



Slika 8: Prikaz Arduino Mega ploče.

Izvor: https://store.arduino.cc/products/arduino-mega-2560-rev3

3.2.3.4. Usporedba ploča

Radi svojih malih dimenzija, niske potrošnje energije i zadovoljavanja svih tehničkih uvjeta projekta, Arduino Nano se pokazuje kao idealna ploča za upravljanje PIR senzorom i paljenjem i gašenjem Raspberry Pi-a. Njegova kompaktna veličina omogućuje jednostavnu integraciju u sustave s ograničenim prostorom, dok sposobnost preciznog upravljanja digitalnim priključcima osigurava pouzdanu kontrolu nad perifernim uređajima. Arduino Nano podržava sve potrebne funkcije, uključujući stanje dubokog sna, što je ključno za održavanje energetske učinkovitosti u projektu. Ova ploča nudi optimalnu kombinaciju performansi i funkcionalnosti, što je čini nezamjenjivim dijelom sustava dizajniranog u ovom završnom radu.

Tablica 1: Usporedba različitih Arduino ploča

Pločica		Arduino Uno R3	Arduino Nano	Mega2560 Rev3
Mikroupravljač		ATmega328p	ATmega328p	ATmega2560
Vrsta USB-a		USB-B	Mini-B USB	USB-B
Ulazno/izlazni priključci	Digitalni	14	14	54
	Analogni ulaz	6	8	16
	Analogni izlaz	0	0	0
	PWM	6	6	15
Komunikacija	UART	Da	Da	Da
	I2c	Da	Da	Da
	SPI	Da	Da	Da
	CAN	Ne	Ne	Ne
	Bluetooth	Ne	Ne	Ne
	WIFI	Ne	Ne	Ne
Napajanje	U/I napon (V)	5	5	5
	Nominalni ulazni napon	7 - 12	8 - 12	9 - 12
	Struja po U/I priključku (mA)	20	20	20
	Vrsta priključka napajanja	5.5x2.5 priključak(Barrel Jack)	GPIO priključak	5.5x2.5 priključak(Barrel Jack)
	Baterijsko napajanje	Ne	Ne	Ne
Brzina procesora (MHz)		16	16	16
USB u serijski pretvarač		ATmega16U2 16MHz	ATmega16U2 16MHz	ATmega16U2 16MHz
Memorija	Flash (KB)	32	32	256
	SRAM (KB)	2	2	8
	EEPROM (KB)	1	1	4
Dimenzije	Masa (g)	25	5	37
	Širina (mm)	53.4	18	53.3
	Dužina (mm)	68.6	45	101.5

Izvor: Radolović, P. (2023). Arduino C programski jezik u mehatronici (Završni rad).

4. Hardverske komponente automatske kamere

U ovom završnom radu korištene su različite komponente kako bi se izradio funkcionalan sustav za nadzor divljih životinja. Odabir komponenti je ključan za postizanje optimalnih performansi i pouzdanosti sustava. Uključene su ploče kao što su Raspberry Pi 4B za obradu podataka i Arduino Nano za upravljanje perifernim uređajima. Za snimanje visokokvalitetnih fotografija koristi se Raspberry Pi HQ kamera, dok je za osvjetljenje u uvjetima slabog svjetla upotrijebljena infracrvena LED ploča. Napajanje sustava osigurava 12 V baterija kapaciteta 7000 mAh u kombinaciji s silazno ulaznim pretvaračem napona, koji prilagođava napon potrebama sustava. Qualcomm MDM9600 modem omogućuje pouzdanu internetsku vezu, a 12 VDC priključak s otvorenim krajevima koristi se za spajanje baterije na konverter. Svaka od ovih komponenti ima specifičnu ulogu u sustavu i zajedno omogućuju njegovu učinkovitu i dugotrajnu funkcionalnost.

4.1. Elektroničke komponente

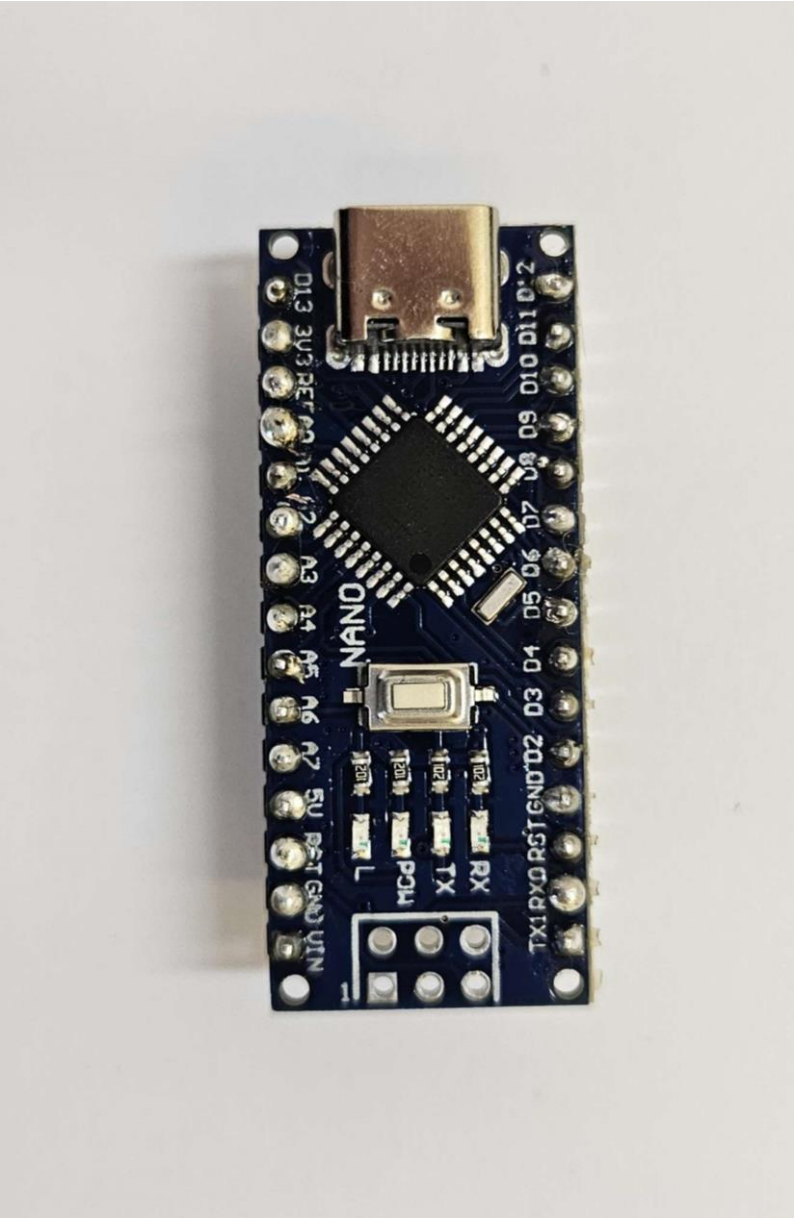
4.1.1. Raspberry Pi 4B

Raspberry Pi 4B je moćno mikroračunalo koje služi kao središnji procesor u ovom projektu, odnosno služi za obradu podataka, te slanje samih na udaljeni poslužitelj. Kao što je prije navedeno opremljen je četverojezgrenim ARM Cortex-A72 procesorom na 1.5 GHz, te sa 4 GB RAM memorije (Raspberry Pi Foundation, n.d.).

Slika 9: Raspberry Pi 4B mikroračunalo.

4.1.2. Arduino Nano

Arduino Nano je kompaktni mikroupravljač temeljen na Atmega328P čipu. U ovom projektu se koristi radi njegove iznimno male veličine što ga čini idealnim za korištenje u ovom sustavu radi ograničenog prostora unutar kućišta sustava, služi za paljenje i gašenje Raspberry Pi 4B-a i infracrvene LED ploče putem releja kad dobije signal s PIR senzora.



Slika 10: Arduino Nano mikroupravljač.

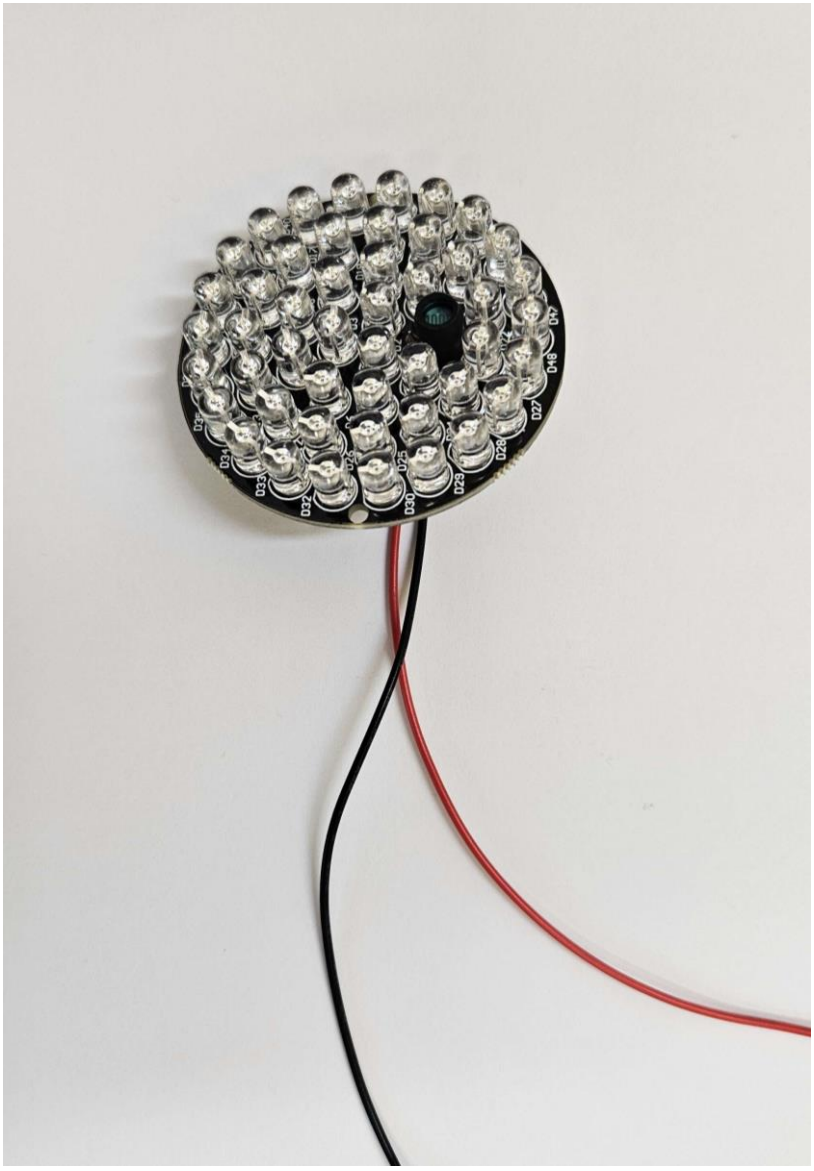
4.1.3. Raspberry Pi HQ kamera modul

Raspberry Pi HQ kamera je visokokvalitetna kamera dizajnirana za Raspberry Pi, opremljena Sony IMX477R senzorom rezolucije 12,3 megapiksela. Omogućuje snimanje fotografija i videozapisa visoke rezolucije, te se koristi u ovom projektu za prikupljanje vizualnih podataka o okolišu. HQ kamera podržava različite objektive, što pruža fleksibilnost u odabiru vidnog polja i fokusa (Raspberry Pi Foundation, n.d.).

Slika 11: Raspberry Pi HQ kamera s objektivom.

4.1.4. Infracrvena LED ploča

Infracrvena LED ploča se koristi za osvjetljavanje okoliša u slabom osvjeljenju ili totalnom mraku kao što je u toku noćnih sati. Ploča emitira infracrvenu svjetlost koja je nevidljiva ljudskom oku, ali omogućava kameri da snima jasno i u potpunom mraku (Zhou, Lin, Young i Ju, 2018). Ploča dobiva napon kad Arduino Nano prilikom detekcije pokreta uključi relej na koji je ploča spojena.



Slika 12: Infracrvena ploča.

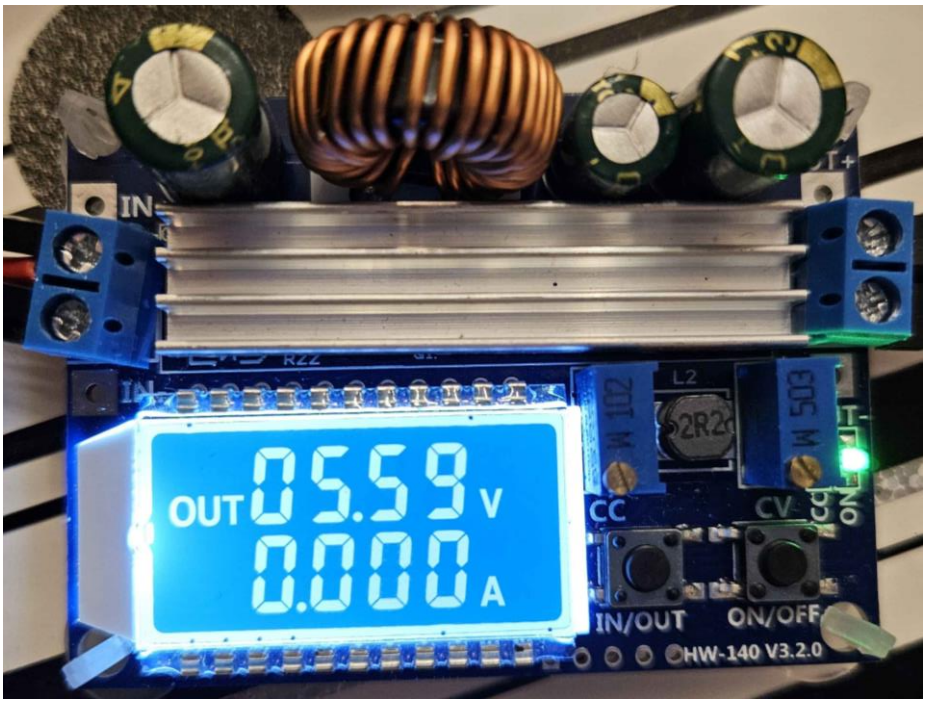
4.1.5. Napajanje

Paket baterija od 12 V i 7000 mAh pruža dugotrajno napajanje cijelom sustavu, omogućujući mu rad na terenu. Veliki kapacitet baterije u kombinaciji sa korištenjem učinkovitih komponenti omogućava dugotrajni rad ove baterije.

Slika 13: Baterija za napajanje sustava.

4.1.6. Silazno-uzlazni pretvarač

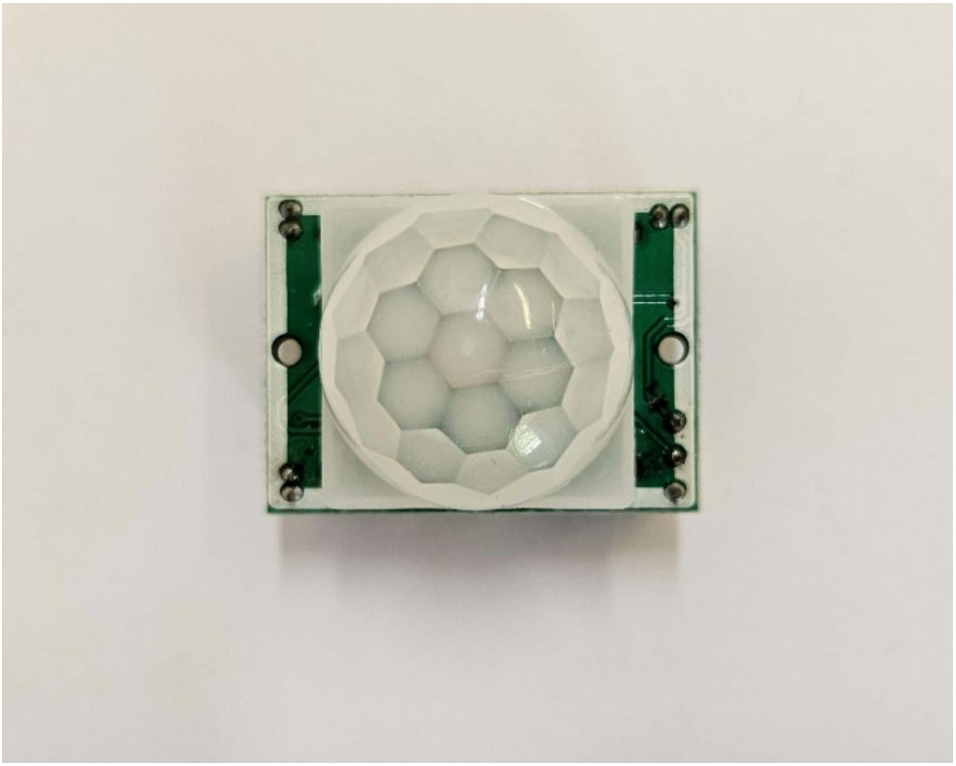
Silazno-uzlazni pretvarač je uređaj koji prilagođava napon iz izvora kako bi odgovarao potrebama Raspberry Pi-a i Arduino Nano-a. U ovom projektu smanjuje ulazni napon od 12 V na 5 V u svrhe napajanja prije spomenutih mikroračunala i mikroupravljača (Hwu i Peng, 2012).



Slika 14: Silazno-uzlazni pretvarač napona.

4.1.7. Pasivni infracrveni senzor

PIR senzor jest uređaj koji detektira infracrveno zračenje emitirano od strane objekata u njegovom vidnom polju, može detektirati pomake do 7 metara udaljenosti, te s kutom od 110 stupnjeva. Radi tako što mjeri promjene u infracrvenom zračenju koje uzrokuju pokreti u okolini, aktivirajući odgovarajuće izlazne signalne. U ovom projektu PIR senzor služi za detektiranje pokreta životinja ili ljudi u okolini sustava, te svojim izlaznim signalom budi Arduino Nano koji naknadno pokreće Raspberry Pi 4B i snima fotografiju (Juan, Kim, Sa, Kim i Cha, 2016).



Slika 15: PIR senzor.

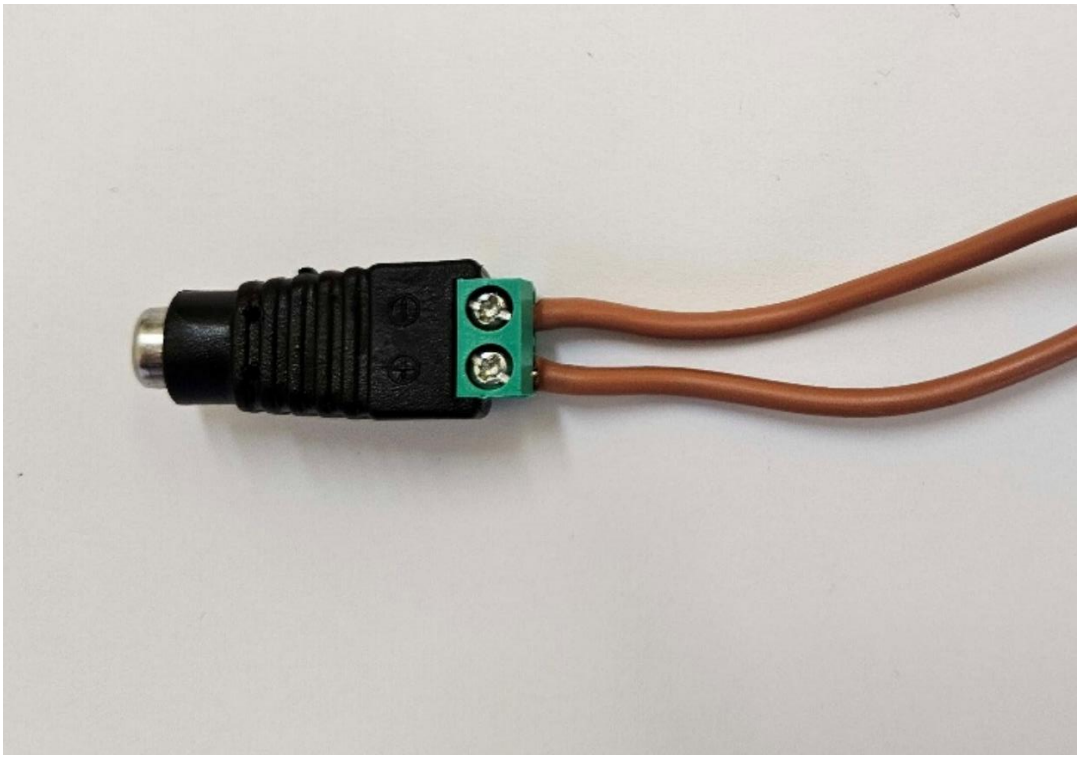
4.1.8. Qualcomm MDM9600 internet modem

Qualcomm MDM9600 internet modem omogućuje bežičnu komunikaciju putem LTE mreže. U ovom projektu modem je postavljen u USB adapteru za lakše povezivanje. Ovaj modem je ključan dio za prijenos podataka prikupljenih od strane Raspberry Pi-a na udaljeni poslužitelj, te pruža brzi prijenos podataka i stabilnu vezu što je ključno za rad ovog sustava (Jotrin Electronics, n.d.).

Slika 16: Qualcomm MDM9600 modem na USB adapteru.

4.1.9. DC priključak za napajanje

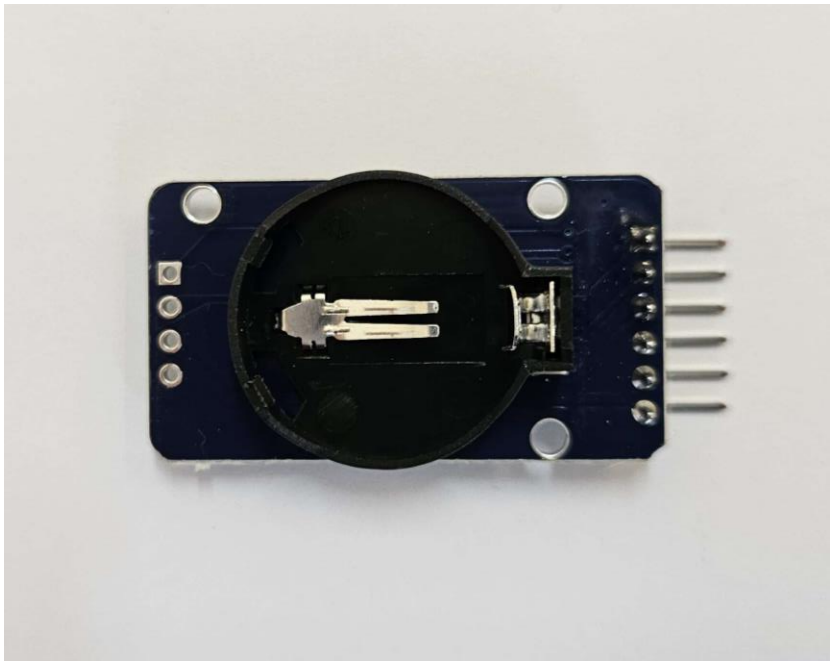
DC priključak za napajanje ovaj priključak koristi se da spojimo bateriju sa ostatkom sustava i sa silazno-uzlaznim pretvaračem napona, te također omogućuje lako isključivanje baterije od ostatka sustava budući da je cijeli sustav spojen na isti priključak.



Slika 17: DC priključak za napajanje.

4.1.10. Sat stvarnog vremena

Sat stvarnog vremena (*engl. Real-Time Clock*, RTC) je integrirani krug koji održava točno vrijeme i datum, čak i kad je sustav isključen. Radi tako da koristi kristalni oscilator za generiranje stabilnih vremenskih impulsa na frekvenciji od 32.768 kHz. Ovi impulsi se koriste za pokretanje unutarnjeg brojača koji prati sekunde, minute, sate, dan, datum, mjesec i godinu. RTC čip sadrži registarske setove gdje se ovi podaci pohranjuju i ažuriraju u stvarnom vremenu. Koristi vlastitu bateriju tipa CR2032 koja omogućuje njegovu autonomnu funkcionalnost održavanja datuma i bilježenja vremena bez prekida (Jin-jun, 2007). U svrhu ovog projekta RTC spojen na Raspberry PI 4B putem I2C sučelja se koristi da bi precizno održavao vrijeme na Raspberry Pi-u koji to isto vrijeme bilježi na fotografijama koje šalje na udaljeni poslužitelj tako da bi korisnik imao točno vrijeme u kojem je fotografija snimljena.



Slika 18: RTC (bez baterije).

4.1.11. Relej

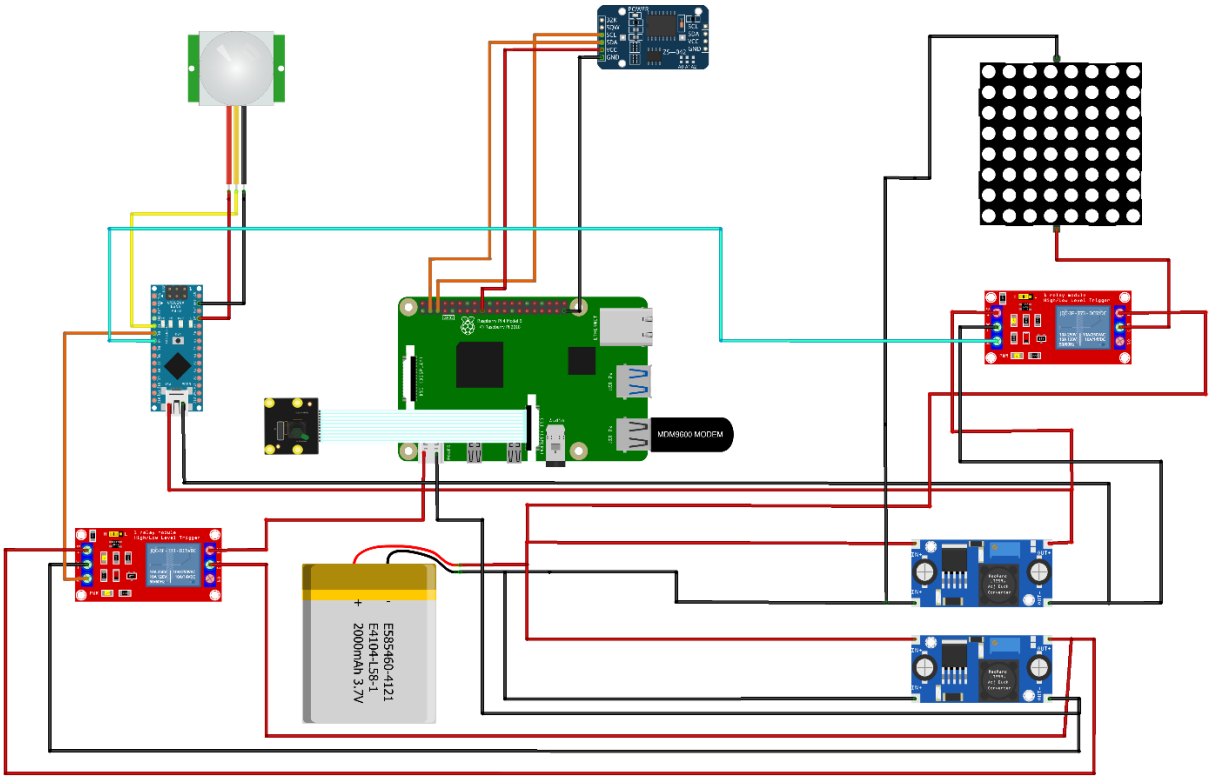
Relej predstavlja ključnu komponentu u ovom sustavu, omogućujući preciznu kontrolu nad strujnim krugovima, uključujući napajanje Raspberry Pi-a i IR ploče. Opremljen je s tri osnovna kontakta: C (Zajednički kontakt), NO (Normalno otvoreni kontakt), i NC (Normalno zatvoreni kontakt). Ovi kontakti omogućuju upravljanje protokom struje, gdje C služi kao zajednički priključak, NO omogućuje protok struje kada je relej aktiviran, dok NC održava protok kada je relej neaktivan. Relej je sposoban prebacivati struje do 10 A, pri čemu podržava do 250 VAC ili 30 VDC (Chipoteka, 2024).

Slika 19: Relej.

4.2. Način povezivanja elektroničkih komponenti

Cilj ove hardverske implementacije je povezati sve komponente u sustavu na način koji osigurava sigurno napajanje, pouzdanu kontrolu i funkcionalnost sustava (vidjeti sliku 20). Potrebno je implementirati sustav koji će omogućiti upravljanje napajanjem svih ključnih komponenti, kao što su: Arduino Nano, Raspberry Pi 4B, IR ploča. Sustav mora biti sposoban detektirati pokrete u okolini putem PIR senzora, te pokrenuti Raspberry Pi 4B putem releja kojime upravlja Arduino Nano pomoću izlaznog signala PIR senzora pri detekciji pokreta. Ključna komponenta u ovom sustavu je silazno-uzlazni pretvarač napona koji osigurava prilagođeno i sigurno napajanje za sve komponente.

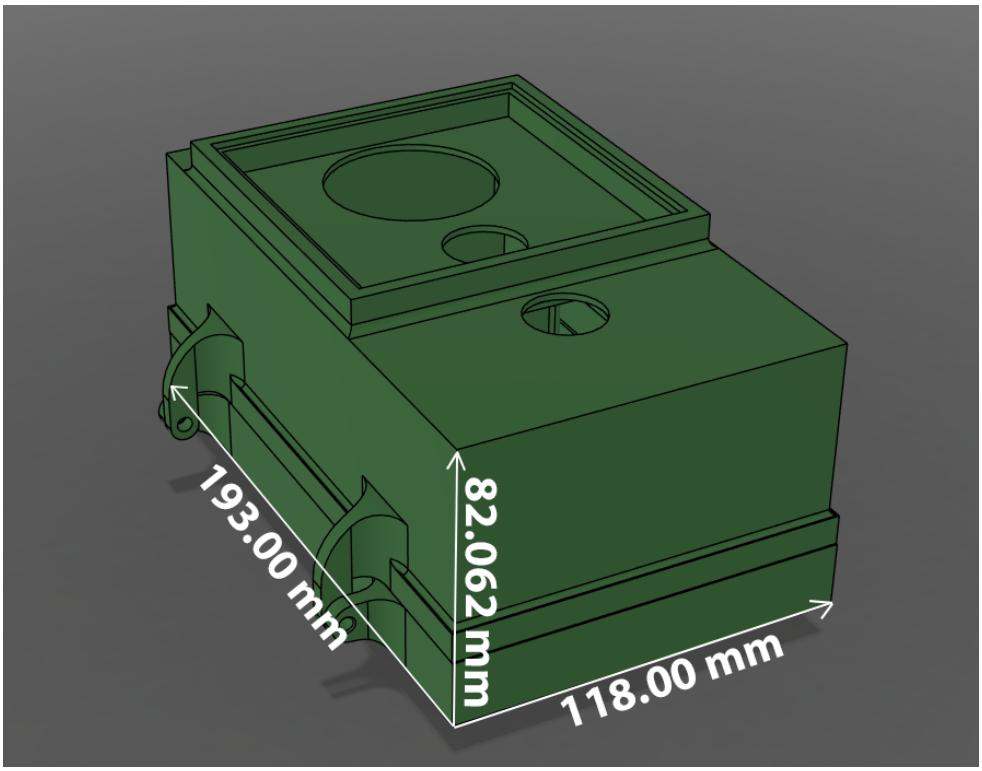
12 V baterija kapaciteta 7000 mAh služi kao primarni izvor napajanja, povezana putem DC priključka s otvorenim krajevima koji omogućuje jednostavno i sigurno povezivanje s ostalim komponentama. Iz DC priključka, napajanje se dijeli paralelno na dvije grane: jedna grana vodi prema IR ploči koja koristi 12V za svoje napajanje, dok druga grana vodi do silazno-uzlaznog pretvarača napona, koji smanjuje ulazni napon s 12 V na 5 V kako bi osigurao napajanje za Arduino Nano i Raspberry Pi 4B. Arduino Nano i Raspberry Pi 4B napajaju se paralelno, oba putem USB-C kabela, pri čemu Arduino Nano upravlja napajanjem Raspberry Pi-a pomoću releja, kontrolirajući protok struje kroz USB-C kabel. Kada PIR senzor detektira pokret, šalje signal na digitalni priključak 3 Arduino Nano-a, koji zatim aktivira relej i omogućuje napajanje Raspberry Pi-a. Kada se Raspberry Pi pokrene, Arduino Nano uključuje relej na kojem je spojena IR ploča, osiguravajući da se ploča uključi samo kada je to potrebno. Osim toga, RTC modul, spojen na SDA i SCL priključke Raspberry Pi-a, održava točno vrijeme u sustavu. Qualcomm MDM9600 internet modem, spojen na USB 2.0 utor Raspberry Pi-a, omogućuje uspostavljanje internetske veze, dok je Raspberry Pi HQ kamera spojena na CSI port Raspberry Pi-a, omogućujući visokokvalitetno snimanje fotografije i videozapisa. Ova integracija osigurava učinkovit rad sustava s optimiziranim upravljanjem napajanjem i funkcionalnostima.



Slika 20: Shema povezivanja elektroničkih komponenti automatske kamere.

4.3. Izrada kućišta

U svrhu ovog završnog rada izrađeno je kućište koje osigurava pouzdanu funkcionalnost u svim vremenskim uvjetima. Kućište kamere je oblika kvadra dimenzija 193x118x82.062 mm prikazano na slici 21. Kućište je izrađeno tehnologijom 3D printanja uz korištenje polilaktične kiseline (*engl. Polylactic acid*, PLA), plastike koja je otporna na različite ekološke uvjete (vidjeti sliku 22). Za dodatnu zaštitu komponenti sustava, poklopac kućišta je opremljen kanalima u koje je dodatno ugrađena brtva promjera 2 mm izrađena od nitrline gume (NBR) koja je učvršćena ljepilom unutar kanala. Za zaštitu kamera modula i infracrvene LED ploče izrađen je utor u koji se postavlja akrilno staklo dimenzija 100x100 mm. Ovi faktori omogućuju sustavu da izdrži varijacije temperature, vlage i drugih vremenskih uvjeta.



Slika 21: Kućište kamere.

Slika 22: Kamera s automatskim okidanjem postavljena u divljinu.

5. Softverska implementacija automatske kamere

Osim hardverske implementacije, ključni korak je osigurati komunikaciju i kontrolu između komponenti putem programskog koda. Kod se dijeli između Arduino Nano-a, koji je programiran u Arduino C programskom jeziku, i Raspberry Pi 4B, gdje je situacija nešto složenija. Na Raspberry Pi-u se prvo koristi Bash skripta koja se automatski pokreće pri paljenju uređaja. Ova skripta snima fotografiju, otiskuje na nju trenutno vrijeme i datum, te komprimira fotografiju kako bi se smanjila potrošnja internetskog prometa. Nakon toga slijedi Python kod koji šalje podatke, odnosno fotografije, na udaljeni poslužitelj putem SSH protokola (SSH Communications Security, 2024). U nastavku će se kod analizirati u dijelovima.

5.1. Arduino Nano kod

Na samom početku potrebno je uključiti biblioteku „LowPower.h“ za podršku funkcije dubokog sna koja je ključna za smanjenje potrošnje energije cijelog sustava (Segawa, Shirota, Fujisaki, Kimura i Kanai, 2014):

```
#include <LowPower.h>
```

U nastavku slijedi definiranje fizičkih priključaka Arduina u kodu s odgovarajućim varijablama, pri čemu se koristi „const int“ kako bi se osiguralo da se vrijednosti tih varijabli, koje predstavljaju brojeve priključaka, ne mogu promijeniti tijekom izvođenja programa:

```
const int pirPin = 2;
const int relejPin = 3;
```

Sljedeća linija definira varijablu koja označava detekciju pokreta putem PIR senzora, te postavljenu na „false“. Korištenje ključne riječi „volatile“ osigurava da se vrijednost ove varijable uvijek pravilno očitava iz memorije, čak i kada se mijenja unutar prekidne rutine, što sprječava moguće greške u radu programa:

```
volatile bool pirDetekcija = false;
```

Funkcija „setup()“ inicijalizira Arduino postavke koje su potrebne za rad s PIR senzorom i relejem. Prvo, postavlja pinove: „pirPin“ kao ulazni jer će očitavati signale s PIR senzora, a „relejPin“ kao izlazni jer će kontrolirati relej. Zatim, postavlja početno stanje „relejPin“ na „LOW“, čime se relej isključuje. Na kraju, funkcija „attachInterrupt“ omogućuje prekidnu rutinu koja će se pokrenuti svaki put kad PIR senzor detektira pokret (rastući (RISING) signal na „pirPin“). Ta rutina će aktivirati funkciju „pirISR“, koja postavlja zastavicu da je pokret detektiran. Dio koda koji opisuje prethodno nalazi se ispod:

```
void setup() {
  pinMode(pirPin, INPUT);
  pinMode(relejPin, OUTPUT);
  digitalWrite(relejPin, LOW);
  attachInterrupt(digitalPinToInterrupt(pirPin), pirISR,
RISING);
}
```

Funkcija „loop()“ započinje sa provjerom uvjeta unutar „if“ naredbe koja ispituje da li je pir senzor detektirao pokret, odnosno da li je varijabla „pirDetekcija“ postavljena na stanje „True“. Ako je stanje postavljeno na „true“, relej se uključuje pomoću „digitalWrite“ funkcije koja postavlja „relejPin“ na „HIGH“. Zatim funkcija „delay“ održava relej uključenim jednu minutu kako bi osigurali da Raspberry Pi ima dovoljno vremena za slanje podataka. Nakon isteka vremena isključuje se relej pomoću „digitalWrite“ funkcije, a varijabla „pirDetekcija“ postavlja se na „false“, što signalizira da je detekcija pokreta obrađena i sustav je spreman za novu detekciju. Dio koda koji opisuje prethodno dan je sljedećim linijama:

```
void loop() {
  if (pirDetekcija) {
    digitalWrite(relejPin, HIGH);
     delay(1 * 60 * 1000);
     digitalWrite(relejPin, LOW);
     pirDetekcija = false;
  }
```

Sljedeća funkcija koristi „for“ petlju kako bi se mikroupravljač postavio u stanje dubokog sna radi uštede energije. Petlja se ponavlja 38 puta a u svakoj iteraciji mikroupravljač

se postavlja u stanje sna na 8 sekundi pomoću funkcije „LowPower.powerDown“, ova funkcija isključuje analogno u digitalni pretvarač (*engl. Analog-to-Digital Converter*, ADC), i detekciju niskog napona (*engl. Brown-out Detection*, BOD) kako bi se dodatno smanjila potrošnja energije tokom sna. Ukupno trajanje sna iznosi približno 5 minuta (38 ponavljanja po 8 sekundi), što sprječava stalno paljenje Raspberry Pi-a radi lažnih detekcija:

```
for (int i = 0; i < 38; i++) {
  LowPower.powerDown(SLEEP_8S, ADC_OFF, BOD_OFF);
 }
```

Funkcija „pirISR()“ postavlja varijablu „pirDetekcija“ u stanje „true“, ova funkcija se aktivira pozivom iz „setup()“ funkcije kad PIR senzor da signal Arduino Nano-u da se dogodila detekcija:

```
void pirISR() {
  pirDetekcija = true;
}
```

5.2. Raspberry Pi kod

5.2.1. Bash skripta

Na samom početku datoteke označava se da se skripta treba izvršavati korištenjem Bash interpretera. Kad se skripta pokrene, operacijski sustav koristi tu liniju, odnosno „shebang“ kako bi znao koji program treba koristiti za izvođenje skripte, ova linija osigurava da će se skripta ispravno pokrenuti neovisno o okruženju (Prakash, 2021). Nakon toga slijedi funkcija „sleep“ koja uvodi pauzu od 5 sekundi prije nego li se ostatak skripte provede. Ova pauza je veoma korisna jer omogućuje dovoljno vremena Raspberry Pi-u koji se tek pokrenuo da osposobi sve sustave i servise, te tako osigura ispravan rad skripte:

```
#!/bin/bash
sleep 5
```

Sljedeća linija pokreće program „libcamera“, te njegovu funckiju „still“ koja služi za snimanje fotografije pomoću Raspberry Pi HQ kamere koja je spojena na CSI port Raspberry Pi-a. Opcija „-n“ isključuje prikaz pregleda fotografije, to se koristi tako da se fotografija snimi odmah, bez prikaza na zaslonu. Opcija „-o“ označava ime datoteke i format u kojem će se fotografija snimiti. Opcije „—width“ i „—height“ služe za specificiranje rezolucije u kojoj želimo snimiti fotografiju, bez njih fotografija bi se snimila u maksimalnoj mogućoj rezoluciji što bi bilo značajno više nego potrebno, te bi trošilo puno više internetskog prometa (Raspberry Pi Foundation, n.d.):

```
libcamera-still -n -o image.jpg --width 1024 --height 768
```

Nakon što je fotografija snimljena da bi uštedjeli na internetskom prometu potrebno je komprimirati fotografiju, to se izvodi pokretanjem dodatne Python skripte putem „python3“ komande, ova skripta će biti opisana u Python dijelu koda. Nakon što je fotografija komprimirana postavlja se varijabla „Image“ sa novim nazivom fotografije kako bi se dalje u skripti moglo upravljati fotografijom. Navedeno je dano sljedećim kodom:

```
python3 /home/admin/compress.py
image="compressed_image.jpeg"
```

Prije nego li se može upravljati svojstvima fotografije potrebno je postaviti visinu i širinu fotografije, odnosno rezoluciju. To se izvodi sljedećim instrukcijama gdje varijable „width“ i „height“ uz pomoću naredbe „identify“ izvlače širinu i visinu fotografije u pikselima tako da se može koristiti u daljnjim naredbama u skripti:

```
width=$(identify -format "%w" "$image")
height=$(identify -format "%h" "$image")
```

Za prikazivanje informacija o trenutnom vremenu i datumu, te imenu Autora, a da se očuva izvorna fotografija potrebno je dodati prostor ispod fotografije gdje će se te informacije ispisati. Za dodavanje tog prostora koristi se prethodno dobivena informacija o visini fotografije u pikselima te se množi sa 0.07, odnosno 7% ukupne visine fotografije. Rezultat ovog postupka pohranjuje se u varijablu „overlay_height“. Nakon toga potrebno je maknuti decimalni dio iz izračunate vrijednosti da bi fotografiju

bilo moguće spremiti. Dio koda koji opisuje prethodno nalazi se ispod:

```
overlay_height=$(echo "$height * 0.07" | bc)
overlay_height=${overlay_height%.*}
```

Da bi se na fotografiju dodalo trenutno vrijeme i datum kako bi korisnik mogao znati točno vrijeme kad je fotografija snimljena prvo je potrebno dobiti trenutno vrijeme. To se izvršava pomoću varijabli „current_date_overlay“ i „current_time_overlay“, prva varijabla dobiva trenutni datum u formatu DD/MM/YYYY, dok druga varijabla dobiva trenutno vrijeme u 24-satnom formatu. Nakon toga se te varijable kombiniraju u varijabli „text“ koja će biti prikazana na završenoj fotografiji. Prethodno je ostvareno sljedećim kodom:

```
current_date_overlay=$(date +"%d/%m/%Y")
current_time_overlay=$(date +"%H:%M:%S")
text="Datum: $current_date_overlay Vrijeme:
$current_time_overlay"
```

Također je potrebno sličnu radnju napraviti i za varijable koje će se koristiti za spremanje naziva datoteke. Radi potreba organiziranja datoteka na poslužitelju format u kojem će se spremati datum je YYYY/MM/DD. Na kraju se sprema finalno ime fotografije u varijablu „final_filename“. Navedene radnje dane su sljedećim linijama koda:

```
current_date_filename=$(date +"%Y%m%d")
current_time_filename=$(date +"%H%M%S")

final_filename="PICT_${current_date_filename}_${current_time_fi
lename}.jpg"
```

Za generiranje privremenog prekrivanja koje će sadržavati datum, vrijeme i ime autora, a koje će kasnije biti integrirano u konačnu fotografiju, potrebno je definirati odgovarajuću fotografiju za prekrivanje. U tu svrhu, naziv prekrivanja se pohranjuje u varijablu „overlay_image“. Zatim se pomoću naredbe „convert“, u kombinaciji s varijablama „width“ i „overlay_height“, određuju dimenzije prekrivanja. Visina prekrivanja određena je kao 7% visine originalne fotografije, dok je širina jednaka širini originalne fotografije. Ovaj prostor prekrivanja ispunjava se bijelom bojom, a pomoću

definiranog fonta Helvetica veličine 24 piksela i crne boje, u donjem lijevom kutu prekrivanja ispisuju se podaci o datumu i vremenu koji su pohranjeni u varijabli „text“. U donjem desnom kutu prekrivanja ispisuje se ime autora. Tako generirano prekrivanje koristi se kao privremena fotografija koja će biti spojena s glavnom fotografijom u završnom koraku obrade fotografije:

```
overlay_image="overlay.jpg"
convert -size "${width}x${overlay_height}" xc:white \
    -font Helvetica -pointsize 24 -fill black \
    -gravity SouthWest -annotate +10+5 "$text" \
    -gravity SouthEast -annotate +10+5 "Luka Zuzic" \
    $overlay_image
```

Spajanje privremene fotografije s glavnom fotografijom se izvršava pomoću naredbe „convert“, koja spaja te dvije fotografije i daje im novo ime koje je prije definirano u varijabli „final_filename“:

```
convert "$image" "$overlay_image" -append "$final_filename"
```

Nakon spremanje konačne fotografije potrebno je izbrisati privremene fotografije koje su ostale iz procesa, odnosno izvornu fotografiju i privremeno prekrivanje. Ovo se obavlja korištenjem funkcije „rm“, te imenima fotografija koje su spremljene u varijablama „overlay_image“ i „image“:

```
rm $overlay_image
rm "$image"
sleep 5
```

Na samom kraju se pokreće Python skripta pomoću komande „python3“. Ova skripta prenosi konačnu fotografiju na poslužitelj:

```
python3 /home/admin/jpgpload.py
```

5.2.2. Python kod

5.2.2.1. Slanje fotografije

Na samom početku potrebno je uvesti biblioteke „os“ i „paramiko“. „os“ se koristi za radnje sustava, odnosno u ovom slučaju za pronalaženje fotografija na Raspberry Pi-u, ispisivanje njihovih odredišta, te na kraju za uklanjanje fotografija. „Paramiko“ je biblioteka koja omogućuje povezivanje na poslužitelj putem SSH protokola, te jednostavno slanje fotografija preko istog:

```
import os
import paramiko
```

Da bi se moglo povezati sa poslužiteljem potrebno je definirati parametre za prijavu, javnu adresu poslužitelja, te odredište za spremanje fotografija:

```
hostname = 'byceba.com'
port = 22
username = 'korisničko-ime'
password = 'zaporka'
remote_path = 'odredište'
```

Dalje se određuje radni direktorij skripte kako bi skripta znala u kojem direktoriju tražiti fotografije:

```
script_directory = os.path.dirname(os.path.abspath(__file__))
os.chdir(script_directory)

local_path = os.path.dirname(os.path.abspath(__file__))
```

Za pristup poslužitelju potrebno je postaviti SSH klijent i prihvatiti/dodati nove ključeve pri povezivanju na poslužitelj, ovo se izvršava putem funkcija iz paramiko biblioteke:

```
ssh_client = paramiko.SSHClient()
ssh_client.set_missing_host_key_policy(paramiko.AutoAddPolicy()
)
```

Na kraju slijedi „try“ funkcija koja se na početku povezuje na poslužitelj korištenjem podataka koji su zadani na početku koda, ukoliko su uneseni podaci krivi ispisuje se greška u terminal. Nakon povezivanja pretražuje se za lokalni put do fotografija, te ako postoji jedna ili više fotografija ispisuje se njihovo odredište i započinje se sa prijenosom na poslužitelj. Nakon uspješnog ili neuspješnog prijenosa fotografije

zatvara se veza sa poslužiteljem i kod se završava. Navedeno je ostvareno sljedećim kodom:

```
try:

    ssh_client.connect(hostname, port, username, password)
    sftp = ssh_client.open_sftp()


    for filename in os.listdir(local_path):

        if filename.lower().endswith(".jpg"):
            local_filepath = os.path.join(local_path, filename)
            remote_filepath = os.path.join(remote_path,
filename)

            if os.path.exists(local_filepath):
                print(f"Pronađeno {local_filepath}, pripremanje
za prijenos...")
                try:
                    print(f"Upload {filename}...")
                    sftp.put(local_filepath, remote_filepath)

                    print(f"Uklanjanje {local_filepath}...")
                    os.remove(local_filepath)
                except Exception as upload_error:
                    print(f"Prijenos nije uspio {filename}:
{upload_error}")
            else:
                print(f"Datoteka nije pronađena:
{local_filepath}")

    sftp.close()
    ssh_client.close()
    print("Uspješan prijenos i uklanjanje.")
except Exception as e:
    print(f"Dogodila se greška: {e}")
```

5.2.2.2. Komprimiranje fotografija

Na samom početku moraju se uvesti potrebne biblioteke za komprimiranje fotografija i za sistemske radnje, to su „PIL“ i „os“:

```
from PIL import Image
import os
```

Za komprimiranje fotografija potrebno je definirati funkciju koja će obavljati tu radnju sa parametrima koji su kasnije zadani u glavnoj petlji koda. Funkcija „compress_image“ dobiva varijable koje joj govore izvorni put do fotografije, izlazno odredište, razinu kompresije, te veličinu fotografije. Nakon završenog komprimiranja vraća destinaciju na kojoj je spremljena fotografija:

```
def compress_image(file_path, output_path, quality, max_size):
    with Image.open(file_path) as img:
        if max_size:
            img.thumbnail(max_size)
        img.save(output_path, format='JPEG', quality=quality)
    return output_path
```

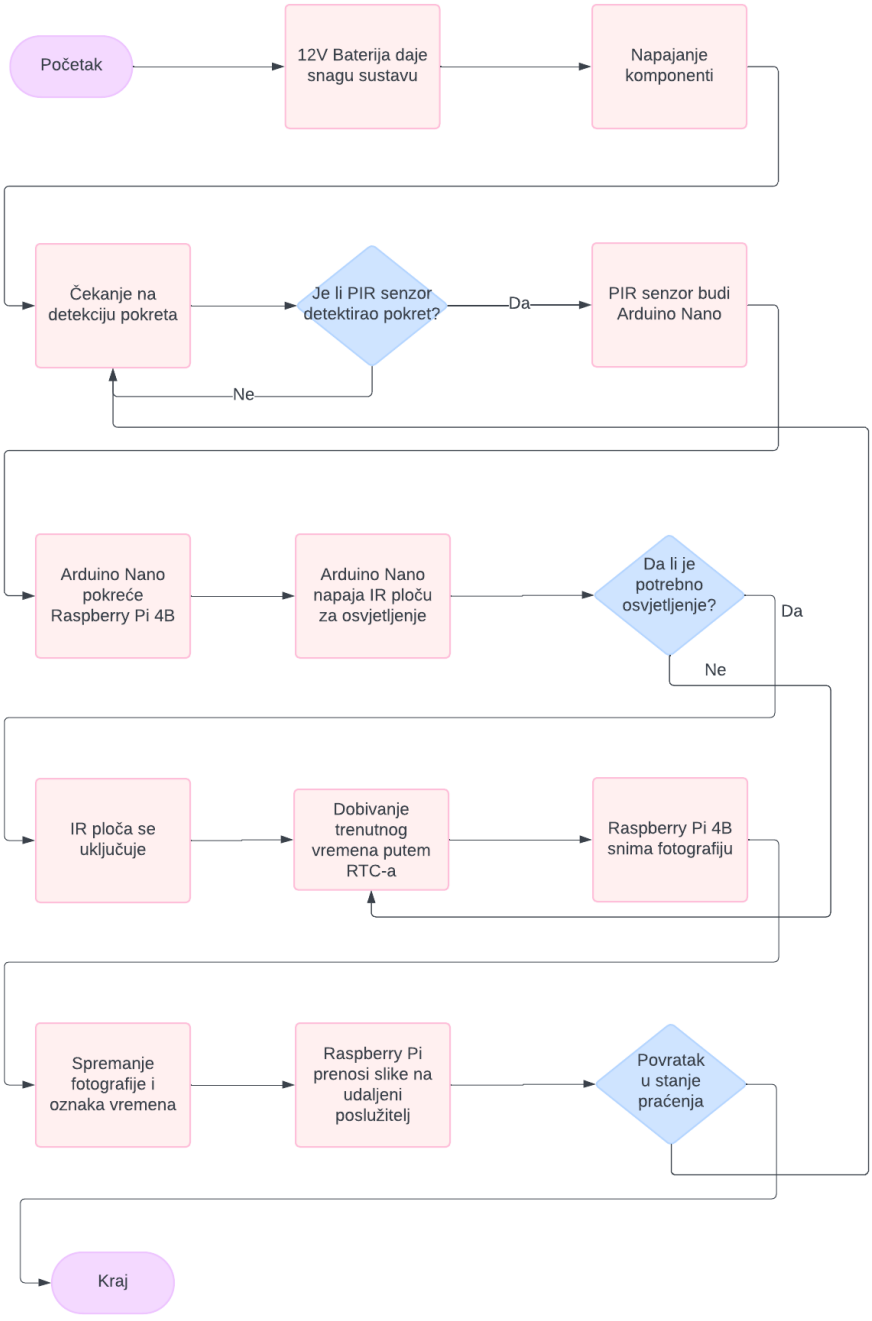
Glavna funkcija ovog koda je „if“ funkcija koja osigurava da se blok koda unutar nje pokreće samo ako je skripta izravno pokrenuta, a ne kad se uvozi kao modul u neki program. Prvo se postavljaju fiksne varijable sa odredištem ulazne fotografije i odredištem na koju će komprimirana fotografija biti spremljena. Nakon toga postavlja se razina kompresije u varijabli „quality“ od 1 do 100 (manja vrijednost je veća kompresija), te u „max_size“ se postavlja rezolucija izvorne fotografije u pikselima. Nakon postavljenih varijabli pokreće se funkcija za komprimiranje fotografije, te se uklanja izvorna fotografija. Dio koda koji se odnosi na prethodno nalazi se ispod:

```
if __name__ == "__main__":
    input_image_path = "/home/admin/image.jpg"
    output_image_path = "/home/admin/compressed_image.jpeg"
    quality = 55
    max_size = (1024, 768)

    compressed_image_path = compress_image(input_image_path,
output_image_path, quality, max_size)
    os.remove(input_image_path)
    print(f"Fotografija je spremljena na:
{compressed_image_path}")
```

5.3. Dijagram toka rada automatske kamere prilikom detekcije divlje životinje

Sustav počinje s radom i napaja Arduino Nano putem baterije koji čeka signal PIR

senzora o detekciji pokreta toplokrvne životinje da bi izašao iz stanja dubokog sna. Kad PIR senzor detektira pokret, Arduino Nano pokreće Raspberry Pi mikroračunalo koje započinje s fotografiranjem ili snimanjem videozapisa okoline, ovisno o odabiru korisnika. Nakon završetka snimanja ili fotografiranja datoteka se komprimira neovisno o tome da li je riječ o slici ili video zapisu, te se učitava na udaljeni poslužitelj. U slučaju noćnog snimanja postoji međukorak uključivanja ploče sa infracrvenim LED diodama. Prethodni postupak je prikazan na slici 23.



Slika 23: Dijagram toka rada sustava.

6. Poslužitelj

6.1. VPS poslužitelj

Za potrebe ovog završnog rada korišten je virtualni privatni poslužitelj (*engl. Virtual Private Server*, VPS) poslužitelj s javnom IP adresom, iznajmljen od kompanije Contabo (Contabo, 2024). Poslužitelj raspolaže s 4 jezgri virtualne središnje jedinice za obradbu (*engl. Virtual Central Processing Unit*, vCPU), što omogućuje paralelnu obradu više zadataka i osigurava dovoljno računalne snage za učinkovito upravljanje zahtjevima aplikacije. Pored toga, poslužitelju je na raspolaganju 6 GB RAM memorije, koja je dovoljna za pokretanje svih potrebnih servisa i aplikacija, uključujući bazu podataka, web poslužitelj, i druge potrebne komponente, bez ugrožavanja performansi. Poslužitelj također ima ograničenje na mjesečni internetski promet od 32 TB, što omogućuje slobodan prijenos velike količine podataka, uključujući učitavanje i preuzimanje fotografija i drugih datoteka.

6.2. Web stranica

Web stranica postavljena na VPS-u koristi Apache2 kao osnovu sustava. Apache2 je zadužen za posluživanje statičkih i dinamičkih sadržaja, te za upravljanje HTTPS zahtjevima. Backend je razvijen korištenjem pomoću Flask-a, lagano radno okruženje za razvoj web aplikacija napisan u Pythonu. Za potrebe ovog završnog rada Flask se koristi za izgradnju dinamičkih dijelova stranice, upravljanje korisničkim sesijama sa kolačićima, te integraciju sa bazom podataka za jednostavno sortiranje fotografija u bazu podataka, prijavljivanje i odjavljivanje s web stranice, upravljanje svojim kamerama i pristup fotografijama korisnicima. Za pohranu korisničkih podataka koristi se SQLite baza podataka, koja je vrlo jednostavna, ali učinkovita za aplikacije koje ne zahtijevaju složene funkcionalnosti koje pruža klasična strukturirana baza podataka (*engl. Structured Query Language*, SQL). U bazi podataka pohranjuju se ključni podaci, kao što su korisnički podaci za prijavu, podaci o broju kamera korisnika, podaci o odredištima fotografija. Ovakav pristup je ključan za rad web stranice, odnosno omogućuje korištenje web stranice za više korisnika, a ne samo jednog.

6.2.1. Flask konfiguracija

Flask pruža niz funkcionalnosti, uključujući autentifikaciju korisnika, upravljanje kamerama, prikaz galerije fotografija i interaktivnu kartu. U nastavku će biti detaljno opisane glavne rute aplikacije i način na koji one funkcioniraju unutar sustava.

Na samom početku konfiguracije, aplikacija je postavljena korištenjem Flask klase. Prilikom inicijalizacije, definirane su putanje do direktorija s predlošcima (template_folder) i statičkim datotekama (static_folder). Također potrebno je definirati tajni ključ (secret_key) koji se koristi za osiguranje sesija unutar aplikacije. Ovaj ključ osigurava integritet i sigurnost sesijskih podataka. Dio koda koji se odnosi na prethodno prikazan je ispod:

```
app = Flask(__name__, template_folder=template_dir,
static_folder=template_dir)
app.secret_key = 'tajni-ključ'
```

Nakon tajnog ključa potrebno je definirati gdje se nalazi baza podataka od koje se vuču podaci o korisnicima i potrebno je definirati lokaciju pretinca gdje se nalaze korisničke fotografije, što je dano sljedećim linijama koda:

```
app.config['DATABASE'] = database_path
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
```

Funkcija „get_db()“ koristi se za povezivanje s bazom podataka, a „close_connection()“ osigurava zatvaranje veze s bazom nakon što su svi zahtjevi obrađeni. Kako bi se osigurao kontinuirani rad aplikacije, implementirana je funkcija „retry_db_execute()“ koja ponavlja pokušaj izvršenja SQL upita. Opisanome odgovaraju sljedeće linije koda:

```
def get_db():
    db = getattr(g, '_database', None)
    if db is None:
        db = sqlite3.connect(app.config['DATABASE'])
        db.row_factory = sqlite3.Row
        db.execute("PRAGMA journal_mode=WAL")
    return db

@app.teardown_appcontext
def close_connection(exception):
```

```
    db = getattr(g, '_database', None)
    if db is not None:
        db.close()

def retry_db_execute(db, query, params, retries=10, delay=0.5):
    for i in range(retries):
        try:
            db.execute(query, params)
            return
        except sqlite3.OperationalError as e:
            if 'locked' in str(e):
                time.sleep(delay)
                logging.warning(f"Retry {i+1}/{retries} for
query: {query} due to database lock")
            else:
                raise
    raise sqlite3.OperationalError("Baza podataka je
zakljucana")
```

Funkcija „login()“ implementira osnovnu funkcionalnost prijave korisnika. Korisnici unose svoje korisničko ime i lozinku putem forme na početnoj stranici. Nakon slanja forme metodom POST, aplikacija dohvaća korisničke podatke iz baze podataka i uspoređuje unesenu lozinku s haširanom (*engl. hashing*) lozinkom pohranjenom u bazi koristeći „bcrypt“ algoritam. Ako su podaci točni, korisnik se uspješno prijavljuje u sustav, a njegova sesija se pohranjuje u session objektu. Ako prijava nije uspješna, korisnik se vraća na početnu stranicu s porukom o grešci. Opisano je prikazano sljedećim kodom:

```
@app.route('/', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username'].strip().lower()
        password = request.form['password']
        db = get_db()
        cursor = db.execute('SELECT * FROM users WHERE
LOWER(username) = ?', (username,))
        user = cursor.fetchone()
        if user and bcrypt.checkpw(password.encode('utf-8'),
user['password_hash'].encode('utf-8')):
            session['user_id'] = user['id']
            return redirect(url_for('select'))
        else:
            flash('Invalid credentials', 'error')
            return redirect(url_for('login'))
```

```
        return render_template('index.html')
```

Funkcija „logout()“ na rutu „/logout“ omogućuje korisnicima da se odjave iz sustava. Ona uklanja „user_id“ iz sesije i preusmjerava korisnika na stranicu za prijavu:

```
@app.route('/logout')
def logout():
    session.pop('user_id', None)
    return redirect(url_for('login'))
```

Funkcija „select()“ na ruti „/select“ prikazuje korisniku sve kamere koje su povezane s njegovim računom. Ova funkcija koristi bazu podataka kako bi dohvatila sve kamere koje su registrirane na korisnika te ih prikazuje putem web sučelja:

```
@app.route('/select')
@login_required
def select():
    user_id = session['user_id']
    db = get_db()
    cursor = db.execute('SELECT * FROM cameras WHERE user_id =
?', (user_id,))
    cameras = cursor.fetchall()
    try:
        return render_template('select.html', cameras=cameras)
    except UnicodeDecodeError as e:
        return str(e)
```

Funkcija „add_camera()“ na ruti „/add_camera“ omogućuje korisnicima da dodaju nove kamere u sustav. Korisnik unosi identifikator kamere i ime kamere putem forme (slika 26). Nakon validacije unesenih podataka (provjerava se je li identifikator kamere brojčan i pozitivan), aplikacija pohranjuje nove podatke u bazu:

```
@app.route('/add_camera', methods=['GET', 'POST'])
@login_required
def add_camera():
    if request.method == 'POST':
        camera_id = request.form['camera_id']
        camera_name = request.form['camera_name']
        user_id = session['user_id']

        if not camera_id.isdigit() or int(camera_id) <= 0:
```

```
            flash('Broj kamere mora biti pozitivan broj.',
'error')
            return render_template('add_camera.html')

        db = get_db()
        try:
            db.execute("INSERT INTO cameras (user_id,
camera_id, camera_name) VALUES (?, ?, ?)", (user_id, camera_id,
camera_name))
            db.commit()
            flash('Camera added successfully.')
            return redirect(url_for('select'))
        except sqlite3.IntegrityError:
            flash('Camera ID already exists.', 'error')
            return render_template('add_camera.html')
    return render_template('add_camera.html')
```

Funkcija „delete_cameras()“ na ruti „/delete_cameras“ omogućuje korisnicima brisanje jedne ili više kamera pomoću gumba postavljenog na stranici. Korisnik odabire kamere koje želi ukloniti, a zatim aplikacija briše podatke o odabranim kamerama iz baze. Odgovarajući kod se nalazi ispod:

```
@app.route('/delete_cameras', methods=['POST'])
@login_required
def delete_cameras():
    camera_ids = request.form.getlist('camera_ids')
    user_id = session['user_id']
    db = get_db()

    if not camera_ids:
        flash('Niste odabrali kameru.', 'error')
        return redirect(url_for('select'))

    try:
        for camera_id in camera_ids:
            db.execute("DELETE FROM cameras WHERE camera_id = ?
AND user_id = ?", (camera_id, user_id))
        db.commit()
        flash('Odabrane kamere su obrisane.', 'success')
    except sqlite3.Error as e:
        flash(f'An error occurred while deleting cameras: {e}',
'error')

    return redirect(url_for('select'))
```

Funkcija „gallery()“ na ruti „/gallery“ prikazuje korisnicima sve fotografije snimljene na svim kamerama koje su dodane na korisnički profil. Aplikacija dohvaća putanje fotografija iz baze podataka i prikazuje ih na web stranici kojoj se pristupa pritiskom na gumb „Galerija“ (slika 25). Prethodno opisano dano je sljedećim linijama koda:

```
@app.route('/gallery')
@login_required
def gallery():
    user_id = session['user_id']
    db = get_db()
    cursor = db.execute('SELECT file_paths FROM cameras WHERE
user_id = ?', (user_id,))
    file_paths = []
    for row in cursor.fetchall():
        if row['file_paths']:
            file_paths.extend(row['file_paths'].split(','))
    file_paths = [url_for('static', filename=fp.strip()) for fp
in file_paths]
    try:
        return render_template('gallery.html',
file_paths=file_paths)
    except UnicodeDecodeError as e:
        return str(e)
```

Funkcija „gallery_images()“ na ruti „/gallery_images“ vraća popis svih fotografija u JSON formatu, omogućujući aplikaciji da dinamički prikaže fotografije na stranici galerije. Prije dohvaćanja fotografija, funkcija poziva „update_image_database()“ koja pretražuje pretinac s fotografijama za datoteke koje završavaju sa „.png“, „.jpg, „.jpeg“, „.gif“, te kojima ime počinje s „PICT“ kako bi se osiguralo da su svi podaci u bazi ažurirani, odnosno da se prikazuju sve fotografije. Spomenuto je ostvareno sljedećim kodom:

```
def update_image_database():
    db = get_db()
    user_id = session.get('user_id')

    for root, dirs, files in os.walk(image_folder):
        for file in files:
            if file.lower().endswith(('.png', '.jpg', '.jpeg',
'.gif')):
                camera_id_match =
re.search(r'PICT_(\d{8})_(\d+)', file)
```

```
                if camera_id_match:
                    camera_id = camera_id_match.group(2)
                    file_path = os.path.join(root, file)
                    relative_path = os.path.relpath(file_path,
template_dir)

                    cursor = db.execute("SELECT user_id,
file_paths, camera_name FROM cameras WHERE camera_id = ?",
(camera_id,))
                    result = cursor.fetchone()

                    if result:
                        existing_paths = result['file_paths']
or ''
                        if relative_path not in
existing_paths.split(','):
                            new_paths = (existing_paths + ',' +
relative_path).strip(',')
                            retry_db_execute(db, "UPDATE
cameras SET file_paths = ? WHERE camera_id = ?", (new_paths,
camera_id))
                    else:
                        camera_name = f"Camera {camera_id}"
                        if user_id:
                            retry_db_execute(db, "INSERT INTO
cameras (user_id, camera_id, camera_name, file_paths) VALUES
(?, ?, ?, ?)", (user_id, camera_id, camera_name,
relative_path))
    db.commit()

    cursor = db.execute("SELECT camera_id, file_paths FROM
cameras")
    cameras = cursor.fetchall()
    for camera in cameras:
        if camera['file_paths']:
            file_paths = camera['file_paths'].split(',')
            valid_file_paths = [fp for fp in file_paths if
os.path.exists(os.path.join(template_dir, fp))]
            new_file_paths = ','.join(valid_file_paths)
            if new_file_paths != camera['file_paths']:
                retry_db_execute(db, "UPDATE cameras SET
file_paths = ? WHERE camera_id = ?", (new_file_paths,
camera['camera_id']))
    db.commit()
```

Funkcija „map()“ na ruti „/map“ vraća web stranicu koji sadrži interaktivnu kartu. Na karti se prikazuju lokacije kamera koje su prethodno dodane putem gumba u donjem

desnom kutu za dodavanje kamera:

```
@app.route('/map')
@login_required
def map():
    return render_template('map.html')
```

Funkcija „get_camera_locations()“ na ruti „/get_camera_locations“ vraća popis svih dodanih kamera i njihovih lokacija u JSON formatu. Ove informacije se koriste za prikaz markera na karti u aplikaciji:

```
@app.route('/get_camera_locations', methods=['GET'])
@login_required
def get_camera_locations():
    user_id = session['user_id']
    db = get_db()
    cursor = db.execute('SELECT camera_id, camera_name,
location FROM cameras WHERE user_id = ?', (user_id,))
    cameras = cursor.fetchall()

    camera_data = []
    for camera in cameras:
        camera_data.append({
            'id': camera['camera_id'],
            'name': camera['camera_name'],
            'location': camera['location']
        })

    return jsonify(camera_data)
```

Funkcija „add_camera_marker()“ na ruti „/add_camera_marker“ omogućuje korisnicima dodavanje novih kamera na kartu uz pomoć gumba postavljenog u donjem desnom kutu web stranice. Kada korisnik odabere lokaciju na karti, funkcija pohranjuje tu lokaciju u bazu podataka i ažurira kartu s novim markerom. Ako kamera nije dodana, ažurira se baza podataka s novom lokacijom kamere i označava da je kamera dodana. Nakon uspjeha, vraća se potvrda s podacima kamere, dok se u suprotnom vraća poruka o grešci:

```
@app.route('/add_camera_marker', methods=['POST'])
@login_required
def add_camera_marker():
```

```
    lat = request.form.get('lat')
    lng = request.form.get('lng')
    camera_id = request.form.get('camera_id')

    if not lat or not lng or not camera_id:
        print(f"Missing data: lat={lat}, lng={lng},
camera_id={camera_id}")
        return jsonify({'success': False, 'message': 'Missing
data.'}), 400

    user_id = session['user_id']

    db = get_db()
    cursor = db.execute('SELECT * FROM cameras WHERE camera_id
= ? AND user_id = ? AND added_to_map = 0', (camera_id,
user_id))
    available_camera = cursor.fetchone()

    if available_camera:
        location = f"{lat},{lng}"
        db.execute('UPDATE cameras SET location = ?,
added_to_map = 1 WHERE camera_id = ?', (location, camera_id))
        db.commit()
        print(f"Camera marker added: camera_id={camera_id},
location={location}")
        return jsonify({'success': True, 'camera_name':
available_camera['camera_name'], 'camera_id': camera_id})
    else:
        print(f"Camera not found or already added:
camera_id={camera_id}")
        return jsonify({'success': False, 'message': 'This
camera is already added or does not exist.'}), 400
```

Funkcija „delete_camera_marker()“ na ruti „/delete_camera_marker“ omogućuje korisnicima brisanje kamera s karte klikom na gumb koji je prikazan klikom na marker. Funkcija ažurira bazu podataka tako da ukloni lokaciju kamere i marker s karte. Funkcija započinje dohvaćanjem identifikacijskog broja kamere (camera_id) putem POST zahtjeva. Nakon toga, dohvaća se trenutni korisnik iz sesije te se otvara veza s bazom podataka. Funkcija zatim provjerava postoji li kamera u bazi podataka povezana s korisnikom, te je li već dodana na kartu. Nakon uspješnog brisanja lokacije, funkcija vraća JSON odgovor sa kodom koji označava da je brisanje uspjelo. Spomenutim radnjama odgovara sljedeći kod:

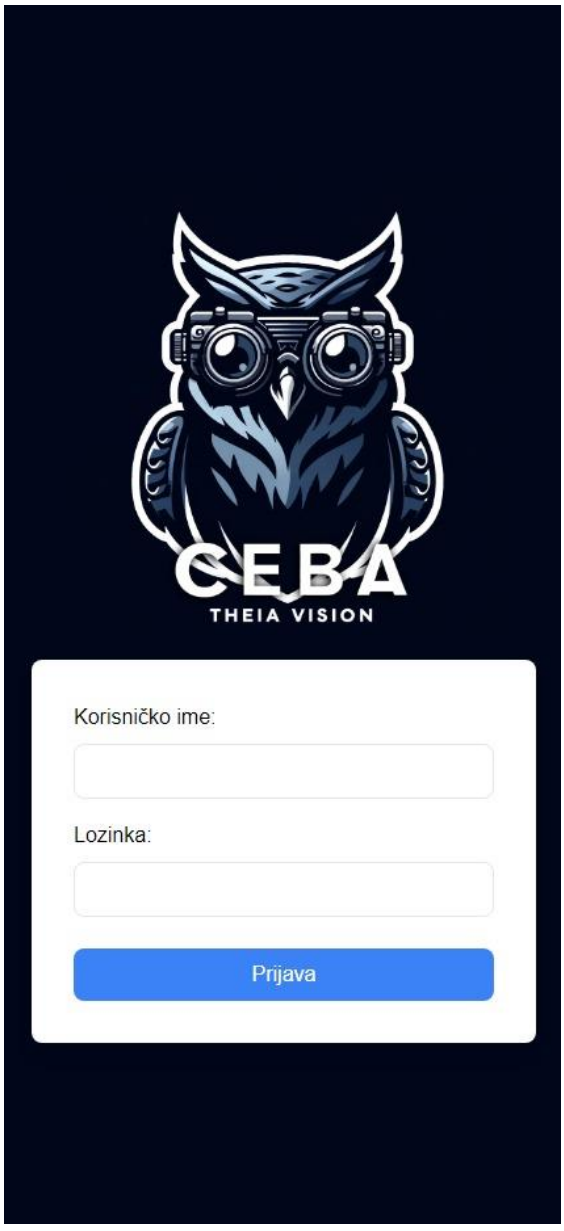
```
@app.route('/delete_camera_marker', methods=['POST'])
```

```
@login_required
def delete_camera_marker():
    camera_id = request.form.get('camera_id')

    if not camera_id:
        print("No camera ID received.")
        return jsonify({'success': False, 'message': 'Camera ID
is required.'}), 400

    user_id = session['user_id']
    db = get_db()

    cursor = db.execute('SELECT * FROM cameras WHERE camera_id
= ? AND user_id = ? AND added_to_map = 1', (camera_id,
user_id))
    camera = cursor.fetchone()

    if camera:
        db.execute('UPDATE cameras SET location = NULL,
added_to_map = 0 WHERE camera_id = ?', (camera_id,))
        db.commit()
        print(f"Camera ID {camera_id} deleted successfully.")
        return jsonify({'success': True})
    else:
        print(f"Camera ID {camera_id} not found or not
authorized.")
        return jsonify({'success': False, 'message': 'Camera
not found or not authorized.'}), 400
```

6.2.2. Korisničko sučelje

Korisničko sučelje započinje s početnom stranicom koja služi kao ulazna točka za korisnike sustava. Sučelje je napravljeno korištenjem HTML-a u kombinaciji sa kaskadnim stilskim listovima (*engl. Cascading Style Sheets*, CSS) i Tailwindom. Stranica sadrži logotip projekta, te formu za prijavu korisnika centriranu na sredinu stranice ispod logotipa (vidjeti sliku 24). Fotografija se prikazuje putem „img“ elementa koji koristi „url_for“ funkciju za dinamičko dohvaćanje datoteka iz statičkog direktorija. Forma za prijavu korisnika je smještena unutar „div“ elementa, koji je stiliziran da se pojavi u sredini ekrana s bijelom pozadinom i zaobljenim rubovima, kako bi se stvorio vizualno odvojen blok. Unutar forme nalaze se dva polja za unos: jedno za korisničko ime i drugo za lozinku. Oba polja su obavezna i imaju definirane atribute za automatsko popunjavanje (*engl. autocomplete*) kako bi se korisnicima olakšalo ponavljanje prijava. Elementi forme su stilizirani pomoću Tailwind CSS klasa, što omogućava jednostavno

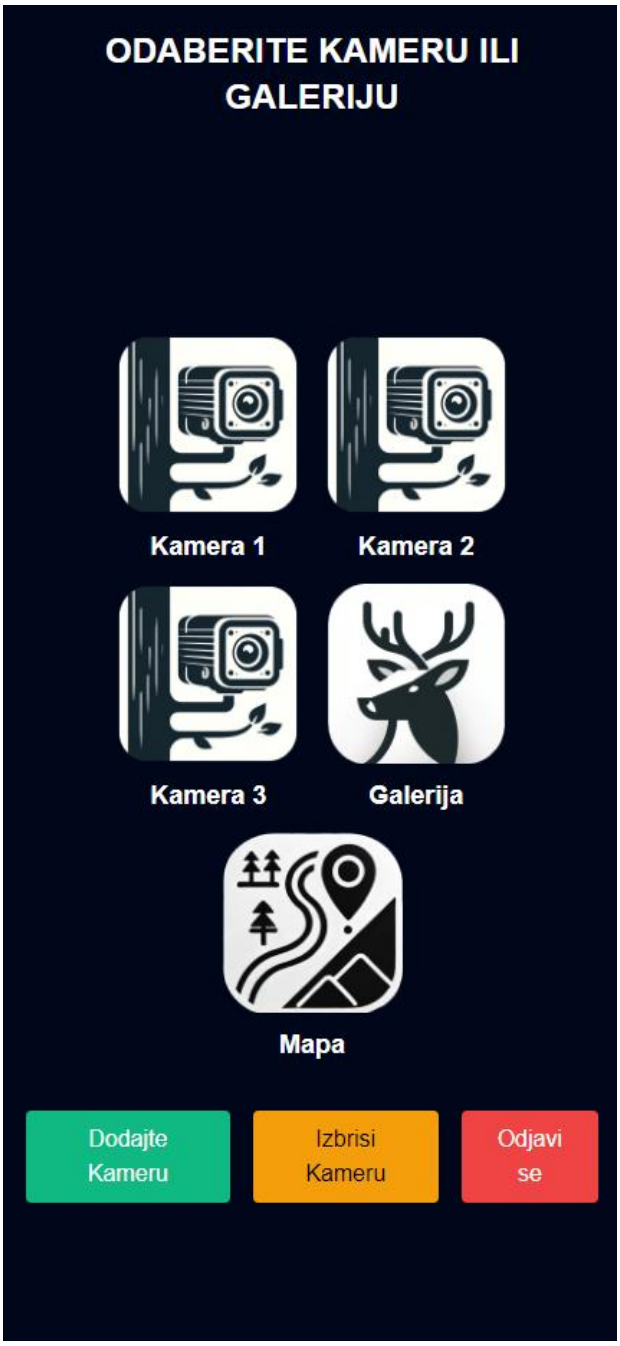
upravljanje izgledom stranice, uključujući interakciju poput promjene boje okvira prilikom fokusiranja na unos. Prijava se vrši putem „submit“ gumba koji je stiliziran tako da bude vizualno istaknut, s efektima promjene boje pri prelasku mišem (hover) i tranzicijom. Sučelje je prilagodljivo, te je tako zagarantirana namijenjena funkcionalnost na svakoj platformi i rezoluciji.



Slika 24: Prikaz sučelja web stranice za prijavu, prilagođeno mobitelu.

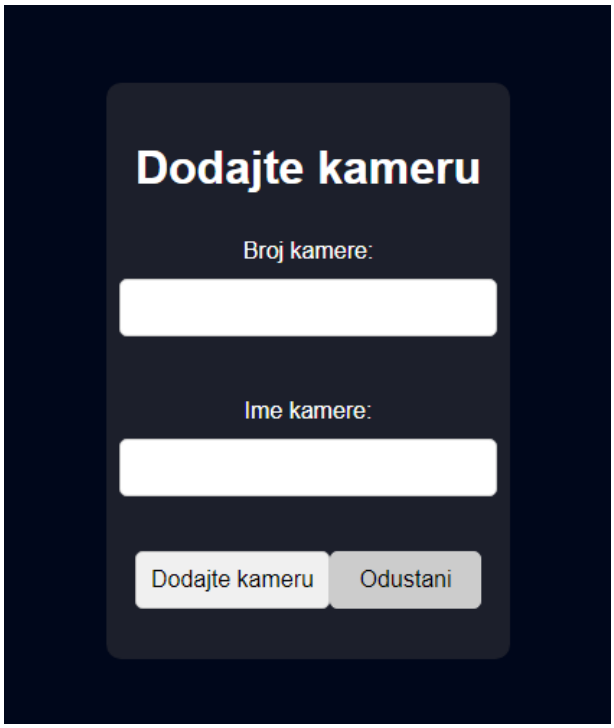
Nakon što se korisnik uspješno prijavi u sustav, prikazuje se početna stranica koja omogućuje pregled i upravljanje svim kamerama povezanima s njegovim računom, kao i pristup galeriji fotografija i interaktivnoj mapi (vidjeti sliku 25). Stranica je dizajnirana da bude jednostavna za navigaciju, intuitivna i vizualno privlačna krajnjem korisniku, koristeći kombinaciju HTML-a, CSS-a i JavaScript-a. Svaka kamera

povezana s korisnikovim računom prikazana je putem ikone kamere, s imenom ispisanim ispod. Ikone kamera su organizirane u rešetki koja se automatski prilagođava veličini ekrana, što omogućuje ispravan prikaz na različitim uređajima. Pored ikona kamera, korisnik ima mogućnost pristupa galeriji koja sadrži sve fotografije na kamerama putem ikone galerije te putem ikone mape interaktivnoj mapi koja sadrži lokacije postavljenih kamera ukoliko ih je korisnik postavio.



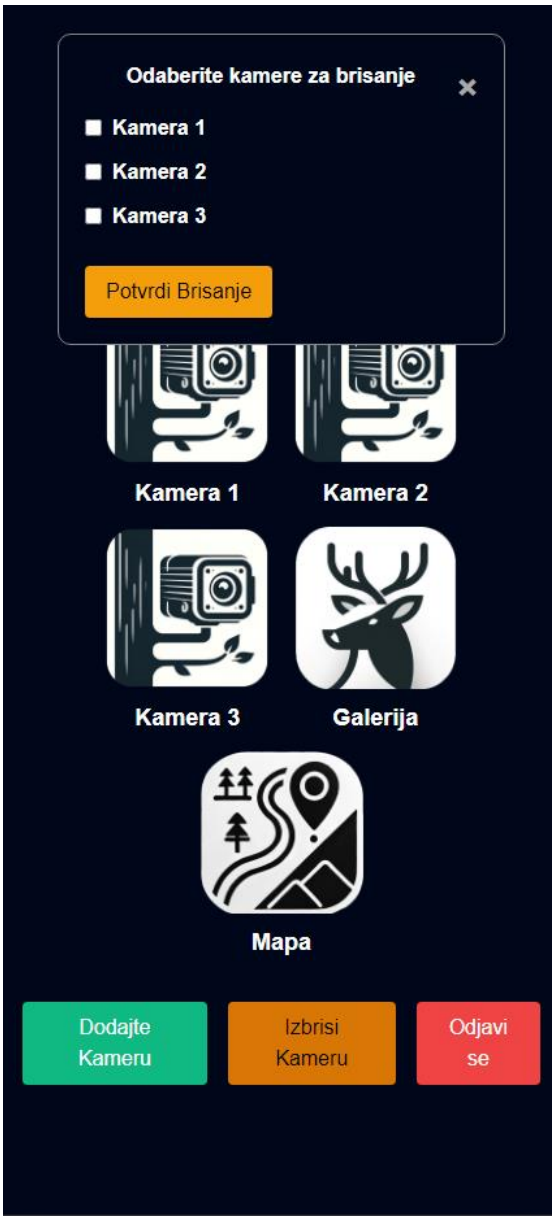
Slika 25: Prikaz sučelja web stranice za odabir kamera, prilagođeno mobitelu.

Kamere se dinamički prikazuju na stranici pomoću petlje unutar HTML-a koja iterira kroz sve kamere pridružene korisničkom računu. Ova funkcionalnost je implementirana putem logike u Python Flask okruženju, gdje se informacije o kamerama dohvaćaju iz baze podataka i šalju HTML-u prilikom učitavanja stranice. Svaka kamera se dodaje kao novi element u HTML strukturi, koristeći „for“ petlju. Ova petlja generira HTML kod za svaku kameru, uključujući ikonu, naziv kamere, te link koji vodi na stranicu koja sadrži sve fotografije koje je kamera fotografirala. Na dnu stranice nalaze se tri glavna gumba: gumb za dodavanje nove kamere, gumb za brisanje postojećih kamera i gumb za odjavu. Gumb za dodavanje kamera vodi korisnika na novu stranicu sa sučeljem u kojem je potrebno upisati 12-brojčani identifikator kamere, te ime kamere (vidjeti sliku 26). Identifikator kamere je brojčana vrijednost koja mora biti točno 12 znamenki, dok ime kamere može biti proizvoljan tekstualni unos. Da bi se osiguralo da je unos identifikatora ispravan, stranica uključuje JavaScript validaciju koja provjerava da je identifikator pozitivan broj s točno 12 znamenki. Ako uneseni identifikator ne zadovoljava ove kriterije, korisnik će biti obaviješten putem poruke o pogrešci. Nakon što korisnik unese potrebne podatke i potvrdi unos klikom na gumb "Dodajte kameru", forma se šalje poslužitelju putem POST zahtjeva. Akcija forme definirana je pomoću Flask funkcije „url_for('add_camera')“, koja šalje podatke poslužitelju gdje se obrađuju. Na poslužitelju, Python backend obrađuje primljene podatke. Kamera se dodaje u bazu podataka s navedenim identifikatorom i imenom.



Slika 26: Prikaz funkcije dodavanja novih kamera.

Gumb za brisanje kamera otvara modalni prozor u kojem korisnik može označiti kameru ili kamere koje želi ukloniti sa svog računa, modalni prozor se pojavljuje u središtu ekrana (vidjeti sliku 27). Na kraju ostaje gumb za odjavu korisnika koji putem Flask backenda briše podatke korisničke sesije i vraća korisnika na početnu stranicu. Izvorni kod ove stranice bez CSS-a, odnosno Tailwinda prikazan je u daljnjem tekstu radi boljeg razumijevanja principa rada.



Slika 27: Prikaz funkcije brisanja kamera*, prilagođeno* mobitelu.

Unutar "head" bloka HTML koda postavljaju se osnovne informacije i resursi za web stranicu. Definira se skup znakova kao „UTF-8“, osigurava se da stranica izgleda kao što je zamišljeno na svim uređajima pomoću "viewport" oznake, a stranica dobiva naslov "THEIA VISION". Također su dodane ikonice različitih rezolucija za prikaz na mobilnim uređajima i web preglednicima, koristeći „url_for“ funkciju koja omogućuje dinamičko dohvaćanje fotografija iz datoteka na poslužitelju. Opisano je prikazano sljedećim kodom:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>THEIA VISION</title>
      <link  rel="apple-touch-icon"  sizes="180x180"  href="{{
url_for('static', filename='webicon180x180.png') }}">
     <link  rel="icon"  type="image/png"  sizes="32x32"  href="{{
url_for('static', filename='webicon32x32.png') }}">
    <style>
</style>
</head>
```

Ovaj HTML kod kreira korisničko sučelje za upravljanje kamerama i pristup galeriji. Unutar <body> elementa, koristi se „flexbox“ raspored kako bi se elementi prilagodili ekranu. U zaglavlju se prikazuje naslov "ODABERITE KAMERU ILI GALERIJU", a glavni dio sadrži prikaz dostupnih kamera i opcije za galeriju i mapu, kroz koje korisnici mogu jednostavno navigirati putem klika na ikonu. Svaka kamera prikazana je u obliku fotografije, a ispod fotografije se nalazi ime kamere. Također, tu je gumb za dodavanje nove kamere, brisanje postojećih i odjavu sa sustava. Odgovarajući HTML kod dan je ispod:

```
<body class="min-h-screen flex flex-col">
    <header class="fixed top-0 w-full header-bg text-white p-4
z-50">
         <h1 class="text-center text-2xl md:text-3xl lg:text-
4xl">ODABERITE KAMERU ILI GALERIJU</h1>
    </header>
    <main class="flex-grow flex flex-col items-center justify-
center pt-24 md:pt-28 lg:pt-32 px-4">
         <div class="w-full max-w-6xl flex flex-wrap justify-
center gap-3 mt-8">
            {% for camera in cameras %}
             <div class="w-1/3 md:w-1/3 lg:w-1/4 flex flex-col
items-center">
                                <a  href="{{  url_for('home',
camera_id=camera.camera_id) }}" class="block">
                               <img  src="{{  url_for('static',
filename='cameraicon.webp')  }}"  class="mx-auto  icon-small"
alt="Kamera {{ camera.camera_id }}">
                </a>
                      <h2  class="text-center  text-lg  mt-2">{{
camera.camera_name }}</h2>
```

```
            </div>
            {% endfor %}
             <div class="w-1/3 md:w-1/3 lg:w-1/4 flex flex-col
items-center">
               <a href="{{ url_for('gallery') }}" class="block">
                                  <img  src="{{  url_for('static',
filename='galleryicon.png')  }}"  class="mx-auto  icon-small"
alt="Galerija">
                </a>
              <h2 class="text-center text-lg mt-2">Galerija</h2>
            </div>
             <div class="w-1/3 md:w-1/3 lg:w-1/4 flex flex-col
items-center">
                <a href="{{ url_for('map') }}" class="block">
                                  <img  src="{{  url_for('static',
filename='mapicon.png')   }}"   class="mx-auto   icon-small"
alt="Mapa">
                </a>
                <h2 class="text-center text-lg mt-2">Mapa</h2>
            </div>
        </div>
        <div class="mt-8 flex gap-4">
            <button class="bg-green-500 hover:bg-green-600 text-
white  py-2  px-4  rounded"  onclick="window.location.href='{{
url_for('add_camera') }}'">Dodajte Kameru</button>
              <button class="bg-yellow-500 hover:bg-yellow-600
text-black  py-2  px-4  rounded"  id="deleteCameraBtn">Izbrisi
Kameru</button>
           <button class="bg-red-500 hover:bg-red-600 text-white
py-2    px-4    rounded"    onclick="window.location.href='{{
url_for('logout') }}'">Odjavi se</button>
        </div>
    </main>
    <div id="deleteCameraModal" class="modal">
        <div class="modal-content">
            <span class="close" id="closeModal">&times;</span>
             <h2 class="text-center mb-4">Odaberite kamere za
brisanje</h2>
                                  <form   id="delete-camera-form"
action="/delete_cameras" method="POST">
                <div class="camera-list">
                    {% for camera in cameras %}
                     <div class="camera-item flex items-center
gap-2 mb-2">
                        <label for="camera_{{ camera.camera_id
}}" class="flex items-center gap-2 cursor-pointer">
                                        <input  type="checkbox"
name="camera_ids"  id="camera_{{  camera.camera_id  }}"  value="{{
camera.camera_id }}">
```

```
                                {{ camera.camera_name }}
                            </label>
                        </div>
                        {% endfor %}
                    </div>
                        <button type="submit" class="bg-yellow-500
hover:bg-yellow-600 text-black py-2 px-4 rounded mt-4">Potvrdi
Brisanje</button>
                </form>
            </div>
        </div>
```

Sljedeći JavaScript kod osigurava ispravnu interakciju korisnika s modalom za brisanje kamera. Kada korisnik klikne na gumb "Izbriši Kameru", modal se otvara, omogućujući izbor kamera za brisanje. Korisnik može zatvoriti modal klikom na ikonu zatvaranja ili izvan modala. Dio koda koji se odnosi na JavaScript-u nalazi se ispod:

```
        <script>
            var modal = document.getElementById("deleteCameraModal");
            var btn = document.getElementById("deleteCameraBtn");
            var span = document.getElementById("closeModal");

            btn.onclick = function() {
                modal.style.display = "block";
            }

            span.onclick = function() {
                modal.style.display = "none";
            }

            window.onclick = function(event) {
                if (event.target == modal) {
                    modal.style.display = "none";
                }
            }
        </script>
    </body>
    </html>
```

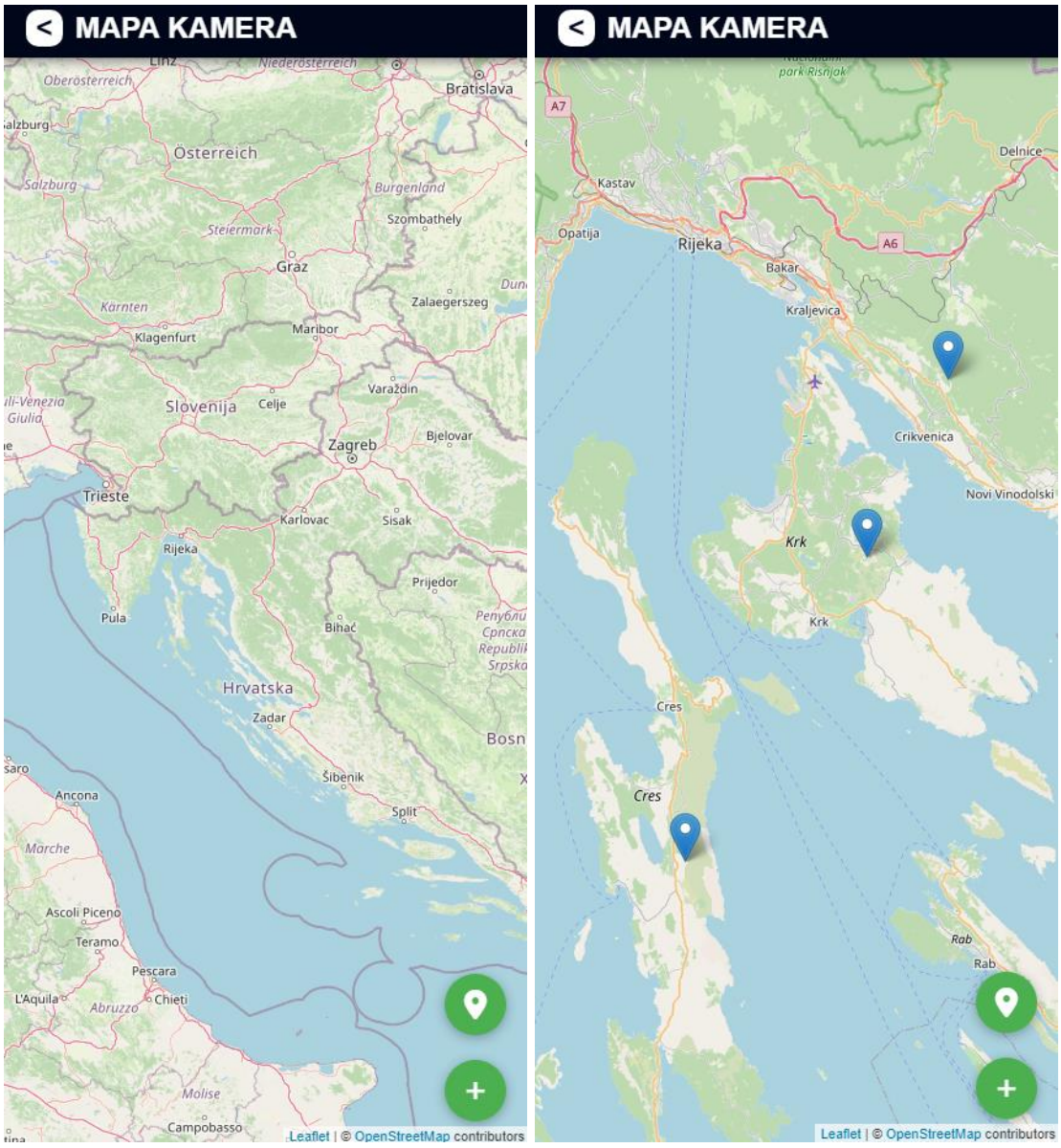
Stranica kamere omogućuje korisnicima pregled svih fotografija koje su snimljene određenom kamerom. Fotografije su organizirane u kategorije na temelju datuma snimanja, a korisnici mogu pregledavati, zumirati i vidjeti detalje svake pojedine fotografije. Glavni dio stranice zauzima mreža fotografija koja prikazuje sve fotografije

povezane s odabranom kamerom (vidjeti sliku 28). Fotografije su podijeljene u nekoliko kategorija na temelju datuma snimanja, uključujući "Danas", "Jučer", "Zadnjih 7 dana" i "Starije od 7 dana". Svaka kategorija prikazuje fotografije u obliku mreže s mogućnošću klika na fotografiju radi uvećanja i detaljnijeg pregleda. Nakon što se stranica učita, JavaScript API šalje zahtjev na rutu „/userimages/<camera_id>“, gdje „camera_id“ predstavlja identifikator odabrane kamere. Ovaj zahtjev vraća JSON objekt s URL-ovima fotografija. JavaScript zatim sortira fotografije prema datumu u nekoliko kategorija ("Danas", "Jučer", "Zadnjih 7 dana", "Starije od 7 dana"). Svaka fotografija se uspoređuje s trenutnim datumom kako bi se odredilo kojoj kategoriji pripada. Nakon kategorizacije, fotografije se dinamički dodaju na stranicu unutar odgovarajućih sekcija, čime se kreiraju HTML elementi kao što su „div“ i „img“, koji se zatim ubacuju u odgovarajuće dijelove stranice. Kada korisnik klikne na bilo koju fotografiju, otvara se modalni prozor koji prikazuje uvećanu verziju fotografije. JavaScript upravlja otvaranjem i zatvaranjem modala, omogućujući korisnicima da pregledaju fotografiju u punoj veličini.

Slika 28: Prikaz sučelja web stranice kamere koja sadrži fotografije, prilagođeno mobitelu.

Stranica s mapom, koja je sastavni dio korisničkog sučelja, omogućuje korisnicima pregled i upravljanje lokacijama kamera unutar sustava. Ova funkcionalnost je ostvarena korištenjem Leaflet knjižnice za interaktivne mape, koja u kombinaciji s OpenStreetMap podlogom omogućuje jednostavno pregledavanje i upravljanje

kamerama u stvarnom vremenu. Nakon učitavanja stranice, prikazuje se interaktivna karta koja zauzima cijeli prostor zaslona (vidjeti sliku 29). Mapa je inicijalno centrirana na zadanu geografsku lokaciju s koordinatama 45.1, 15.2, pri čemu je razina zumiranja postavljena na 7, što omogućuje pregled Republike Hrvatske. Za prikaz mape koristi se Leaflet knjižnica, koja omogućuje učitavanje kartografskih slojeva putem OpenStreetMap servisa. Ova mapa je interaktiva pa korisniku omogućava pregledavanje, zumiranje, dodavanje nove i brisanje stare kamere. Funkcionalnost dodavanja novih kamera na mapu započinje klikom na gumb za dodavanje kamere u donjem desnom kutu stranice, što otvara padajući izbornik u kojem su prikazane sve kamere dostupne za dodavanje. Ovaj izbornik dinamički se popunjava podacima dohvaćenim s backend poslužitelja putem asinkronih JavaScript i XML (*engl. Asynchronous JavaScript and XML*, AJAX) zahtjeva na rutu „/get_available_cameras“. Nakon što korisnik odabere kameru, može kliknuti na željenu lokaciju na mapi gdje želi postaviti kameru. Klikom na mapu, koordinate lokacije zajedno s identifikatorom odabrane kamere šalju se na backend putem POST zahtjeva na rutu „/add_camera_marker“. Backend provjerava valjanost podataka te ažurira bazu podataka s novom lokacijom kamere. Kamera se zatim prikazuje na mapi kao marker, zajedno s pop-up prozorom koji nudi opcije brisanja markera ili pregleda fotografija kamere. U slučaju da korisnik već ima postavljene lokacije kamera, pri učitavanju stranice automatski se šalje zahtjev poslužitelju za dohvaćanje svih kamera koje su već postavljene na mapi. Ruta „/get_camera_locations“ na backendu vraća podatke o kamerama u obliku JSON objekta, uključujući ID, naziv i geografske koordinate kamera. Ti podaci se koriste za postavljanje markera na mapi na odgovarajućim lokacijama. Svaka kamera prikazana na mapi može se jednostavno obrisati putem pop-up prozora markera. Klikom na gumb "Obriši lokaciju", unutar pop-up prozora, šalje se POST zahtjev poslužitelju na rutu „/delete_camera_marker“, gdje se traži brisanje kamere iz baze podataka. Poslužitelj provjerava autentičnost korisnika, te se kamera briše iz baze podataka, a marker se uklanja s mape. Stranica također podržava funkcionalnost lociranja korisnika. Klikom na gumb za lociranje, pokreće se HTML5 geolokacijski API koji dohvaća GPS koordinate korisnika. Nakon što se koordinate uspješno dobiju, karta se automatski centrirana na tu lokaciju, a na mapu se dodaje privremeni marker koji označava trenutnu poziciju korisnika.



Slika 29: Prikaz mape bez dodanih kamera (lijevo), s dodanim kamerama (desno), prilagođeno mobitelu.

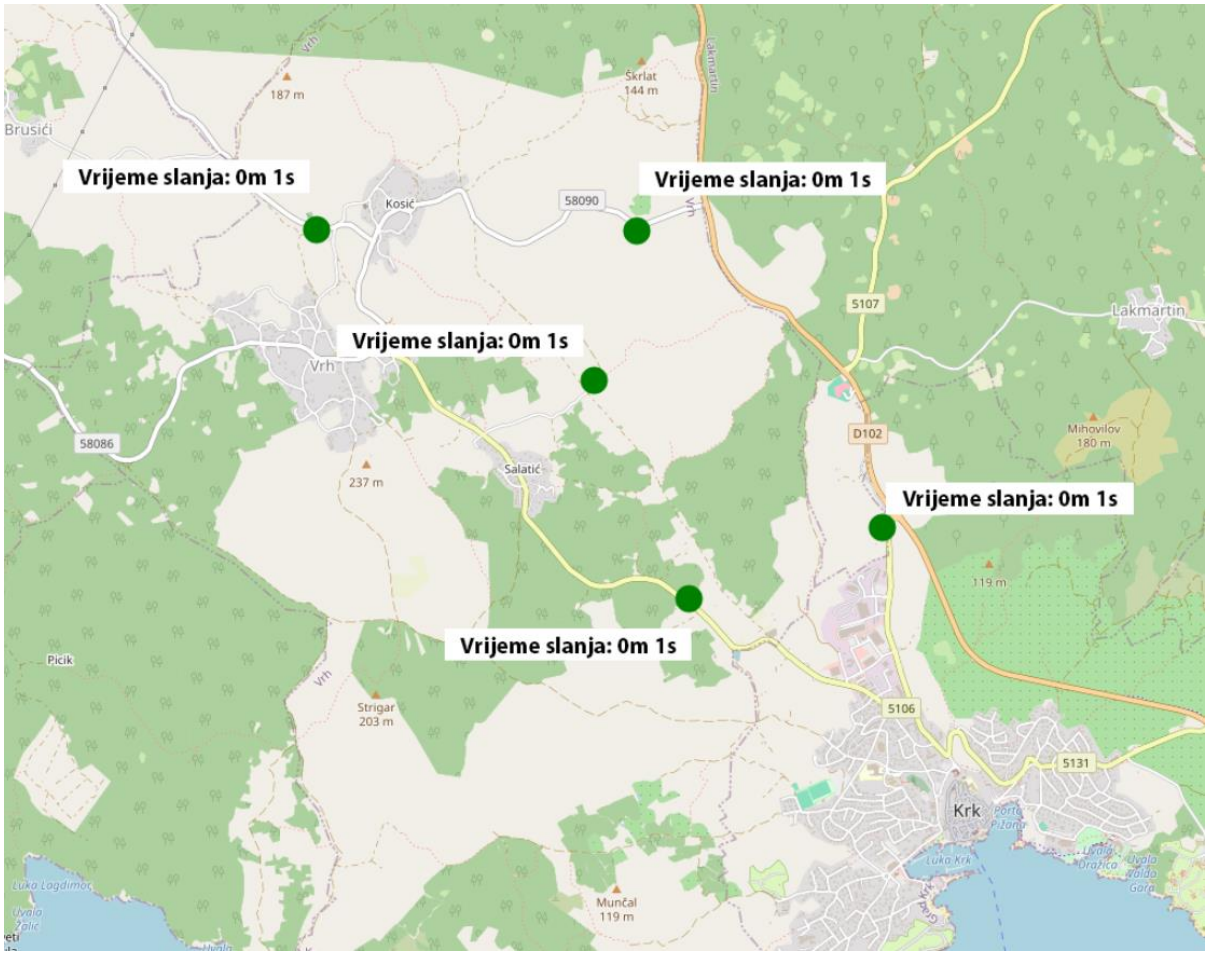
7. Metoda mjerenja

Za mjerenje vremena prijenosa podataka na poslužitelj, provedena su mjerenja na različitim lokacijama na otoku Krku. Na Raspberry Pi 4B mikroračunalu, razvijena je Bash skripta koja precizno mjeri trajanje izvršenja Python koda zaduženog za slanje fotografija i videozapisa na poslužitelj. Svaki uspješan prijenos bilježi se zajedno s trajanjem prijenosa. Na svakoj odabranoj lokaciji postupak slanja fotografija ponovljen je pet puta, nakon čega je izračunato prosječno vrijeme prijenosa za svaku lokaciju. Mjerenje prosječnog trajanja prijenosa provedena je na isti način i za prijenos videozapisa trajanja 30 sekundi, koristeći istu metodu. Mjerenja su provedena 16. srpnja 2024. između 9:00 i 12:00 h.

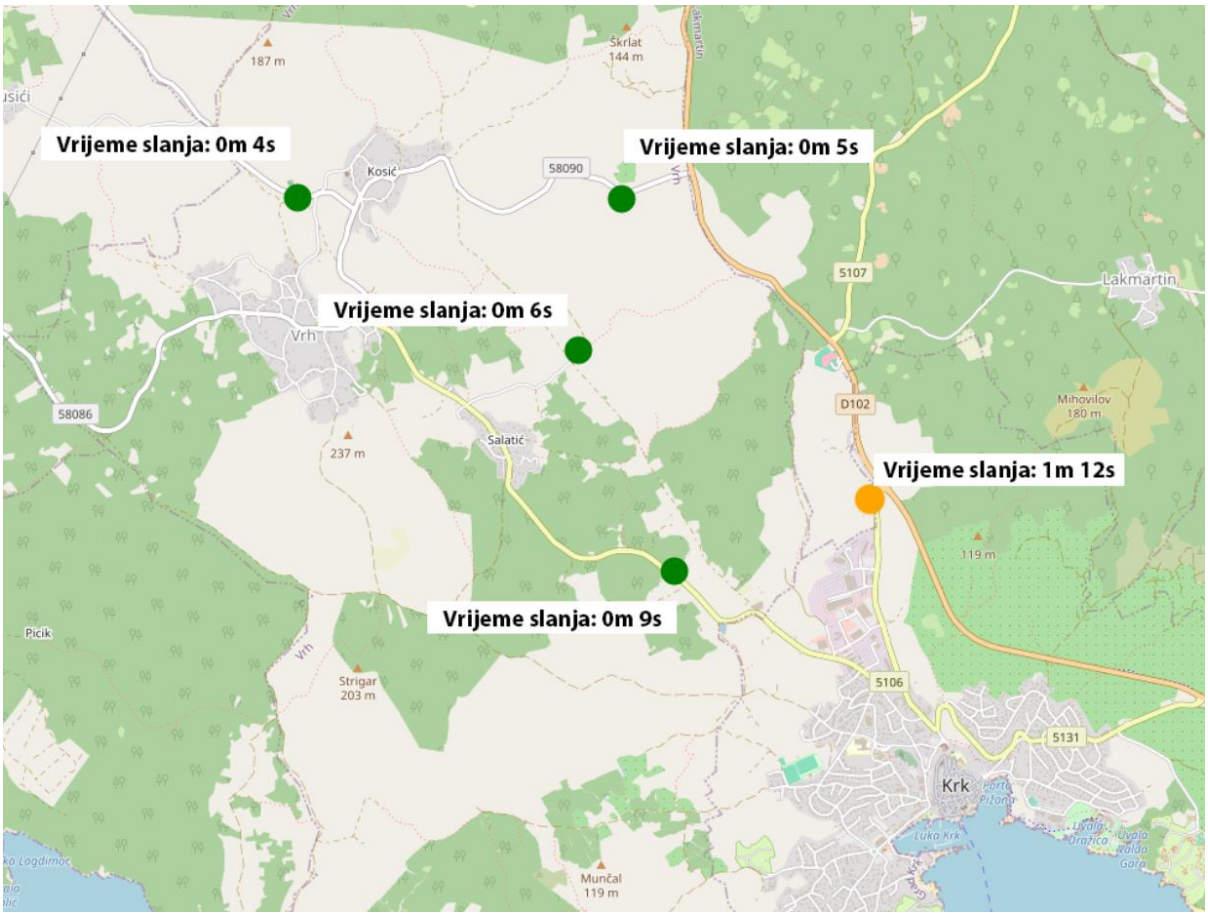
8. Rezultati

8.1. Rezultati mjerenja

Za potrebe prijenosa podataka korištena je SIM kartica operatera Hrvatski Telekom radi njihove pokrivenosti. Veličina učitanog videozapisa od 30 sekundi iznosila je 4.5 megabajta, dok je veličina učitane fotografije iznosila 111 kilobajta. Budući da je korištena 4G mreža učitavanje je veoma brzo, to je prikazano na slici 30, gdje se vidi da je prosječno vrijeme slanja fotografija iznosilo oko 1 sekundu na svakoj lokaciji, te prosječnim vremenom slanja videozapisa od 6 sekundi ne uzimajući u obzir anomaliju od 1m i 12s koja je nastala radi opterećenja mreže prilikom prijenosa dva od pet videozapisa (vidjeti sliku 31).



Slika 30: Mjerenja slanja fotografija.



Slika 31: Mjerenja slanja videozapisa.

8.2. Prikaz snimljenih fotografija

Prilikom testiranja fotografije su snimljene i prenesene na poslužitelj u stvarnim uvjetima, tijekom dana i noći. Važno je napomenuti da su ove fotografije uključene prvenstveno kako bi ilustrirale tehničku funkcionalnost razvijenog sustava za praćenje divljih životinja. Snimljene fotografije ne prikazuju divlju životinju, unatoč tome, funkcionalnost sustava je temeljito ispitana i potvrđena korištenjem dostupnih subjekata, u ovom slučaju ptice (vidjeti sliku 32), što je omogućilo validaciju ključnih komponenti kao što su detekcija pokreta, noćno snimanje (vidjeti sliku 33), prijenos podataka putem LTE mreže te integracija s poslužiteljem i web sučeljem. Snimljene fotografije stoga ne služe kao prikaz stvarnih aplikacija sustava za praćenje divljih životinja, već su uključene kao dokaz uspješnog funkcioniranja određenih tehničkih aspekata sustava u različitim uvjetima rada.

Slika 32: Fotografija snimljena automatskom kamerom tijekom dana.

Slika 33: Fotografija snimljena automatskom kamerom tijekom noći.

9. Diskusija

Ovaj završni rad uspješno je ostvario postavljeni cilj izrade sustava za praćenje divljih životinja, temeljenog na Raspberry Pi 4B mikroračunalu i Arduino Nano mikroupravljaču. Sustav se pokazao pouzdanim u prikupljanju i slanju podataka na poslužitelj zahvaljujući korištenju 4G mreže, koja omogućuje brz i stabilan prijenos podataka s terena na centralni poslužitelj. Korištenje PIR senzora za detekciju pokreta te Raspberry Pi HQ kamera za snimanje fotografija u kombinaciji sa infracrvenom LED pločom osiguralo je da sustav radi i u uvjetima slabog osvjetljenja, uključujući noćne sate, što je ključno za praćenje aktivnosti divljih životinja u njihovom prirodnom okruženju.

Iako je sustav uspješno realiziran, jedno od uočenih nedostataka je korištenje kućišta izrađenog tehnologijom 3D printanja od PLA plastike. PLA je popularan izbor za 3D printanje zbog svoje dostupnosti i jednostavnosti upotrebe, no ima nekoliko značajnih nedostataka kada se koristi u vanjskim uvjetima. Naime nije otporna na visoke temperature koliko njezine alternative te je podložna bržem propadanju pod utjecajem UV zračenja i vlage. Zbog toga, preporučuje se razmotriti korištenje materijala poput akrilonitril butadien stirena (*engl. Acrylonitrile butadiene styrene*, ABS) ili polietilen terefalat glikola (*engl. Polyethylene terephtalate glycol*, PETG), koji imaju bolje mehaničke karakteristike, veću otpornost na temperature i otpornost na atmosferske uvjete (Mendenhall i Eslami, 2023). Time bi se produžila dugovječnost i pouzdanost sustava u različitim okolišnim uvjetima.

Iako je Raspberry Pi 4B mikroračunalo idealan izbor radi svoje snage, njegova cijena i potrošnja električne energije predstavljaju izazov, posebno kada je riječ o izradi više uređaja za praćenje. Radi toga, predlaže se istražiti mogućnosti prilagodbe sustava na platformu poput ESP32 mikrokontrolera. ESP32 je znatno jeftiniji i troši manje električne energije od Raspberry Pi 4B-a, ali pruža sve ključne funkcionalnosti potrebne za ovaj projekt, uključujući mogućnost spajanja na mrežu, rad sa senzorima i kamerama, te dovoljnu procesorsku snagu za obradu podataka i upravljanje kamerama. Adaptiranjem sustava na ESP32 mogla bi se značajno smanjiti cijena po uređaju, što bi bilo korisno za masovnu implementaciju.

Ovaj sustav se pokazao boljim ili barem jednako učinkovitim kao postojeća

komercijalna rješenja, nadogradnjom na 4G mrežu osigurano je pouzdano slanje podataka, što većina postojećih rješenja ne može garantirati budući da se oslanjaju na 2G mrežu. Međutim, postoji prostor za dodatna poboljšanja, osobito u smislu kućišta, optimizacije troškova korištenjem alternativnih platformi poput ESP32, te implementacija solarnog punjenja baterija za duži rad sustava.

10. Zaključak

U ovom radu predstavljen je sustav za nadzor divljih životinja u prirodnom staništu koji se sastoji od kamere s automatskim okidanjem, udaljenog poslužitelja s bazom podataka i web stranicom putem koje korisnici mogu pristupiti snimljenim fotografijama i videozapisima. Kamera je izrađena korištenjem Raspberry Pi mikroračunala i Arduino Nano mikroupravljača, uz integraciju PIR senzora, Raspberry Pi HQ kamere i LTE modula za prijenos podataka. Okidanje kamere se vrši putem PIR senzora koji detektira pokret toplokrvne životinje. Arduino Nano prilikom pozitivne detekcije uključuje Raspberry Pi mikroračunalo, koje pokreće fotografiranje ili snimanje videozapise divlje životinje koja se nalazi u dometu kamere, te po završetku radnje datoteke učitava na poslužitelj pomoću LTE prijenosa podataka. Automatska kamera na taj način omogućava pouzdano i energetski učinkovito praćenje životinja, uključujući fotografiranje i snimanje videozapisa tijekom noćnih sati zahvaljujući infracrvenom osvjetljenju.

Rezultati testiranja prijenosa fotografija i videozapisa s kamere na poslužitelj pokazali su da je sustav brz i pouzdan, s prosječnim vremenom prijenosa fotografija od oko 1 s i videozapisa između 4 do 9 s, uz izuzetak jednog mjerenja gdje je došlo do većeg odstupanja. Na taj način je krajnjim korisnicima (lovcima i istraživačima) putem web sučelja omogućen pristup fotografijama i videozapisima praktički u realnom vremenu.

Ovim sustavom doprinosi se očuvanju okoliša i bioraznolikosti omogućavanjem neinvazivnog praćenja populacija u prirodnim staništima. Precizno praćenje omogućuje bolje razumijevanje migracijskih obrazaca i ponašanja životinja, što može biti ključno za sprječavanje prekomjernog povećanja populacija ili rano otkrivanje invazivnih vrsta koje ugrožavaju ekosustav. Sustav se može koristiti kao alat u istraživačkim i zaštitnim programima, čime doprinosi održivom upravljanju populacijama divljih životinja.

Moguće nadogradnje uključuju proširenje funkcionalnosti s dodatnim senzorima za praćenje okolišnih čimbenika poput temperature i vlage, kao i primjenu naprednih algoritama za prepoznavanje životinjskih vrsta. Uz to, daljnji razvoj softvera mogao bi omogućiti poboljšanu analizu podataka te automatsku identifikaciju i izvještavanje o promjenama u populacijama ili ponašanju životinja.

Literatura

Arduino. (2024). Getting started with Arduino IDE 2.0. Arduino Documentation. Preuzeto 30. kolovoza 2024. s https://docs.arduino.cc/software/ide-v2/tutorials/getting-started-ide-v2/

Arduino. (n.d.). Arduino Mega 2560 Rev3. Arduino Store. Preuzeto 30. kolovoza 2024. s https://store.arduino.cc/products/arduino-mega-2560-rev3

Arduino. (n.d.). Arduino Nano. Arduino Store. Preuzeto 30. kolovoza 2024. s https://store.arduino.cc/en-hr/products/arduino-nano

Arduino. (n.d.). Arduino Uno Rev3. Arduino Store. Preuzeto 30. kolovoza 2024. s https://store.arduino.cc/products/arduino-uno-rev3

Chipoteka. (2024). Kompatibilni 5V relej modul za Arduino. Chipoteka. Preuzeto 4. rujna 2024. s https://www.chipoteka.hr/kompatibilni-5v-relej-modul-za-arduino-8090229155

Badamasi, Y. (2014). The working principle of an Arduino. 2014 11th International Conference on Electronics, Computer and Computation (ICECCO), 1-4. https://doi.org/10.1109/ICECCO.2014.6997578

Balon, B., & Simic, M. (2019). Using Raspberry Pi Computers in Education. 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 671-676. https://doi.org/10.23919/MIPRO.2019.8756967

Cai, Y., & Arney, T. (2018). Scripting for Administration, Automation and Security. Proceedings of the 19th Annual SIG Conference on Information Technology Education. https://doi.org/10.1145/3241815.3241829

Contabo. (2024). Contabo: Cloud VPS, dedicated servers, and colocation. Preuzeto 30. kolovoza 2024. s https://contabo.com/en/

Dmitrović, S. (2020). What is C++? U Modern C++ for Absolute Beginners. https://doi.org/10.1007/978-1-4842-6047-0_2

Ellis, M., Thompson, G., & Haskell-Dowland, P. (2019). FROM MODEL B TO MICRO – TEACHING COMPUTER SCIENCE. EDULEARN19 Proceedings. https://doi.org/10.21125/EDULEARN.2019.0744

Galadima, A. (2014). Arduino as a learning tool. 2014 11th International Conference on Electronics, Computer and Computation (ICECCO), 1-4. https://doi.org/10.1109/ICECCO.2014.6997577

Hart-Davis, G. (2017). Choosing Operating Systems for Raspberry Pi. , 55-69. https://doi.org/10.1007/978-1-4842-2304-8_3

Hodgson, D. (2021). Mechatronics and Physical Computing. , 453-477. https://doi.org/10.1016/b978-0-12-815073-3.00020-x

Hwu, K., & Peng, T. (2012). A novel buck–boost converter combining KY and buck converters. IEEE Transactions on Power Electronics, 27(5), 2236-2241. https://doi.org/10.1109/TPEL.2011.2182208

Jin-jun, L. (2007). Interface design of AVR microcomputer with serial clock DS3231. International Electronic Elements.

Jordan, D. (1990). Implementation benefits of C++ language mechanisms. Communications of the ACM, 33(7), 61-64. https://doi.org/10.1145/83880.84460

Jotrin Electronics. (n.d.). MDM9600 product information. Preuzeto 30. kolovoza 2024. s https://www.jotrin.com/product/parts/MDM9600

Juan, R., Kim, J., Sa, Y., Kim, H., & Cha, H. (2016). Development of a Sensing Module for Standing and Moving Human Body Using a Shutter and PIR Sensor. , 11, 47-56. https://doi.org/10.14257/IJMUE.2016.11.7.05

Khan, S., Chowdhury, M., & Nandy, U. (2023). LTE/LTE-A Based Advanced Wireless Networks. Journal of Engineering Research and Reports. https://doi.org/10.9734/jerr/2023/v25i101012

Loukides, M., & Oram, A. (1996). Programming with GNU software. . https://doi.org/10.1016/s0898-1221(97)90157-7

Mendenhall, R., & Eslami, B. (2023). Experimental Investigation on Effect of Temperature on FDM 3D Printing Polymers: ABS, PETG, and PLA. *Applied Sciences*. https://doi.org/10.3390/app132011503.

Microsoft. (2021). What is Windows 10 IoT Core? Windows IoT Core documentation. Preuzeto 30. kolovoza 2024. s https://learn.microsoft.com/en-us/previous-versions/windows/iot-core/windows-iot-core#what-is-windows-10-iot-core

Oliphant, T. (2007). Python for Scientific Computing. Computing in Science & Engineering, 9. https://doi.org/10.1109/MCSE.2007.58

Pan, T., & Zhu, Y. (2018). Getting Started with Arduino. , 3-16. https://doi.org/10.1007/978-981-10-4418-2_1

Prakash, A. (2021). What is Shebang in Linux Shell Scripting? Linux Handbook. Preuzeto 30. kolovoza 2024. s https://linuxhandbook.com/shebang/

Raspberry Pi Foundation. (n.d.). Camera software. Raspberry Pi Documentation.

Preuzeto 30. kolovoza 2024. s https://www.raspberrypi.com/documentation/computers/camera_software.html

Raspberry Pi Foundation. (n.d.). Raspberry Pi High Quality Camera. Raspberry Pi. Preuzeto 30. kolovoza 2024. s https://www.raspberrypi.com/products/raspberry-pi-high-quality-camera/

Raspberry Pi Foundation. (n.d.). Specifikacije Raspberry Pi 4 Model B. Raspberry Pi. Preuzeto 30. kolovoza 2024. s https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/

Ritchie, D. M. (1993). The development of the C language. ACM SIGPLAN Notices, 28(3), 201-208. https://doi.org/10.1145/154766.155580

Saari, M., Baharudin, A., & Hyrynsalmi, S. (2017). Survey of prototyping solutions utilizing Raspberry Pi. 2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 991-994. https://doi.org/10.23919/MIPRO.2017.7973568

Segawa, J., Shirota, Y., Fujisaki, K., Kimura, T., & Kanai, T. (2014). Aggressive use of Deep Sleep mode in low power embedded systems. 2014 IEEE COOL Chips XVII, 1-3. https://doi.org/10.1109/CoolChips.2014.6842956

SSH Communications Security. (2024). SSH command. SSH Academy. Preuzeto 30. kolovoza 2024. s https://www.ssh.com/academy/ssh/command

Watkiss, S. (2016). More Input and Output: Infrared Sensors and LCD Displays. , 89-126. https://doi.org/10.1007/978-1-4842-1898-3_5

Zhou, M., Lin, H., Young, S., & Yu, J. (2018). Hybrid sensing face detection and registration for low-light and unconstrained conditions.. Applied optics, 57 1, 69-78 . https://doi.org/10.1364/AO.57.000069

Žužić, L., & Vale, D. (2024). Razvoj nadzornog sustava za lovce s prijenosom podataka na daljinu. Rad prihvaćen za objavljivanje.

Popis slika

Popis tablica