



VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

FUNDAMENTINIŲ MOKSLŲ FAKULTETAS

INFORMACINIŲ SISTEMŲ KATEDRA

Raimondas Sadkevičius

**AUTONOMINĖS VAIZDO STEBĖJIMO SISTEMOS VAIZDAMS ATPAŽINTI
KŪRIMAS, PAREMTAS MIKROVALDIKLIU
DEVELOPMENT OF AN AUTONOMOUS VIDEO SURVEILLANCE SYSTEM FOR
IMAGE RECOGNITION BASED ON A MICROCONTROLLER**

Baigiamasis magistro darbas

Informacijos ir informacinių technologijų saugos studijų programa, valstybinis kodas 6211BX014

Informatikos inžinerijos studijų kryptis

Vilnius, 2025

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

FUNDAMENTINIŲ MOKSLŲ FAKULTETAS

INFORMACINIŲ SISTEMŲ KATEDRA

Raimondas Sadkevičius

**AUTONOMINĖS VAIZDO STEBĖJIMO SISTEMOS VAIZDAMS ATPAŽINTI
KŪRIMAS, PAREMTAS MIKROVALDIKLIU
DEVELOPMENT OF AN AUTONOMOUS VIDEO SURVEILLANCE SYSTEM FOR
IMAGE RECOGNITION BASED ON A MICROCONTROLLER**

Baigiamasis magistro darbas

Informacijos ir informacinių technologijų saugos studijų programa, valstybinis kodas 6211BX014

Informatikos inžinerijos studijų kryptis

Vadovas

prof. dr. Nikolaj Goranin

(Moksl. laipsnis/pedag. vardas, vardas, pavardė)

Lietuvių kalbos konsultantas dr. Vaida Buivydienė

(Moksl. laipsnis/pedag. vardas, vardas, pavardė)

Vilnius, 2025

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
FUNDAMENTINIŲ MOKSLŲ FAKULTETAS
INFORMACINIŲ SISTEMŲ KATEDRA

Studijų kryptis: Informatikos inžinerija

Studijų programa: Informacijos ir informacinių technologijų sauga , valstybinis kodas
6211BX014

Specializacija: Informacijos ir informacinių technologijų sauga

TVIRTINU

Katedros vedėjas

Nikolaj Goranin

2025-05-27

MAGISTRO BAIGIAMOJO DARBO UŽDUOTIS

Nr. ITSfm-23-10867

Vilnius

Studentas (-ė): Raimondas Sadkevičius

Baigiamojo darbo tema: Autonominės vaizdo stebėjimo sistemos vaizdams atpažinti kūrimas, paremtas mikrovaldikliu

Baigiamojo darbo užbaigimo terminas pagal numatytą studijų kalendorinį grafiką.

BAIGIAMOJO DARBO UŽDUOTIS:

Tikslas: Sukurti autonomiškai veikiančią video stebėjimo sistemą, veikiančią mikrokontrolerio (pvz. Raspberry) ir akumuliatoriaus pagrindu, galinčią aptikti tam tikro tipo objektus ir automatiškai pranešti apie juos. Taip pat turi būti atlikti bandymai siekiant kuo ilgiau užtikrinti sistemos autonominio veikimo laiką.

Uždaviniai:

1. Atlikti probleminės srities ir analogiškų sistemų analizę.
2. Suprojektuoti sistemą.
3. Įgyvendinti projektą, išbandyti sistemos veikimą su skirtingais DI modeliais, optimizuoti sistemos veikimą pagal veikimo laiko parametą, atlikti testimą.

Planuojamas rezultatas: Autonominė vaizdo stebėjimo sistema su objektų atpažinimu.

Vadovas profesorius Nikolaj Goranin

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

Raimondas Sadkevičius, 20203289

(Studento vardas ir pavardė, studento pažymėjimo Nr.)

Fundamentinių mokslų fakultetas

(Fakultetas)

Informacijos ir informacinių technologijų sauga, ITSfm-23

(Studijų programa, akademinė grupė)

BAIGIAMOJO DARBO (PROJEKTO)

SĄŽININGUMO DEKLARACIJA

2025 m. gegužės 27 d.

Patvirtinu, kad mano baigiamasis darbas tema „Autonominės vaizdo stebėjimo sistemos vaizdams atpažinti kūrimas, paremtas mikrovaldikliu“ yra savarankiškai parašytas. Šiame darbe pateikta medžiaga nėra plagijuota. Tiesiogiai ar netiesiogiai panaudotos kitų šaltinių citatos pažymėtos literatūros nuorodose.

Prenkant ir įvertinant medžiagą bei rengiant baigiamąjį darbą, mane konsultavo mokslininkai ir specialistai: daktaras Vaida Buivydienė. Mano darbo vadovas profesorius daktaras Nikolaj Goranin.

Kitų asmenų indėlio į parengtą baigiamąjį darbą nėra. Jokių įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs (-usi).

Raimondas Sadkevičius

(Parašas)

(Vardas ir pavardė)

Vilniaus Gedimino technikos universitetas		ISBN	ISSN
Fundamentinių mokslų fakultetas		Egz. sk.	
Informacinių sistemų katedra		Data-.....-.....	

Antrosios pakopos studijų Informacijos ir informacinių technologijų saugos programos magistro baigiamasis darbas	
Pavadinimas	Autonominės vaizdo stebėjimo sistemos vaizdams atpažinti kūrimas, paremtas mikrovaldikliu
Autorius	Raimondas Sadkevičius
Vadovas	Nikolaj Goranin

	Kalba: lietuvių
--	------------------------

Anotacija
Baigiamajame magistro darbe nagrinėjama autonominių vaizdo stebėjimo sistemų su objektų atpažinimų problematika, nagrinėjama sistemos sandara, jos veikimo būdai bei objektų atpažinimo algoritmai. Pateikta autonominės vaizdo stebėjimo sistemos palyginamoji analizė, pabrėžti sistemos privalumai ir trūkumai. Išanalizuoti egzistuojantys sprendimai ir išryškinta projekto problematika – sistemos veikia nedideliu atstumu, bei veikia tik prisijungus prie bevielio tinklo, arba atpažįsta objektus ne pagal jų klasifikaciją naudojant DI, o pagal jų formą bei veikia su jutikliais, kurie pagerina jų veikimo laiką, tačiau tai neatlieka pagrindinės siekiamos užduoties – užfiksuoti objektą atokioje vietoje, kurioje neįmanoma prijungti bevielio tinklo arba kur reikia tiksliai nustatyti analizuojamą objektą. Tiriamojoje dalyje parinkti sistemos komponentai. Atlikti sistemos bandymai realiomis sąlygomis, kurie parodė, kad veikimo laikas ir stabilumas gali būti toliau tobulinami priklausomai nuo poreikių. Eksperimentinis metodas ir iteracinis tobulinimas parodė, kad net riboto galimybių mikrovaldiklis gali tapti pagrindu inovatyviems stebėjimo sprendimams.

Prasminiai žodžiai: vaizdo stebėjimo sistema, mikrovaldiklis, vaizdo apdorojimas ir atpažinimas, autonominė kamera, Raspberry Pi, objektų atpažinimas
--

Vilnius Gediminas Technical University		ISBN	ISSN
Faculty of Fundamental Sciences		Copies No.	
Department of Information Systems		Date-.....-.....	

Master Degree Studies Information and Information Technologies Security study programme Master Graduation Thesis	
Title	Development of an Autonomous Video Surveillance System Based on a Microcontroller with Object Recognition
Author	Raimondas Sadkevičius
Academic supervisor	Nikolaj Goranin

	Thesis language: Lithuanian
--	------------------------------------

Annotation
The final master thesis deals with the problems of autonomous video surveillance systems with object recognition, the structure of the system, its methods of operation and object recognition algorithms. A comparative analysis of the autonomous video surveillance system is presented, the advantages and disadvantages of the system are highlighted. Existing solutions are analyzed and the problems of the project are highlighted - the systems operate at short distances and only work when connected to a wireless network, or they recognize objects by their shape rather than by their classification using IoT, and they work with sensors which improve their lifetime, but this does not fulfil the main objective of capturing an object in a remote location where it is not possible to connect to the wireless network, or where it is necessary to pinpoint the object under analysis. In the exploratory part, the components of the system have been selected. The system has been tested under realistic conditions, showing that uptime and stability can be further improved depending on the needs. The experimental approach and the iterative refinement have shown that even a microcontroller with limited capabilities can provide the basis for innovative monitoring solutions.

Keywords: video surveillance system, microcontroller, image processing and recognition, autonomous camera, Raspberry Pi, object recognition
--

TURINYS

IVADAS.....	12
1. AUTONOMINIŲ VAIZDO STEBĖJIMO SISTEMŲ ANALIZĖ.....	14
1.1. Autonominių vaizdo stebėjimo sistemų problematika.....	14
1.1.1. Techniniai projekto įgyvendinimo iššūkiai.....	15
1.1.2. Ryšio ir duomenų valdymo problemos.....	15
1.2. Autonominių vaizdo stebėjimo sistemų analizė.....	15
1.2.1. Autonominių vaizdo stebėjimų sistemų kritinės taikymo sritys.....	16
1.2.1. Objektų atpažinimo technologijų analizė.....	16
1.3. Autonominės vaizdo stebėjimo sistemų sandara ir energijos poreikis.....	19
1.3.1. Mikrovaldiklis.....	19
1.3.2. Kameros modulis.....	20
1.3.3. Vaizdo įrašų valdymo programinė įranga.....	22
1.3.4. Programinės priemonės.....	23
1.3.5. Perspėjimai.....	25
1.4. Autonominių vaizdo stebėjimo sistemų palyginamoji analizė.....	25
1.5. Skyriaus išvados.....	27
2. AUTONOMINĖS VAIZDO STEBĖJIMO SISTEMOS SU OBJEKTŲ ATPAŽINIMU PROJEKTAVIMAS.....	29
2.1. Projekto įgyvendinimo užduočių planavimas.....	29
2.1.1. Sistemos funkciniai reikalavimai.....	30
2.1.2. Sistemos nefunkciniai reikalavimai.....	30
2.1.3. Sistemos panaudos atvejų diagrama.....	31
2.1.4. Sistemos sekų diagrama.....	31
2.2. Projektuojamos sistemos raiškūs paveikslėlis.....	33
2.2.1. Esamos sistemos raiškūs paveikslėlis.....	33
2.2.2. Siekiamos sistemos raiškūs paveikslėlis.....	33
2.3. Projekto realizacijai reikalingos aparatinės įrangos parinkimas.....	34
2.3.1. Mikrovaldiklio parinkimas.....	34
2.3.2. Kameros parinkimas.....	34
2.3.3. Pranešimų siuntimo aparatinės įrangos parinkimas.....	35

2.3.4.	<i>Akumulatoriaus parinkimas</i>	36
2.4.	Autonominės vaizdo stebėjimo sistemos su objektų atpažinimu architektūra.....	37
2.5.	Mikrovaldiklio ir atpažinimo sistemos konfigūracija.....	38
2.5.1.	<i>Mikrovaldiklio pradinė konfigūracija</i>	39
2.5.2.	<i>Vaizdo apdorojimo ir atpažinimo sistemos konfigūracija</i>	39
2.6.	Skyriaus išvados.....	40
3.	VAIZDO STEBĖJIMO SISTEMOS SU OBJEKTŲ ATPAŽINIMU REALIZAVIMAS.....	41
3.1.	Sistemos konfigūracija.....	41
3.2.	Sistemos autonomiškumas.....	47
3.2.1.	<i>Pirmasis sistemos veikimo bandymas</i>	48
3.2.2.	<i>Antrasis sistemos veikimo bandymas</i>	48
3.2.3.	<i>Trečiasis sistemos veikimo bandymas</i>	52
3.2.4.	<i>Ketvirtasis sistemos veikimo bandymas</i>	54
3.3.	Suprojektuotos sistemos aktualumas.....	56
3.4.	Skyriaus išvados.....	56
	IŠVADOS.....	58
	LITERATŪROS SĄRAŠAS.....	59

ILIISTRACIJŲ SĄRAŠAS

1.1 pav. Autonominė vaizdo kamera Eufy SoloCam C210 (Šaltinis: https://www.eufy.com/products/t8b00122?variant=42541391446202).....	22
1.2 pav. Autonominė vaizdo kamera EZVIZ CS-EB3 4G (Šaltinis: https://ezvizlife.lt/produktas/ezviz-cs-eb3-4g-lauko-akumuliatorine-kamera/).....	23
2.1 pav. Sistemos panaudos atvejų diagrama.....	27
2.2 pav. Sistemos sekos diagrama.....	28
2.3 pav. Esamos būsenos raiškūs paveikslėlis.....	29
2.4 pav. Siekiamos sistemos raiškūs paveikslėlis.....	30
2.5 pav. OV5647 kameros modulis (Šaltinis: https://www.anodas.lt/odseven-kamera-hd-ir-cut-ov5647-5mpx-dienos-nakties-ir-skirta-raspberry-pi-ir-moduliai?search=RPi%20%20kamera&page=3).....	31
2.6 pav. Raspberry Pi SIM800 GSM GPRS modulis.....	31
2.7 pav. Li-ion 18650 akumuliatoriai.....	32
2.8 pav. Ličio jonų ličio akumuliatoriaus įkrovimo modulis ir DC/DC įtampos keitiklis.....	32
2.9 pav. Sukonstruota autonominė vaizdo stebėjimo sistema.....	33
2.10 pav. Sistemos architektūros schema.....	34
3.1 pav. RPi operacinės įrangos diegimas į SD kortelę.....	37
3.2 pav. YOLOv5 reikalavimai.....	39
3.3 pav. Naršyklės lange matomas tiesioginės transliacijos vaizdas.....	40
3.4 pav. SIM kortelės konfigūracija.....	41
3.5 pav. AT pagrindinės komandos.....	42
3.6 pav. Komandinės eilutės atsakas.....	42
3.7 pav. MMS žinutės gavimas su aptiktu vaizdu.....	43
3.8 pav. Log'ai sistemos paleidimo metu.....	45
3.9 pav. Log'ai sistemos veikimo metu.....	45
3.10 pav. Prisijungimas prie RPi iš kito kompiuterio naudojant <i>Putty</i>	45
3.11 pav. Prisijungimas prie RPi iš kito kompiuterio.....	46
3.12 pav. Virtualaus ekrano prijungimas.....	46
3.13 pav. Vaizdo paleidimo kartu su akumuliatoriaus prijungimu komanda.....	47
3.14 pav. Testuojamas vaizdo paleidimas.....	47
3.15 pav. Sistemos miego režimo išjungimas.....	47
3.16 pav. Objektų žymėjimo gautose nuotraukose įdiegimas.....	49
3.17 pav. Originalus vaizdas bei apdorotas vaizdas su objekto žymėjimu.....	49
3.18 pav. Akumuliatoriaus tūrio mažėjimo laiko atžvilgiu grafikas.....	51

LENTELIŲ SĄRAŠAS

1.1 lentelė. Mikrokompiuterių palyginimas	20
1.2 lentelė. Kamerų palyginimas.....	21
1.3 lentelė. „Java“ ir „Python“ kriterijų palyginimas	24
1.4. lentelė. Autonominių vaizdo stebėjimo sistemų su objektų atpažinimu palyginimas.....	27
2.1. lentelė. Autonominės vaizdo stebėjimo sistemos su objektų atpažinimu projektavimo užduočių diagrama	29
3.1 lentelė. Pirmojo akumuliatorių tūrio vertės matavimo rezultatai	48
3.2 lentelė. Antrojo akumuliatorių tūrio vertės matavimo rezultatai	51
3.3 lentelė. Trečiojo akumuliatorių tūrio vertės matavimo rezultatai	54

SANTRUPOS

CCTV (angl. *Closed-Circuit Television*) – uždaros grandinės televizija;
DVRs (angl. *Digital Video Recorders*) – skaitmeninis vaizdo įrašymo įrenginys;
NVR (angl. *Network Video Recorders*) – tinklo vaizdo įrašymo įrenginiai;
RCNN (angl. *Regional Based Convolutional Neural Network*) – regiono pasiūlymu pagrįstas metodas;
DCNN (angl. *Deep Convolutional Neural Network*) – gilusis neuroninis tinklas;
CNN (angl. *Convolutional Neural Networks*) – konvoliucinis neuroninis tinklas;
VOS (angl. *Video Object Segmentation*) – vaizdo įrašų objektų segmentavimas;
MOTS (angl. *Multi-Object Tracking and Segmentation*) – kelių objektų sekimas ir segmentavimas;
ATMS (angl. *Automated Traffic Surveillance Systems*) – automatizuotos eismo stebėjimo sistemos;
AUV (angl. *Autonomous Underwater Vehicle*) – povandeninės autonominės transporto priemonės;
SMD (angl. *Singapore Maritime Dataset*) – Singapūro jūrų duomenų rinkinys;
SVM (angl. *Support vector machine*) – pagalbinio vektoriaus mašinos;
VOR (angl. *Video object referencing*) – vaizdo objektų nuoroda;
YOLO (angl. *You Only Look Once*) – pažiūrėti tik vieną kartą;
COCO (angl. *Common Objects in Context*) – įprasti objektai kontekste;
GMM (angl. *Gaussian Mixture Model*) – Gauso mišinio modelis;
RAM (angl. *Random-access Memory*) – operatyvioji atmintis;
IDE (angl. *Integrated Development Environment*) – integruotų programų kūrimo aplinka;
SBC (angl. *Single Board Computer*) – vieno lusto sistema arba vienos plokštės kompiuteriai;
MSE (angl. *Mean Squared Error*) – vidutinio kvadratinio nuokrypio algoritmas;
UML (angl. *Unified Modeling Language*) – vieninga modeliavimo kalba;
DI (angl. *Artificial intelligence*) – dirbtinis intelektas.

IVADAS

Vaizdo stebėjimo sistemos nuo senų laikų naudojamos ne tik saugumo ir stebėjimo tikslais, bet ir siekiant aptikti objektą, įrašyti užfiksuotą vaizdą ir analizuoti aplinką. Laikui bėgant vaizdo stebėjimo sistemos tobulėjo, atsirado didelės galios, gebančios veikti nuotoliniu būdu, atpažinti objektus bei suteikiančios puikios raiškos vaizdą kameros.

Per pastaruosius metus pasaulis stebėjo pokyčius įvairiuose socialiniuose ir ekonominiuose sektoriuose, dėl kurių gyvenimas tapo sudėtingesnis įvairiais aspektais, įskaitant asmenų saugumą ir komfortą. Taigi, poreikis būti apsaugotam ir stebimam tapo būtinybe. Todėl viešose ir privačiose erdvėse įrengtos stebėjimo kameros yra tinkamas sprendimas saugumui užtikrinti (Elharrouss et al., 2021).

Vaizdo stebėjimas – tai viešųjų ar privačių erdvių stebėjimas nuotoliniu būdu naudojant vaizdo stebėjimo sistemas. Šiomis kameromis užfiksuoti vaizdai paprastai perduodami į valdymo centrą, juos analizuoja ir iškart peržiūri operatoriai arba jie yra įrašomi, o vėliau analizuojami. Valdymo centre turėtų būti galimybė atpažinti objektą, pvz., gyvūną, žmogų ar kitą judantį objektą, be to, sistema turėtų informuoti operatorių, kad patikrintų filmuotą medžiagą (Tsakanikas & Dagiuklas, 2018).

Vaizdo stebėjimo sistemos turi daugybę pritaikymo sričių. Jos dažniausiai naudojamos asmenų saugumo tikslais tokiose srityse kaip: namų saugumas, biurai, bankai ir viešosios erdvės, siekiant stebėti ir atgrasyti galimas grėsmes. Jos taip pat naudojamos žmonių ir krovinių transportavimo sistemose, tokiose kaip oro uostai, traukinių stotys ir greitkeliai, siekiant stebėti ir užtikrinti visuomenės saugumą. Be to, vaizdo stebėjimo sistemos randa savo pritaikymą mažmeninėse prekybos parduotuvėse siekiant užkirsti kelią vagystėms ir stebėti klientų elgesį. Jos taip pat gali būti naudojamos pramoninėse aplinkose stebint gamybos procesus ir užtikrinant darbo vietos saugumą.

Pavyzdžiui, viena iš sričių, kurioje vaizdo stebėjimas yra labai svarbi priemonė saugumo ir ekonomiško sumetimais, yra statybų aikštelės. Į statybvietais dažnai patenka nusikaltėliai, vandalai ir kiti leidimo neturintys asmenys, ieškantys galimybės pavogti įrangą, įrankius ir kitus vertingus daiktus. Daugumoje statybviečių saugomas šimtatūkstantinės vertės turtas ir medžiagos, kartais su mažai arba visai be jokių saugumo sistemų. Dėl vagysčių statybos įmonės gali patirti didelių nuostolių ir dėl to gali vėluoti projektai, taip pat tai įtakoja planavimo ir pirkimo išlaidas. Siekiamos kovoti su šia problema, daugelis statybų įmonių kreipiasi į mobiliųjų stebėjimo įrenginių tiekėjus, kad apsaugotų savo turtą (Zhang et al., 2020).

Šiame magistro darbe bus kuriama autonominė vaizdo stebėjimo sistema skirta vaizdams atpažinti paremta mikrovaldikliu, siekiant užtikrinti vaizdo stebėjimą vietose, kur neįmanomas kameros prijungimas prie maitinimo šaltinio, tačiau vaizdo stebėjimas ir atpažinimas būtinas siekiant užtikrinti saugumą.

Tyrimo aktualumas – poreikis spręsti problemas, susijusias su konkrečių naudotojų dominančių objektų stebėjimu bei būtent jų fiksavimu vaizde sudėtingomis sąlygomis, kai tradicinės technologijos neužtikrina patikimo ryšio arba neveikia be tiesioginio maitinimo bei jų funkcionalumo ribos kelia didelius iššūkius.

Tyrimo metodika – analitinėje darbo dalyje buvo naudojama su tema susijusių mokslinių straipsnių analizė ir jų palyginimas.

Tyrimo problema – egzistuojantys sprendimai nepadengia visų reikiamų autonominei vaizdo stebėjimo sistemai keliamų reikalavimų, per siauras objektų atpažinimo spektras bei ribotas ryšio pasiekiamumas neužtikrina reikiamo rezultato, o siauras sistemų pasirinkimas neužtikrina visų poreikių padengimo.

Tyrimo objektas – autonominės vaizdo stebėjimo sistemos su objektu atpažinimu paremtos mikrovaldikliu.

Tyrimo tikslas – sukurti autonomine mikrovaldiklių paremtą vaizdo stebėjimo sistemą, galinčią užtikrinti objektų atpažinimą.

Uždaviniai tikslui pasiekti:

1. Atlikti probleminės srities analizę, išanalizuoti literatūrą apie esamas sistemas ir mokslinius darbus, susijusius su vaizdo stebėjimo sistemomis su objektu atpažinimu.
2. Parengti autonominės vaizdo stebėjimo sistemos su objektu atpažinimu specifikaciją, ją suprojektuoti.
3. Įgyvendinti suprojektuotą sistemą, atlikti sistemos eksperimentus ir įvertinti gautus rezultatus.

Tyrimo metodai – optimizuoti įrangos ir procesų energijos suvartojimą autonominėje vaizdo stebėjimo sistemoje su objektu atpažinimu kartotiniu metodu nuolat tobulinant ir optimizuojant sistemą, remiantis grįžtamuoju ryšiu ir realaus pasaulio testavimu. Derinant eksperimentinį vertinimą ir kartotinį tobulinimą, galima užtikrinti išsamų sistemos galimybių įvertinimą ir laikui bėgant tobulinti.

Darbo aprobacija:

Svarbiausieji darbo rezultatai pagal magistro tezių tematiką pateikti pranešime, skaitytame:

1. 2025 m. balandžio 16 d., „Vilnius Tech“ vykusioje 28-ojoje Lietuvos jaunųjų mokslininkų konferencijos „Mokslas – Lietuvos ateitis“ teminėje konferencijoje „Informacinių technologijų sauga ir informacinės sistemos“.

1. AUTONOMINIŲ VAIZDO STEBĖJIMO SISTEMŲ ANALIZĖ

Šioje dalyje atliekama literatūros šaltinių analizė, kurios tikslas išanalizuoti autonominių vaizdo stebėjimo sistemų paklausą, apžvelgti skirtingų autorių požiūrius į vaizdo stebėjimo sistemas, įvertinti jų taikomas metodikas ir algoritmus.

1.1. Autonominių vaizdo stebėjimo sistemų problematika

Per pastaruosius dešimtmečius, dėl pasaulyje vykusių permainų įvairiuose socialiniuose ir ekonominiuose sektoriuose, žmonių gyvenimas tapo labiau sudėtingas ir reikalaujantis didesnės apsaugos. Taigi, poreikis būti saugiam ir stebimam tapo savotiška būtinybė. Paprastai aplinkos ir žmonių stebėjimui būdavo pasitelkiama žmogiškaisiais ištekliais. Patalpų ir kitų vietovių stebėjimas 24 valandas per parą yra brangus ir ekonomiškai neefektyvus sprendimas, dėl to atsirado poreikis sukurti autonominę vaizdo stebėjimo sistemą, kuri nereikalautų pastovaus monitoringo. Atsiradusios vaizdo įrašymo sistemos buvo labiau ekonomiškas variantas, tačiau dėl pavojaus netekti informacijos kameros sugadinimo metu didžiojoje dalyje atvejų jų atsisakyta.

Autonominės vaizdo stebėjimo sistemos su objektu atpažinimu yra labiausiai tinkamos sistemos siekiant ekonomiškai ir efektyviai naudoti tiek žmogiškuosius išteklius, tiek informacijos kiekį. Pagrindinis autonominės sistemos vaidmuo yra padėti operatoriui atlikti skirtingas stebėjimo užduotys vienu metu. Šios užduotys gali būti susijusios su įvairiomis programomis, tokiais kaip eismo valdymas, nelaimingų atsitikimų numatymas, nusikaltimų prevencija, judesio aptikimas ir aplinkos apsauga.

Šiuolaikinė visuomenė patiria vis didesnę susidomėjimą sauga ir saugumu, todėl sukaupia daug patirties, susijusios su vaizdo stebėjimo sistemų diegimu. Integruoti į *Smart Spaces* šie vaizdo jutiklių tinklai suteikia intelektualiai sistemai galimybę stebėti aplinką, o kartu – aptikti ir nustatyti įvairias neįprastas situacijas, kurios gali kilti. Šiuo metu yra daugybė vaizdo stebėjimo sistemų, kurios naudojamos įvairiose srityse, tokiose kaip įsibrovimų aptikimas ar eismo stebėjimas. Tačiau autonominis įspėjimų ir neįprastų situacijų aptikimas vis dar yra primityviame etape (Calavia et al., 2012).

Tam tikrose gyvenimo srityse, poreikis užtikrinti vaizdo stebėjimą yra būtinas, pavyzdžiui bankuose, viešosiose vietose, oro uoste ar parduotuvėse, kad išvengtų įvairių pavojų tokių kaip: vagystės, teroro išpuoliai, smurtas ir nelaimingi atsitikimai. Dėl to viešose ir privačiose erdvėse įrengtos stebėjimo kameros yra tinkamas sprendimas saugumui užtikrinti. Tačiau, pasitaiko atvejų, kai yra sudėtinga arba net neįmanoma prijungti kamera prie maitinimo šaltinio ir vaizdo stebėjimas įprastu būdu tampa neįmanomas. Tokiais atvejais geriausia rinktis autonomines vaizdo stebėjimo sistemas.

Projektuojant autonomine vaizdo stebėjimo sistemą su objektų atpažinimu susiduriama su problemomis, kurios riboja sistemos efektyvumą, patikimumą bei platų taikymą. Toliau darbe aptarsiu pagrindines problemines vietas, kurios gali sukelti sunkumu sistemos realizavime:

1.1.1. Techniniai projekto įgyvendinimo iššūkiai

Vaizdo stebėjimo sistemos, ypač tos, kurios naudojamos lauko sąlygomis susiduria su objektų atpažinimo tikslumo problema. Prastos oro sąlygos tokios kaip: šviesos pokyčiai (šešėliai), lietus, rūkas ar sniegas gadina vaizdo kokybę bei reikalauja galingesnio objektų atpažinimo modelio. Silpnasis objektų atpažinimo modelis taip pat gali įtakoti paklaidą, kuomet užfiksuotame vaizde nėra mus dominančio objekto tačiau pranešimas yra gaunamas, ši problema taip pat pasireiškia dabartinėse sistemose, kur nėra įdiegtas objekto atpažinimas o sistema veikia su jutikliais. Prie techninio iššūkio taip pat priskiriamas daug resursų naudojantis realaus laiko apdorojimas, kuris lėtina reakciją bei įtakoja vaizdo delsimą, dėl ko galima nespėti laiku sureaguoti į iškilusią grėsmę.

Autonominėms vaizdo stebėjimo sistemoms su objektų atpažinimu labai svarbus aspektas yra energijos suvartojimas. Problema projekto įgyvendinimui sudaro didelės dirbtinio intelekto (DI, angl. *Artificial intelligence*) ir vaizdo apdorojimo sąnaudos, objektų atpažinimui yra sunaudojama žymiai daugiau energijos nei jutikliais paremtos sistemos, dėl ko sumažėja kameros veikimo laikas.

1.1.2. Ryšio ir duomenų valdymo problemos

Dar viena probleminė autonominių vaizdo stebėjimo sistemų su objektų atpažinimu vieta yra duomenų perdavimas. Silpnas ryšys gali sutrikdyti išėjimų siuntimą o didelis duomenų kiekis stipriai užkrauna tinklą bei sunaudoja didelį kiekį energijos, kas trumpina sistemos veikimo laiką.

1.2. Autonominių vaizdo stebėjimo sistemų analizė

Šiuo metu rinkoje vyrauja aibė įvairių baterijomis maitinamų autonominių vaizdo stebėjimo sistemų, tačiau mokslinių straipsnių atlikta nėra. Nors autonominės vaizdo stebėjimo sistemos, maitinamos baterijomis ir atsinaujinančios energijos šaltiniais (pvz., saulės panelėmis), sparčiai plėtojamos, mokslinėje literatūroje išlieka reikšmingų spragų – ypač kalbant apie baterijų veikimo laiko optimizavimą. Daugelis tyrimų (pvz., *IEEE Sensors Journal*, *ACM IoT* publikacijos) fokusuojasi į sistemų architektūrą, vaizdo analizės algoritmus, tačiau nepakankamai nagrinėja, kaip išplėsti baterijų veikimą esant realioms sąlygoms. Pavyzdžiui, nors DI pagrindu sukurti judesio aptikimo metodai sumažina energijos suvartojimą, reti straipsniai analizuoja, kaip šie metodai veikia ilgalaikėje perspektyvoje, kai baterijos talpa nyksta.

Be to, trūksta išsamių tyrimų, lyginančių skirtingų baterijų technologijų (*Li-ion*, *LiFePO₄*, superkondensatorių) efektyvumą ilgalaikėse autonomiškose sistemose. Kol kas dauguma komercinių sprendimų (pvz., *Hikvision* ar *Reolink* kameros) remiasi bendromis energijos vadybos praktikomis, o ne moksliškai pagrįstomis metodikomis. Todėl ši sritis lieka perspektyvi tolimesniems tyrimams, ypač siekiant sukurti ištis ilgaamžes sistemas.

1.2.1. Autonominių vaizdo stebėjimų sistemų kritinės taikymo sritys

Viena plačiausiai autonominių vaizdo stebėjimų sistemų panaudojimo sričių yra medžioklė, aplinkos ir sienos apsauga. Autonominės sistemos su *Li-ion* baterijomis ir DI atpažinimu yra plačiai panaudojamos apsaugoti nykstančias rūšis. Miškuose, parkuose bei atokiose gamtos vietose, kur nėra galimybės įjungti kamera prie tiesioginio elektros šaltinio, bei kur siekiant užkirsti kelią nelegaliam brakonieriavimui arba norint stebėti retų, nykstančių gyvūnų buveines tokios kameros gali operatyviai DI pagalba atpažinti bei pranešti apie galimą grėsmę arba tiriamo objekto judėjimą.

Kameros su judesio ir objektų atpažinimu yra tinkamos taip pat siekiant nustatyti nelegalius atliekų išmetimus miškų ir saugomų teritorijų pakraščiuose, atokiose vietose, kur pastovus maitinimas dažnai neįmanomas dėl infrastruktūros trūkumo. Taip pat pasienio zonose, stabdant nelegalų pabėgėlių patekimą į šalį.

Dar viena, gana plačiai naudojama pritaikymo sritys yra statybų objektų apsauga. Vykdamas statybos darbus teritorijoje ne visuomet yra galimybė prijungti kamera prie tiesioginio maitinimo šaltinio, nes iki statybvietsės nėra atvestas elektros tinklas, tai apsunkina tokių objektų apsaugą. Tačiau su autonominės vaizdo stebėjimo sistemos pagalba galima užfiksuoti vandalizmo arba vagysčių atvejus net atokiuose objektuose be elektros tiekimo.

1.2.1. Objektų atpažinimo technologijų analizė

Šiuo metu objektų atpažinimo sistemos yra plačiai ištirtos ir efektyviai naudojamos įvairiuose srityse, pavyzdžiui: transporto sistemose, vyriausybinių įstaigose, pramoninėje aplinkoje, bankuose, prekybos centruose, viešosiose pastatuose ir kt. Dauguma viešose ir privačiose vietose veikiančių stebėjimo sistemų priklauso nuo žmogaus operatoriaus stebėtojo, kuris aptinka bet kokią įtartina veiklą stebėdamas vaizdą (Sadollah & Sinha, 2020).

Statistinės analizės panaudojimas, vaizdinei informacijai apdoroti, yra aptartas Rezaei straipsnyje, jame analizuojant eismo sąlygas, statistinė eismo srauto ir vartotojų sąveikos analizė buvo analizuojama naudojant atstumo stebėjimą (Rezaei et al., 2023).

Kaip nurodyta Sharma straipsnyje, gilaus mokymosi modeliai mokosi išskirti ir analizuoti požymius, kad būtų galima tiksliai prognozuoti objektų buvimą ir tapatybę vaizdiniuose duomenyse. Pasak autorių, objektų aptikimas apima užduotį nustatyti objektų vietą, ir tada klasifikuoti juos ribojančiose ribose, kuriomis vaizduojami objektai yra pažymėti (Sharma et al., 2023).

Vaizdo įrašų pagrįstas realaus laiko objektų identifikavimas atliekant į kompiuterinę viziją orientuotus tyrimus pastaruoju metu išaugo dėl plačiai taikomo regiono pasiūlymu pagrįsto metodo (angl. *Regional Based Convolutional Neural Network*, RCNN). Šiuo požiūriu konvoliucinis neuroninis tinklas (angl. *Convolutional Neural Networks*, CNN) ir gilusis neuroninis tinklas (angl. *Deep Neural Networks*, DNN) yra baziniai modeliai. FRCNN (greitesnis CNN) ir CNN modelius tyrėjai pripažino dėl savo greičio ir tikslumo (Maheswari et al., 2022).

DNN sistemos yra tinklų klasė, kurioje kiekvienas sluoksnis gali atlikti sudėtingas operacijas, tokias kaip vaizdavimas ir abstrakcija, kad suprastų teksto, balso ir vaizdo duomenis (Sharma et al., 2023). Geriausias šios sistemos panaudojimas pateiktas Rezaei straipsnyje. Šiame straipsnyje jie teigia, kad per pastarąjį dešimtmetį DNN diegimas padarė didelę pažangą objektų aptikimo srityje. Jau nekalbant apie tokių modelių efektyvumą ir tikslumą sudėtingose, dinamiškose, triukšmingose ir perpildytose eismo aplinkose, jose rezultatai nėra tokie geri, kaip objektų aptikimas kontroliuojamuose patalpų aplinkose. Jie plėtoja DNN pagrįstą modelį, atsižvelgdami į mokymo duomenis, mokymąsi ir hiperparametrus, bei modelio vertinimą pagal testavimo duomenis (Rezaei et al., 2023).

Remiantis prieš tai paminėtais straipsniais, dažniausiai naudojamas metodas vaizdinei informacijai apdoroti yra CNN. Šis galingas modelis, gali automatiškai išmokti ir išgauti vaizdų funkcijas, todėl galima tiksliai ir efektyviai aptikti ir atpažinti objektus. Tas pats metodas naudojamas ir Maheswari straipsnyje. Šiame tyrime naudojamas metodas apima DCNN, kuris yra mašininio mokymosi algoritmo tipas, kad būtų galima analizuoti ir klasifikuoti stebėjimo kamerų užfiksuotus vaizdus ar vaizdo įrašus (Maheswari et al., 2022). Kompiuterinio matymo algoritmų ir CNN metodo taikymas bepiločių orlaivių vaizdo įrašų vaizdiniais duomenims analizuoti geriausiai aptariamas Schiano ir Fortes straipsniuose. Abiem atvejais sistema nustato konkrečius dominančius objektus naudojant iš dronų gautą vaizdinę medžiagą pvz. suskaičiuoja tuščias stovėjimo vietas automobilių stovėjimo aikštelėje arba aptinka ir seka balandžius ant pastatų (Schiano et al., 2022) (Fortes et al., 2021).

Park'as savo straipsnyje apie objektų atpažinimą ir sekimą judančiuose vaizdo įrašuose, skirtuose jūriniams autonominiams paviršiniams laivams, analizuoja regresija/klasifikavimu pagrįstą metodą – YOLO (angl. *You Only Look Once*) (H. Park et al., 2022). Cintas straipsnyje trumpai paaiškina YOLO modelį. Jų žodžiais tariant, ši sistema yra konvoliucinė neuroninio tinklo architektūra, siūloma objektų aptikimui realiuoju

laiku didelės spartos vaizdo įrašuose. YOLO tinklas aptikimo problemą paverčia regresijos problema, kad vaizdas naudojamas kaip įvesties tinklas tik vieną kartą, o ne atskirai apdorojamas kiekvienai klasei. Todėl vaizdo kadruose galima nustatyti aptiktų objektų vietą žymiai greičiau nei tradiciniais objektų aptikimo algoritmais (Çintaş et al., 2020).

Mahendru ir Dubey straipsnyje moksliniai tyrimai apima YOLO ir YOLO_v3 algoritmų įgyvendinimą ir garso grįžtamojo ryšio integravimą, kad būtų galima gauti informaciją apie aptiktus objektus realiuoju laiku (Mahendru & Dubey, 2021). Šio tipo tyrimai taip pat pateikiami Xiang straipsnyje. Jis naudoja YOLO_v2 modelį. Nors gilaus mokymosi algoritmai duoda tikslius objektų aptikimo ir sekimo problemų rezultatus, jiems reikalingos didelio našumo skaičiavimo sistemos (Xiang et al., 2021).

Be to, nemažai autorių taikė metodą, apimančią YOLO tinklo naudojimą ir kumuliacinio vidurkio metodo taikymą objekto atpažinimo rezultatams pagerinti. Pavyzdžiui Park'as ir Kang'as savo straipsnyje naudojo YOLO tinklą žuvims atpažinti povandeniniuose vaizdo įrašuose (Park & Kang, 2020). Çintas naudojo šį metodą, kuris apima kompiuterinio matymo metodų naudojimą, kad būtų galima sekti tikslinių povandeninių autonominių transporto priemonių (AUV – angl. *Autonomous Underwater Vehicle*) judėjimą (Çintaş et al., 2020). Parkas taip pat sutelkė dėmesį į algoritmų ir metodų kūrimą, leidžiančius AUV tiksliai aptikti ir sekti objektus realiuoju laiku iš vaizdo srautų (H. Park et al., 2022). Autonominių vaizdo stebėjimo sistemų analizei Xiang ir Sabater straipsniuose taip pat naudojami YOLO algoritmai (Xiang et al., 2021) (Sabater et al., 2020). Šis metodas taip pat Mahendru ir Dubey straipsnyje apima YOLO ir YOLO_v3 algoritmų įgyvendinimą ir garso grįžtamojo ryšio integravimą, kad būtų galima gauti informaciją apie aptiktus objektus realiuoju laiku (Mahendru & Dubey, 2021).

Szymak ir Gasiorowski straipsnyje taip pat plačiai aptariamos povandeninės vaizdo stebėjimo sistemos su objektų atpažinimu. Čia autoriai naudoja *AlexNet* modelį, kuris buvo apmokytas didelio masto duomenų rinkiniui, ir tiksliai suderina jį su konkrečiu povandeninių vaizdų duomenų rinkiniu (Szymak & Gasiorowski, 2020). Kita vertus, Kannan savo straipsnyje Gauso modelį derina su formų derinimo technikomis, vadovaujamas evoliuciniu povandeninių vaizdų apdorojimo ir objektų atpažinimo algoritmu (Kannan, 2020).

Apibendrinant, dauguma autorių, nepaisant skirtingų vaizdo stebėjimo ir objektų atpažinimo metodų, turėjo labai panašius numatymui naudojamus požymius, tokius kaip gilaus mokymosi modelio išmoktos vizualinės savybės, įskaitant spalvą, tekstūrą ir formą, straipsniuose pasiekti daugiausia tie patys rezultatai teikiant metodus, didinančius vaizdo objektų aptikimo sistemų patikimumą ir efektyvumą.

1.3. Autonominės vaizdo stebėjimo sistemų sandara ir energijos poreikis

Per pastarąjį dešimtmetį vaizdo stebėjimo sistemos virto nuo paprastų vaizdo gavimo ir rodymo sistemų iki išmaniųjų (pusiau) autonominių sistemų, galinčių atlikti sudėtingas procedūras. Šiais laikais vaizdo stebėjimo sistema gali integruoti įvairius sudėtingiausius vaizdo ir vaizdo analizės algoritmus iš tyrimų sričių, tokių kaip: klasifikavimas (pvz., neuroniniai tinklai ar stochastiniai modeliai), modelių atpažinimas, sprendimų priėmimas, vaizdo gerinimas ir keletas kitų. Taigi, modernią stebėjimo sistemą sudaro vaizdo įrašymo ir vaizdo gavimo įrenginiai, duomenų apdorojimo-analizės moduliai ir saugojimo blokai, komponentai, kurie yra labai svarbūs sistemos darbo eigai (Tsakanikas & Dagiuklas, 2018).

Vaizdo stebėjimas yra kompleksinė sistemos užduotis, kuriai reikalingos gerai apgalvotos dalys. Pagrindinė autonominės vaizdo stebėjimo sistemos paremtos mikrovaldikliu sandara susideda iš pačio mikrovaldiklio, kuris pajėgus atlikti reikiamus veiksmus, kameros, vaizdo įrašų valdymo programinės įrangos, programinių priemonių ir perspėjimo sistemos. Visos šios dalys yra išsamiau išdėstytos tolimesniuose skyreliuose.

1.3.1. Mikrovaldiklis

Yra daug mikrovaldiklių, kurių kiekvienas turi skirtingas specifikacijas, todėl jie yra optimalūs naudoti konkrečiose srityse. Praktinis panaudojimas, galingumas ir prieinamumas rinkoje yra pagrindiniai pasirinkimo veiksniai, kurie įtakoja mikrovaldiklio pasirinkimą šiam projektui. 1.1 lentelėje yra palyginti penki šiuo metu rinkoje esantys populiariausi mikrovaldikliai.

Pagal lentelėje pateiktus rezultatus bei atliktą išsamią analizę pasirinkti trys mikrokompiuteriai kurie išsiskiria energijos efektyvumu, DI našumu bei suderinamumu su specifiniais projekto reikalavimais.

Geriausias pasirinkimas galingumo atžvilgiu yra *NVIDIA Jetson Nano* mikrokompiuteris. Jis palaiko *128 CUDA* branduolių bei tinkamas sudėtingiems objektų atpažinimo uždaviniams, tačiau suvartoja pakankamai daug energijos dėl ko reikalauja didesnio akumulatoriaus tūrio.

Ekonomiškiausias variantas iš aukščiau pateiktų yra *BeagleBone Black*, jis turi minimalų energijos suvartojimą (~2-4 W) tačiau jis nėra pritaikytas objektų atpažinimui bei turi silpną DI našumą.

Šiam projektui geriausiai pritaikomas bei geriausias ekonomiško ir našumo atžvilgiu yra *Raspberry Pi 4* (RPI) mikrokompiuteris. Jis turi plačią DI palaikymo plėtrą, vidutiniškai sunaudoja ~2-6 W energijos, palaiko daug bibliotekų (tokių kaip: *OpenCV*, *TensorFlow Lite* ir kt.)

1.1 lentelė. Mikrokompiuterių palyginimas

	Banana Pi M5	Raspberry Pi 4	ODROID N2	NVIDIA Jetson Nano	BeagleBone Black
Šaltinis	https://docs.banana-pi.org/en/BPI-M5/BananaPi_BPI-M5	https://kamami.pl/15505-raspberry-pi-4-model-b	https://www.hardkernel.com/shop/odroid-n2-with-4gbyte-ram/	https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano/	https://beagleboard.org/black
Pranašumai	Integruota eMMC atmintis, Gigabit Ethernet tinklo lizdas, WiFi, Bluetooth 5.0	Gigabit Ethernet tinklo lizdas, WiFi, Bluetooth 5.0, 2x microHDMI lizdai su 4K palaikymu	Gigabit Ethernet tinklo lizdas, WiFi, HDMI IR imtuvas	Galingas GPU, puikus DI projektams	Integruota eMMC atmintis, platus I/O galimybių spektras
Procesorius	Amlogic S905X3, keturių branduolių Cortex-A55	Broadcom BCM2711, Quad-core Cortex-A72 (ARM v8) 64-bitų SoC @1.5GHz	Amlogic S922X, Quad-core Cortex-A73 @1.8GHz, Dual-core Cortex-A53 @1.9GHz	Quad-core ARM Cortex-A57	AM335x 1GHz ARM® Cortex-A8
Operacinė sistema	Android, Linux	Windows 10, Mac, Linux (Raspbian, Ubuntu, Kali ir kt.)	Linux (Ubuntu, KODI), Android 9.0	Linux (Ubuntu, NVIDIA JetPack SDK)	Linux (Debian, Ångström)
Operatyvioji atmintis	4GB LPDDR4	2/4/8GB LPDDR4	2/4GB DDR3	4GB LPDDR4	512MB DDR3
USB jungtys	4x USB A3.0, 1x USB A2.0	2x USB A3.0, 2x USB A2.0	4x USB A3.0, 1x USB A2.0	4x USB A3.0, 1x USB 2.0 Micro-B	1x USB 2.0
Energijos suvartojimas	~5-7 W	~2.5-6 W	~6-8 W	~5-10 W	~2-4 W

1.3.2. Kameros modulis

Kameros tai pagrindiniai sistemos komponentai, fiksuojantys stebimos srities vaizdo įrašus. Jie gali būti fiksuoti arba reguliuojami ir gali turėti tokias funkcijas kaip: pasisukimas, pakreipimas ir mastelio keitimas.

Renkantis kamerą autonominei vaizdo stebėjimo sistemai su objektų atpažinimu naudojant mikrokompiuterį, labiausiai atkreipiamas dėmesys į energijos efektyvumą, vaizdo kokybę, bei kadrų per sekundę kiekį, taip pat svarbus aspektas yra lengvas naudojimas bei funkcionalumas.

Žemiau esančioje 1.2 lentelėje yra rekomenduojamų kamerų skirtų mikrokompiuterių projektams palyginimas:

1.2 lentelė. Kamerų palyginimas

Kamera	Raspberry Pi Camera Module V2	Raspberry Pi High Quality Camera (HQ)	Arducam IMX219 Wide Angle Module	Arducam OV5647 Camera Module	NoIR Camera Module V2
Šaltinis	https://www.anodas.lt/raspberry-pi-kamera-v2-8mp	https://lt.farnell.com/raspberry-pi/rpi-hq-camera/rpi-high-quality-camera-12-3-mp/dp/3381605?&CMP=KNC-GLT-GEN-SKU-MDC-Others	https://www.anodas.lt/en/arducam-imx219-8mp-camera-module-for-raspberry-v2-and-nvidia-jetson-nano-noir-arducam-b0188?gad_source=1&gclid	https://www.anodas.lt/en/arducam-5mp-ov5647-camera-board-w-cs-mount-lens-fully-compatible-with-raspberry-pi?gad_source=1&gclid	https://www.lemona.lt/kamera-8mp-kompiuteriui-raspberry-pi-noir.html?utm_source=google&utm_medium
Raiška	8MP (Sony IMX219)	12.3MP (Sony IMX477)	8MP (Sony IMX219)	5MP (OmniVision OV5647)	8MP (Sony IMX219)
Kadrai Per Sekundę (FPS)	1080p30, 720p60	Aukšti kadrai dažniai	1080p60, 720p180	Standartinis	1080p30, 720p60
Regos Kampas	Standartinis	Konfigūruojamas	Iki 160°	Standartinis	Standartinis
Optiniai elementai	Fiksuotas objektyvas	Keičiami objektyvai (CS ir C-mount)	Fiksuotas objektyvas	Fiksuotas objektyvas	Fiksuotas objektyvas
Privalumai	Gera vaizdo kokybė, plačiai naudojama, patikima integruotas IR	Aukšta vaizdo kokybė, keičiami objektyvai	Platus regos kampas, puiki kokybė	Ekonomiškas pasirinkimas, tinkamas pradiniam projektams	Be IR filtro, tinkamas naktiniam matymui
Energijos suvartojimas	~0,5-1,5 W	~1,5-3 W	~0,7-2 W	~0,5-1,2 W	~0,5-1,5 W

Pagal lentelę matyti, kad renkantis kamera šiam projektui svarbu atsižvelgti į kelis pagrindinius kriterijus:

- Vaizdo kokybei labai svarbu yra raiška, t. y. kuo didesnis pikselių skaičius, tuo geresnė vaizdo kokybė bei kadrai per sekundę (FPS), ypač jeigu reikia fiksuoti greitai judančius objektus, aukštesnis FPS yra būtinas.
- Objektyvo kokybė gali užtikrinti geresnį vaizdo aiškumą su mažiau iškraipymų, o platus regos kampas leidžia apimti didesnę plotą, tačiau gali sumažinti objekto detales.
- Kameros šviesos jautrumas užtikrina aiškesnį vaizdą prasto apšvietimo sąlygomis bei naktinio matymo galimybe padeda aiškiai fiksuoti vaizdus tamsoje.
- Svarbus kriterijus yra kameros suderinamumas su mikrokompiuteriu bei bibliotekų palaikymas.

- Kameros dydis, montavimo ant įvairių paviršių galimybes bei kameros kaina taip pat gana svarbus aspektas kameros parinkime.

Apibendrinus ir palyginus pateiktas populiariausias šiuo metu rinkoje esančias kameras, jos tarpusavyje yra labai panašios, tačiau pagal šiam projektui taikomus reikalavimus, kad sunaudotų nedaug energijos bei tiktų testuojamai sistemai geriausias pasirinkimas yra *Arducam OV5647 Camera Module* – ji yra ekonomiška ir paprasta kamera turinti ribotą funkcionalumą tačiau tinkama mažo biudžeto projektams kuriems nereikia didelės kokybės. *Raspberry Pi Camera Module V2* turi gera raišką bei nedidelį energijos suvartojimą tačiau neturi naktinio matymo funkcijos, kuri šiuo atveju būtų reikalingas. *Raspberry Pi HQ* kamera yra galinga ir teikia aukštą vaizdo kokybę, tačiau turi kelis niuansus, kaip aukšta kaina ir atskiri objektyvai jei suvartoja reikšmingai daugiau energijos. *Arducam IMX219 Wide Angle Module* teikia gera vaizdo kokybę bei tinka didelių teritorijų stebėjimui dėka plataus kampo. *NoIR Camera Module V2* kamera yra be IR filtro, todėl puikiai tinka naktiniam matymui, tačiau labiausiai pritaikomas esant blogam apšvietimui arba vaizdo stebėjimui tamsoje.

1.3.3. Vaizdo įrašų valdymo programinė įranga

Stebėjimas ir atkūrimas tai sistemos funkcija, kuri leidžia stebėti vaizdą realiuoju laiku. Ji taip pat suteikia galimybę peržiūrėti ir atkurti įrašytą filmuotą medžiagą, leidžiančią vartotojams analizuoti įvykius.

Vaizdo valdymo sistema (angl. *Video Surveillance Management Systems*, VMS) organizuoja stebėjimo darbo eigą integruodama su kameromis, koduotuvais, įrašymo sistemomis, pagrindine saugojimo infrastruktūra, klientų darbo stotimis, tinklų sąsajos sistemomis ir analizės programine įranga, daugiausia suteikdama vieną vaizdo stebėjimo infrastruktūros valdymo sąsają.

Pagrindinės VMS funkcijos:

1. *Tiesioginis vaizdo stebėjimas*: VMS sistemos leidžia apsaugos darbuotojams žiūrėti tiesioginius vaizdo įrašus iš stebėjimo kamerų realiuoju laiku, vietoje arba nuotoliniu būdu.
2. *Vaizdo įrašymas ir saugojimas*: VMS sistemos leidžia įrašyti vaizdo įrašus iš kelių kamerų, o įrašytas vaizdo įrašas paprastai saugomas serveryje arba tinklo saugojimo įrenginyje. VMS sistema taip pat suteikia galimybę ieškoti ir gauti įrašytą vaizdo įrašą.
3. *Vaizdo įrašų analizė*: VMS sistemos gali įtraukti vaizdo analizę, kad automatiškai aptiktų ir įspėtų apsaugos darbuotojus apie įtartina ar neįprastą elgesį.
4. *Integravimas su kitomis apsaugos sistemomis*: VMS sistemas galima integruoti su kitomis apsaugos sistemomis, tokiomis kaip prieigos kontrolės ir signalizacijos sistemos, todėl galima naudoti centralizuotą saugumo valdymo platformą.

5. *Vartotojų valdymas*: VMS sistemos leidžia valdyti vartotojus, todėl administratoriai gali priskirti skirtingus prieigos lygius skirtingiems vartotojams.

Apskritai VMS yra visapusiškas vaizdo stebėjimo ir saugumo valdymo sprendimas, leidžiantis organizacijoms efektyviai stebėti savo patalpas, turtą ir žmones realiuoju laiku (Isarsoft, 2024).

Vaizdo įrašų valdymo programinė įranga gali žymiai įtakoti autonominės vaizdo stebėjimo sistemos energijos suvartojimą, dėl to labai svarbu pasirinkti tinkamus parametrus, kad energijos suvartojimas atitiktų projektui reikalingas vertes.

Labiausiai energijos suvartojimą įtakoja vaizdo raiška, yra numatyta, kad 1080p suvartoja dvigubai daugiau energijos nei 720p. Įrašymo režimas taip pat stipriai įtakoja energijos suvartojimą, pavyzdžiui judesio aktyvuotas įrašymas sunaudoja iki 80% mažiau energijos nei nuolatinis vaizdo įrašymas. DI apmokyti objekto atpažinimą atliekantys modeliai taip pat įtakoja bendrą energijos suvartojimą, dėl to pasirenkant tinkama vaizdo įrašų valdymo programinę įrangą reikia atsižvelgti į energijos efektyvumo aspektus.

1.3.4. Programinės priemonės

Dabartinių technologijų laikais yra ganėtinai platus programavimo kalbų pasirinkimas, kaip ir bibliotekų, bei platformų, kurie palengvina programinio kodo rašymą.

Įgyvendinant programavimo užduotis su mikrovaldikliu, reikia gerai apsvarstyti, kokios programavimo kalbos bus naudojamos, nes planuojama įgyvendinti sudėtingą objektų atpažinimą naudojant pasirinktą mikrovaldiklį. Su tokiais uždaviniais geriausiai susitvarkytų *Python* arba *Java* programavimo kalbos.

Vidinio programavimo sąsaja

Python programavimo kalba yra populiari ir plačiai naudojama, ji yra draugiška pradedantiesiems programuotojams. Ši kalba yra nemokama ir gana lanksti, ją naudoja tokie populiariūs tinklalapiai, kaip: *Youtube*, *Quora* ir *Instagram* (PYPL Popularity of Programming Language, 2024). Ši programavimo kalba pasižymi keliais esminiais privalumais: turi žymiai trumpesnius ir lengvai skaitomus kodo fragmentus, nes yra mažiau griežtos sintaksės taisyklės, kas leidžia greičiau kurti ir testuoti programas. Labai populiari *Python* biblioteka - *OpenCV*. *Python* taip pat plačiai naudojamas mašiniam mokymuisi ir dirbtiniam intelektui su tokiomis bibliotekomis, kaip *TensorFlow*, *Keras* ir *PyTorch*, kurios yra integruotos su vaizdo atpažinimo įrankiais. Yra plačiai naudojamas *Raspberry Pi* projektuose. Daugiau naudojamas dėl paprastumo ir greito prototipavimo. Leidžia greitai kurti ir testuoti modelius, ypač DI ir mašininio mokymosi užduotimis. Paprasta integracija su dideliu skaičiumi bibliotekų ir įrankių, naudojamų tiek mokslinėje veikloje, tiek pramonėje. Tačiau ši sistema, kaip ir bet kuri kita turi savo trūkumų, nepaisant savo privalumų *Python* yra interpretacinė kalba, kuri vykdoma lėčiau nei sudaryta kalba.

Dar viena vidinio programavimo kalba yra *Java*. Ji yra viena iš labiausiai paplitusių šiuolaikinių kompiuterių programavimo kalbų. Dažnai naudojama kuriant žiniatinklį ir programas. *Java*, kaip vidinį programavimą, naudoja kelios populiarios svetainės, tokios kaip: *Amazon*, *Twitter* ir *Youtube* (Ben, 2023).

Šios sistemos privalumas tas, kad ji sudaryta ir paleidžiama naudojant JIT (angl. *Just-In-Time*) kompiliatorių, kuris optimizuoja programos vykdymą kiekvieną kartą paleidžiant. *Java* paprastai veikia greičiau nei *Python* dėl savo kompiliavimo į baitkodus ir vykdymo *Javos* virtualioje mašinoje. Gali būti geresnis pasirinkimas, kai reikalingas didelis našumas ir optimizavimas.

Pagrindinis šios kalbos trūkumas yra tas, kad reikalauja daugiau šabloniško kodo ir yra daugiau apribojimų sintaksėje, dėl ko *Java* kodas gali tapti ilgesnis ir sunkiau skaitomas bei rašomas, ypač pradedančiajam. Yra bibliotekų, pavyzdžiui, *OpenCV*, taip pat naudojamos vaizdo atpažinimo projektams, bet jos nėra taip plačiai optimizuotos ir naudojamos *Java* kontekste. Ją galima integruoti su įvairių rūšių sistemomis ir bibliotekomis, tačiau *Python* turi didesnę pasirinkimą specialiai vaizdo atpažinimo ir mašininės vizijos srityje. 1.3 lentelėje yra pavaizduotas *Java* ir *Python* kriterijų palyginimas.

1.3 lentelė. „Java“ ir „Python“ kriterijų palyginimas

Aspektas	Java	Python
Vykdymo greitis	Greitesnis dėl JIT kompiliavimo	Lėtesnis dėl interpretacinio pobūdžio
Kūrimo greitis ir paprastumas	Reikalauja daugiau šabloninio kodo, sudėtingesnis	Paprastesnis ir greitesnis, mažesnė mokymosi kreivė
Bibliotekos vaizdo atpažinimui	„OpenCV“, mažiau optimizuota	„OpenCV“, „TensorFlow“, „Keras“, „PyTorch“, plačiai naudojama
Palaikymas RPi	Suderinamas, bet mažiau bendruomenės išteklių	Plačiai naudojamas su daug bendruomenės išteklių
Naudojimo intensyvumas	Geras našumui ir optimizavimui	Geras greitam prototipų kūrimui ir testavimui
Integravimas su kitais įrankiais	Galimas, bet ribotesnė biblioteka pasirinkimo	Paprasta integracija su plačiu bibliotekų sąrašu
Bendruomenė ir dokumentacija	Mažiau pavyzdžių ir išteklių	Didelė bendruomenė, daug pavyzdžių ir dokumentacijos
Populiarumas mašininio mokymosi ir DI srityse	Mažiau populiarus	Labai populiarus dėl didelio kiekio įrankių

Vienas iš kone svarbiausių dalykų programuojant – naudoti tinkamą teksto redaktorių. Su *Python* ir *Java* kalba, gan lengvai susidoroja didelė dalis tekstinių redaktorių. Kumar straipsnyje tarp populiariausių

redaktorių buvo paminėti *PyCharm* ir *VS code*, kurie pritaikomi su abiem programavimo kalbomis (Kumar, 2020).

PyCharm yra nemokamas, galingas ir lengvai pritaikomas teksto redaktorius. Jame yra daug integruotų įrankių ir funkcijų, reikalingų jūsų projekto plėtrai. *PyCharm* privalumas yra kodo sintaksės ir klaidų paryškinimas; turi integravimus, kurie leidžia lengvai ir efektyviai naršyti kode, galima lengvai naršyti tarp projekto failų, klasių, metodų ir t.t. be to palaiko daug įvairių bibliotekų.

VS Code yra tekstinio kodo redaktorius, tinkamas naudoti *Windows*, *Linux* ir *MacOS*. Sukurta *Microsoft*, *VS Code* palaiko daugybę programavimo kalbų, reikia tik įdiegti pasirinktos programavimo kalbos plėtinį. Pagrindinis privalumas yra įvairių modulių palaikymas, turi integruotą komandų eilutę kodui paleisti bei šis redaktorius leidžia dirbti su *GitHub* projektais, turi tiesioginio bendrinimo funkciją, leidžiančia nuotoliniu būdu valdyti įrenginį.

1.3.5. *Perspėjimai*

Vaizdo stebėjimo sistemas galima sukonfigūruoti taip, kad jie praneštų kai įvyksta konkretūs įvykiai, pvz., judesio aptikimas, mėginimas sunaikinti ar neteisėta prieiga. Tai padeda vartotojams būti informuotiems ir prireikus nedelsiant imtis veiksmų.

Svarbus profesionalios stebėjimo sistemos aspektas yra nuolat būti informuotam apie tai, kas vyksta kameros stebėjimo diapazone. Nepertraukiamą tiesioginį kamerų srautą visada galima peržiūrėti naudojant kamerų stebėjimo programinę įrangą ir nemokamą išmaniojo telefono programą, skirtą apsaugos kameroms.

CCTV apsaugos kamerų sistemos gali automatiškai pranešti vartotojui, kai aptinkamas judesys. Norint pateikti išmaniuosius judesio aptikimo įspėjimus, pirmiausia reikia įjungti judesio aptikimą DVR arba NVR įrašymo įrenginyje arba IP kameroje. Kad įgyvendinti pranešimų gavimą būtina įgyvendinti tam tikras sąlygas.

Būtinios sąlygos yra turėti DVR, NVR arba IP kamerą (įrenginį) prijungtą prie interneto, ją pridėjus prie išmaniojo telefono programėlės, galima peržiūrėti apsaugos kameras iš išmaniojo telefono ir įrenginyje įjungus judesio aptikimą (Khodadin & Pudaruth, 2020).

1.4. Autonominių vaizdo stebėjimo sistemų palyginamoji analizė

Autonominės vaizdo stebėjimo sistemos tampa neatsiejama šiuolaikinės saugos technologijų dalimi, sujungdamos išmanųjų vaizdo analizės potencialą, energetinį efektyvumą ir prisitaikymą įvairiomis sąlygomis. Šių sistemų populiarumą skatina ne tik jų gebėjimas veikti be pastovaus žmogaus įsikišimo, bet ir pažangios funkcijos, tokios kaip DI pagrįstas objektų atpažinimas, ilgalaikis veikimas nuo baterijų ar autonominis duomenų perdavimas per 4G/5G tinklus. Tačiau rinkoje siūlomi sprendimai labai skiriasi tiek techniniais

parametrais, tiek pritaikomumu konkrečioms situacijoms – nuo miesto stebėjimo iki nuotolinių sunkiai pasiekiamų vietų apsaugos. Šiame skyriuje atlikta palyginamoji analizė atskleidžia, kurios sistemos efektyviausiai balansuoja tikslumą, patikimumą ir energijos sąnaudas, remiantis objektyviais kriterijais: veikimo autonomiškumu, DI modelių tikslumu, integracijos su IoT galimybėmis ir ilgalaikiu techniniu palaikymu. Tyrimo rezultatai padeda ne tik suprasti esamas technologijų ribas, bet ir numatyti tendencijas, kurios formuos ateities saugos sistemas.

Šiame skyriuje palyginsiu dvi šiuo metu rinkoje populiarias autonomines vaizdo stebėjimo kameras: *Eufy SoloCam C210* ir *EZVIZ CS-EB3 4G* - jos skiriasi pagal techninius parametrus, paskirtį ir funkcionalumą, tačiau abi yra populiarios renkantis autonomine vaizdo stebėjimo sistemą, kuri informuotų gavėją apie užfiksuotą judesį.

Eufy SoloCam C210 – tai belaidė, akumuliatorinė išmanioji kamera, skirta lauko stebėjimui. Ji pasižymi aukšta 2K raiška, integruotu dirbtiniu intelektu ir vietiniu įrašų saugojimu. Vienas iš kameros trūkumų yra tai, kad kamera nepalaiko 4G ryšio ir veikia tik su WiFi, dėl ko ji gali būti naudojama tik ribotose vietose, kuriuose pasiekiamas belaidis internetas. Sistemos funkcionalumas leidžia jai veikti nuo judesio aptikimo, kai judesio daviklis aptinka judėjimą, jis įgalina integruotą DI, kuris aptinka objektą bei atsiunčia pranešimą. Tai leidžia prailginti kameros veikimo laiką iki 3 mėnesių.



1.1 pav. Autonominė vaizdo kamera Eufy SoloCam C210 (Šaltinis:

<https://www.eufy.com/products/t8b00122?variant=42541391446202>)

EZVIZ CS-EB3 4G – tai belaidė kamera, veikianti per 4G LTE ryšį, ideali vietovėms be Wi-Fi prieigos. Kamera tinkama lauko sąlygomis, turi saulės panelės palaikymą, kas gali dar prailginti veikimo laiką. Bendrai ši kamera yra puikus pasirinkimas siekiant apsaugoti turtą vietose, kur nėra įdiegto WiFi ryšio. Kamera atpažįsta objektą iki 15 m atstumu, siunčia pranešimus apie užfiksuotą objektą. Kamera be įkrovimo veikia ~100 dienų, tai pasiekama pasinaudojant AOV (angl. *Always-on protection*) metodu. Kamera 24 val. per parą

stebi aplinką, ir jeigu joje nėra judesio, ji kas tam tikrą laiką daro ekrano nuotrauką, o paros pabaigoje yra sukuriama laiko intervalais paremtas vaizdo įrašas. Sistema neatpažįsta konkretaus objekto, o tik jo formą t. y. Orientuojasi į jo poziciją bei fiksuoja žmogų arba transporto priemonę.



1.2 pav. Autonominė vaizdo kamera EZVIZ CS-EB3 4G (Šaltinis: <https://ezvizlife.lt/produktas/ezviz-cs-eb3-4g-lauko-akumuliatorine-kamera/>)

1.1 lentelėje matomas aptartų sistemų bei projektuojamos sistemos charakteristikos. Sistemų veikimas paremtas skirtingomis logikomis, tačiau visų funkcija ta pati – atpažinti objektą.

1.4. lentelė. Autonominių vaizdo stebėjimo sistemų su objektų atpažinimu palyginimas

Funkcija	<i>Eufy SoloCam C210</i>	<i>EZVIZ CS-EB3 4G</i>	Projektuojama sistema
Ryšys	Wi-Fi	4G LTE	3/4 G LTE
Vaizdo raiška	2304 × 1296 (2K)	2304 × 1296 (2K)	2592 × 1944 (5K)
DI funkcijos	Žmogaus atpažinimas (kiti objektai su <i>HomeBase</i> atnaujinimu)	Žmogaus ir transporto priemonių formos atpažinimas	Pagal naudotojo parinktus nustatymus
Objekto atpažinimas	Taip	Ne	Taip
Atmintis	8 GB integruota	MicroSD / Debesų saugykla	MicroSD
Distancija	IR iki 10 m	15 m	~ iki 100 m

Apibendrinant, išanalizavus egzistuojančius sprendimus, matoma, kad jie turi tam tikrų trūkumų, pvz.: sistemos veikia nedideliu atstumu, bei veikia tik su WiFi pagalba, arba atpažįsta objektus ne pagal jų klasifikaciją su DI pagalba, o pagal jų formą bei veikia su jutiklių pagalba, kas pagerina jų veikimo laiką, tačiau tai neatlieka pagrindinės siekiamos užduoties – užfiksuoti objektą atokioje vietoje, kurioje neįmanoma prijungti WiFi arba kur reikia tiksliai nustatyti analizuojamą objektą. Projektuojama sistema turėtų veikti su

3/4G ryšiu, kad būtų pasiekama iš įvairiausių vietų bei turėtų atpažinti objektus pagal jų klasifikaciją bei naudotojo poreikius ir užfiksuoti objektą su maksimaliu tikslumu iš didelio atstumo.

1.5. Skyriaus išvados

1. Atlikus problemines srities analize nustatyta, kad egzistuoja poreikis autonominėms vaizdo stebėjimo sistemoms, kurios gali ne tik pranešti apie užfiksuota judesį, bet ir atpažinti vaizde atsiradusi objektą. Kad tai realizuoti šiam momentui trūksta tinkamai ištirtos ir ištestuotos sistemos, veikiančia su ribotais ryšio ištekliais, su vaizdo atpažinimo algoritmais, bei minimaliu energijos suvartojimu.
2. Siekiant padengti dabartinių sistemų trūkumus yra rekomenduojama naudoti energetiškai efektyvias mikrovaldikliu pagristas sistemas, kurios gebėtų atlikti objektų atpažinimą ir tuo pačių naudotų minimaliai energijos bei galėtų veikti nuo akumuliatoriaus.
3. Aprašyta iš ko turėtų susidaryti autonominė vaizdo stebėjimo sistema ir nustatyta, koks sistemos komponentų energijos suvartojimas. Palygintos rinkoje populiariausios autonominės vaizdo kameros gebančios atpažinti objektą DI pagalba.

2. AUTONOMINĖS VAIZDO STEBĖJIMO SISTEMOS SU OBJEKTŲ ATPAŽINIMU PROJEKTAVIMAS

Šiame skyriuje aptartas būsimos sistemos projektavimas: planavimas, sistemos raiškūs paveikslėlis, funkciniai ir nefunkciniai sistemos reikalavimai, projekto realizavimui reikalingų priemonių pasirinkimas, mikrokompiuterio pradiniai nustatymai bei vaizdo apdorojimo ir atpažinimo sistemos kūrimas.

2.1. Projekto įgyvendinimo užduočių planavimas

Projektuojant autonominę vaizdo stebėjimo sistemą su vaizdo atpažinimu, naudojant mikrokompiuterį, būtina atlikti kelis skirtingus žingsnius. Žemiau pateikiama užduočių diagrama, kurioje susistemintas skirtingų žingsnių procesas.

2.1. lentelė. Autonominės vaizdo stebėjimo sistemos su objektų atpažinimu projektavimo užduočių diagrama

Projekto planavimas ir reikalavimų analizė	Apibrėžiami projekto tikslai, funkcionalumas ir techniniai reikalavimai
Aparatinės įrangos parinkimas	Tinkamo mikrokompiuterio ir kitų priedų parinkimas
Mikrokompiuterio pradinė konfigūracija	Operacinės sistemos parinkimas ir pradinių nustatymų konfigūravimas
Vaizdo apdorojimo ir atpažinimo programinės įrangos kūrimas	Mikrokompiuterio konfigūracija užtikrinant vaizdo stebėjimą, vaizdo apdorojimo funkcijos, objektų atpažinimo modelio parinkimas bei realiojo laiko analizė

Sistemos tikslas – automatiškai aptikti ir klasifikuoti objektus, vietovėse, kuriuose nėra tiesioginio elektros tiekimo.

Pagrindinė šio projekto problematika yra įrangos ir procesų energijos suvartojimas. Įprastos sistemos yra projektuojamos su sąlyga, kad bus užtikrintas pastovus sistemos maitinimas, tačiau, kai kuriais atvejais to padaryti neįmanoma, tačiau yra būtina užtikrinti saugumą ir vaizdo stebėjimą.

Kad tinkamai įgyvendinti projektą būtina teisingai parinkti aparatinę įrangą, kuri būtų atsakinga ne tik už teisingą objektų atpažinimą bet ir veiktų autonomiškai, o tai reiškia, sunaudotų nedaug energijos. Taip pat

svarbu tinkamai nustatyti mikrovaldiklio pradinę konfigūraciją, kuri būtų atsakinga už sklandų sistemos veikimą. Operacinės sistemos, programinės kalbos ir kt. parinkimas įtakos tolimesnį projekto vystymąsi ir sklandžią jo realizaciją, o tinkama mikrokompiuterio konfigūracija užtikrins vaizdo stebėjimą, tinkamai parinkti vaizdo atpažinimo modeliai užtikrins reikiamą sistemos veikimą.

Sistemos valdyme numatomas vienas naudotojas, kuris turėtų prieigą prie pagrindinių sistemos funkcijų, tačiau reikalui esant, tokio pačio lygio prieiga būtų suteikta kitiems asmenims (naudojant vaizdo stebėjimo kamera namuose, prie sistemos prieiga galima būtų suteikti visiems šeimos nariams).

2.1.1. *Sistemos funkciniai reikalavimai*

Sistemos funkciniai reikalavimai apibrėžia konkretų sistemos elgesį t. y. ką sistema turi daryti, kokias funkcijas vykdyti ir kaip reaguoti į vartotojo arba aplinkos veiksmus. Kad autonominė vaizdo stebėjimo sistema su objektų atpažinimu tinkamai veiktų, ji turi atitikti šias pagrindines funkcijas:

FR1: Sistema turi įrašyti vaizdą iš kameros bent 2–30 FPS su 720p raiška.

FR2: Sistema turi naudojant dirbtinį intelektą užtikrinti automatizuotą objektų aptikimą.

FR3: Sistema turi identifikuoti ir klasifikuoti objektus naudojant modelius (pvz.: YOLO, *TensorFlow Lite* ir kt.).

FR4: Sistema turi leisti pasirinkti, kokį objektą aptikti.

FR5: Sistema turi siųsti aiškius pranešimus (SMS, MMS) aptikus tam tikrus objektus ar įvykius.

FR6: Sistema turi pažymėti aptiktus objektus kadre.

FR7: Sistema turi automatiškai išsaugoti įrašus su pažymėtais objektais.

2.1.2. *Sistemos nefunkciniai reikalavimai*

Nefunkciniai reikalavimai apibrėžia kokybinius sistemos parametrus – tai yra, kaip sistema turi veikti, kokias savybes turėti ir kokios sąlygos turi būti tenkinamos. Jie neapibūdina konkrečių funkcijų, tačiau nurodo bendrus veiklos reikalavimus. Pagrindiniai šios sistemos nefunkciniai reikalavimai būtų:

NFR1: Sistema turi veikti greitai ir patikimai.

NFR2: Sistema turi būti realizuota naudojant mikrovaldiklį.

NFR3: Objektų atpažinimas turi veikti realių laiku.

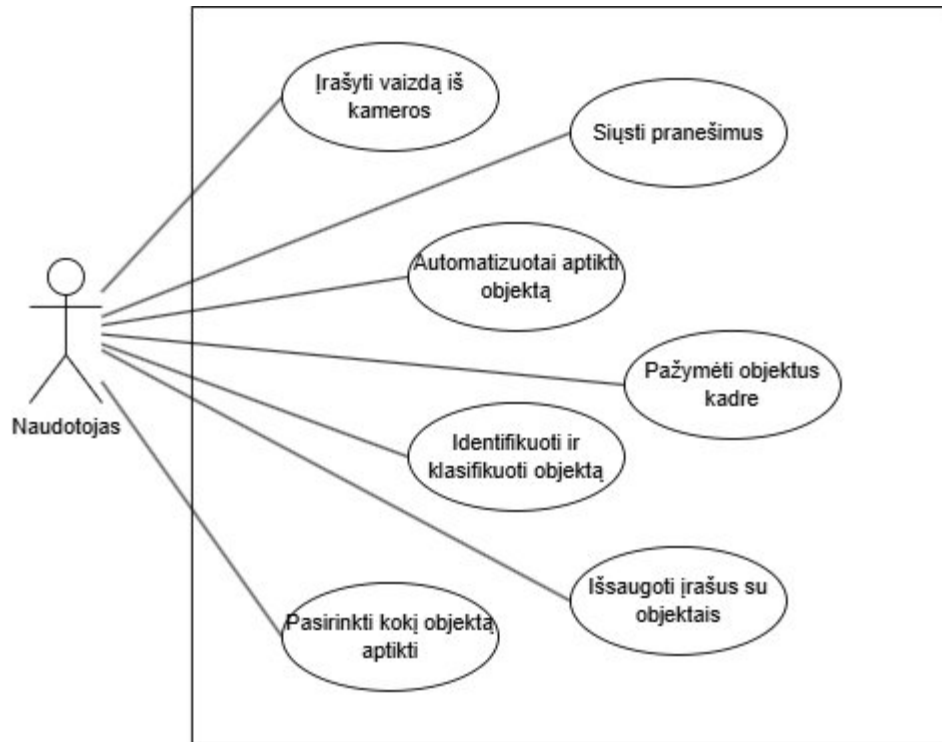
NFR4: Sistemos konfigūracija turi būti paprasta.

NFR5: Sistema turi veikli pilnai autonomiškai naudodama akumuliatorius.

Optimalus balansas tarp šių reikalavimų užtikrina patikimą ir ilgą autonomines vaizdo stebėjimo sistemos su objektu atpažinimu veikimą.

2.1.3. Sistemos panaudos atvejų diagrama

Panaudos atvejų (*Use Case*) diagrama vaizduoja, kaip vartotojai sąveikauja su sistema ir kokias funkcijas (panaudos atvejus) sistema atlieka. Ji apibrėžia sistemos funkcionalumą iš vartotojo perspektyvos. 2.1 paveiksle pavaizduota panaudos atvejų diagrama projektuojamam autonominės vaizdo stebėjimo sistemos projektui:



2.1 pav. Sistemos panaudos atvejų diagrama

Žmogeliukas ir burbuliukai iš funkcinių reikalavimų

Pagrindiniai pavaizduotos sistemos panaudos atvejai yra sistemos paleidimas, aptinkamų objektų konfigūracija bei pageidaujamos pranešimų siuntimo formos pasirinkimas. Kai aptinkamas objektas sistema automatiškai apdoroja vaizdą, išsaugo vaizdą su pažymėtais objektais bei valdo energiją. Objektų aptikimas įgalina įrašų išsaugojimą ir SMS/MMS siuntimą.

2.1.4. Sistemos sekų diagrama

Sekos diagrama vaizduoja sistemoje vykstančių veiksmų ir procesų seką.

Pagrindiniai sistemos komponentai:

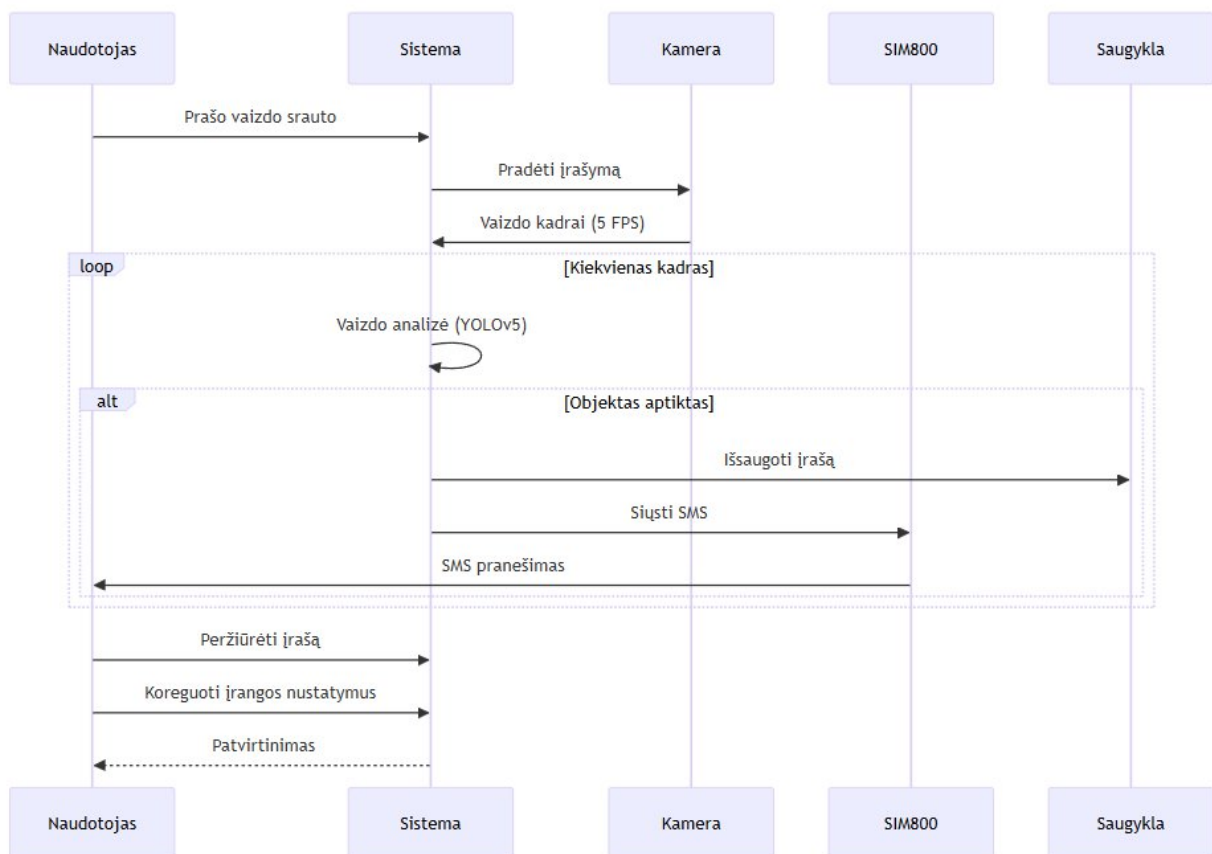
- Vartotojas (*Naudotojas*);
- *Raspberry Pi 4* (*Sistema*);
- *OV5647* Kamera (*Kamera*);

- SIM800 Modulis (*SIM800*);
- Duomenų saugykla (*Saugykla*).

Seka vyksta tokia tvarka:

- *Naudotojas* įjungia *Sistemą*;
- *Sistema* išsiunčia *Kamerai* signalą pradėti įrašymą;
- *Kamera* gražina į *Sistemą* užfiksuotus vaizdo kadrus;
- *Sistema* analizuoja kadrus su YOLOv5 pagalba;
- YOLOv5 pateikia rezultatą *Sistamai* (pvz.; Aptiktas žmogus);
- *Sistema* siunčia signalą *SIM800* apie atpažintą objektą;
- *SIM800* siunčia pranešimą *Naudotojui*;
- *Naudotojas* peržiūri įrašą, jei reikia koreguoja įrangos nustatymus, sistema patvirtina keitimus.

2.2 paveiksle pavaizduota sekos diagrama projektuojamam autonominės vaizdo stebėjimo sistemos projektui:



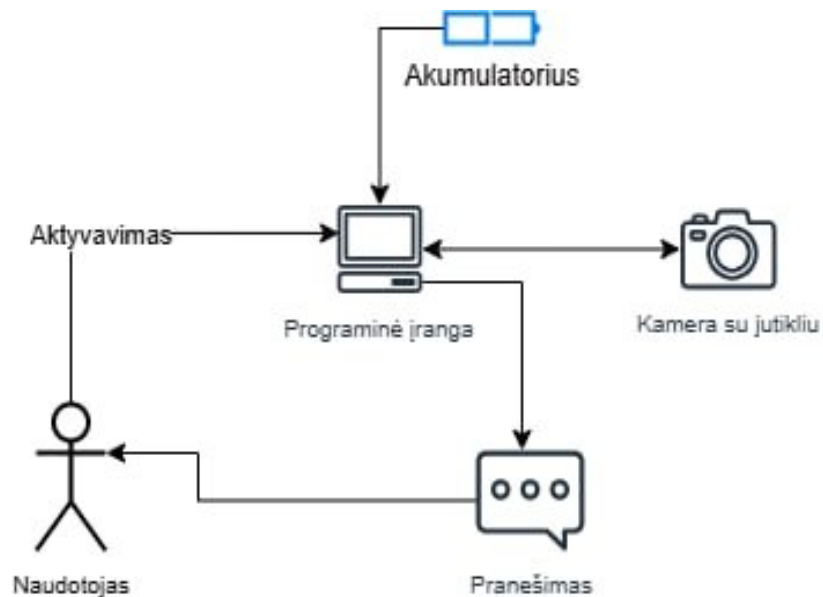
2.2 pav. Sistemos sekos diagrama

2.2. Projektuojamos sistemos raiškūs paveikslėlis

Šiame skyriuje pateikiami du raiškieji paveikslėliai, paaiškinantys sistemos pagrindines problemas ir jų sprendimo galimybes. Esamos būsenos raiškiajame paveikslėlyje pateikiamos su autonominio vaizdo stebėjimo sistemos su objektų atpažinimu siejamos ir šiame darbe nustatytos autonomiškumo pagrindinės problemos, o siekiamos būsenos raiškiajame paveikslėlyje pateikiamas šių problemų galimas sprendimo būdas įgyvendinant projektuojamą sistemą.

2.2.1. Esamos sistemos raiškūs paveikslėlis

Autonominės vaizdo stebėjimo sistemos dažniausiai apima labai primityvią sistemą, kurioje kamera veikia jutiklio pagrindu. Vartotojas įjungia kamerą, kai kamera pastebi judesį, ji siunčia signalą programai, kuri nedelsiant informuoja naudotoją apie pastebėtą veiklą kameros stebėjimo diapazone. Pagrindinė problema tokios sistemos, kad ji negali atpažinti, koks objektas sužadino pranešimą. Dėl šio neefektyvumo, apkraunamas vartotojas, nes jis, norėdamas pamatyti, koks objektas yra užfiksuotas kadre, privalo patikrinti mobiliąją aplikaciją arba internetinę svetainę. 2.3 paveikle yra pavaizduotas esamos būsenos raiškūs paveikslėlis, kuriame atsispindi sistemos trūkumai.

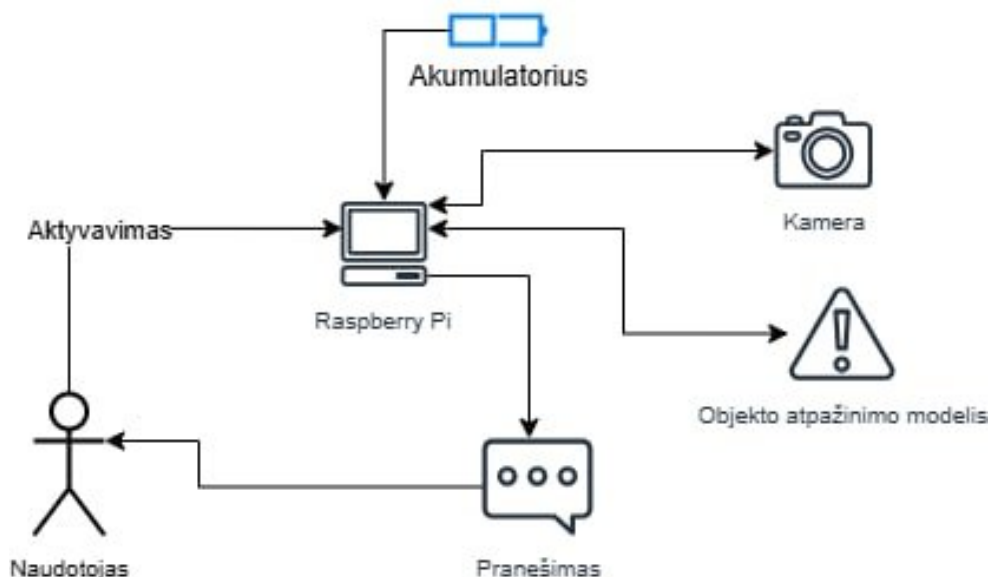


2.3 pav. Esamos būsenos raiškūs paveikslėlis

2.2.2. Siekiamos sistemos raiškūs paveikslėlis

Siekiant pagerinti autonominę vaizdo stebėjimo sistemą, yra programuojama sistema su tinkamu objektų atpažinimu. Šioje sistemoje atsiranda atskiras objekto atpažinimo modelis, kuris veikdamas kartu su

judėjimo aptikimo modeliu, pagrįstu vidutine kvadratine paklaida (MSE) atpažįsta vartotojui aktualų objektą (pvz., gyvūną, žmogų, automobilį) ir jam siunčia tik aktualų pranešimą su atpažintu objektu. Ši sistema pagerina vartotojo naudojimą, nes anuliuoja netinkamus (pvz. nuo vėjo judančių šakų) objektus bei suteikia patogesnę sistemos naudojimą.



2.4 pav. Siekiamos sistemos raiškūs paveikslėlis

2.3. Projekto realizacijai reikalingos aparatinės įrangos parinkimas

Šiame poskyryje bus nagrinėjama aparatinė įranga, su kurios pagalba bus realizuojama autonominė vaizdo stebėjimo sistema su objektų atpažinimu.

2.3.1. Mikrovaldiklio parinkimas

Analitinėje dalyje atlikus mikrovaldiklių pasiūlos analizę, bei panagrinėjus rinkoje esančius prieinamus mikrokompiuterius nutarta šiam projektui naudoti *Raspberry Pi 4B* (4Gb) mikrokompiuterį. RPi yra optimalus pasirinkimas dėl balanso tarp našumo, energijos efektyvumo ir lankstumo. Jis tinka tiek paprastiems judesio aptikimo projektams tiek sudėtingoms sistemoms su objektų atpažinimu.

RPi turi daug privalumų šiam projektui: palaiko *Python*, jo operacinė sistema turi daug integruotų bibliotekų o veikdamas vidutiniškai sunaudoja vos ~2-6 W energijos, kas leidžia sistemas su jo panaudojimu paversti autonominėmis.

2.3.2. Kameros parinkimas

Analitinėje dalyje taip pat išanalizavus autonominei vaizdo stebėjimo sistemai su objektų atpažinimu tinkamas vaizdo stebėjimo kameras nutarta naudoti *Arducam OV5647 Camera Module* – ji yra ekonomiška ir paprasta kamera turinti ribotą funkcionalumą tačiau tinkama mažo biudžeto projektams, kuriems nereikia didelės kokybės (2.5 pav.).



2.5 pav. OV5647 kameros modulis (Šaltinis: <https://www.anodas.lt/odseven-kamera-hd-ir-cut-ov5647-5mpx-dienos-nakties-ir-skirta-raspberry-pi-ir-moduliai?search=RPi%204%201kamera&page=3>)

Šis 5MP modulis su *OmniVision OV5647* sensoriumi yra visiškai suderinamas su RPi mikrokompiuteriu bei suvartoja mažiau energijos (~0.5–1 W) nei aukštos raiškos kameros, todėl tinka autonominėms sistemoms. Kamera lengvai valdoma per *Python* bibliotekas, o paprastas sąsajos dizainas sumažina diegimo sudėtingumą.

2.3.3. Pranešimų siuntimo aparatinės įrangos parinkimas

Pranešimams su atpažintu objektų siuntimui šiame projekte bus naudojamas RPi *SIM800 GSM GPRS* modulis (2.6 pav.).



2.6 pav. Raspberry Pi SIM800 GSM GPRS modulis

Šis modulis yra visiškai pritaikytas RPi sąsajai, ir leidžia prisijungti prie mobiliųjų tinklų, siųsti SMS, skambinti ar naudoti internetą per GPRS. Jis yra puikus sprendimas autonominėms sistemoms, kurios turi veikti ten, kur nėra WiFi ir *Ethernet* prijungimo. Vienas iš šio modulio trūkumų yra jo energijos poreikis, jis veikia su 3.7–4.4 V įtampa, bei reikalauja išorinės antenos ir gali sunaudoti iki 2A srovės per ryšio užmezgimą, todėl tinkamiausias tik trumpalaikiam ryšiui pvz., siųsti išpėjimus SMS.

2.3.4. Akumuliatoriaus parinkimas

Maitinimo šaltinis tyrimui buvo pasirinktas remiantis vidutiniu sistemos energijos suvartojimu. *Li-ion* 18650 yra vienas populiariausių akumuliatorių, kuris plačiai naudojamas nešiojamoje elektronikoje, elektroniniuose įrenginiuose ir autonominėse sistemose. Jie yra patvarus, veikia 300-500 pilnų įkrovimo/iškrovimo ciklų su nedideliu pradinės talpos sumažėjimu. Jie yra plačiai prieinami, ištestuoti bei nebrangus, tas patogu testuojamai sistemai. Papildomai su akumuliatoriais bus naudojamas *Li-ion* ličio baterijų įkrovimo modulis bei DC/DC įtampos keitiklis (2.7 ir 2.8 pav.).



2.7 pav. Li-ion 18650 akumuliatoriai



2.8 pav. Ličio jonų ličio akumuliatoriaus įkrovimo modulis ir DC/DC įtampos keitiklis

Pagal aukščiau pasirinktą aparatinę įrangą buvo sukonstruota autonominė vaizdo stebėjimo sistema su objektų atpažinimu bei pranešimu siuntimu paremta mikrovaldikliu. Sukonstruota sistema pavaizduota 2.9 paveiksle.



2.9 pav. Sukonstruota autonominė vaizdo stebėjimo sistema

2.4. Autonominės vaizdo stebėjimo sistemos su objektų atpažinimu architektūra

Autonominės vaizdo stebėjimo sistemos su objektų atpažinimu architektūrą galima suskirstyti į 4 pagrindinius sluoksnius: įrenginių lygmuo, duomenų apdorojimas, ryšio valdymas ir vartotojo sąsaja.

Pagrindinis yra įrenginių lygmuo, jis susideda iš: *Raspberry Pi 4* – centrinio valdymo vieneto (4 GB RAM, 128 GB SD kortelė), *OV5647* kameros vaizdo įvedimui (720p, 5 FPS), *SIM800* modulio (SMS/MMS pranešimų siuntimui) ir akumuliatorių Li-ion (~16,6 V, ~17Ah).

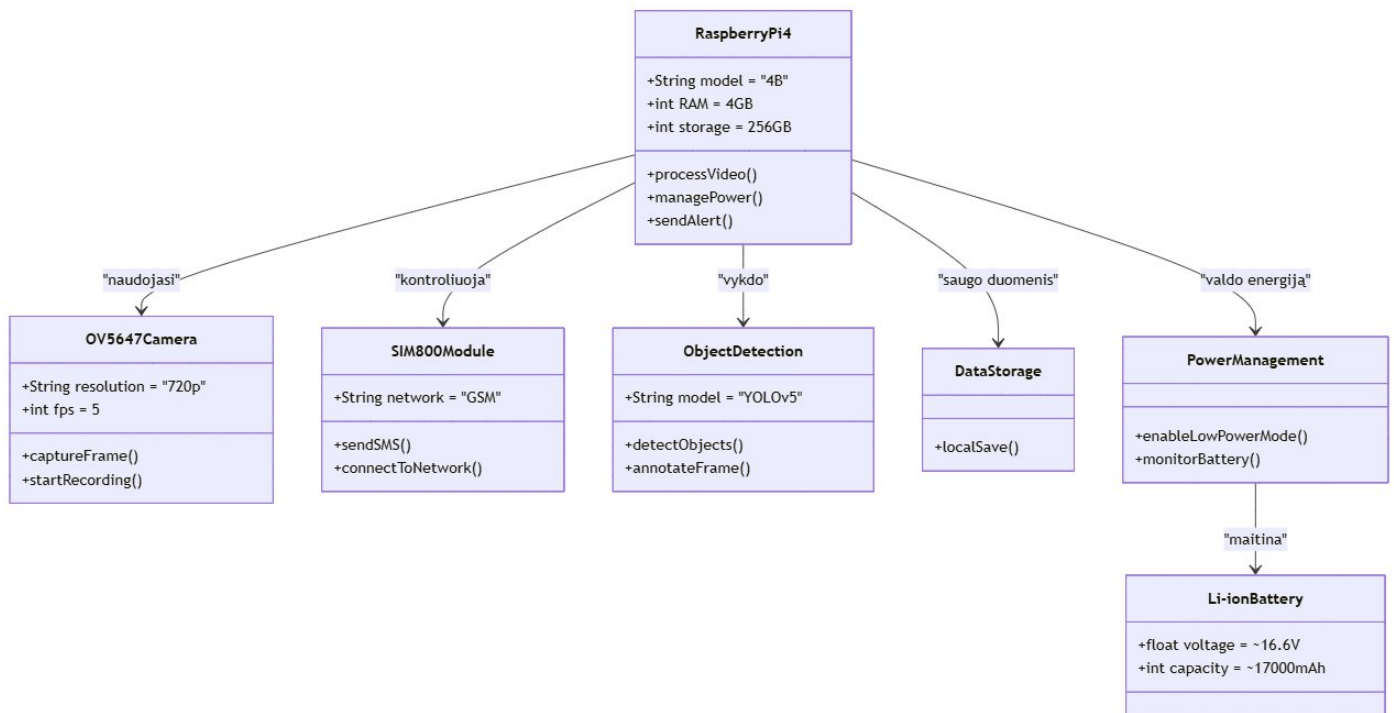
Duomenų apdorojimo lygmens funkcijos yra: vaizdo gavimas – kamera perduoda vaizdą RPi per CSI-2 sąsaja, objektų atpažinimas – *YOLOv5 Mobilenet FTLite* atpažista duominantį objektą, duomenų saugojimas – SD kortelėje.

Ryšio valdymas susideda iš *SIM800* modulio – siunčia SMS ir MMS įspėjimus aptikus objektą (pvz.: patebėtas objektas: žmogus – 0.75) bei naudoja AT komandas valdymui.

Vartotojo sąsaja nustato sistemos valdymo būdus – siunčia pranešimus į telefoną.

Maitinimo valdymas DC/DC įtampos keitiklis – adaptuoja įtampa pagal prietaiso reikalavimus, įkrovimo modulis – užtikrina saugų įkrovimą/iškrovimą, saugo nuo perkrovos ir greito išsikrovimo.

Žemiau pavaizduota šiam projektui pritaikyta sistemos architektūros schema (2.10 pav.).



2.10 pav. Sistemos architektūros schema

2.5. Mikrovaldiklio ir atpažinimo sistemos konfigūracija

Ankstesniame skyriuje atlikus programavimo kalbų analize nustatyta, kad *Python* programavimo kalba yra optimaliausias pasirinkimas projektui su RPi mikrokompiuteriu, kadangi ji nėra tokia sudėtinga ir yra viena iš pagrindinių programavimo kalbų, jos pagalba bus lengviau pasiekti numatytą darbo tikslą. Taip pat, norint struktūrizuoti kodą, reikia pasirinkti karkasą, kurio pagalba bus lengviau tai įgyvendinti (Globaldev.tech, 2022).

Apžvelgus tinkamiausius teksto redaktorių buvo pasirinktas *PyCharm*. Nors *VS Code* bendrai yra geresnis kodo redaktorius, tačiau jis yra pritaikytas daugybei programavimo kalbų, tuo tarpu *PyCharm* yra pritaikytas labiau *Python* kalbai. Be to, *PyCharm* palaiko populiariausias operacines sistemas. Anot Šarūno Navicko (Navickas, 2024), *PyCharm*, pasirūpina jog *Python* kalbos žingsniai nesiskirtų, nesvarbu kokia operacinė sistema būtų naudojama.

2.5.1. Mikrovaldiklio pradinė konfigūracija

Projekto įgyvendinimui pasirinkus RPi mikrokompiuterį, būtina užtikrinti jo tinkamą veikimą. RPi puikiai veikia su nemokama *Raspbian* operacine sistema. *Raspbian* yra *Debian* pagrindu sukurta operacinė

sistema, specialiai pritaikyta RPi aparatinei įrangai. Tai optimizuota ir visą programinės įrangos paketą reikalingą įvairių projektų kūrimui ir naudojimui RPi platformoje turinti operacinė sistema.

2.5.2. Vaizdo apdorojimo ir atpažinimo sistemos konfigūracija

Vaizdo stebėjimo sistemose su vaizdo atpažinimu, naudojant RPi, kameros objektų atpažinimas reikalauja įvairių technologijų ir įrankių, tokių kaip vaizdo apdorojimas, dirbtinis intelektas ir integruotų bibliotekų.

Kad kamera atpažintų objektą būtina sudaryti tam tikras sąlygas. Pirmiausia, kamera turi turėti galimybę užfiksuoti vaizdo srautą arba atskirus kadrus realiu metu. Kad kamera tiksliau atpažintų objektus, dažnai naudojamas vaizdo skiriamosios gebos ir kadrų dažnio nustatymas. Taip pat reikalingas yra tinkamas kameros modulis.

Vaizdo apdorojimui naudojamos integruotos bibliotekos tokios kaip *OpenCV*, joje atliekami pirminiai vaizdo apdorojimo žingsniai tokie kaip: spalvų transformacijos, fono triukšmo mažinimas, filtravimas, kraštinių aptikimas ir kt. Naudojant bibliotekas vaizdo iš lygių spalvų RGB galima pereiti į pilkąsias spalvas, taip sumažinant apdorojimo apkrovą. Taip pat vaizdo apdorojimo metu atliekami kiti paruošiamieji veiksmai, tokie kaip vaizdo normalizavimas ar dydžio keitimas.

Objektų atpažinimui bus naudojamas objektų atpažinimo modelis YOLOv5. Objektų atpažinimui galima naudoti tiek jau paruoštus modelius, tiek modifikuoti jau esamus modelius. Paprastesnis objektų atpažinimo metodas yra naudoti jau iš anksto modifikuotus modelius, tokius kaip *MobileNet*, *Inception* ar *ResNet*. Šie modeliai dažniausiai yra apmokyti su tūkstančiais nuotraukų ir turi gerą atpažinimo tikslumą. Tačiau, jeigu reikia specifinių objektų aptikimo, galima apmokyti modelį naudojant *TensorFlow*, *Keras* ar *PyTorch*. Tai reikalauja specifinio duomenų rinkinio ir galingo kompiuterio modifikavimui bei daugiau laiko apdorojimui, kas naudoja žymiai daugiau energijos.

Kad galima būtų objektą aptikti realiuoju laiku, įdiegus modelį, vaizdai ar vaizdo srautai yra siunčiami į modelį. Modelis tuo tarpu apdoroja vaizdą ir grąžina aptiktų objektų klasių etiketes ir jų vieta kadre (*bounding box*). Aptikti objektai gali būti sužymėti užrašu ir ribojančiomis dėžutėmis vaizdo kadre. Sistema turės siųsti pranešimus ar atlikti veiksmus (pvz.: išsiųsti perspėjimą el. paštu ar SMS) kaip atsaką į aptiktus objektus.

2.6.Skyriaus išvados

1. Atliktas būsimos autonominės vaizdo stebėjimo sistemos su vaizdo atpažinimu projektavimas, pavaizduota užduočių diagrama, panaudos atvejų diagrama, sekų diagrama, išryškinti funkciniai ir nefunkciniai reikalavimai, pavaizduotas sistemos raiškūs paveikslėlis.

2. Atlikus projekto realizavimui reikalingos aparatinės įrangos analizę naudojimui pasirinktas *Raspberry Pi 4* 4GB mikrokompiuteris su *OV5647* kamera, *SIM800* moduliu ir keturiais *Li-ion 18650* akumuliatoriais.
3. Technologijų analizės pagrindu pasirinkta *Python* programavimo kalba ir *PyCharm* teksto redaktorius. RPi veiks *Rasbian* operacinės sistemos pagrindu, o vaizdo apdorojimo ir atpažinimo sistema veiks su YOLOv5 modulio pagalba ir jame įdiegtomis bibliotekomis.

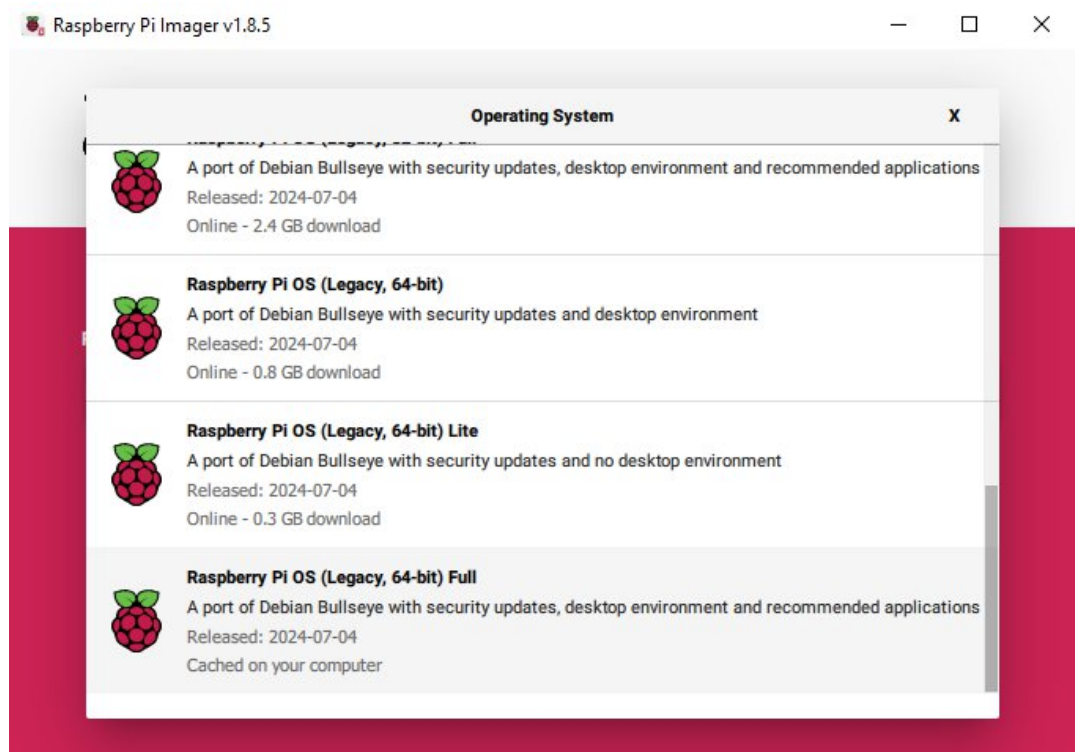
3. VAIZDO STEBĖJIMO SISTEMOS SU OBJEKTŲ ATPAŽINIMU REALIZAVIMAS

3.1. Sistemos konfigūracija

Magistro darbo įgyvendinimas buvo atliktas *Rasbian* OS aplinkoje, kuriant sistemas su *Python* programavimo kalba. Ši kalba yra prieinama ir gana lanksti, nes ji turi daugybę galimybių, padedančių lengviau įgyvendinti paskirtą užduotį.

Siekiant sukurti autonominę vaizdo stebėjimo sistemą naudojant RPi mikrokompiuterį, pirmiausia į SD kortelę yra įrašoma RPi OS vadinama *Raspbian*. RPi OS į mikrokompiuterį įrašoma su *Raspberry Pi Imager*. RPi *Imager* tai įrankis, skirtas lengvai įrašyti operacines sistemas į SD kortelę arba kitą atminties įrenginį. Su šiuo įrankiu galima greitai pasirinkti ir įrašyti įvairias RPi OS versijas bei kitas suderinamas operacines sistemas į savo atminties laikmeną.

Šiame projekte naudojama RPi *Imager* v1.8.5 versija, kurioje pasirinkta RPi OS 64-bit su *Debian Bullseye* versija. Operacinė sistema įrašoma į 128 Gb talpos SD kortelę siekiant užtikrinti saugų ir efektyvų mikrokompiuterio veikimą.



3.1 pav. RPi operacinės įrangos diegimas į SD kortelę

Kitas žingsnis yra RPi kompiuterio paleidimas. RPi plokštė prijungiama prie interneto tinklo (WiFi modulio), monitoriaus ir USB klaviatūros pagrindinei sąrankai. Į mikrokompiuterį įdedama paruošta SD kortelė su *Raspbian* OS ir prijungiamas maitinimo šaltinis.

Paleidus *Raspbian* yra atliekamas sąrankos procesas, aktyvuojama SSH (angl. *Secure Shell*) - *Linux* funkcija, leidžianti efektyviai atidaryti RPi mikrokompiuterio seansą iš pagrindinio kompiuterio komandinės eilutės.

Kad galima būtų naudoti objektų atpažinimą *Raspbian* nustatymuose būtina aktyvuoti kamera. Šiam projektui naudojamas OV5647 kameros modulis pritaikytas dirbti su RPi.

Atlikus pradinę RPi kompiuterio ir kameros konfigūraciją *Raspbian* konsolėje atliekamos *sudo apt update* ir *sudo apt upgrade -y* funkcijos siekiant valdyti paketus ir atnaujinti sistemą. Taip pat naudojama *sudo apt-get autoremove -y* funkcija norint išvalyti nereikalingus paketus. Kai yra diegiama programinė įranga per paketų valdymo sistemą, ji kartais įdiegia ir papildomus paketus, kurie yra būtini tos programinės įrangos veikimui. Kai programinė įranga pašalinama, šie papildomi paketai kartais lieka sistemoje, nors jau nėra reikalingi. Komanda „autoremove“ identifikuoja tokius paketus ir juos pašalina.

Kad būtų galima įgyvendinti *Flask* pagrindu veikiančią vaizdo stebėjimo sistemą su objektu atpažinimu reikalingi tam tikri programinės įrangos paketai.

RPi konsolėje pirmiausia yra įrašoma viena naujausių objektų aptikimo modelių YOLOv5. Naudojant *cd ~* ir *git clone <https://github.com/Ultralytics/yolov5.git>* funkcijas iš *GitHub* duomenų bazės į RPi OS yra klonuojama YOLOv5 kodo bazė. Vėliau iš šios bazės su funkcijomis *cd ~/yolov5* ir *pip3 install -r requirements.txt* į operacinę sistemą yra perkeltami YOLOv5 reikalavimai, kurie yra pavaizduoti žemiau esančiame paveikslėlyje (Pav. 3.2)

```

# YOLOv5 requirements
# Usage: pip install -r requirements.txt

# Base -----
gitpython>=3.1.30
matplotlib>=3.3
numpy>=1.23.5
opencv-python>=4.1.1
pillow>=10.3.0
psutil # system resources
PyYAML>=5.3.1
requests>=2.32.0
scipy>=1.4.1
thop>=0.1.1 # FLOPs computation
torch>=1.8.0 # see https://pytorch.org/get-started/locally (recommended)
torchvision>=0.9.0
tqdm>=4.64.0
ultralytics>=8.2.34 # https://ultralytics.comS
# protobuf<=3.20.1 # https://github.com/ultralytics/yolov5/issues/8012

# Logging -----
# tensorboard>=2.4.1
# clearml>=1.2.0
# comet

# Plotting -----
pandas>=1.1.4
seaborn>=0.11.0

# Export -----
# coremltools>=6.0 # CoreML export
# onnx>=1.10.0 # ONNX export
# onnx-simplifier>=0.4.1 # ONNX simplifier
# nvidia-pyindex # TensorRT export
# nvidia-tensorrt # TensorRT export
# scikit-learn<=1.1.2 # CoreML quantization
# tensorflow>=2.4.0,<=2.13.1 # TF exports (-cpu, -aarch64, -macos)
# tensorflowjs>=3.9.0 # TF.js export
# openvino-dev>=2023.0 # OpenVINO export

```

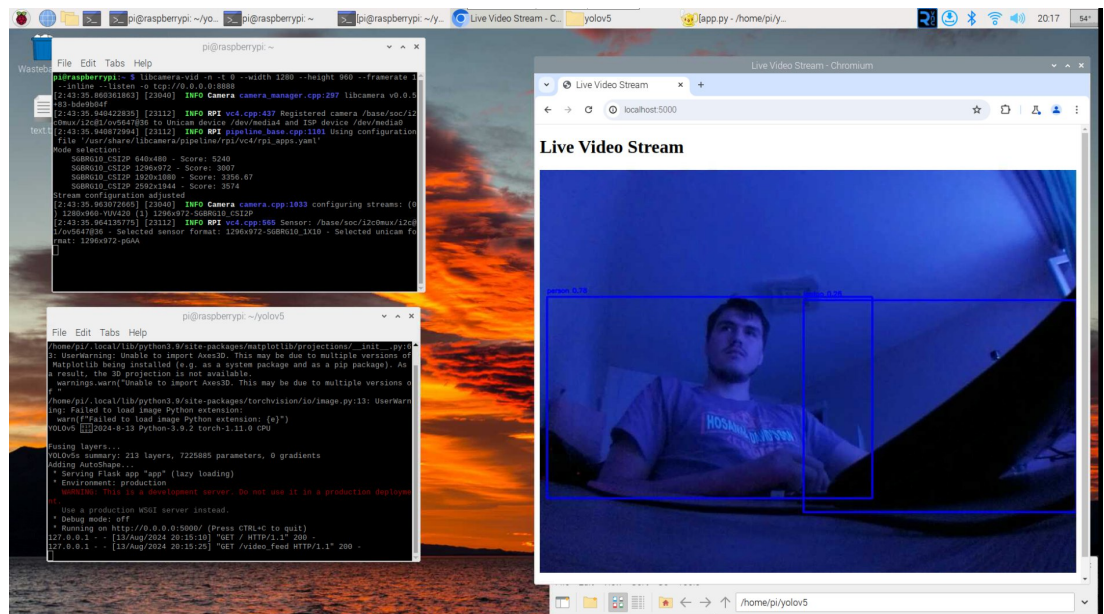
3.2 pav. YOLOv5 reikalavimai

Vėliau naudojant *pip3 uninstall torch torchvision* ir *pip3 install torch==1.11.0 torchvision==0.12.0* funkcijas į operacinę sistemą yra įrašomos *PyTorch* ir *Torchvision* bibliotekos. Jos skirtos atlikti vaizdo atpažinimo užduotis.

Norint įjungti TCP srautus per SSH reikia su *sudo nano ~/yolov5/detect.py* ir *is_url = source.lower().startswith(('rtsp://', 'rtmp://', 'http://', 'https://', 'tcp://'))* funkcijomis modifikuoti *is_url* kad ji įgalintų tiesioginio vaizdo stebėjimą.

Siekiant gauti bendrą kamerų valdymo sistemą į OS įrašoma *Libcamera*. Naudojant *libcamera-vid -n -t 0 --width 1280 --height 960 --framerate 1 --inline --listen -o tcp://127.0.0.1:8888* funkciją į OS įdiegiamas papildinys leidžiantis paleisti TCP.

Paleidus *python3 app.py* funkciją, RPi leidžia į naršyklės langą įrašius *localhost:5000* paleisti tiesioginę transliaciją.



3.3 pav. Naršyklės lange matomas tiesioginės transliacijos vaizdas

Vaizdo transliavimas tiesiogiai sunaudoja labai daug resursų, todėl darbe bus naudojamas MSE (angl. *Mean Squared Error*) algoritmas, kad palyginti vaizdus bei limituoti gaunamų pranešimų apie objektų atpažinimą kiekį.

Jei MSE viršija nustatytą slenkstį (šiuo atveju 100), galima teigti, kad vaizdas pasikeitė pakankamai reikšmingai, kad būtų verta atlikti objekto atpažinimą pagal objekto klasę (pvz., žmogus, transporto priemonė, gyvūnas), tuomet sukonstruojama žinutė, rodanti, kiek jų buvo aptikta, ir tai atvaizduojama konsolėje. Jei objektų nenustatyta, išveda "Šiuo metu ramu" funkcija. Įdiegus šią funkciją, sistema ne tik sumažina apdorojimo apkrovą, bet veikia efektyviau, nes laiko išteklių naudojami tik tada, kai tai tikrai būtina.

Naudojant algoritmus yra patikrinama, ar įvyko skirtumai tarp kadrų. Jei skirtumas pastebimas, kadras perduodamas YOLOv5n modeliui, ir atpažinti objektai pavaizduojami kadre.

Konsolėje pridėjama funkcija `cv2.waitKey(200)`, kuri užtikrina, kad programai reikia palaukti 200 milisekundžių tarp kiekvieno kadro, kas atitinka 5 kadrus per sekundę ($1000 \text{ ms} / 5 = 200 \text{ ms}$). Tai taikoma tam, kad užtikrinti minimalų kiekį „netikrų“ išpėjimų. Kodo struktūra išlieka ta pati, su galimybe aptikti žmones, transporto priemones ir gyvūnus, kaip buvo numatyta anksčiau.

Funkcija patogi tuo, kad jei vaizdai yra identiški, MSE bus 0. Jei vaizdai skiriasi pagal pikselius, MSE didės. Kuo didesnis MSE skaičius, tuo didesni skirtumai tarp vaizdų.

Naudojant MSE funkcija, apskaičiuojamas skirtumas tarp pirminio ir dabartinio vaizdo. Priklausomai nuo nustatyto slenksčio, nustatoma, ar įvyko reikšmingas skirtumas tarp dviejų vaizdų bei vadovaujantis

patikrinimo logika, kuri fiksuoja objektus tik tuomet, kai jie atpažįstami (pvz. Žmogus arba gyvūnas), į konsolę išvedamas pranešimas su objekto informacija: *print(f"TIKRAS OBJEKTAS - {label}")*.

Pranešimų su užfiksuotą judesiu bei atpažintu objektų siuntimui darbe naudojamas RPi *SIM800 GSM GPRS* modulis.

Šis modulis yra pritaikytas RPi sąsajai, kuri suderinama su SIM800 keturių dažnių *GSM / GPRS / BT* moduliu. Moduliui siunčiamos AT (angl. *Attention commands*) komandos naudojant RPi nuoseklųjį prievadą, su jo pagalba gali būti realizuotos tokios funkcijos kaip skambinimas ir atsiliepimas į skambučius, pranešimų siuntimas ir priėmimas bei naršymas internete. Pajungus SIM800 modulį, komunikacija įgyvendinau su GPIO (angl. *General Purpose Input/Output*).

Tuomet, su vaizdo atpažinimo funkcijos pagalbą nustatomi aktualius vaizdo atpažinimo objektai pvz. Žmogus. Kad pašalinti vaizdo transliavimo delsimą ~30 sek, iš pradžių įvedama logika 5k/s, tačiau norint pašalinti pastovaus vaizdo apdorojimo delsimą, nustatoma, kad MSE algoritmas apdorotą vaizdą su principu, kai paimamas paskutinis aktualus kadras (tai arba pirmas kadras paleidus sistemą, arba paskutinis kadras prieš tai, kai buvo pakitimas vaizdo kadre) ir jis lyginamas su kitais toliau einančiais kadrų, kad sistema jų neapdorotų papildomai bei nebūtų vaizdo apdorojimo vėlavimo. Jeigu MSE algoritmas suveikia, paleidžiamas vaizdo įrašo apdorojimas ir tikrinama vaizdo pakitimo priežastis pvz. Vaizde atsiradęs žmogus, gyvūnas ar kitas mums neįdomus objektas. Jeigu apdorojant vaizdą nustatoma, kad aptiktas mus dominantis objektas, sistema automatiškai su SIM800 modulio pagalba siunčia trumpąją SMS žinutę apie užfiksuotą objektą arba MMS žinutę su užfiksuotu objektu.

Naudojant SIM800 modulį, komunikacija vyksta su AT komandų pagalba. Kad išsiųsti vaizdą su užfiksuotu objektu pirmiausia yra suvedami reikiami duomenys, tokie kaip: SIM kortelės mobiliojo ryšio *mmsc_url*, *mms_proxy*, *port* (prievadas) APN tinklas.

```
# MMS ir interneto nustatymai
mms_settings = {
    "profile_name": "TELE2 MMS",
    "apn": "internet.tele2.lt",
    "mmsc": "http://mmsc.tele2.lt", #be "http://"
    "proxy": "mmsproxy.tele2.lt",
    "proxy_ip": "192.12.40.29", #tarpinis serveris
    "port": "8080" # Galima naudoti 9201 arba 8080
}

internet_settings = {
    "profile_name": "TELE2 internetas",
    "apn": "internet.tele2.lt",
    "apn_ip": "10.62.22.193",
    "port": "", # Turi būti tuščia
    "user": "wap",
    "pass": "wap"
}
```

3.4 pav. SIM kortelės konfigūracija

Šiuos duomenis su AT komandų pagalba išsiunčiame tiesiogiai iš RPI į SIM800 modulį per GPIO jungtį ir sutikriname dėl galimai pasitaikančių klaidų.

```
# Nustatyti MMS centrą
ser.write(f'AT+CMMSURL="{mms_settings["mmsc"]}"\r'.encode())
time.sleep(1)
response = check_response(ser)
print(response)

# Nustatyti tinklo parametras (bearer context ID)
ser.write(b'AT+CMMSID=1\r')
time.sleep(1)
response = check_response(ser)
print(response)

# Nustatyti MMS proxy ir prievadą
ser.write(f'AT+CMMSPROTO="{mms_settings["proxy_ip"]}",{mms_settings["port"]}\r'.encode())
time.sleep(1)
response = check_response(ser)
print(response)

# Nustatyti ryšio tipą
ser.write(b'AT+SAPBR=3,1,"Contype","GPRS"\r')
time.sleep(1)
response = check_response(ser)
print(response)

# Nustatyti APN
ser.write(f'AT+SAPBR=3,1,"APN",{internet_settings["apn"]}\r'.encode())
time.sleep(1)
response = check_response(ser)
print(response)

# Nustatyti vartotojo vardą
ser.write(f'AT+SAPBR=3,1,"USER",{internet_settings["user"]}\r'.encode())
time.sleep(1)
response = check_response(ser)
print(response)

# Nustatyti slaptažodį
ser.write(f'AT+SAPBR=3,1,"PWD",{internet_settings["pass"]}\r'.encode())
time.sleep(1)
response = check_response(ser)
print(response)
```

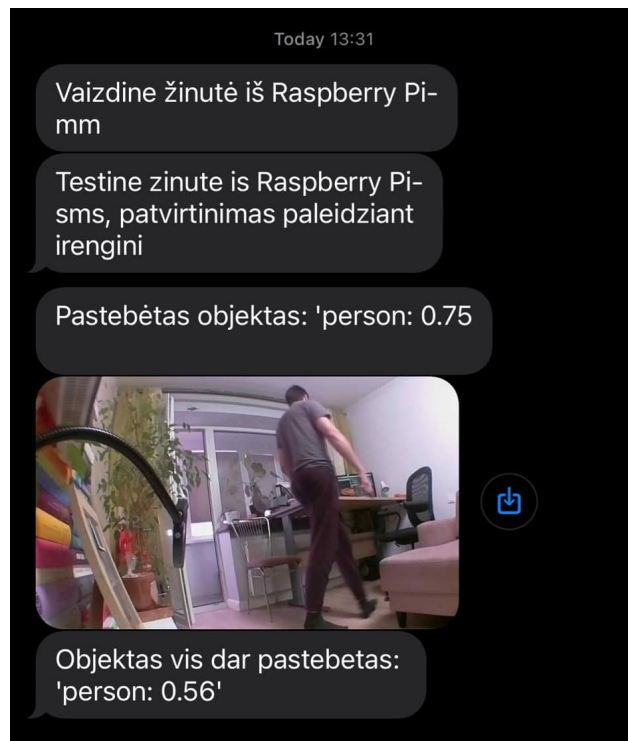
3.5 pav. AT pagrindinės komandos

Neaptikus klaidų sistema įkelia teigiamą atsaką – OK. Tuomet apdorojus ir nusistačius reikiamus parametrus su AT komandos pagalba į modulį yra įkeliamas siunčiamo pranešimo turinys bei vaizdinė nuotrauka bin64 formatu. Sugeneravus pranešimą yra vykdomas MMS žinutės siuntimas gavėjui.

```
Skriptas sėkmingai paleistas
Nustatome SMS režimą...
Atsakymas po SMS režimo nustatymo: AT+CMGF=1
OK
Siunčiame SMS numerį...
Atsakymas po SMS numerio siuntimo: AT+CMGS="+37062306540"
>
Siunčiame SMS tekstą...
Atsakymas iš SIM800 po SMS siuntimo: Testine zinuote is Raspberry Pi-sms
>
```

3.6 pav. Komandinės eilutės atsakas

Galiausiai naudotojas gauna MMS vaizdinę nuotrauką su tam tikru pranešimu apie užfiksuotą objektą.



3.7 pav. MMS žinutės gavimas su aptiktu vaizdu

Pagrindinis failas yra pavadintas „vaizdas.py“, kuris vykdo vaizdo apdorojimą kartu su MSE ir 5k/s logika. *vaizdas.py*, kai gaunamas pranešimas jog objektas yra tikras, per *mms_siuntimas.py* failą išsiunčia 1 MMS žinutę kas 2 minutes (laiką galima pakeisti). Aptikus vaizdą toliau kas 20 sek siunčia SMS žinutę jog objektas vis dar yra kadre – taip taupant naudotojo pinigines lėšas. *mms_siuntimas.py* failas apdoroja AT komandas, kurios susijusios su MMS pranešimu siuntimu, taip pat įrašo tekstą (koks objektas yra pastebėtas), bei išsiunčia vaizdine nuotrauka. *sms_siuntimas.py* yra paleidžiamas tik tuo metu, kol nėra siunčiamos MMS žinutes arba pasikeitus objektui. MMS konfigūracijas turime paleisti tik 1 kartą (paleidžiant programą), jų nereikia įvedinėti kiekvieną kartą siunčiant, išsiuntimo metu kinta tik vaizdinė nuotrauka ir tekstas.

3.2. Sistemos autonomiškumas

Šiame skyrelyje bus siekiama realizuoti projektuojamos sistemos autonomiškumą bei atlikti bandymus ir sistemos energijos optimizavimo veiksmus, kurių dėka bus siekiama kuo labiau prailginti sistemos veikimo laiką.

3.2.1. Pirmasis sistemos veikimo bandymas

Tyrimui buvo naudojami 4 ličio jonų akumuliatoriai *Li-ion 18650*, kuriems baterijų energijai sujungti ir kartu naudoti naudojamas *Li-ion* ličio baterijų įkrovimo modulis bei DC/DC įtampos keitiklis, naudojamas sumažinti įtampą iki 5 V, kuri yra RPi nominali darbinė galia.

Siekiant nustatyti akumuliatorių tūrį buvo naudojamas „test battery“ įrenginys, kuris veikia iškraunant/pakraunant baterijas per kelis ciklus, kad nustatyti akumuliatorių tūrį. Atlikus matavimus nustatyta, kad 4 akumuliatorių tūris yra apie 16953mAh.

3.1 lentelė. Pirmojo akumuliatorių tūrio vertės matavimo rezultatai

1 akumuliatorius 4,16V	04237mAh
2 akumuliatorius 4,17 V	04238mAh
3 akumuliatorius 4,16 V	04234mAh
4 akumuliatorius 4,17 V	04244mAh
Suma	16953mAh

Nustačius akumuliatorių tūrį, buvo paleidžiamas RPi su esama konfigūracija, kad nustatyti, kiek laiko veikia kamera su objektu atpažinimu pakraunama baterijomis. Bandymai atlikti optimaliomis sąlygomis uždaroje erdvėje, kurioje vyraavo 10-15°C temperatūra. Vidutiniškai valandos bėgyje fiksuojama buvo ~23 objektai. Atlikus pirmąjį bandymą nustatyta, kad RPi veikė 4 valandas.

RPi energijos poreikis paskaičiuotas padalinus mAh iš išdirbto laiko ir gauta 4238 mA. T.y. RPi su pradinėmis konfigūracijomis srovės sunaudojimas yra 4,238 A.

3.2.2. Antrasis sistemos veikimo bandymas

Atlikus pirmąjį bandymą, siekiant prailginti kameros veikimo laiką, buvo atlikti programinio kodo keitimai.

Pirmiausia buvo atjungtas MMS siuntimas t.y. nustatoma, kad bandymo metu paleidus MMS failą, programa jį apdorotų iki to momento kai yra *image* įkėlimas prieš MMS siuntimą, tačiau pačio MMS siuntimo sistema nevykdo, yra tik išsiunčiama SMS žinutė su patvirtinimu apie užfiksuotą objektą.

Kadangi kamera yra sumontuota vietoje, kuri nėra pastoviai prižiūrima, kad sužinoti tikslų kameros veikimo laiką nuspręsta užprogramuoti, kad paleidimo metu automatiškai būtų sukuriamas aplankalas, kuriame bus išsaugomos nuotraukos pavadinime fiksuojant datą ir laiką. Taip pat laiko užfiksavimui buvo panaudojami *log'ai* (3.8 ir 3.9 pav.).

```

2025-04-01 10:35:18 ## Programa paleista
2025-04-01 10:37:12 ## MMS - apdorojus - image_10-37-10.jpg
2025-04-01 10:37:18 ## SMS-po-MMS išsiustas - image_10-37-10.jpg
2025-04-01 10:40:37 ## Programa paleista
2025-04-01 10:43:13 ## MMS - apdorojus - image_10-42-41.jpg
2025-04-01 10:43:19 ## SMS-po-MMS išsiustas - image_10-42-41.jpg
2025-04-01 10:47:14 ## MMS - apdorojus - image_10-46-52.jpg
2025-04-01 10:47:49 ## SMS-po-MMS išsiustas - image_10-46-52.jpg
2025-04-01 10:52:22 ## MMS - apdorojus - image_10-52-00.jpg

```

3.8 pav. Log'ai sistemos paleidimo metu

```

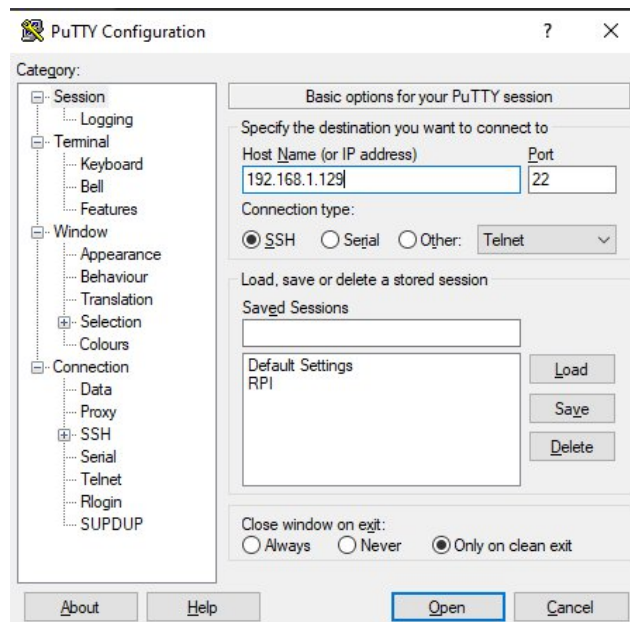
2025-04-15 07:15:38 ## MMS - apdorojus: image_07-15-04.jpg | Aptikta: person: 0.28
2025-04-15 07:15:44 ## SMS-po-MMS išsiustas: image_07-15-04.jpg | Aptikta: person: 0.28
2025-04-15 07:15:56 ## SMS išsiustas: image_07-15-51.jpg | Aptikta: person: 0.28
2025-04-15 07:19:53 ## MMS - apdorojus: image_07-19-18.jpg | Aptikta: person: 0.29
2025-04-15 07:20:18 ## SMS-po-MMS išsiustas: image_07-19-18.jpg | Aptikta: person: 0.29
2025-04-15 07:24:18 ## Sistema veikia - 5 min nuo paskutinio įrašo
2025-04-15 07:29:18 ## Sistema veikia - 5 min nuo paskutinio įrašo
2025-04-15 07:30:40 ## MMS - apdorojus: image_07-30-18.jpg | Aptikta: person: 0.29
2025-04-15 07:31:16 ## SMS-po-MMS išsiustas: image_07-30-18.jpg | Aptikta: person: 0.29
2025-04-15 07:35:18 ## Sistema veikia - 5 min nuo paskutinio įrašo

```

3.9 pav. Log'ai sistemos veikimo metu

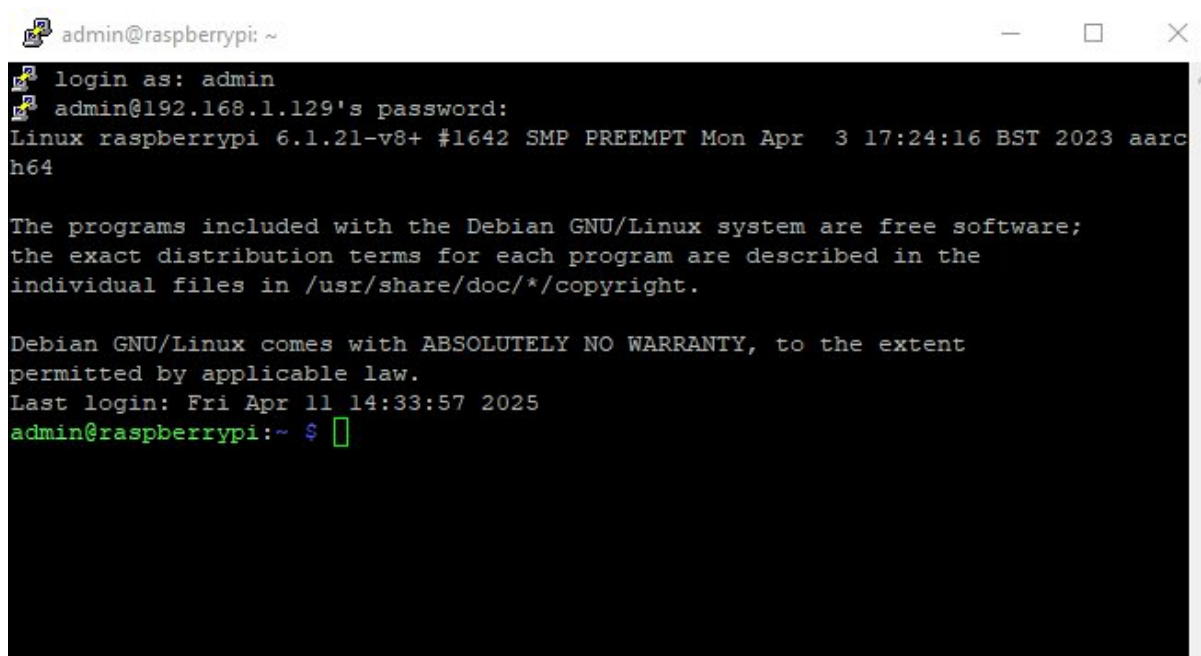
Taip pat, siekiant prailginti sistemos veikimo laiką nuspręsta perjungti RPI GUI vartotojo sąsaja į *Raspberry Pi OS Lite* versiją. Sėkmingai pašalinus GUI komponentus, bei išvalius nereikalingus paketus sistema buvo perkrauta. Tai atlikus turėtų pagerėti resursų naudojimas, nes sistema veiks be grafines sąsajos.

Kad įgalinti RPi komunikaciją iš kito kompiuterio įjungiamas SSH. RPi konfigūracijose nustatomas pagrindinio kompiuterio pavadinimas (IP adresas), kuris yra 192.168.1.129. Tuomet vykdomas prisijungimas naudojant nuotolinio serverių valdymo ir duomenų perdavimo atviro kodo programa *PuTTY* (3.10 pav.).



3.10 pav. Prisijungimas prie RPi iš kito kompiuterio naudojant *PuTTY*

Įvedus gautą IP adresą prisijungiama naudojant vartotojo vardą ir slaptažodį (3.11 pav.).

A terminal window titled 'admin@raspberrypi: ~' with standard window controls. The terminal shows a login sequence: 'login as: admin', 'admin@192.168.1.129's password:', followed by system information including 'Linux raspberrypi 6.1.21-v8+ #1642 SMP PREEMPT Mon Apr 3 17:24:16 BST 2023 aarch64'. It then displays the Debian GNU/Linux license text and the last login time 'Fri Apr 11 14:33:57 2025'. The prompt 'admin@raspberrypi:~ \$' is shown with a cursor.

```
admin@raspberrypi: ~
login as: admin
admin@192.168.1.129's password:
Linux raspberrypi 6.1.21-v8+ #1642 SMP PREEMPT Mon Apr 3 17:24:16 BST 2023 aarch64

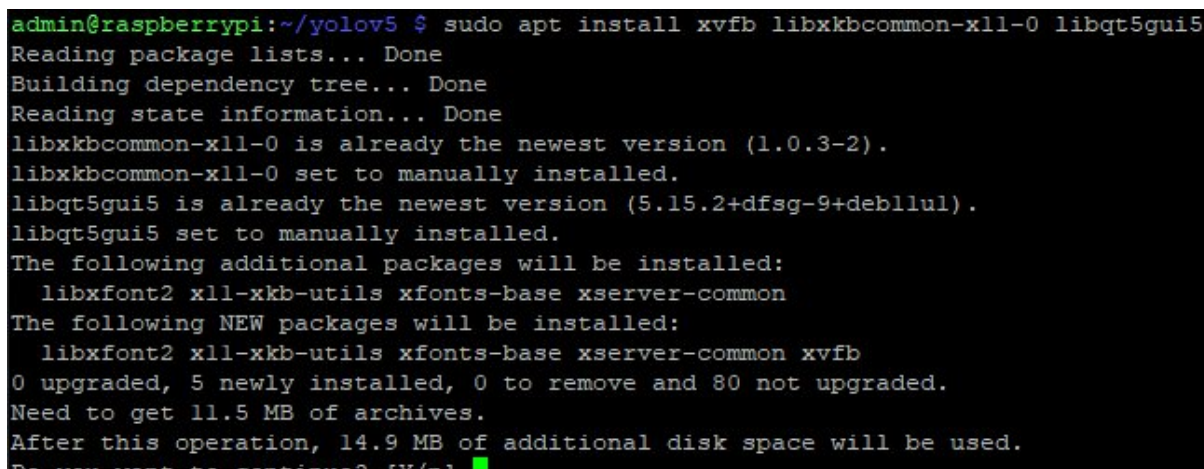
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Apr 11 14:33:57 2025
admin@raspberrypi:~ $
```

3.11 pav. Prisijungimas prie RPi iš kito kompiuterio

Kad galima būtų prisijungti prie RPi bet kur ir bet kada sukurtas statinis IP adresas 192.168.1.129. Tai darant būtina įsitikinti kad maršrutizatoriuje IP adresas nebūtų užimtas bei turėtų galimybę prie jo prisijungti per WiFi.

Kadangi prieš tai naudota *Python* programa naudojo fizinį ekraną, suinstaliuojamas papildomas virtualus ekranas, su minimaliu X11 palaikymu.

A terminal window showing the command 'sudo apt install xvfb libxkbcommon-x11-0 libqt5gui5' and its output. The output indicates that libxkbcommon-x11-0 and libqt5gui5 are already installed, and lists additional packages to be installed: libxfont2, x11-xkb-utils, xfonts-base, and xserver-common. It also shows the disk space requirements for the installation.

```
admin@raspberrypi:~/yolov5 $ sudo apt install xvfb libxkbcommon-x11-0 libqt5gui5
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
libxkbcommon-x11-0 is already the newest version (1.0.3-2).
libxkbcommon-x11-0 set to manually installed.
libqt5gui5 is already the newest version (5.15.2+dfsg-9+deb11u1).
libqt5gui5 set to manually installed.
The following additional packages will be installed:
  libxfont2 x11-xkb-utils xfonts-base xserver-common
The following NEW packages will be installed:
  libxfont2 x11-xkb-utils xfonts-base xserver-common xvfb
0 upgraded, 5 newly installed, 0 to remove and 80 not upgraded.
Need to get 11.5 MB of archives.
After this operation, 14.9 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

3.12 pav. Virtualaus ekrano prijungimas

Tuomet, paleidžiama programėle ir tikrinama ar viskas pasileidžia, kaip priklauso.

Kad sutaupyti energijos ir RPi veiktų dar ilgiau, nusprendžiama naudoti YOLOv5n su *MobileNet TFLite* integracija. Tai įgyvendinat instaliuojama biblioteka *pip install tensorflow opencv-python numpy*.

Kode teko pakeisti modelio interferencijos dalį (iš *PyTorch* į *TFLite*), visas likęs funkcionalumas (MMS/SMS, log'ų rašymas, bei katalogų kūrimas) išliko nepakitę.

Šio būdo dėka *TFLite* modelis yra optimizuotas RPi (*int8 kvantizacija*), naudojami 4 branduoliai (*num_threads=4*), modelis užima tik ~1MB atminties, išlaikomas efektyvus kadrų analizės metodas su MSE. Tuomet eksportuojama YOLOv5n versija į *TFLite* konsolėje.

Įjungiamo funkcija, kad prijungus akumulatorius prie RPi, jis pats paleistų failą *vaizdas.py* (Pav. 4.18).

```
admin@raspberrypi:~ $ sudo nano /etc/systemd/system/yolov5_vaizdas.service
```

3.13 pav. Vaizdo paleidimo kartu su akumulatoriaus prijungimu komanda

Norint patikrinti ar viskas suveikė tinkamai, atliekamas sistemos testavimas:

```
admin@raspberrypi:~ $ journalctl -u yolov5_vaizdas -f
-- Journal begins at Wed 2025-01-29 11:33:10 EET. --
Apr 13 22:31:15 raspberrypi systemd[1]: Started YOLOv5 vaizdo analizė su Xvfb.
Apr 13 22:32:29 raspberrypi xvfb-run[23774]: INFO: Created TensorFlow Lite XNNPA
CK delegate for CPU.
```

3.14 pav. Testuojamas vaizdo paleidimas

Bei išjungiamas sistemos užmigimo režimas:

```
admin@raspberrypi:~ $ sudo systemctl mask sleep.target suspend.target
Created symlink /etc/systemd/system/sleep.target → /dev/null.
Created symlink /etc/systemd/system/suspend.target → /dev/null.
```

3.15 pav. Sistemos miego režimo išjungimas

Atlikus visas sistemos konfigūracijas atliekamas antrasis bandymas. Antrojo bandymo metu sistema veikė 12 valandų ir 31 minutę. Celių pakrovimo metu gauti rezultatai:

3.2 lentelė. Antrojo akumuliatorių tūrio vertės matavimo rezultatai

1 akumulatorius 4,16V	04263mAh
2 akumulatorius 4,17 V	04268mAh
3 akumulatorius 4,16 V	04273mAh
4 akumulatorius 4,17 V	04284mAh
Suma	17088mAh

Gautą rezultatą padalinus iš išdirbto laiko nustatyta, kad su dabartinėmis konfigūracijomis RPi kamera sunaudojo 1,367 A energijos. Vidutiniškai valandos bėgyje fiksuojami buvo ~33 objektai.

3.2.3. Trečiasis sistemos veikimo bandymas

Siekiant dar labiau pagerinti gautą rezultatą, buvo nuspręsta dar patobulinti esamą sistemą. Kad nustatyti energijos taupymo režimą t.y. sumažinti CPU apkrovą buvo panaudota *sudo apt install cpufrequtils* ir *sudo cpufreq-set -g powersave* komanda. Išjungiamos tokios RPi funkcijos, kurios nėra naudojamos kameros veikimo metu, kad sutaupyti galios: internetas, spausdintuvo valdymas/mobilusis *Ethernet* 4G 5G, modemų valdymas (USB), GPIO mygtukų valdymas, *Windows RDP* prijungimas, taip pat sustabdomi nereikalingi *background* procesai, *Bluetooth*, nuotolinio darbalaukio priėjimą VNC, atsitiktinių skaičių generavimas, LED lemputės veikimas, HDMI signalą, garso modulis.

Taip pat taupant energiją sumažinama GPU atmintis bei sisteminiai laikmačiai ir *Python time/datetime* laikmačio dažnis. Tai padarius gali kilti laiko netikslumo problema, tačiau funkcija *datetime.now()* nėra reikšminga, o suvartoja apie 16-583 mW.

Atliekant bandymus pastebėta, kad gautose rezultatuose ne visuomet galima atskirti, kur yra pastebėtas objektas. Todėl nuspręsta įdiegti naują objekto atpažinimo funkcionalumą – objektų žymėjimą.

Tam tikslui atliekami MSE algoritmo pakeitimai. Nustatoma, kad pirmiausia, kai yra aptinkamas objektas jis yra siunčiamas į *capture_image* funkciją, kad toliau apdoroti. Nuotrauka apdorojama ir sukuriamą pažymėta nuotrauka – kurioje yra pažymėtas atpažintas objektas, ji kartu su originalia nuotrauka išsaugomas tos dienos kataloge, tai leidžia įsitikinti jog DI nesuklydo ir pažymėjo tinkamą objektą.

```

def capture_image(frame, boxes=None, scores=None, classes=None, class_names=None):
    """Išsaugome gautą vaizdą su pažymėtais objektais"""
    folder_path = get_current_date_folder()
    timestamp = datetime.now().strftime("%H-%M-%S")

    if not send_lock.locked():
        # Originalus vaizdas (visada išsaugomas)
        original_path = os.path.join(folder_path, f"image_{timestamp}_original.jpg")
        cv2.imwrite(original_path, frame)

        # Jei yra aptikimų (jau patikrinta pagrindiniame cikle)
        if boxes is not None and scores is not None and classes is not None:
            marked_path = os.path.join(folder_path, f"image_{timestamp}_marked.jpg")
            marked_frame = frame.copy()

            for i in range(len(scores)):
                cls = int(classes[i])
                # Tikriname ar klasės ID yra tinkamas
                if cls < len(class_names): # Pridėta apsauga
                    ymin, xmin, ymax, xmax = boxes[i]
                    xmin = int(xmin * marked_frame.shape[1])
                    xmax = int(xmax * marked_frame.shape[1])
                    ymin = int(ymin * marked_frame.shape[0])
                    ymax = int(ymax * marked_frame.shape[0])

                    cv2.rectangle(marked_frame, (xmin, ymin), (xmax, ymax), (0, 255, 0), 2)
                    label = f"{class_names[cls]}: {scores[i]:.2f}"
                    cv2.putText(marked_frame, label, (xmin, ymin-10),
                                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,255,0), 2)

            cv2.imwrite(marked_path, marked_frame)
            print(f"Vaizdai išsaugoti: {original_path} ir {marked_path}")
            return marked_path

        print(f"Originalus vaizdas išsaugotas: {original_path}")
        return original_path
    return None

```

3.16 pav. Objektų žymėjimo gautose nuotraukose įdiegimas

Atlikus šiuos nežymius pakeitimus tapo šiek tiek lengviau nustatyti, ar kamera fiksuojant objektą tikrai nesuklysta bei aiškiai pastebėti objektą kadre. Objekto atpažinimas bei pažymėjimas pavaizduoti 3.17 paveiksle.



3.17 pav. Originalus vaizdas bei apdorotas vaizdas su objekto žymėjimu

Atlikus šiuos energijos taupymo veiksmus buvo paleistas RPi bei gautas rezultatas 14 valandų ir 36 minutės. Trečiojo celių pakrovimo metu gautos vertės:

3.3 lentelė. Trečiojo akumuliatorių tūrio vertės matavimo rezultatai

1 akumulatorius 4,16V	04263mAh
2 akumulatorius 4,17 V	04268mAh
3 akumulatorius 4,16 V	04273mAh
4 akumulatorius 4,17 V	04284mAh
Suma	17088mAh

Gautas analogiškas celių matavimo rezultatas, kaip po antrojo bandymo, tačiau paskaičiavus gautą rezultatą, nustatyta, kad RPi sunaudojo 1,178 A energijos. Vidutiniškai valandos bėgyje fiksuojama buvo ~47 objektai.

3.2.4. Ketvirtasis sistemos veikimo bandymas

Po trečiojo bandymo nuspręsta padaryti galutinius keitimus bei paleisti sistemą. Pakeista kodo struktūra: nenaudojamas dinamiškas FPS keitimas, tai yra labiau energetiškai efektyvus variantas su paprastesne logika. Optimizuotas CPU *governor* – į *powersave* režimą su komanda `os.system("echo powersave | sudo tee /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor")` ir USB valdymas (nustatytas automatinis valdymas – kai veikia tik tuomet, kai įjungiamo jungtis į USB).

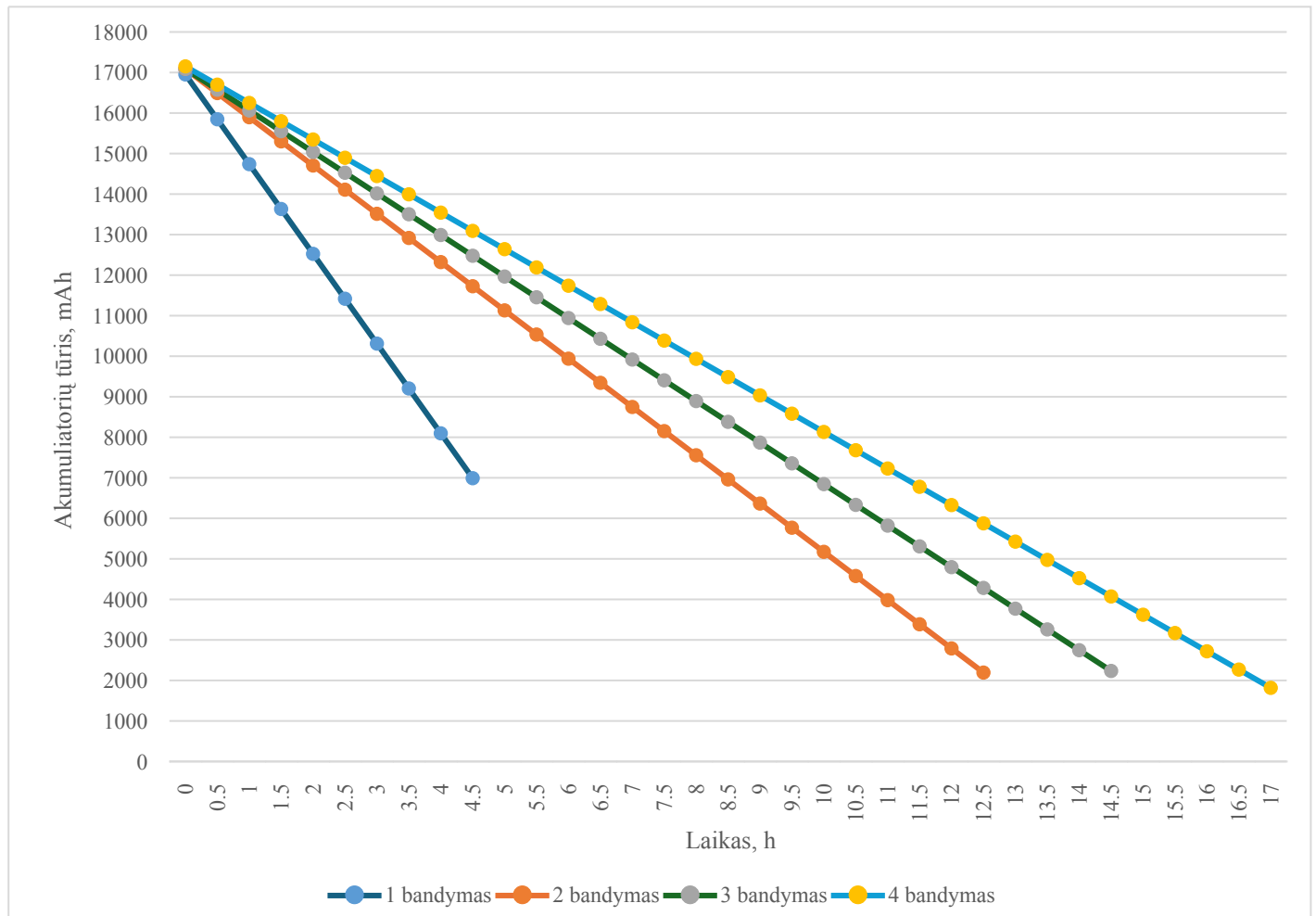
Vėliau siekiant pagerinti CPU veikimą nustatytas maksimalus CPU dažnis. Kad tai atlikti, pirmiausia su komanda `cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_available_frequencies` patikrinti RPi modulio galimi dažniai. Panaudota *stress-ng* komanda, kurios pagalba atliktas 30 sekundžių testas 1 branduoliui *stress-ng --cpu 1 --timeout 30 --metrics*, bei tuo pačiu metu stebėta RPi temperatūra *watch -n 2 vcgencmd measure_temp*. Pagal gautus duomenis pasirinktas optimalus CPU dažnis kuris yra 800MHz.

Modelio apkrovos sumažinimui, optimizuotas *TFLite* modelis iki 2 branduolių (*num_threads=2*). SIM800 modulis įvedamas į miego režimą, prieš siunčiant MMS/SMS komandas, yra išsaukiamas modulis su tuščia komanda ir tada išjungiamas modulio *powersave* režimas. Kad pažadinti modulį naudojama `ser.write(b'\r\n')` su komanda `time.sleep(0.5)` paliekamas laiko tarpas, kad įsitikinti, kad sistema tikrai pabudo bei siunčiama komanda `ser.write(b'AT+ CSCLK=0\r\n')`, kad išsijungtų miego režimas. Tuomet išsiuntus pranešimą sistema vėl siunčiama į miego režimą su komanda `ser.write(b'AT+ CSCLK=1\r\n')` taip taupant įrenginio energiją.

Atlikus paskutiniuosius energijos taupymo veiksmus buvo paleistas RPi bei gautas rezultatas 17 valandų ir 06 minutės. Gautas analogiškas celių matavimo rezultatas, kaip po ketvirtojo bandymo, tačiau

paskaičiavus gautą rezultatą, nustatyta, kad RPi sunaudojo 1,036 A energijos. Vidutiniškai valandos bėgyje fiksuojama buvo ~40 objektų.

Atlikus 4 bandymus gauti duomenys pavaizduoti 3.18 paveiksle.



3.18 pav. Akumuliatoriaus tūrio mažėjimo laiko atžvilgiu grafikas

Pagal gautus duomenis matome, kad atliekant 1-ąjį bandymą, RPi išsikrovė ankščiau nei tikėtasi, nors akumuliatoriaus likutis (mAh) dar buvo pakankamas veikimui. Tai galėjo atsitikti dėl to, kad įvyko trumpalaikis srovės šuolis bei akumuliatorius negalėjo tiekti pakankamai srovės, įtampa pradėjo greitai kristi bei RPi išsijungė net turint pakankamai akumuliatoriaus likučio darbui. Atliekant pirmąjį bandymą RPi turėjo didžiausią energijos suvartojimą ~2214 mAh/h, tai įtakojo didelę įrenginio apkrova, kuomet veikė daug neefektyvių procesų.

Atlikus antrąjį bandymą matomas geresnis RPi veikimas, mikrokompiuteris veikė 12,5 h, vidutiniškai suvartodamas per 1 valandą ~1191 mAh. Šis rezultatas pasiektas perjungiant RPi GUI vartotojo sąsaja į *Raspberry Pi OS Lite* versiją bei įdiegiant YOLOv5n su *MobileNet TFLite* integracija.

Trečiasis bandymas gavosi su lėtu ir stabiliu išsikrovimu. Sistema veikė 14,5 valandos vidutiniškai per vieną valandą suvartodama ~995 mAh. Tai pasiekta nustačius energijos taupymo režimą bei atjungus nereikalingą funkcionalumą bei pridėjus objektų pažymėjimą kadre, kuris neįtakojo energijos suvartojimo.

Ketvirtasis ir bene labiausiai energetiškai efektyvus bandymas, kai sistema veikė 16,5 valandos bei vidutiniškai per valandą sunaudojo ~903 mAh, gavosi po to, kai buvo optimizuotas veikimo dažnis bei pagerintas energijos taupymo režimas.

3.3. Suprojektuotos sistemos aktualumas

Autonominės vaizdo stebėjimo sistemos su integruotu objektų atpažinimu yra itin aktualios šiuolaikiniame pasaulyje, kur didėja poreikis efektyviai, ekonomiškai ir automatiškai valdyti saugumo, stebėjimo ar pramonės procesus. Sukurtos sistemos pagrindas – RPi mikrovaldiklis, *Raspbian* OS ir *Python* programavimo kalba – leidžia sukurti lengvai pritaikomą, modulinį sprendimą, kuris gali būti naudojamas įvairiose srityse: nuo namų apsaugos iki gyvūnų stebėjimo sunkiai prieinamose vietose arba sudėtingomis sąlygomis.

Sistemos, veikiančios nepriklausomai nuo išorinių serverių, maitinimo ar pastovios priežiūros yra ypač svarbios vietovėse su ribotu tinklo ryšiu ir sudėtingu pasiekiamumu. Optimizuotas veikimo laikas (nuo 4 val. iki 17 val.) patvirtina, kad RPi su tinkamai sukonfigūruotu programiniu kodu gali dirbti ilgai net su ribotais resursais, kai energijos vartojimo optimizavimas pasiektas per eksperimentinį kodavimą, modelių supaprastinimą ir fono procesų valdymą. Objektų aptikimas ir pranešimų siuntimas SMS/MMS formatu užtikrina momentinį informavimą apie įvykius, kas kritiška apsaugos sistemose.

Suprojektuota sistema yra aktuali tuo, kad ji fiksuoja objektą dideliu atstumu (~60 m.) bei ją pritaikius su DI pagalba galima atpažinti bet kokią dominantę objektą. Sistema savo paprastumu ir efektyvumu gali būti pritaikyta galybėje projektų pvz., stebint retus gyvūnus, analizuojant aplinką bei saugant turtą, kad ir kur jis bebūtų. Kamera dėl jos sandaros galima lengvai užmaskuoti naudojant 3D technologijas.

3.4. Skyriaus išvados

1. Sukonfigūruota vaizdo stebėjimo sistema, naudojant *Raspberry Pi* mikrovaldiklį, veikiančią taikant *Raspbian* OS ir *Python* programavimo kalbą, kuri sugeba užfiksuoti objektą vaizdo sraute, jį atpažinti ir SMS arba MMS žinute informuoti naudotoją apie užfiksuotą įvykį.
2. Atlikti autonominės vaizdo stebėjimo sistemos su objektų atpažinimų bandymai derinant eksperimentinį ir kartotinį tobulinimą, siekiant kuo labiau optimizuoti sistemos veikimo laiką.

3. Įgyvendinant suprojektuotą sistemą, atlikti sistemos eksperimentai bei gautas rezultatas, kad autonomiškai veikiančios kameros su *Raspberry Pi* pagrindu bei objektų atpažinimu veikimo laikas pagerintas nuo 4 valandų iki 17 valandų.

IŠVADOS

1. Remiantis atlikta autonominių vaizdo stebėjimo sistemų analize, nustatyta pagrindinė šios srities problematika – jau egzistuojančios sistemos veikia nedideliu atstumu, bei veikia tik prisijungus prie bevielio tinklo, arba atpažįsta objektus ne pagal jų klasifikaciją DI, o pagal jų formą bei veikia su jutikliais, o tai pagerina jų veikimo laiką, tačiau tai neatlieka pagrindinės siekiamos užduoties – užfiksuoti objektą atokioje vietoje, kurioje neįmanoma prijungti belaidžio tinklo arba kur reikia tiksliai nustatyti analizuojamą objektą. Taip pat atlikta jau esamų sistemų analizė, poreikis, taikymo sritys bei aprašytos sistemos sudėtinės dalys ir kaip jos lemia energijos suvartojimą.
2. Vadovaujantis atliktu būsimos autonominės vaizdo stebėjimo sistemos su vaizdo atpažinimu projektavimu, paruoštos projektui įgyvendinti reikalingos užduotys, pavaizduotos UML diagramos, nustatyti sistemos funkciniai ir nefunkciniai reikalavimai, pagal kuriuos buvo atliktas tolesnis projektas. Parinkta projektui realizuoti reikalinga aparatinė įranga: *Raspberry Pi 4* mikrokompiuteris su *OV5647* kamera, *SIM800* moduliu ir keturi *Li-ion 18650* akumuliatoriai. Išanalizavus pasirinkta *Python* programavimo kalba ir *PyCharm* teksto redaktorius, kurie veikia *Rasbian* operacinės sistemos pagrindu, o vaizdo apdorojimo ir atpažinimo sistema veiks su *YOLOv5* modulio pagalba ir jame įdiegtomis bibliotekomis.
3. Įgyvendinant projektą buvo suprojektuota sistema, apimanti akumuliatoriais maitinamą *Raspberry Pi* pagrindu veikiančią kamerą, kuri sugeba atpažinti vartotoją dominantį objektą. Sistemos optimizacijos metu perjungus RPI GUI vartotojo sąsaja į *Raspberry Pi OS Lite* versiją bei įdiegus *YOLOv5n* su *MobileNet TFLite* integraciją ir įgyvendinus visus energijos taupymo sprendimus, sistemos autonomiškumą pavyko padidinti nuo 4 iki 17 valandų.

Ateities darbai

Atliekant tolesnius autonominės vaizdo stebėjimo sistemos su objektų atpažinimu tyrimus, norint užtikrinti didesnę sistemos autonomiškumą, reikėtų apsvarstyti galimybę naudoti didesnės talpos akumuliatorius arba integruoti saules paneles. Su dabartinėmis funkcijomis sistema yra maksimaliai optimizuota, pasiektas pakankamas veikimo laikas su pastoviu vaizdo stebėjimu, tačiau norint dar labiau prailginti sistemos veikimą rekomenduojama įdiegti IR bei judesio daviklius, bei siekiant gauti kuo realistiškesnius rezultatus reikėtų atsižvelgti į oro temperatūros svyravimus ir jų įtaką sistemos veikimo laikui.

LITERATŪROS SĄRAŠAS

- Arun Kumar. (2020). *10+ Best Python IDEs for Linux*. <https://www.fossilinux.com/27003/best-python-ides-for-linux.htm>
- Ben. (2023). *Java vs. Python: Coding Battle Royale*. <https://stackify.com/java-vs-python/>
- Botland.com.pl*. (2024). <https://botland.store/>
- Calavia, L., Baladrón, C., Aguiar, J. M., Carro, B., & Sánchez-Esguevillas, A. (2012). A semantic autonomous video surveillance system for dense camera networks in Smart Cities. *Sensors (Switzerland)*, 12(8), 10407–10429. <https://doi.org/10.3390/s120810407>
- Çintaş, E., Özyer, B., & Şimşek, E. (2020). Vision-Based Moving UAV Tracking by Another UAV on Low-Cost Hardware and a New Ground Control Station. *IEEE Access*, 8, 194601–194611. <https://doi.org/10.1109/ACCESS.2020.3033481>
- Corey Schafer. (2024). *Github.com*. <https://github.com/CoreyMSchafer>
- Elharrouss, O., Almaadeed, N., & Al-Maadeed, S. (2021). A review of video surveillance systems. *Journal of Visual Communication and Image Representation*, 77, 103116. <https://doi.org/10.1016/J.JVCIR.2021.103116>
- Fortes, S., Kulesza, R., & Li, J. J. (2021). A Case Study of Object Recognition from Drone Videos. *2021 4th International Conference on Information and Computer Technologies (ICICT)*, 84–87. <https://doi.org/10.1109/ICICT52872.2021.00021>
- Globaldev.tech. (2022). Top 13 Python web frameworks to learn in 2023. *Globaldev*. <https://globaldev.tech/blog/top-10-python-web-frameworks-to-learn>
- Hussain, S. J. (2015). *Smart Surveillance System using Thing Speak and Raspberry Pi*. <https://doi.org/10.17148/IJARCCE.2015.4749>
- Isarsoft. (2024). What is a Video Management System (VMS)? *KNOWLEDGE HUB ARTICLE*. [https://www.isarsoft.com/knowledge-hub/vms#:~:text=A%20Video%20Management%20System%20\(VMS\)%20is%20a%20software%2Dbased,scale%20surveillance%20and%20security%20monitoring](https://www.isarsoft.com/knowledge-hub/vms#:~:text=A%20Video%20Management%20System%20(VMS)%20is%20a%20software%2Dbased,scale%20surveillance%20and%20security%20monitoring)
- Kamami.pl*. (2024). <https://kamami.pl/en/>
- Kannan, S. (2020). Intelligent object recognition in underwater images using evolutionary-based Gaussian mixture model and shape matching. *Signal, Image and Video Processing*, 14(5), 877–885.

<https://doi.org/10.1007/s11760-019-01619-w>

- Karl Hughes. (2024). *GitHub vs. Bitbucket vs. GitLab - Help me decide*. <https://stackshare.io/stackups/bitbucket-vs-github-vs-gitlab>
- Khodadin, F., & Pudaruth, S. (2020). An intelligent camera surveillance system with effective notification features. *International Journal of Computing and Digital Systems*, 9(6), 1251–1261. <https://doi.org/10.12785/ijcds/0906022>
- Koorsen Fire & Security. (2022). WHAT IS A VIDEO SURVEILLANCE SYSTEM? *Koorsen Fire & Security*. <https://blog.koorsen.com/what-is-a-video-surveillance-system>
- Mahendru, M., & Dubey, S. K. (2021). Real Time Object Detection with Audio Feedback using Yolo vs. Yolo_v3. *2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, 734–740. <https://doi.org/10.1109/Confluence51648.2021.9377064>
- Maheswari, M., Josephine, M. S., & Jeyabalaraja, V. (2022). Customized deep neural network model for autonomous and efficient surveillance of wildlife in national parks. *Computers and Electrical Engineering*, 100, 107913. <https://doi.org/10.1016/J.COMPELECENG.2022.107913>
- Mahipal Nehra. (2019). *Top 5 Python Frameworks for Web Development in 2020*. <https://www.decipherzone.com/blog-detail/top-python-frameworks>
- Nathan Segal. (2016). *10 Common Uses of Bootstrap*. <https://www.htmlgoodies.com/html5/10-common-uses-of-bootstrap/>
- Park, H., Ham, S.-H., Kim, T., & An, D. (2022). Object Recognition and Tracking in Moving Videos for Maritime Autonomous Surface Ships. *Journal of Marine Science and Engineering*, 10(7). <https://doi.org/10.3390/jmse10070841>
- Park, J.-H., & Kang, C. (2020). A Study on Enhancement of Fish Recognition Using Cumulative Mean of YOLO Network in Underwater Video Images. *Journal of Marine Science and Engineering*, 8(11). <https://doi.org/10.3390/jmse8110952>
- Paul, M., Haque, S. M. E., & Chakraborty, S. (2013). Human detection in surveillance videos and its applications - a review. *EURASIP Journal on Advances in Signal Processing*, 2013(1), 176. <https://doi.org/10.1186/1687-6180-2013-176>
- PYPL PopularitY of Programming Language*. (2024). <https://pypl.github.io/PYPL.html>
- Rezaei, M., Azarmi, M., & Mir, F. M. P. (2023). 3D-Net: Monocular 3D object recognition for traffic monitoring.

- Expert Systems with Applications*, 227, 120253. <https://doi.org/10.1016/J.ESWA.2023.120253>
- Sabater, A., Montesano, L., & Murillo, A. C. (2020). Robust and efficient post-processing for video object detection. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 10536–10542. <https://doi.org/10.1109/IROS45743.2020.9341600>
- Sadollah, A., & Sinha, T. (2020). *Recent Trends in Computational Intelligence*. IntechOpen. <https://books.google.lt/books?id=MJn8DwAAQBAJ>
- Šarūnas Navickas. (2024). *Paprastas puslapis su Flask*. <https://www.griaustinis.lt/video/python-paprastas-puslapis-su-flask>
- Schiano, F., Natter, D., Zambrano, D., & Floreano, D. (2022). Autonomous Detection and Deterrence of Pigeons on Buildings by Drones. *IEEE Access*, 10, 1745–1755. <https://doi.org/10.1109/ACCESS.2021.3137031>
- Sharma, P., Gupta, S., Vyas, S., & Shabaz, M. (2023). Retracted: Object detection and recognition using deep learning-based techniques. *IET Communications*, 17(13), 1589–1599. <https://doi.org/https://doi.org/10.1049/cmu2.12513>
- SQLite*. (2024). <https://www.sqlite.org/index.html>
- Szymak, P., & Gasiorowski, M. (2020). Using pretrained alexnet deep learning neural network for recognition of underwater objects. *Nase More*, 67(1), 9–13. <https://doi.org/10.17818/NM/2020/1.2>
- Tsakanikas, V., & Dagiuklas, T. (2018). Video surveillance systems-current status and future trends. *Computers & Electrical Engineering*, 70, 736–753. <https://doi.org/10.1016/J.COMPELECENG.2017.11.011>
- Verkada. (n.d.). Secure Remote Access. *Verkada Overview, Consideration & FAQs*. Retrieved January 29, 2024, from <https://info.verkada.com/surveillance-features/secure-remote-access/>
- Xiang, Z., Seeling, P., & Fitzek, F. H. P. (2021). You Only Look Once, But Compute Twice: Service Function Chaining for Low-Latency Object Detection in Softwarized Networks. *Applied Sciences*, 11(5). <https://doi.org/10.3390/app11052177>
- Zhang, M., Shi, R., & Yang, Z. (2020). A critical review of vision-based occupational health and safety monitoring of construction site workers. *Safety Science*, 126, 104658. <https://doi.org/https://doi.org/10.1016/j.ssci.2020.104658>