ДВНЗ «Донецький національний технічний університет»

Навчально-науковий інститут комп'ютерно-інформаційних технологій та автоматизації

(повне найменування інституту, назва факультету)

Кафедра автоматики та телекомунікацій

(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри автоматики та телекомунікацій
_____ Валерій ПОЦЕПАЄВ
(підпис) (ім'я та прізвище)
«___» _____ 2025 р.

Випускна кваліфікаційна робота

бакалавра

(освітньо-кваліфікаційний рівень)

на тему «Підвищення ефективності динамічних сонячних панелей з застосуванням технології машинного зору»

Виконав студент 4 курсу, групи АКТ-21

(шифр групи)

спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології

(шифр і назва спеціальності)

Набока Ярослав Геннадійович

(Прізвище, Ім'я та По-батькові) (підпис)

Керівник зав. каф. АТ, к.т.н., доц. _____ Валерій ПОЦЕПАЄВ

(посада, науковий ступінь, вчене звання, прізвище та ім'я) (підпис)

Рецензент к.т.н., доц., доц. каф. ЕТіКІ _____ Ганна ШЕЇНА

(посада, науковий ступінь, вчене звання, прізвище та ім'я) (підпис)

Нормоконтроль:
Дар'я ЖУКОВСЬКА
12.06.2025 р.

Засвідчую, що у цій випускній кваліфікаційній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Дрогобич – 2025 р.

ДВНЗ «Донецький національний технічний університет»
Навчально-науковий інститут комп'ютерно-інформаційних технологій та автоматизації
(повне найменування інституту, назва факультету)
Кафедра автоматики та телекомунікацій
(повна назва кафедри)

Захист відбувся _____
(дата)
з оцінкою_____
Секретар ЕК _____
(підпис)

Випускна кваліфікаційна робота

бакалавра

Тема: «Підвищення ефективності динамічних сонячних панелей з застосуванням технології машинного зору»

Виконавець, студент
гр. АКТ-21 _____ Ярослав НАБОКА
(підпис, дата, Ім'я ПРІЗВИЩЕ)
Керівник _____ Валерій ПОЦЕПАЕВ
(підпис, дата, Ім'я ПРІЗВИЩЕ)
Консультанти: _____ Гліб СТУПАК
(підпис, дата, Ім'я ПРІЗВИЩЕ)
_____ Олександр СЕРГІЄНКО
(підпис, дата, Ім'я ПРІЗВИЩЕ)

Нормоконтроль _____ Дар'я ЖУКОВСЬКА
(підпис, дата, Ім'я ПРІЗВИЩЕ)

Дрогобич – 2025 р.

Державний вищий навчальний заклад

"Донецький національний технічний університет"

Навчально-науковий інститут комп'ютерно-інформаційних технологій та автоматизації

Кафедра Автоматика та телекомунікації

Освітньо-кваліфікаційний рівень бакалавр

Галузі знань 15 Автоматизація та приладобудування
(шифр і назва)

Спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри АТ
Валерій ПОЦЕПАЄВ
«____» _____2025 року

З А В Д А Н Н Я
НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

Набоки Ярослава Геннадійовича
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Підвищення ефективності динамічних сонячних панелей з застосуванням технології машинного зору
керівник проекту (роботи) Поцепаєв Валерій Валерійович к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від 11.04.2025 року № 115
2. Строк подання студентом проекту (роботи)_____

3. Вихідні дані до проекту (роботи)_____
Технічна документація та матеріали з переддипломної практики

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Аналіз існуючих динамічних сонячних панелей. Теоритичне обгрунтування доцільності застосування машинного зору. Розробка системи трекінгу на основі машинного зору. Експерементальні дослідження.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Схема моделі двох осьової динамічної сонячної панелі.Електрична схема підключення.Повністю зібрана сонячна панель.Приклад аналізу зображення функцією find_brightest_area. Графік потужності (Вт) динамічної панелі протягом дня.

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1,4	Поцепаєв В.В., зав. каф. АТ		
2,3	Ступак Г.В., ст. викладач каф. АТ		
Додаток А	Сергієнко О. І. к.т.н., доц. каф. УГВіОП		
Нормо-контроль	Жуковська Д.О., ст. викладач каф. АТ	-	12.06.2025 р.

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Аналіз існуючих динамічних сонячних панелей	09.05.2025	
2	Теоритичне обгрунтування доцільності застосування машинного зору	13.05.2025	
3	Розробка системи трекінгу на основі машинного зору	20.05.2025	
4	Експеременталні дослідження	27.06.2025	
5	Охорона праці та безпека під час надзвичайних ситуаціях на підприємстві	04.05.2025	

Студент _____ **Ярослав НАБОКА**
(підпис) (прізвище та ініціали)

Керівник роботи _____ **Валерій ПОЦЕПАЄВ**
(підпис) (прізвище та ініціали)

ЛИСТ ЗАУВАЖЕНЬ

Посада, Ім’я та ПРІЗВИЩЕ	Суть зауваження, оцінка та підпис

АНОТАЦІЯ

Набока Ярослав Геннадійович. Підвищення ефективності динамічних сонячних панелей з застосуванням технології машинного зору / Випускна кваліфікаційна робота на здобуття освітнього ступеня «бакалавр» зі спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології. – ДВНЗ «ДонНТУ», Дрогобич, 2025.

Пояснювальна записка: 66 стор., 27 рис., 1 табл., 14 посилання.

У цій випускній кваліфікаційній роботі розглянуто спосіб підвищення ефективності динамічних сонячних панелей шляхом впровадження системи машинного зору. Актуальність теми обумовлена значними руйнуваннями енергетичної інфраструктури України внаслідок воєнних дій, що спричинило зростання попиту на автономні джерела живлення, зокрема сонячні установки. Основною метою дослідження стало створення інтелектуальної системи орієнтації сонячної панелі із використанням одноплатного комп'ютера Raspberry Pi та алгоритмів OpenCV для визначення просторового положення Сонця. У ході роботи було розроблено програмно-апаратний прототип, реалізовано алгоритми комп'ютерного зору для аналізу зображень, а також проведено експериментальні випробування з метою оцінки ефективності запропонованого підходу. Наукова новизна проєкту полягає в застосуванні методів машинного зору як альтернативи традиційним фоторезисторним системам. Практичне значення роботи полягає в можливості використання запропонованої системи для підвищення енергоефективності та автономності як побутових, так і промислових сонячних установок. Основні результати були представлені на онлайн-семінарі кафедри автоматики та телекомунікацій.

Ключові слова: сонячна панель, машинний зір, інтелектуальна система керування, динамічне позиціонування сонячної панелі, алгоритм аналізу зображення, мікрокомп'ютер raspberry pi, керування у реальному часі.

ЗМІСТ

ВСТУП……. 9
1 АНАЛІЗ ІСНУЮЧИХ ДИНАМІЧНИХ СОНЯЧНИХ ПАНЕЛЕЙ…….. 11
1.1 Розгляд статичних сонячних панелей …….. 11
1.2 Розгляд динамічних сонячних панелей ……... 12
1.2.1 Огляд контролерів на основі фоторезисторів та їх проблеми …… 13
1.2.2 Огляд контролерів на основі слідкуванням за Сонцем та їх проблеми 14
Висновки до розділу 1 ……. 16
2 ТЕОРИТИЧНЕ ОБГРУНТУВАННЯ ДОЦІЛЬНОСТІ ЗАСТОСУВАННЯ МАШИННОГО ЗОРУ……. 18
2.1 Проблематика традиційних систем……... 18
2.2 Принцип роботи машинного зору та бібліотеки OpenCV…….. 18
2.3 Суть та переваги машинного зору в контексті сонячного трекінгу …….. 21
Висновки до розділу 2 ……. 22
3 РОЗРОБКА СИСТЕМИ ТРЕКІНГУ НА ОСНОВІ МАШИННОГО ЗОРУ…. 24
3.1 Розробка тестової моделі …….. 24
3.1.1 Вибір контролера системи керування …….. 25
3.1.2 Вибір сонячної панелі……. 26
3.1.3 Вибір камери ……... 27
3.1.4 Вибір датчика струму та напруги ……. 29
3.1.5 Розробка схеми підключення …… 31
3.1.6 Проблеми конструкції …… 33
3.2 Розробка програмного забезпечення…… 36
3.2.1 Програмна архітектура системи …… 36
3.2.2 Використання OpenCV …….. 39
3.2.3 Розрахунок нахилу сонячної панелі …….. 40
3.2.4 Опис логіки збору статистичних даних …… 40
3.2.5 Модуль керування сервоприводами……... 41

3.2.6 Веб інтерфейс …….. 42
Висновки до розділу 3 ……. 44
4 ЕКСПЕРЕМЕНТАЛЬНІ ДОСЛІДЖЕННЯ …… 46
4.1 Умови проведення експерименту ……. 46
4.2 Обробка та аналіз результатів ……... 47
Висновки до розділу 4 ……. 48
ВИСНОВКИ …… 49
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ…….. 50
ДОДАТОК А. Охорона праці та безпека в надзвичайних ситуаціях……. 52
ДОДАТОК Б. Компонентів які були використані…… 56
ДОДАТОК В. Програмний код ……. 57

ВСТУП

Актуальність роботи – через війну в Україні було пошкоджено дуже багато енергетичної інфраструктури, що викликало велике зростання встановлення сонячних панелей.

За інформацією члена управління Асоціації сонячної енергетики України Сергій Ручко [1], на початку 2023 року в Україні налічувалося 53 тисячі домогосподарств із сонячними електростанціями.

За даними “Solarpath” [2], 2025 року очікується подальший розвиток сонячної енергетики в Україні. Крім того, міжнародні організації мають намір фінансувати встановлення сонячних систем для лікарень та інших ключових об'єктів, що сприятиме зміцненню енергетичної безпеки країни.

Ці статті показують, що сонячна енергетика не лише займає важливу частину всієї енергетики, а є важливою частиною енергостабільності України. Сонячна енергетика – це галузь, що тільки розвивається в нашій країні, через що попит і кількість сонячних панелей з кожним місяцем зростає, особливо в регіонах з нестабільним електропостачанням, що дозволяє приватним домогосподарствам менше залежати від централізованого енергопостачання.

Таким чином, сонячна енергетика стала важливою частиною не тільки для загальної енергетики України, а також для частих домогосподарств. Враховуючи все сказане вище, можна дійти висновку, що підвищення ефективності динамічних сонячних панелей є актуальною науковою роботою.

Мета роботи – це підвищення ефективності перетворення сонячної енергії шляхом розробки та реалізації системи динамічного стеження за Сонцем.

Завдання проектування/дослідження – це:

Розробити систему машинного зору для визначення положення Сонця.

Побудувати алгоритм орієнтування сонячної панелі. Обрати та налаштувати апаратне забезпечення. Провести експериментальні дослідження. Оцінити ефективність реалізованої системи.

Об’єктом в роботі виступає інтелектуальна система орієнтування

сонячної панелі з використанням машинного зору. Предметом є методи та засоби реалізації динамічного орієнтування сонячної панелі за допомогою машинного зору.

Методи дослідження – це експериментальний метод – використано для дослідження ефективності панелі в різних умовах. Аналітичний метод – застосовано для аналізу зібраних результатів. Метод моделювання – для розробки алгоритмів орієнтування з використанням Python та OpenCV. Інженерний підхід – для побудови всієї системи на базі Raspberry Pi 4 та підключеного устаткування.

Структура та загальний обсяг випускної кваліфікаційної роботи. Робота складається з вступу, чотирьох розділів, висновків роботи, списку використаних джерел та додатків. Загальний обсяг становить ___ сторінок. Практичне значення отриманих результатів – це модернізація системи автоматичного управління динамічною сонячною панеллю, яка може замінити існуючу систему на виробництві або при встановленні нових динамічних сонячних панелей.

Апробація роботи – це представлення основних тез бакалаврської роботи на онлайн-семінарі, що відбувся на кафедрі Автоматики та телекомунікацій.

1 АНАЛІЗ ІСНУЮЧИХ ДИНАМІЧНИХ СОНЯЧНИХ ПАНЕЛЕЙ

1.1 Розгляд статичних сонячних панелей

Рисунок 1.1 – Статична сонячна панель

Статичні сонячні панелі — це панелі, які закріплюються в одному оптимальному положенні і не мають можливості зміни кута нахилу. Є популярним варіантом і найпростішим.

Переваги:

1. Простота конструкції.
2. Відсутність рухомих частин.
3. Низька вартість.
4. Висока надійність.
5. Ідеальні для дахових установок або невеликих автономних систем.
6. Не потребує постійного догляду.

Недоліки:

1. Ефективність знижується у ранкові та вечірні години, а також у зимовий період.
2. Ефективність знижується у зимовий період.
3. Неможливість динамічно адаптуватися до зміни положення Сонця.

4. Залежність від правильно розрахованого орієнтації та кута нахилу.

1.2 Розгляд динамічних сонячних панелей

Динамічні сонячні панелі — сучасні автоматичні системи, які здатні змінювати свій кут нахилу та повороту. Самі сонячні панелі встановлюються на трекери. Основним завданням трекера є стеження за сонцем для отримання максимальної енергії.

Типи трекерів:

Одноосьові трекери – Обертаються навколо однієї осі (зазвичай горизонтальної — за азимутом). – Збільшують виробіток енергії на 15–30%. – Простіші у реалізації порівняно з двоосьовими.

Двоосьові трекери – Мають змогу обертатися навколо двох осей: горизонтальної та вертикальної. – згідно з даними статті [10] одноосьові трекери дають приріст генерації приблизно на 15-30%, двох осьові до 35-40%. Докладнішу статику можна розглянути в цій статті [10].

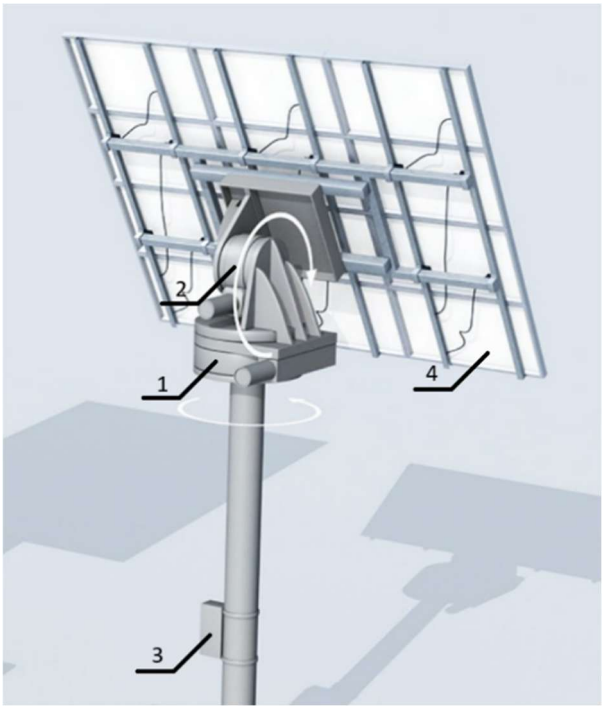

Рисунок 1.2 – Схема моделі двох осьової динамічної сонячної панелі

На рисунку 1.2 зображені елементи:

1. Вісь X.
2. Вісь Y.
3. Контроллер.
4. Сонячний модуль.

1.2.1 Огляд контролерів на основі фоторезисторів та їх проблеми

Трекер на основі фоторезисторів - один із найпростіших типів. Ці пристрої працюють за принципом знаходження найяскравішого напрямку освітлення. Вони складаються з 4 фоторезисторів, розділених непрозорою перегородкою, яка створює затемнення на фоторезисторах, якщо світло падає під кутом. Коли світло падає на рівномірно на фоторезистори, контролер порівнює дані з датчиків і коригує положення сонячної панелі.

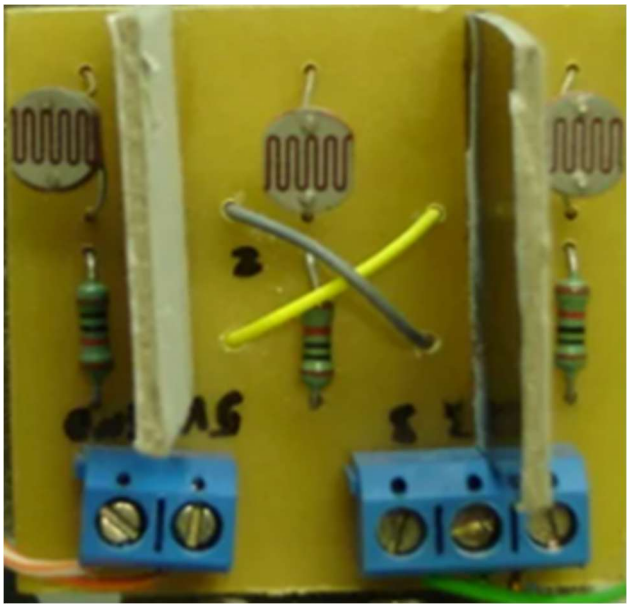

Рисунок 1.3 – Приклад вигляду трекера на 3 фоторезисторах

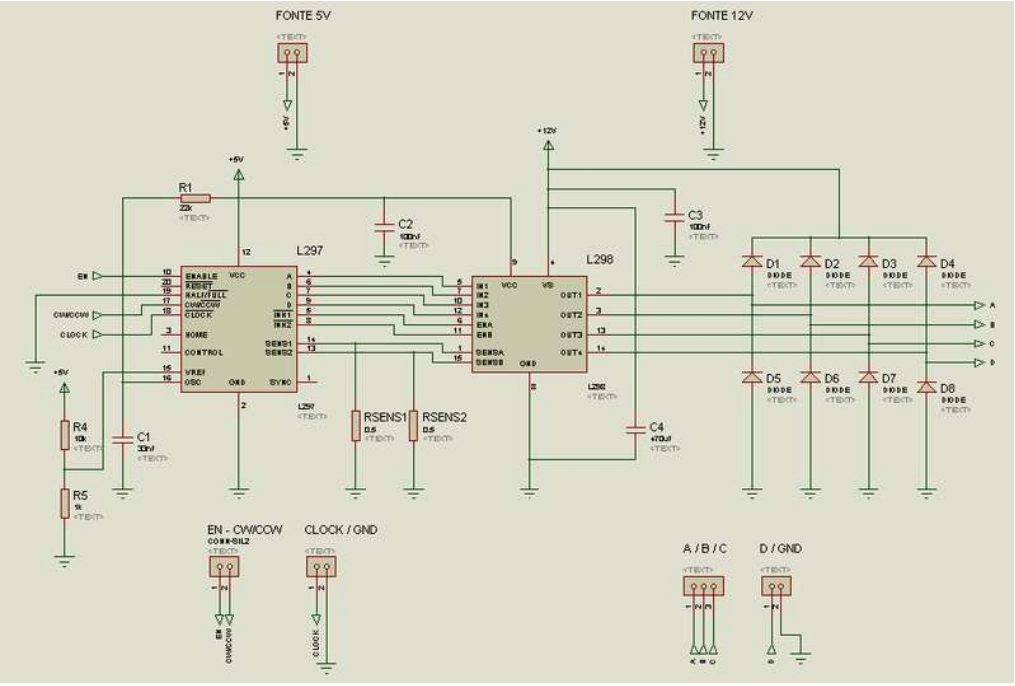

Рисунок 1.4 – Електрична схема трекера на фоторезисторах

Проблемою трекерів на основі фоторезисторів є низька точність у хмарні дні, за наявності хмар або великої кількості розсіяного світла може з'являтися тремтіння та хаотичні рухи у пошуках найяскравішої точки.

Докладніше про принцип роботи таких трекерів та їх програмне забезпечення можна прочитати зі статті [11]

1.2.2 Огляд контролерів на основі слідкуванням за Сонцем та їх проблеми

Астрономічне слідкування за Сонцем ґрунтується на принципах сферичної астрономії — розділі науки, який описує положення небесних тіл на небесній сфері відносно певної точки на Землі. Для розрахунку азимуту та висоти Сонця в небі використовують математичні формули, що враховують:

1. Географічна широта (φ) і довгота (λ)

   Ці координати вказують розташування трекера (пристрою, що відстежує положення Сонця) на поверхні Землі:

   φ — географічна широта.

   λ — географічна довгота.

Ці значення є критично важливими для правильного визначення положення Сонця на небі в певному місці.

2. Юліанська дата (JD) - це безперервний відлік днів, який широко використовується в астрономії. Вона потрібна для обчислення сонячного часу та положення Сонця в будь-який момент.

$$JD = 367Y - \left[\frac{7\left(Y+\left[\frac{M+9}{12}\right]\right)}{4}\right] + \left[\frac{275M}{9}\right] + D + 1721013.5 + \frac{UT}{24} \quad (1.1)$$

Де Y – рік,

M – місяць,

D – день,

UT – це всесвітній час. Він виражається у годинах з десятковою частиною.

3. Сонячне схилення (δ). Кут між напрямком на Сонце та площиною земного екватора. Змінюється щодня, від -23.44 до +23.44 внаслідок нахилу земної осі.

$$\delta = 23.44 \cdot sin\left(\frac{360}{365} \cdot (n + 10)\right) \quad (1.2)$$

Де n – порядковий номер дня в році (1 для 1 січня, 365 для 31 грудня).

4. Сонячна година і часовий кут (H) це кут між напрямком на південь і положенням Сонця в поточний момент:

$$H = \mathrm{ST} \cdot 15 - 180 \quad (1.3)$$

ST — місцевий сонячний час у годинах.

5. Висота Сонця (h)

$$\sin(\mathrm{h}) = \sin(\phi) + \sin(\delta) \cdot \cos(\delta) + \cos(\mathrm{H}) \quad (1.4)$$

Де h – кут висоти Сонця над горизонтом.

Ця формула дозволяє визначити, на якій висоті Сонце знаходиться над горизонтом у конкретний момент часу.

6. Азимут Сонця

$$\cos(A) = \frac{\sin(\delta) - \sin(h) \cdot \sin(\phi)}{\cos(h) \cdot \cos(\phi)} \quad (1.5)$$

Де A — азимут Сонця.

Ця формула дозволяє обчислити, в якому напрямку (азимуті) знаходиться Сонце.

Вище описані всі формули до розрахунку положення сонця на небі. Докладніший опис алгоритмів розрахунку положення Сонця наведено в статті “Solar Position Algorithm for Solar Radiation Applications” [12].

Проблематика такого способу полягає в недостатньому врахуванні атмосферних умов, а саме хмарності, туману, пилу або локальних засвіток. У таких ситуаціях Сонце може бути не самим ефективним місцем. Наприклад, при сході сонця, коли сонце ще не вийшло за горизонт, ми вже можемо спостерігати відблиски від сонця та отримувати енергію.

Висновки до розділу 1

У першому розділі ми розглянули різні типи трекерів. Один з найпосростіших трекерів це трекери на основі фоторезисторів, які реагують інтенсивність світла, вони і є найпростішими і найдешевшими трекерами, але мають велику кількість недоліків і проблем, а саме низьку ефективність у хмарну погоду і мають чутливість до розсіяного світла, можуть робити зайві рухи у пошуках кращого місця і таким чином витрачати енергію.

Другим трекером є стеження за сонцем, який базується на сферичній

астрономії та математичних розрахунках. Має більшу ефективність ніж трекери на фоторезисторах, але також має безліч проблем.

2 ТЕОРИТИЧНЕ ОБГРУНТУВАННЯ ДОЦІЛЬНОСТІ ЗАСТОСУВАННЯ МАШИННОГО ЗОРУ

2.1 Проблематика традиційних систем

Ефективність роботи фотоелектричних систем напряму залежить від того, під яким кутом сонячне випромінювання потрапляє на поверхню панелі. Із теоретичної точки зору, максимальна інтенсивність сонячного потоку досягається при перпендикулярному падінні променів, тобто під кутом 90 відносно площини модуля. Відхилення від цього положення призводить до зниження інтенсивності поглинання світла, а отже — до втрати частини потенційної електроенергії.

Згідно з дослідженням [13], оптимізація кута нахилу сонячних панелей може призвести до збільшення річної продуктивності на 15–20%. Крім того, варіативність кута залежить не лише від географічної широти, а й від сезону року та атмосферних умов, що вказує на важливість динамічної корекції положення панелі протягом доби.

2.2 Принцип роботи машинного зору та бібліотеки OpenCV

Машинне зору — це міждисциплінарна сфера, що об’єднує технології цифрової обробки зображень, комп’ютерної графіки, штучного інтелекту та прикладної математики з метою навчити комп’ютери розпізнавати й аналізувати візуальну інформацію. На відміну від людського ока, яке використовує біологічні рецептори та складну контекстну інтерпретацію, машинне зору працює з цифровими даними — наборами пікселів, отриманими з камер або відеопотоків.

Дає можливість аналізувати зображення а саме розпізнавати об'єкти їх колір, стежити за рухами вимірювати і виявляти найяскравіші об'єкти в

реальному часі.

Приклади використання:

1. Автономні автомобілі: розпізнають пішоходів, розмітку, знаки
2. Розпізнавання обличчя: камера фокусується на обличчі людини
3. Армія: Визначення танків та інших цілей

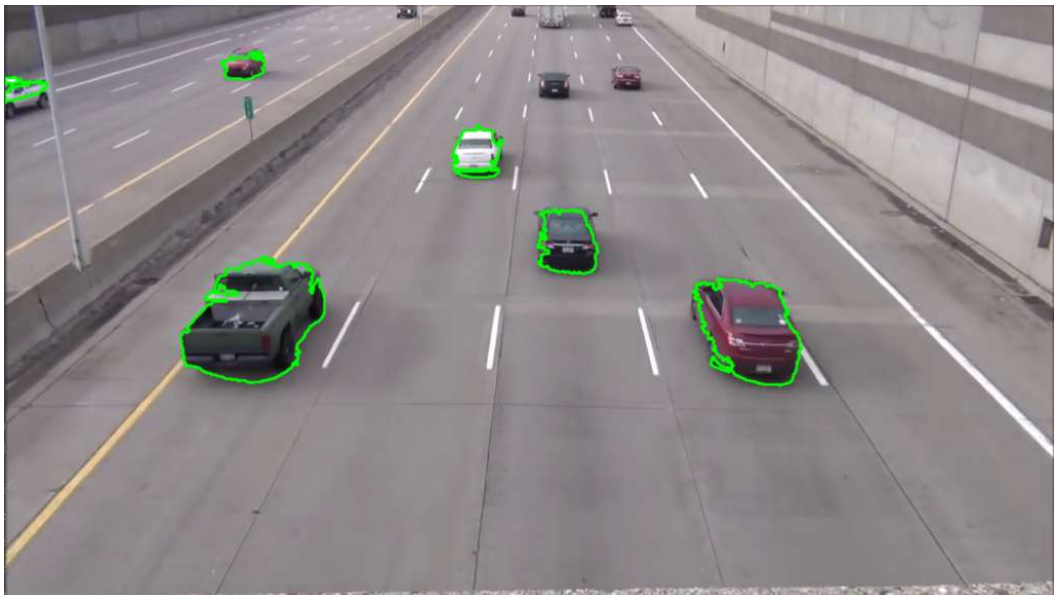

Рисунок 2.1 – Приклад пошуку контурів на зображенні

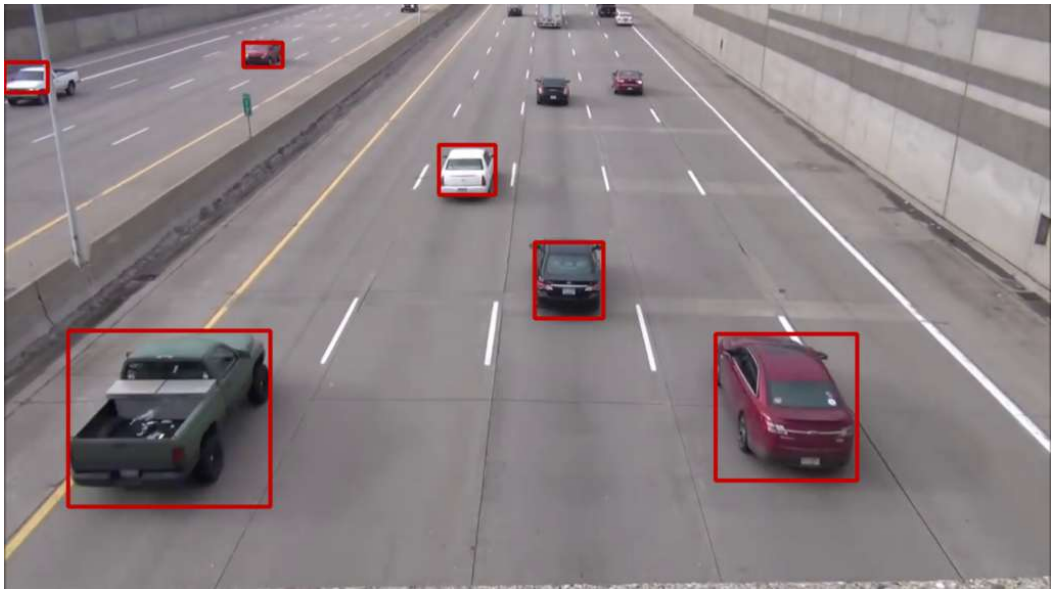

Рисунок 2.2 – Приклад аналізу дорожнього трафіку

На рисунку 2.1 представлено приклад роботи системи машинного зору,

реалізованої з використанням бібліотеки OpenCV. Зображення демонструє процес виявлення та відстеження транспортних засобів на дорозі в реальному часі. Алгоритми комп'ютерного зору автоматично ідентифікують автомобілі на відеопотоці, обводячи їх рамками та призначаючи унікальні ідентифікатори. Це дозволяє системі здійснювати аналіз трафіку, оцінювати щільність потоку та відстежувати рух окремих об'єктів.

Типова система машинного зору складається з низки послідовних етапів обробки даних:

1. Захоплення зображення або відеопотоку.
2. Попередня обробка отриманих даних.
3. Виділення релевантних ознак.
4. Аналіз, класифікація або прийняття рішення.

Ключовим етапом у функціонуванні системи машинного зору є створення цифрового зображення, камера фіксує у вигляді двовимірної матриці пікселів з даними про яскравість і колір кожного з них. Це зображення передається до обчислювального модуля, де проходить початкову обробку. На цьому етапі відбувається видалення шумів, покращення якості, а також масштабування або переформатування зображення — усе це спрямовано на спрощення подальшої аналітики.

Після первинної обробки система переходить до аналізу структури зображення, використовуючи алгоритми для виявлення контурів, меж, кутів, а також текстурних і кольорових відмінностей. На основі цих ознак система ідентифікує об'єкти, важливі для подальших дій — наприклад, визначає об'єкт на фоні, відстежує його переміщення або класифікує за формою й розмірами. У цьому процесі часто застосовуються попередньо навчені моделі машинного навчання, здатні розпізнавати характерні патерни або категорії. Залежно від завдань, система може автоматично ухвалювати рішення — змінити положення виконавчого механізму, надіслати сповіщення оператору або зафіксувати подію.

Одним із найпоширеніших інструментів для реалізації систем машинного зору є бібліотека OpenCV— відкритий та потужний програмний фреймворк,

розроблений для обробки зображень і відео в режимі реального часу. Завдяки підтримці таких мов програмування, як C++, Python та Java, OpenCV легко інтегрується у різноманітні проєкти, забезпечуючи високу гнучкість і масштабованість. Ця бібліотека надає широкий набір інструментів: від базових операцій (зміна колірних моделей, фільтрація, детекція контурів, морфологічні перетворення, масштабування й обертання) до складних методів комп'ютерного зору, таких як розпізнавання облич, трекінг об'єктів, аналіз оптичного потоку й побудова моделей навколишнього середовища. OpenCV активно використовується в наукових, промислових і побутових рішеннях, що потребують високої точності візуального аналізу.

OpenCV функціонує на основі високоефективних алгоритмів цифрової обробки зображень, що працюють безпосередньо з масивами піксельних даних. Ці алгоритми дозволяють швидко та точно виконувати трансформації, необхідні для аналізу візуальної інформації. Наприклад, для виявлення контурів об'єктів застосовується оператор Канні, який знаходить різкі зміни яскравості між суміжними пікселями, що вказує на межі. У разі обробки відео використовується трекінг — послідовний аналіз кадрів із метою визначення переміщення об'єктів у часі.

Особливу перевагу OpenCV демонструє у вбудованих рішеннях — таких як Raspberry Pi або мобільні мікропроцесори, де обмежені обчислювальні ресурси вимагають максимальної ефективності. Завдяки своїй модульній архітектурі й оптимізації під різні апаратні платформи, ця бібліотека дозволяє швидко впроваджувати системи машинного зору в проєкти автоматизації, технічного контролю, робототехніки, автономного транспорту та інших сфер.

2.3 Суть та переваги машинного зору в контексті сонячного трекінгу

Використання машинного зору дозволяє отримувати дані про хмарність та погодні умови в режимі реального часу. У випадку з динамічними сонячними панелями машинний зір виконує функцію сенсора, який може:

1. Знаходити найяскравішу область на зображенні.
2. Коригувати положення сонячної панелі у відповідь на зміни у навколишньому середовищі.
3. Фіксувати появу хмар або затемнень.
4. Аналізувати навколишнє середовище для запобігання помилковим спрацюванням.

Такий підхід на відміну від елементарного аналізу яскравості, надає значно ширший функціонал. Завдяки камері та алгоритмам обробки зображень, система може оцінювати контекст сцени, ідентифікувати джерело світла навіть за часткової хмарності, а також відокремлювати відблиски чи інші об'єкти, що не мають відношення до Сонця. Таким чином, використання комп’ютерного зору дозволяє працювати стабільно та ефективно у складних метеоумовах, де інші методи втрачають точність.

Згідно з даними дослідження [14], використання машинного зору дозволяє досягти до 23% збільшення ефективності у порівнянні з фіксованими панелями, а також суттєво перевершує LDR-трекери за стабільністю роботи в умовах змінного освітлення.

Висновки до розділу 2

Машинний зір є фундаментальною технологією, що відкриває нові можливості для автоматизованих систем у найрізноманітніших сферах. Завдяки здатності аналізувати зображення та відео в реальному часі, ця технологія дозволяє техніці самостійно сприймати, оцінювати та реагувати на зовнішні умови. Одним із найефективніших інструментів реалізації машинного зору є бібліотека OpenCV, яка забезпечує високу точність, швидкість обробки та широку функціональність навіть у системах із обмеженими ресурсами.

У контексті розробки інтелектуальних систем динамічного керування, зокрема для орієнтації сонячних панелей, використання машинного зору в поєднанні з OpenCV дозволяє суттєво підвищити ефективність енергогенерації.

Це доводить актуальність, практичну доцільність і перспективність впровадження машинного зору як одного з ключових елементів сучасних енергоефективних рішень.

Проведений теоретичний аналіз дозволяє зробити висновок, що впровадження систем стеження за Сонцем на основі машинного зору є науково обґрунтованим, технічно доцільним та енергетично вигідним рішенням. На відміну від традиційних трекерів, системи з камерою та обробкою зображень мають здатність адаптуватися до змінного навколишнього середовища, що робить їх універсальними для широкого спектра кліматичних та географічних умов. Крім того, такі системи можуть бути основою для багатофункціональних інтелектуальних платформ, здатних не лише слідкувати за Сонцем.

3 РОЗРОБКА СИСТЕМИ ТРЕКІНГУ НА ОСНОВІ МАШИННОГО ЗОРУ

3.1 Розробка тестової моделі

3.1.1 Вибір контролера системи керування

На даний момент існує дуже багато різних варіацій мікрокомп'ютерів. При виборі відповідного мікрокомп'ютерів варто звернути увагу на:

1. Продуктивність
2. Можливість підключення камери
3. Підтримка I2C інтерфейсу
4. Підтримка широтно-імпульсної модуляції (ШІМ)
5. Підтримка Linux, Python, OpenCV
6. Компактність
7. Ціна

Також дана платформа має активну підтримку операційної системи Linux, багато готових бібліотек під Python. Тож моїм вибором стала Raspberry Pi 4 Model B.

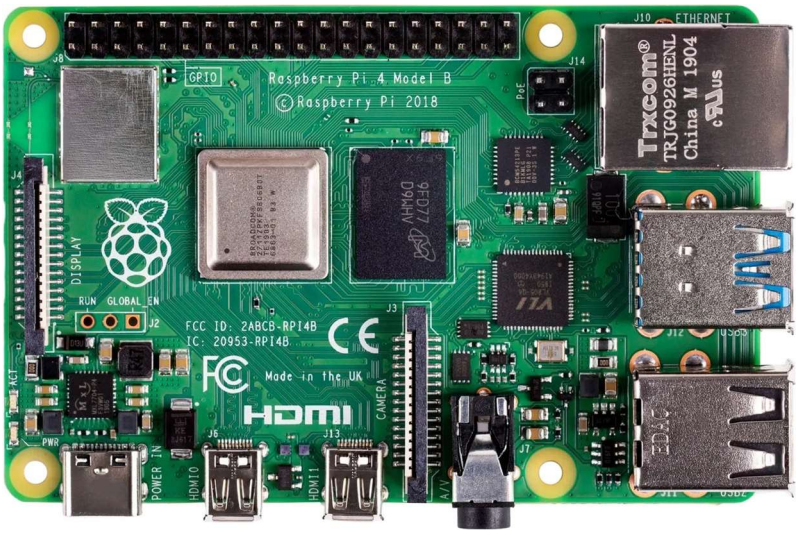

Рисунок 3.1 – Raspberry Pi 4 Model B 4GB

Характеристики модулю:

1. Процесор: Broadcom BCM2711, Quad core Cortex-A72 1.5 GHz
2. Оперативна пам'ять: 4GB LPDDT4-2400
3. Живлення: 5V DC
4. Зберігання даних: SD card 128GB
5. Розміри: 85 * 47 * 15 мм

Докладніші технічні характеристики модулю описані [7].

3.1.2 Вибір сонячної панелі

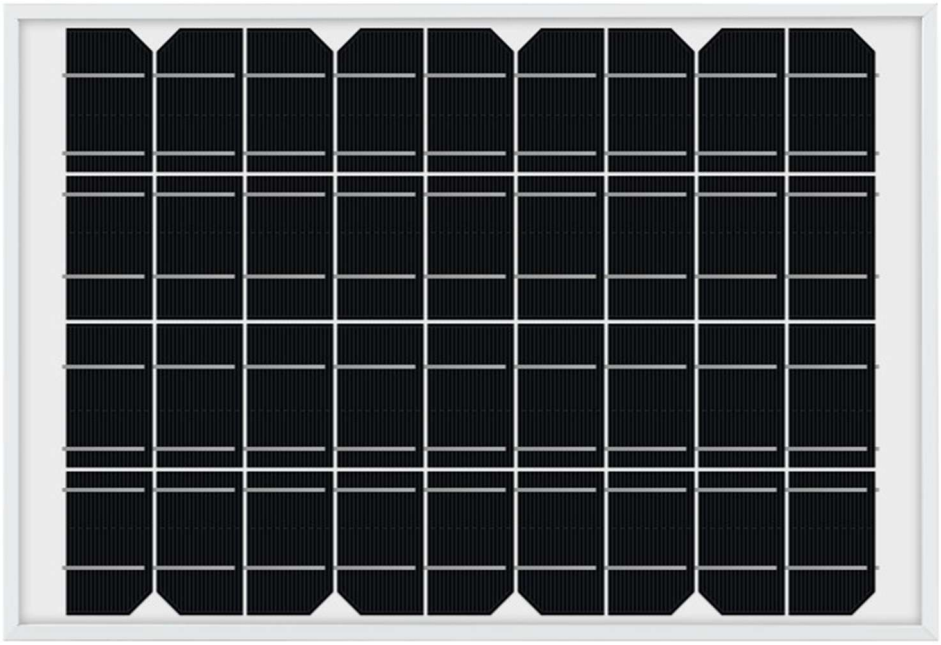

Рисунок 3.2 – Silicon Solar PV module 18V10W

Для правильного вибору типу сонячної панелі важливо ознайомитися з основними їх типами. Зараз існує декілька основних типів сонячних панелей:

1. Монокристалічні сонячні панелі:

Виготовляються с одного кристала кремнію. Вони мають високий ККД, термін служби 25-30 років.

1. Полікристалічні сонячні панелі:

На відміну від монокристалічних полікристалічні виготовляються з дрібних кристалів кремнію. Мають набагато гірший ККД в порівнянні з монокристалічними.

2. Тонкоплівкові сонячні панелі:

Виготовляються шляхом нанесення тонкого слою матеріалу (аморфного кремнію або селеніду кадмію). Мають набагато гірший ККД в порівнянні з монокристалічними. Можна виготовити гнучні панелі що дозволяє використовувати їх в нестандартних умовах.

Була обрана сонячна панель Silicon Solar PV module 18V10W монокристалічного кремнієвого типу, тому що вона має високий ККД, тривалий термін служби 25-30 років. Підходить під розміри проекту і має невелику вагу.

Характеристики модулю:

1. Тип панелі: Монокристалічна кремнієва
2. Максимальна потужність: 10 Вт
3. Робоча напруга: 18 В
4. Робочий струм: ~0.55 А
5. ККД модуля: ~18-20%
6. Розміри: 340 * 250 * 17 мм
7. Матеріал каркасу: Алюміній
8. Вага: 0.935 кг

Докладніші технічні характеристики модулю описані [4].

3.1.3 Вибір камери

Існує безліч камер які сумісні з Raspberry Pi, вони можуть мати різний кут огляду та якість зображення. Нам потрібна камера, яка буде мати як можна більший кут огляду, щоб знімати найбільшу частину неба, що допоможе нам краще знаходити найяскравіші області в небі. Також важливим параметром є роздільна здатність камери, яка відповідає за чіткість і деталізацію зображення.

Рисунок 3.3 – RPi Camera (G), Fisheye Lens

Характеристики модулю:

1. Датчик: OV5647
2. Роздільна здатність: 5 Мп
3. Кут огляду: 200
4. Роздільна здатність зображення: 1080p
5. Розміри: 25 * 24 мм
6. Вага: 0.017 кг

Докладніші технічні характеристики модулю описані [6].

3.1.4 Вибір датчика струму та напруги

Датчик стоку та напруги вибирається виходячи з характеристик сонячної панелі. Важливо щоб максимальне вимірюване струм і напруга не перевищували струм і напруга, що видається сонячною панеллю. Виходячи з даних сонячної панелі відмінно підійде INA219.

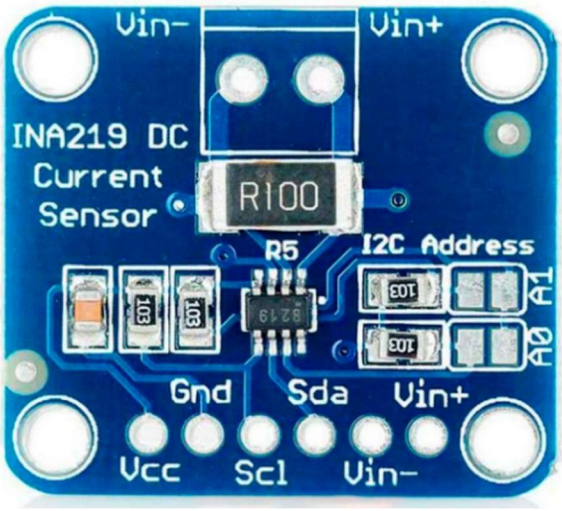

Рисунок 3.4 – INA219

Характеристики модулю:

1. Тип підключення: I2C
2. Розміри: 25 * 22 * 4 мм
3. Максимальний струм, що вимірюється: 3,2А
4. Точність вимірювання струму: 0,8мА
5. Максимальна напруга, що вимірюється: +-26 В
6. Напруга живлення: від 3В до 5В

Докладніші технічні характеристики модулю описані [3].

3.1.5 Вибір сервоприводів та кріплень

Існують різні типи сервоприводів, які поділяються за способом керування:

1. Аналогові – керуються за допомогою ШІМ-сигналу (PWM).
2. Цифрові – мають вбудований мікроконтролер, що забезпечує вищу точність, можливість налаштування та часто підтримку зворотного зв’язку. Зазвичай вони значно дорожчі за аналогові аналоги.

Також важливо звернути увагу на тип редуктора важливо, щоб він був металевим через те, що сонячна панель надаватиме велике навантаження на сервоприводи.

У даному проекті буде використовуватися два аналогові MG996R Tower

Pro на 180 градусів для вертикалі та MG996R Tower Pro на 360 градусів для горизонталі.

Рисунок 3.5 – Сервопривід MG996R Tower Pro

Характеристики модулю:

1. Редуктор: Сталь
2. Кут повороту: 180 або 360
3. Робочий напруга: 4.8 – 7.2 В
4. Діаметр вала: 5.7 мм
5. Крутний момент: 13 кг
6. Розміри: 40.7 * 19.7 * 43 мм
7. Вага: 55 г

Докладніші технічні характеристики модулю описані [5].

Для з'єднання сервоприводів використовуватимемо 2x U-Shaped Long кронштейна та 1x Multi-Purpose кронштейн. Також знадобиться 2 х монтажна втулки. Для більшої стабільності слід використовувати противагу, щоб зменшити навантаження на сервопривід.

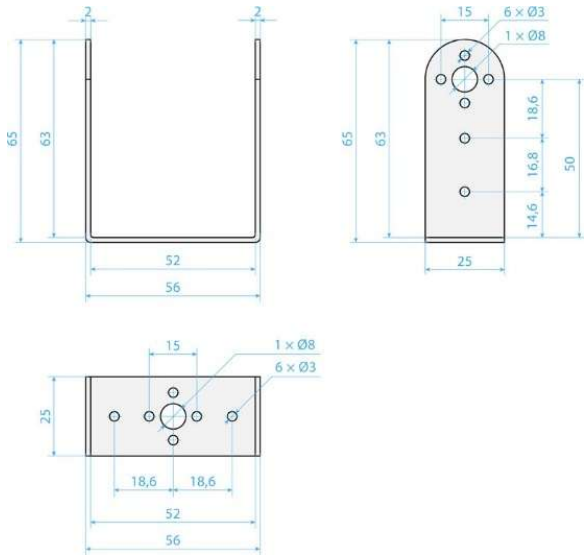

Рисунок 3.6 – Креслення кронштейна U-Shaped Long

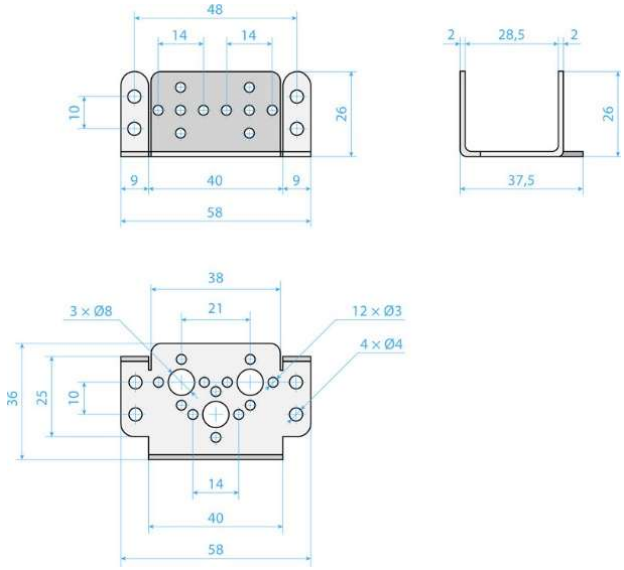

Рисунок 3.7 – Креслення кронштейна Multi-Purpose

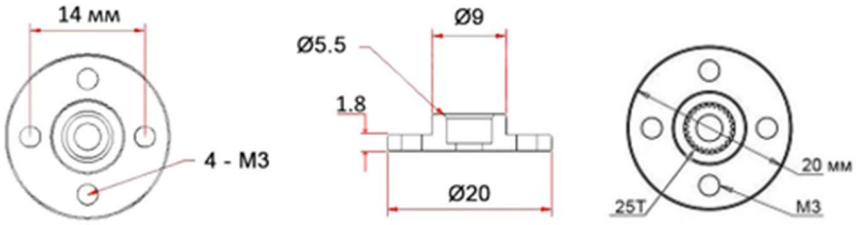

Рисунок 3.8 – Креслення монтажної втулки

Рисунок 3.9 – Зібрана конструкція сервоприводів

Після цього на цю конструкцію буде кріпитися сонячна панель і модуль Raspberry Pi з камерою та датчиком стуруму.

Проблемою даного кріплення є наявність сліпої зони на осі X. У діапазоні 45–90 градусів, сонячна панель фізично не може орієнтуватися на джерело світла через обмеження конструкції. Це може призвести до зниження ефективності системи у певні періоди дня, коли Сонце перебуває в межах цього кутового діапазону.

3.1.5 Розробка схеми підключення

Теперь можна перейти до розробки функціональної схеми підключення всіх модулів до мікрокомп’ютера Raspberry Pi. Основною частиною системи є Raspberry Pi 4 Model B, який виступає в ролі центрального контролера, и отвечает за корректную работу всех модулей. Моніторинг струму здійснюється за допомогою датчика INA219, який підключається через інтерфейс I2C. Для цього використовуються контакти GPIO 2 (SDA) та GPIO 3 (SCL). Цей датчик дозволяє

точно вимірювати силу струму, напругу та потужність.

Для управління положенням сонячної панелі використовуються два сервоприводи, що забезпечують зміну орієнтації по осях X та Y. Вони підключаються до цифрових виходів GPIO 17 і GPIO 27. Ці піни використовуються для генерації широтно-імпульсного сигналу (ШІМ), необхідного для точного керування кутом обертання сервоприводів.

Камера підключена через CSI-2 це промисловий стандарт інтерфейсу, який має високу пропускну здатність до 6 Gbps, що забезпечує високу якість зображення.

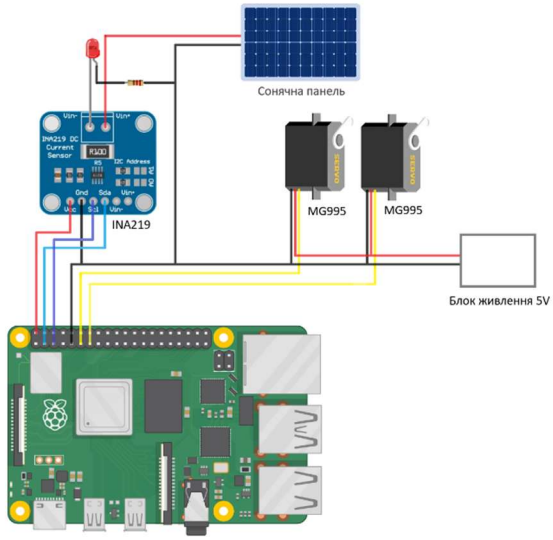

Рисунок 3.10 – Функціональна схема підключення

Розробимо електричну схему підключення у програмному забезпеченні EasyEDA. Для цього використовуватимемо модулі з бібліотеки EasyEDA. Результат роботи можна побачити рисунке 3.11

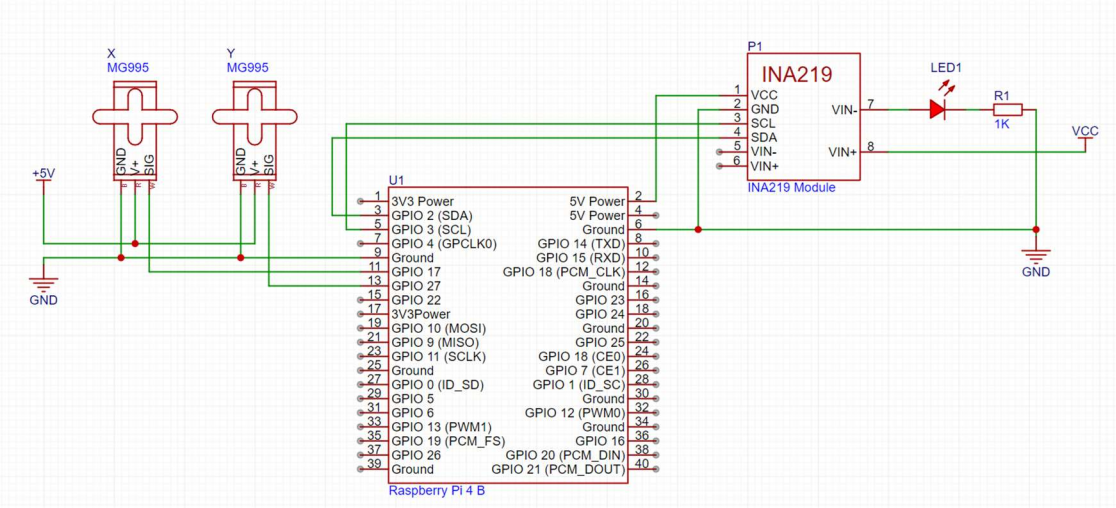

Рисунок 3.11 – Електрична схема підключення

Для коректної роботи двох сервоприводів нам доведеться використовувати додаткове джерело живлення на 5V також потрібно з'єднати GND другого блока живлення з GND Raspberry PI. Для роботи датчика струму і напруги INA219 знадобиться встановити в ланцюг навантаження, в ролі якої виступатиме світлодіод LED1 і резистор на 1 кОм.

3.1.6 Проблеми конструкції

Мертва зона при обертанні (30–90).

У конструкції платформи виявлена критична мертва зона в діапазоні кутів від 30 до 90 по горизонтальній осі (азимуту), в яку механізм фізично не може повернутися. Це обмеження обумовлено самим способом кріплення конструкції — при спробі обертання в зазначений сектор панель впирається у несучі елемент. Рух у цей діапазон стає неможливим, оскільки серводвигун навіть за наявності крутного моменту не здатен подолати механічну перешкоду. Щоб уникнути цього, потрібно або повністю переробити спосіб кріплення, забезпечивши вільний сектор обертання в повному діапазоні, або встановити конструкцію на підйомну опору, яка забезпечить необхідну рухомість.

Перевантаження конструкції

Ще однією серйозною проблемою є вагова асиметрія платформи. Сонячна

панель, як основний елемент системи, має значну масу і розміри. У випадку з використанням монокристалічних панелей навіть порівняно малого формату, вага разом з кріпленням може сягати 2–4 кг. Через те, що панель закріплена лише з одного боку, утворюється перекіс, внаслідок чого постійно навантажуються вали серводвигунів. Відсутність противаги або невірно розрахований її розмір спричиняє появу моменту сили, який систематично виводить механізм з робочого положення, збільшує споживання струму сервомотором та прискорює зношування вузлів. На деяких етапах експлуатації було зафіксовано залипання сервомеханізмів.

Рисунок 3.13 – Зібраний модуль з камерою

З метою вирішення цієї проблеми нами було впроваджено механізм противаги, ретельно підібраний за масою та розміщений симетрично з іншого боку осі обертання. Важливо було досягти такого балансу, щоб центр мас системи знаходився максимально близько до геометричної осі обертання. У результаті вдалося істотно знизити крутний момент, необхідний для повороту платформи, що, в свою чергу, зменшило навантаження на сервоприводи.

Рисунок 3.12 – Зібраний модуль з камерою

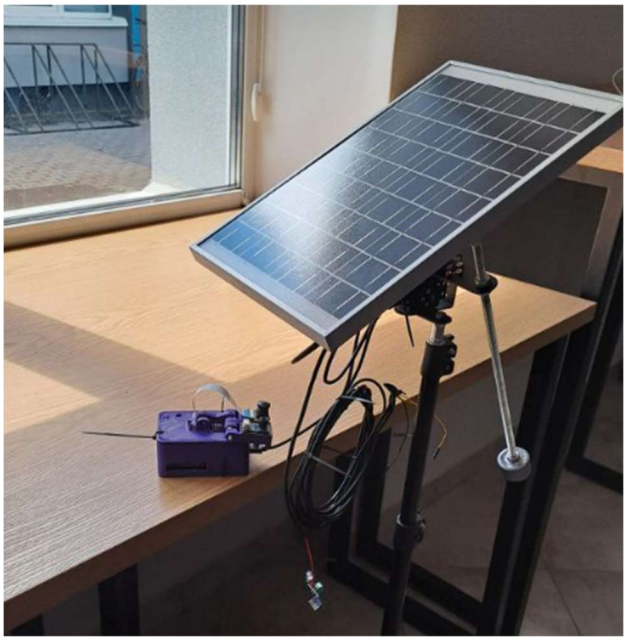

Рисунок 3.14 – Повністю зібрана сонячна панель

Було повністю зібрано всю конструкцію системи динамічного орієнтування сонячної панелі. Усі етапи — від монтажу механічної основи до встановлення електронних компонентів — пройшли успішно. На Рисунку 3.12 зображено модуль з камерою, яка закріплена на платформі для здійснення машинного зору.

Подальший монтаж включав встановлення самої сонячної панелі,

підключення сервоприводів по обох осях — горизонтальній (азимут) і вертикальній (висота), а також підключення датчиків та блока живлення. На Рисунку 3.14 показано повністю змонтовану конструкцію у робочому стані. Особливу увагу було приділено якості з'єднань, міцності кріплень та зручності доступу до ключових вузлів для подальшого обслуговування або налаштування.

3.2 Розробка програмного забезпечення

3.2.1 Програмна архітектура системи

Програмну частину системи побудовано основі операційної системи Linux. Основною мовою програмування є Python, що спрощує процес розробки і дозволяє легко інтегрувати бібліотеки і модулі.

Для реалізації системи стеження було обрано стік технологій.

OpenCV – для аналізу відеопотоку, виявлення яскравих ділянок.

Picamera2 – яка забезпечує доступ до відеопотоку в реальному часі. На відміну від попередньої версії Picamera, ця бібліотека підтримує сучасний API libcamera та надає гнучкі засоби роботи з роздільною здатністю, форматами зображень та експозицією.

Pigpio – для точного керування сервоприводами PWM.

Flask – як мікрофреймворк для створення REST API, що дозволяє дистанційно керувати та моніторити дані.

Astral – для обчислення положення Сонця за географічними координатами, датою та часом.

INA219 – для отримання інформації про струм та напругу.

SQLite – для збереження даних у локальну базу даних.

3.2.2 Використання бібліотеки OpenCV

Для реалізації процесу орієнтації сонячної панелі на джерело світла

критично важливо точно визначити положення Сонця на небі. Оскільки система працює виключно з візуальною інформацією, отриманою з камери, необхідно обробити зображення таким чином, щоб виділити найяскравішу область кадру, яка з відповідає най оптимальнішій позиції . Для цього розроблено функцію find_brightest_area, яка виконує низку операцій попередньої обробки та аналізу зображення.

Загальна логіка функції полягає у наступному:

1. Перетворення у відтінки сірого. Зображення конвертується з кольорового формату в сірий (grayscale) для спрощення обробки. Це дозволяє зосередитись лише на яскравості пікселів, ігноруючи кольорову інформацію, яка у цьому випадку не є релевантною.

Рисунок 3.15 – Приклад конвертації в кольорового формату в сірий

2. Фільтрація шуму. На зображення накладається Гаусовий фільтр (GaussianBlur) із ядром розміром 5x5, що дозволяє згладити дрібні коливання яскравості, викликані шумом або пилом на об’єктиві. Це особливо важливо при змінному освітленні або частковому затіненні неба.

Рисунок 3.16 – Приклад Гаусового фільтру

3. Бінаризація та виділення контурів. Після розмиття застосовується алгоритм threshold або adaptive threshold, що перетворює зображення у двобітну маску, де яскраві зони виділяються білим кольором.

Рисунок 3.17 – Приклад Гаусового фільтру

4. Аналіз контурів. Кожен контур аналізується за площею: якщо вона не перевищує певного мінімального порогу, такий контур ігнорується як шум або випадкове засвічення. Це дає змогу уникнути хибного виявлення яскравих ділянок, що не є Сонцем (наприклад, бліки чи відбиття).

5. Вибір найяскравішої ділянки. Із-поміж усіх валідних контурів обирається той, що має найбільшу середню яскравість і/або найбільшу площу. Визначається його центр (центроїд), який надалі використовується для обчислення кута повороту сервоприводу.

Рисунок 3.18 – Приклад аналізу зображення функцією find_brightest_area

На рисунку 3.14 ми можемо приклад роботи функції find_brightest_area, яка знаходить найяскравішу область на небі. Знайдена область виділена зеленим контуром. Цей результат буде використаний у наступному етапі для розрахунку кута повороту сонячної панелі.

3.2.3 Розрахунок нахилу сонячної панелі

Представлено код функції _run, яка відповідає за основну логіку динамічного керування положенням сонячної панелі. Ця функція працює у вигляді безперервного циклу в фоновому режимі, забезпечуючи постійне оновлення орієнтації панелі. Таким чином, система в реальному часі адаптує свою роботу до змін освітлення, забезпечуючи максимальну ефективність енергозбору.

Алгоритм роботи функції:

1. Захоплення зображення з камери здійснюється за допомогою методу capture, який отримує актуальний кадр для подальшої обробки. У разі, якщо камера тимчасово недоступна або не передає зображення, система призупиняє виконання на короткий проміжок часу та переходить до наступного циклу, забезпечуючи безперервність роботи без помилок.

2. Визначення режиму роботи здійснюється на основі заданих налаштувань: У ручному режимі система використовує координати азимуту та висоти, безпосередньо задані користувачем через інтерфейс керування. У солярному режимі положення Сонця обчислюється за допомогою астрономічного алгоритму на основі дати, часу та географічних координат. В основному режимі, який використовує машинний зір, система аналізує поточне зображення з камери для визначення найяскравішої області на знімку. На основі цього дані передаються на приводи для корекції орієнтації панелі.

3. Виявлення найяскравішої області. Викликається find_brightest_area, яка повертає координати найскравішої області. Якщо така область знайдена, відбувається розрахунок кутових зміщень відносно центру зображення з урахуванням поля зору камери.

4. Передача координат на серводвигуни. Після обчислення координат викликається servo.move_to — що повертає панель у нове положення.

3.2.4 Опис логіки збору статистичних даних

Для забезпечення достовірного аналізу ефективності роботи динамічного позиціювання сонячної панелі необхідно мати чіткий механізм збору та зберігання параметрів генерації електроенергії. У розробленій системі для цього використовується датчик INA219, який дозволяє у реальному часі вимірювати напругу на сонячному модулі, силу струму та розраховувати миттєву потужність.

Збір даних реалізовано за допомогою двох методів: measure та collect. Метод measure відповідає за зчитування поточних значень електричних параметрів. Він формує словник даних, що включає часову мітку у форматі ISO 8601, значення напруги, струму та обчислену потужність. Це дозволяє зв’язати кожне вимірювання з точним моментом часу, що критично важливо для побудови графіків і порівняння результатів між статичним та динамічним режимами.

Метод collect виконує основну роль у циклічному збиранні даних. Цикл працює у фоновому режимі з інтервалом у 60 секунд. У кожній ітерації виконується наступне:

1. Перевіряється дата. Якщо настав новий день, дані попереднього дня очищуються з оперативної пам’яті та повторно завантажуються при потребі.
2. Викликається метод measure, результат якого додається до списку накопичених вимірювань.
3. Вимірювання одразу зберігається у локальну базу даних SQLite шляхом SQL-запиту INSERT INTO power_measurements.

Таким чином, система забезпечує постійний моніторинг продуктивності сонячної панелі, формуючи надійний набір експериментальних даних. Ці дані згодом використовуються для побудови графіків добової потужності, порівняння різних режимів орієнтації (ручного, фіксованого, автоматичного) та верифікації загальної ефективності розробленої системи. Наявність бази даних також дозволяє зберігати архів за декілька днів, що є зручним при довготривалому тестуванні.

3.2.5 Модуль керування сервоприводами

У системі динамічного стеження за Сонцем важливим елементом є механізм керування орієнтацією сонячної панелі. Для цього створено окремий програмний модуль Servo, який здійснює поворот платформи за двома координатами — азимутом (горизонтальна площина) та висотою (вертикальна

площина). Керування відбувається через GPIO-інтерфейс Raspberry Pi з використанням бібліотеки pigpio, що дозволяє формувати високоточні PWM-сигнали.

Цей метод використовується зовнішніми модулями системи (наприклад, головною функцією _run) для подачі команди на переміщення панелі. Він обмежує значення азимуту та кута нахилу до допустимих меж:

азимут: від 100 до 360 градусів, з інверсією осі (360 – азимут) кут

нахилу: від 20 до 180 градусів

Окрему увагу в реалізації методу update приділено обмеженню швидкості повороту сервомеханізмів. Зміна положення відбувається поступово — з кроком у 1 градус та затримкою приблизно 70 мілісекунд між оновленнями. Такий підхід дозволяє уникнути різких стрибків у положенні панелі, які можуть призвести до:

1. Механічного зносу або пошкодження приводів при великих кутах зміщення.
2. Перевантаження по струму під час раптової зміни напрямку.

3.2.6 Веб інтерфейс

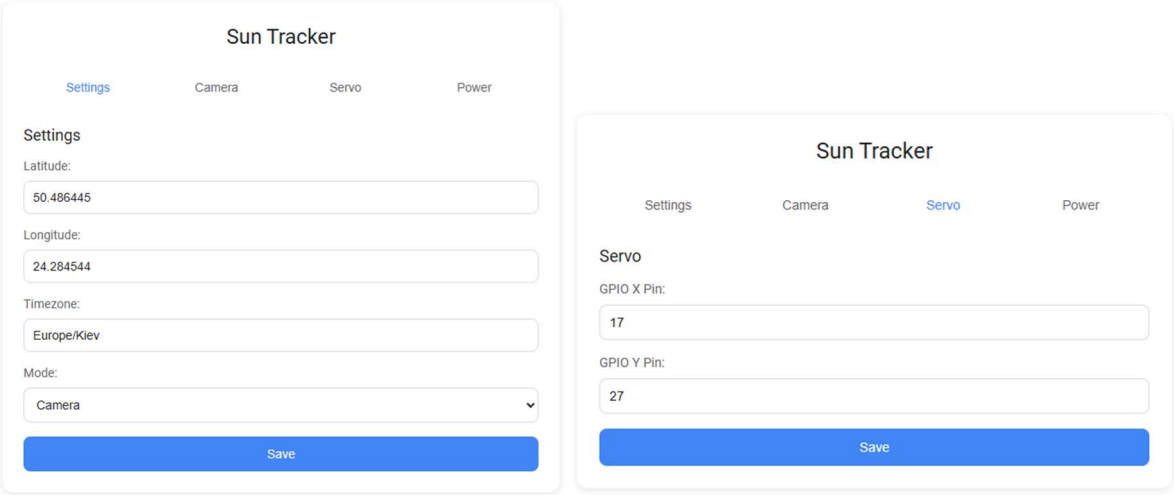

Рисунок 3.19 – Скріншоти web інтерфейсу для сонячної панелі

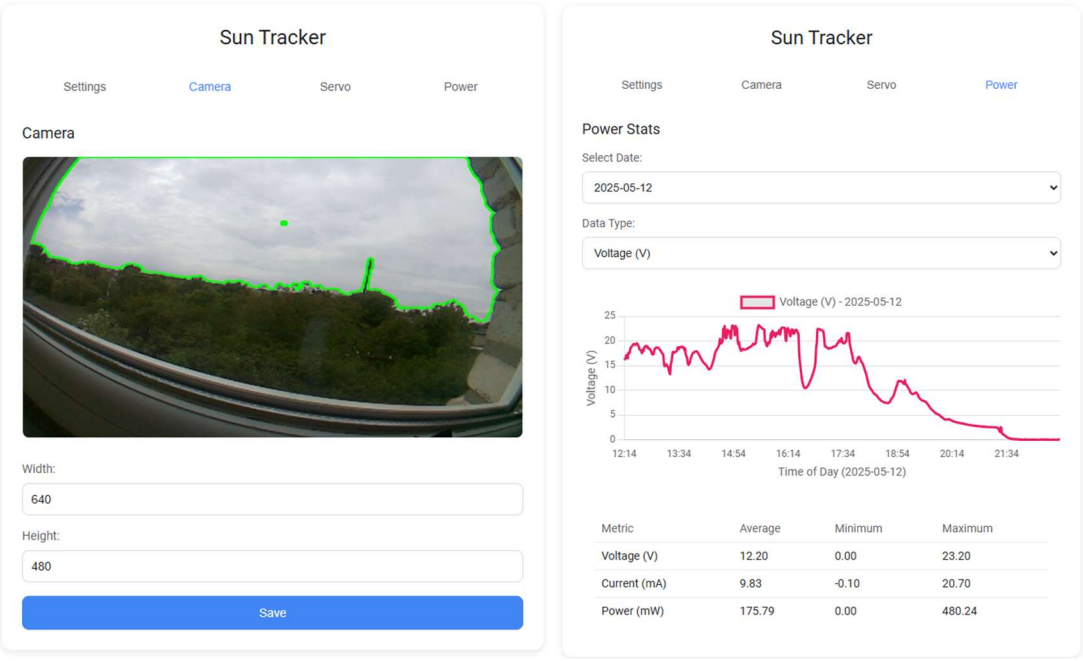

Рисунок 3.20 – Скріншоти web інтерфейсу для сонячної панелі

З метою спрощення налаштування та зручного керування системою динамічного орієнтування сонячної панелі було створено веб-інтерфейс на базі мікрофреймворку Flask із використанням технологій HTML та CSS. Його основне завдання — надати користувачу інтуїтивно зрозумілий доступ до функціоналу системи без необхідності взаємодії з програмним кодом. Інтерфейс дозволяє як спостерігати за роботою в реальному часі, так і управляти ключовими параметрами системи.

Однією з центральних можливостей веб-інтерфейсу є зміна положення панелі через три доступні режими роботи: автоматичний, напівавтоматичний та ручний. В автоматичному режимі система самостійно орієнтує панель на основі аналізу зображення з камери за допомогою алгоритмів машинного зору. Здійснюється оцінка яскравості пікселів, що дозволяє точно визначити положення джерела світла та скоригувати кут нахилу панелі. Напівавтоматичний режим використовує астрономічні обчислення, враховуючи час доби та координати місцевості для прогнозування положення Сонця. У ручному режимі користувач сам задає положення панелі за допомогою елементів керування у браузері.

Інтерфейс також надає можливість налаштовувати параметри відеопотоку, зокрема змінювати роздільну здатність камери. Це дозволяє оптимізувати співвідношення між якістю зображення та швидкістю обробки, залежно від обчислювальних ресурсів та умов освітлення.

Окремо реалізовано функцію візуалізації даних: статистика роботи системи відображається у вигляді графіків, що демонструють зміни положення панелі, рівень освітленості, напругу, струм та інші ключові показники. Такий підхід дає змогу контролювати ефективність роботи в реальному часі та оцінювати вплив зовнішніх факторів на продуктивність.

Особливу увагу приділено трансляції відеопотоку з камери безпосередньо у веб-інтерфейсі. Це зображення не лише відображається для користувача, а й використовується як вхідні дані для обчислювальних алгоритмів. На ньому можна спостерігати, як система ідентифікує яскраві області, обчислює їх координати та ухвалює рішення щодо руху приводу. Таким чином, веб-інтерфейс виконує роль повноцінного центру управління, моніторингу та візуалізації, забезпечуючи зручну взаємодію з системою та підвищуючи її функціональну гнучкість.

Висновки до розділу 3

Розроблено схему підключення всіх необхідних модулів до мікрокомп’ютера Raspberry Pi. У схемі передбачено підключення датчика струму INA219 до інтерфейсу I2C (GPIO 2, 3), сервоприводів до цифрових виходів GPIO 17 та GPIO 27 для керування положенням панелі. Підключення камери реалізовано через CSI-інтерфейс. Окремо враховано особливості живлення системи та стабілізації роботи елементів. Таким чином було розроблено повнофункціональний макет для тестування системи стеження з використанням машинного зору.

Розроблено програмне забезпечення з використанням Python та бібліотек з відкритим вихідним кодом, що забезпечує гнучкість та простоту підтримки

оновлень та масштабованість системи.

Зібрали двох осьову модель динамічної сонячної панелі, розробили програмне забезпечення OpenCV для аналізу зображення в реальному часі. Розробили зручний веб-інтерфейс для управління та стеження за сонячною панеллю.

4 ЕКСПЕРЕМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

4.1 Умови проведення експерименту

Експерименти проводились у реальних умовах на території львівської області.

Умови проведення експерименту:

Дата проведення експеременті: 29.05.2025

Час спостереження: 06:00-19:00

Температура повітря: +9…+17 C

Вологість повітря: 61%

Вітер: 8-19 км/год

Такі умови дозволили забезпечити стабільний доступ до сонячного світла протягом усього циклу спостереження, що є критично важливим для коректної роботи алгоритмів машинного зору та механізму стеження за Сонцем.

Для моніторингу електричних параметрів сонячної панелі використовувався прецизійний цифровий датчик INA219, який забезпечує вимірювання: напруги (V), сили струму (A), миттєвої потужності (W). Зчитування даних здійснювалося з частотою один раз на хвилину, що дозволяє отримати детальну картину змін продуктивності протягом усього періоду експерименту.

Усього за день було зібрано понад 700 повних записів, кожен з яких включав показники напруги, струму та обчисленої потужності. Зібрані дані автоматично передавалися до центрального обчислювального модуля — одноплатного комп’ютера Raspberry Pi, який виконував попередню обробку значень, зберігання в базу даних та побудову графіків у режимі реального часу.

Завдяки цьому стало можливим відстежувати продуктивність панелі залежно від положення Сонця, погодних умов і роботи самої системи стеження. Окрему увагу приділено стабільності передачі даних та точності вимірювання. Перед початком експерименту система була відкалібрована, а датчик INA219 —

протестований для уникнення систематичних похибок. Таким чином, отримані результати є репрезентативними для подібних умов експлуатації й можуть бути використані для подальшого порівняння між статичними та динамічними системами орієнтації.

4.2 Обробка та аналіз результатів

Рисунок 4.1 – Графік потужності (Вт) динамічної панелі протягом дня

На графіку видно чітке переважання зеленої лінії над фіолетовою впродовж усього дня.

Найбільший приріст потужності спостерігався у ранкові години (06:00–11:00) та ввечері (15:00–19:00), коли кут падіння сонячного світла є особливо критичним. У ці періоди динамічна система орієнтується перпендикулярно до Сонця, тоді як статична панель втрачає до 30–40% продуктивності. У період найвищої сонячної активності (11:30–14:30) розрив зменшується, але динамічна панель все одно демонструє перевагу, забезпечуючи більш рівномірне навантаження та кращу компенсацію змін хмарності. Загальна вироблена енергія динамічною панеллю за весь день була в середньому на 27% вищою, ніж у випадку зі статичною конфігурацією.

Висновки до розділу 4

Результати експерементальних досліджень підтвердили ефективність використання машинного зору для орієнтації сонячних панелей. Алгоритми машинного зору, в поєднанні з динамічним позиціюванням, забезпечують точне наведення панелі на джерело світла, що безпосередньо відображається на збільшенні генерації електроенергії. У процесі експерементальних досліджень було зафіксовано збільшення ефективності генерації електроенергії в порівнянні з тридиційними системами стеження.

Завдяки прямому аналізу візуальної інформації система здатна реагувати на зміни інтенсивності освітлення в режимі реального часу. Також завдяки цим даним ми можемо уникати перешкод.

ВИСНОВКИ

У межах виконаної роботи було реалізовано повноцінну систему динамічного орієнтування сонячної панелі з використанням технологій машинного зору.

На першому етапі розглянули проблематику статичних сонячних панелей, розглянули проблеми популярних трекерів. Розглянули доцільність використання машинного зору в контексті сонячних панелей.

На другому етапі було здійснено підбір, закупівлю та збирання всіх необхідних апаратних компонентів, сервоприводів, камер, датчиків напруги та струму (INA219), контролером на базі Raspberry Pi та джерелом живлення. Також було розроблено програмне забезпечення, яке включає модулі обробки відеопотоку, аналізу зображення, розрахунку кутових відхилень, а також керування приводами панелі з урахуванням отриманих координат. Додатково було реалізовано функцію розрахунку положення Сонця на основі астрономічних даних, збір статистики енергетичних параметрів та створено базу даних для зберігання телеметрії.

На третьому етапі було розроблено програмне забезпечення, яке включає модулі обробки відеопотоку, виявлення найяскравішої області зображення, розрахунку кутових відхилень, а також керування приводами панелі з урахуванням отриманих координат. Додатково було реалізовано функцію розрахунку положення Сонця на основі астрономічних даних, збір статистики енергетичних параметрів та створено базу даних для зберігання телеметрії.

На завершальному було проведено експериментальні дослідження в реальних умовах етапі було проведено аналіз зібраної інформації, який підтвердив ефективність застосування машинного зору для задач орієнтації сонячних панелей. Розроблена система продемонструвала стабільну роботу, здатність адаптуватися до змін зовнішніх умов та забезпечила істотне зростання виробленої електроенергії порівняно з фіксованим варіантом.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Українська енергетика. Сонячна електростанція: значні витрати заради сталого енергопостачання [Електронний ресурс]. – Режим доступу: https://ua-energy.org/uk/posts/soniachna-elektrostantsiia-znachni-vytraty-zarady-staloho-enerhopostachannia.

2. SolarPath. Сонячна енергетика України: зростання та виклики у 2025 році [Електронний ресурс]. – Режим доступу: https://solarpath.com.ua/news/sonyachna-energetika-ukrayini-zrostannya-ta-vikliki-u-2025-roczi/.

3. Arduino.ua. Цифровий датчик струму та напруги на INA219 з шиною I2C [Електронний ресурс]. – Режим доступу: https://arduino.ua/prod1661-cifrovoi-datchik-toka-i-napryajeniya-na-ina219-s-shinoi-i2c.

4. Waveshare. Polysilicon Solar Panel (18V 10W), High Conversion Efficiency [Електронний ресурс]. – Режим доступу: https://www.waveshare.com/solar-panel-18v-10w.htm.

5. Arduino.ua. Сервопривід Tower Pro MG995 180 MG995-180 [Електронний ресурс]. – Режим доступу: https://arduino.ua/prod2551-servoprivid-tower-pro-mg995-180.

6. Евоком.юа. Камера RPi OV5647 [Електронний ресурс]. – Режим доступу: https://evo.net.ua/kamera-rpi-ov5647-fisheye-lens-camera/.

7. Евоком.юа. Мікрокомп’ютер Raspberry Pi 4 Model B 8GB [Електронний ресурс]. – Режим доступу: https://evo.net.ua/mikrokomputer-raspberry-pi-4-model-b-8gb/.

8. Arduino.ua. Кронштейни кріплення 2-DOF Pan Tilt Kit для сервоприводів MG995, MG996R, MG945, MG946R [Електронний ресурс]. – Режим доступу: https://arduino.ua/prod5997-kronshteini-krepleniya-2-dof-pan-tilt-kit-dlya-servomotorov-mg995-mg996r-mg945-mg946r.

9. Arduino.ua. Монтажна втулка для серводвигунів 20 мм [Електронний ресурс]. – Режим доступу: https://arduino.ua/prod2910-montajnaya-vtylka-dlya-servodvigatelei-20-mm.

10. U.S. Department of Energy Office of Scientific and Technical Information. Performance of Bifacial Photovoltaic Modules on a Dual-Axis Tracker in a High-Latitude, High-Albedo Environment [Електронний ресурс]. – Режим доступу: https://www.osti.gov/servlets/purl/1641282.

11. Autodesk Instructables. LDR Photovoltaic Solar Tracker [Електронний ресурс]. – Режим доступу: https://www.instructables.com/LDR-Photovoltaic-Solar-Tracker/.

12. National Renewable Energy Laboratory (NREL). Solar Position Algorithm for Solar Radiation Applications [Електронний ресурс]. – Режим доступу: https://docs.nrel.gov/docs/fy08osti/34302.pdf.

13. ScienceDirect. Impact of dust and tilt angle on the photovoltaic performance in a desert environment [Електронний ресурс]. – Режим доступу: https://www.sciencedirect.com/science/article/pii/S0038092X25000027.

14. International Journal of Advance Research, Ideas and Innovations in Technology (IJARIIT). Automatic dual-axis solar tracker system using Image Processing [Електронний ресурс]. – Режим доступу: https://www.ijariit.com/manuscript/automatic-dual-axis-solar-tracker-system-using-image-processing/.

ДОДАТОК А – Охорона праці

У процесі створення та експлуатації будь-якої технічної системи, зокрема системи динамічного стеження за Сонцем, охорона праці є не просто регламентованим обов’язком, а ключовим елементом стійкості, довговічності та безпеки системи. Експлуатація обладнання без належного врахування ризиків створює передумови для аварійних ситуацій, збоїв у роботі й прямих загроз для персоналу. Описані нижче заходи не лише формують базу безпечного функціонування, а й забезпечують зменшення простоїв, підвищення продуктивності й збереження обладнання у справному стані.

А.1 Електробезпека та живлення

У системах, що містять фотоелектричні елементи, контролери, виконавчі механізми (зокрема, серводвигуни), питання електробезпеки постає як пріоритетне. Джерела живлення мають бути обладнані захистом від перенапруги, короткого замикання та перевантаження.

Рекомендовано:

1. Застосування ізольованих джерел живлення з гальванічною розв’язкою;
2. Установлення автоматичних вимикачів та плавких запобіжників у всіх лініях живлення;
3. Вимкнення живлення перед обслуговуванням будь-яких елементів системи;
4. Використання маркування та кольорової індикації кабелів;
5. Перевірка контурів заземлення за допомогою тестерів опору.

Порушення вищезазначених вимог призводило до задокументованих випадків виходу з ладу контролерів, зниження стабільності живлення і, як наслідок, втрати частини добової генерації. У системах, встановлених у

відкритих локаціях, надмірне навантаження призводило також до перегріву кабелів та плавлення ізоляції, що зафіксовано в акті №43/2023 Інституту енергетичних досліджень.

А.2 Стандарти монтажу та ризики при встановленні панелей

Монтаж фотоелектричних панелей супроводжується фізичними навантаженнями, роботою на висоті, використанням інструментів з підвищеним ризиком травматизму. У літній період підвищується небезпека теплового або сонячного удару.

З метою запобігання негативним наслідкам слід:

1. Проводити встановлення в ранкові години або у тіні, з регулярними перервами.
2. Забезпечити наявність питної води, головних уборів і аптечки.
3. Обмежити тривале перебування на відкритому сонці без спецодягу.
4. Організувати контроль з боку відповідального за безпеку робіт.

При виявленні ознак перегріву (головний біль, нудота, сплутаність свідомості) необхідно негайно припинити роботу, перейти в затінене місце, забезпечити охолодження та за необхідності викликати швидку медичну допомогу. Під час навчальних експериментів у червні 2024 року було зафіксовано два випадки перегріву монтажників, один з яких завершився короткочасною госпіталізацією. Це є доказом необхідності дотримання протоколу безпеки при роботі в умовах високої температури.

А.3 Пило та вологозахист компонентів

Електроніка, що експлуатується в умовах вулиці або промислових приміщень, має бути захищена від зовнішніх впливів. Волога, пил, перепади температур є причинами передчасного виходу з ладу сенсорів, приводів, контактних груп.

Для цього рекомендується:

1. Використання герметичних корпусів з рейтингом IP65 або вище;
2. Обробка контактів антикорозійними складами;
3. Застосування силікагелю у внутрішніх відсіках корпусів для боротьби з конденсатом;
4. Організація вентиляції з фільтрацією повітря.

Це підтверджується численними дослідженнями, згідно з якими обладнання з вологозахистом працює у 3–5 разів довше без необхідності сервісного втручання.

А.4 Регламент технічного обслуговування

Кожен об'єкт, що містить рухомі й електронні частини, потребує періодичного обслуговування. Ігнорування графіків профілактики веде до зниження точності роботи трекера, зменшення ККД панелі, збоїв у роботі сенсорів.

Необхідно:

1. Оновлювати прошивку контролера щонайменше раз на 3 місяці;
2. Здійснювати очищення оптики камери раз на 2 тижні;
3. Проводити діагностику сервоприводів із перевіркою люфтів та моменту обертання;
4. Вести журнал обслуговування з підписом відповідального інженера.

А.5 Підготовка персоналу та документація

Усі особи, що працюють із системою, повинні проходити навчання щодо:

1. Програмної архітектури системи.
2. Механіки обертання платформи.
3. Безпечного обслуговування та реагування на помилки.
4. Алгоритмів обробки зображення і їх особливостей.

Навчання фіксується у вигляді сертифікатів, а повторна атестація проводиться щорічно. Всі зміни в системі мають супроводжуватися оновленням документації. Журнали налаштувань, обслуговування та аварій зберігаються не менше 1 року. Наявність актуальної документації дозволяє проводити оперативне оновлення системи, виявляти слабкі місця, а також виступає основою для внутрішнього аудиту безпеки.

А.6 Висновки

Застосування систем динамічного стеження за Сонцем з використанням машинного зору можливе лише за умов дотримання комплексних заходів безпеки. Це охоплює проектування, монтаж, експлуатацію, документацію та навчання персоналу. Досвід експлуатації подібних систем демонструє, що вкладення в охорону праці дозволяє не лише мінімізувати ризики, але й суттєво підвищити стабільність і економічну ефективність функціонування сонячної енергетичної установки. Дотримання вищезазначених вимог — це не тільки нормативна необхідність, а й практичне підтвердження ефективного управління технічними ризиками на всіх етапах життєвого циклу системи.

ДОДАТОК Б – Перелік використаних компонентів

Таблиця Б.1 – Перелік використаних компонентів

Використані модулі	Кількість
Raspberry Pi 4 Model B	1
Резистор 1 кОм	1
Світлодіод	1
Silicon Solar PV module 18V10W	1
Сервопривід MG996R	2
Датчик струму та напруги INA219	1
Кронштейн U-Shaped Long	2
Кронштейн Multi-Purpose	1
Монтажна втулка	2
RPi Camera (G)	1
Штатив для камери	1

ДОДАТОК В – Програмний код

```
import time
import pigpio
import logging

from threading import Thread

from src.config import Config
from src.singleton import Singleton

class Servo(metaclass=Singleton):
    def __init__(self):
        self.config = Config()

        self.pi = pigpio.pi()

        self.gpio_x = int(self.config.gpio_x)
        self.gpio_y = int(self.config.gpio_y)

        self.pi.set_mode(self.gpio_x, pigpio.OUTPUT)
        self.pi.set_mode(self.gpio_y, pigpio.OUTPUT)

        self.current_azimuth = 0
        self.current_altitude = 0
        self.target_azimuth = 0
        self.target_altitude = 0

        self._thread = Thread(target=self.update, daemon=True)
```

```
    self._thread.start()

def move_to(self, azimuth, altitude):
    azimuth = max(100, min(azimuth, 360))
    azimuth = 360 - azimuth

    altitude = max(20, min(altitude, 180))

    self.target_azimuth = azimuth
    self.target_altitude = altitude

def get_current_position(self):
    return self.current_azimuth, self.current_altitude

def update(self):
    step = 1
    delay = 0.07

    while True:
        az_diff = self.target_azimuth - self.current_azimuth
        alt_diff = self.target_altitude - self.current_altitude

        if abs(az_diff) < step and abs(alt_diff) < step:
            time.sleep(delay)
            continue

        az_step = step if az_diff > 0 else -step if az_diff < 0 else 0
        alt_step = step if alt_diff > 0 else -step if alt_diff < 0 else 0

        self.current_azimuth += az_step
```

```
            self.current_altitude += alt_step

            az_pw = 500 + (2000 * int(self.current_azimuth)) // 270
            alt_pw = 500 + (2000 * int(self.current_altitude)) // 180

            az_pw = max(500, min(az_pw, 2500))
            alt_pw = max(500, min(alt_pw, 2500))

            self.pi.set_servo_pulsewidth(self.gpio_x, az_pw)
            self.pi.set_servo_pulsewidth(self.gpio_y, alt_pw)

            time.sleep(delay)

    def release(self):
        self.pi.set_servo_pulsewidth(self.gpio_x, 0)
        self.pi.set_servo_pulsewidth(self.gpio_y, 0)
        self.pi.stop()

        logging.debug("Servo released")

import cv2
import time
import logging
from threading import Thread

from src.config import Config
from src.camera import Camera
from src.astronomy import Astronomy
from src.display import Display
from src.servo import Servo
```

```
class SunTracker:
    def __init__(self):
        self.config = Config()
        self.camera = Camera()
        self.astronomy = Astronomy()
        self.display = Display()
        self.servo = Servo()
        self.running = True
        self._thread = None
        self._last_frame = None

        self._thread = Thread(target=self._run, daemon=True)
        self._thread.start()

    def _run(self):
        while self.running:
            frame = self.camera.capture()
            if frame is None:
                logging.error("No frame captured, skipping")
                time.sleep(0.5)
                continue

            mode = self.config.mode
            azimuth = 0
            altitude = 0
            x = None
            y = None

            if mode == "manual":
```

```
            azimuth = self.config.manual_azimuth
            altitude = self.config.manual_altitude
            logging.debug(f"Manual mode: azimuth={azimuth:.1f},
altitude={altitude:.1f}")
        elif mode == "solar":
            azimuth, altitude = self.astronomy.get_sun_position()
            logging.debug(f"Solar mode: azimuth={azimuth:.1f},
altitude={altitude:.1f}")
        else:
            x, y = self.camera.find_brightest_area(frame)
            if x is not None and y is not None:
                fov_h = 30
                fov_v = 22.5

                norm_x = (x / self.camera.width) - 0.5
                norm_y = (y / self.camera.height) - 0.5

                angular_offset_x = norm_x * fov_h
                angular_offset_y = -norm_y * fov_v

                current_azimuth, current_altitude = self.servo.get_current_position()

                azimuth = current_azimuth + angular_offset_x
                altitude = current_altitude + angular_offset_y

                logging.debug(f"Camera mode: cx={x}, cy={y},
azimuth={azimuth:.1f}, altitude={altitude:.1f}")
            else:
                azimuth, altitude = self.astronomy.get_sun_position()
                logging.debug("No bright area found, using solar position")
```

```
            self.servo.move_to(azimuth, altitude)
            logging.debug(f"Servo moving to: azimuth={azimuth:.1f},
altitude={altitude:.1f}")

            self._last_frame = self.display.draw(frame.copy(), x, y)

            time.sleep(0.5)

    def stream_video(self):
        while self.running:
            frame = self._last_frame
            if frame is None:
                frame = self.camera.capture()
                if frame is None:
                    logging.error("No frame for streaming, retrying")
                    time.sleep(0.1)
                    continue

            ret, jpeg = cv2.imencode('.jpg', frame,
[int(cv2.IMWRITE_JPEG_QUALITY), 90])
            if ret:
                yield (b'--frame\r\n' + b'Content-Type: image/jpeg\r\n\r\n' + jpeg.tobytes()
+ b'\r\n')

            time.sleep(0.1)

    def release(self):
        self.running = False
```

```
        self.camera.release()
        self.display.release()
        self.servo.release()

        logging.debug("SunTracker released")

import os
import cv2
import logging
import numpy as np
from picamera2 import Picamera2

from src.config import Config
from src.singleton import Singleton

class Camera():
    def __init__(self):
        self.config = Config()
        self.width = int(self.config.camera_width)
        self.height = int(self.config.camera_height)
        self.camera = None

        if not os.path.exists("/dev/video0"):
            logging.error("Camera device /dev/video0 not found")
            return

        try:
            if self.camera:
                self.release()
```

```
            self.camera = Picamera2()
            config = self.camera.create_preview_configuration(main={"size": (self.width,
self.height)})
            self.camera.configure(config)
            self.camera.start()

            logging.info(f"Camera initialized: {self.width}x{self.height}")
        except Exception as e:
            logging.error(f"Camera init failed: {e}")
            self.camera = None

    def capture(self):
        if not self.camera:
            logging.error("Camera not initialized")
            return None

        try:
            frame = self.camera.capture_array()
            if frame is None or frame.size == 0:
                logging.error("Captured frame is empty")
                return None
            return cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
        except Exception as e:
            logging.error(f"Capture failed: {e}")
            return None

    def find_brightest_area(self, frame):
        if frame is None:
            logging.error("No frame for sun detection")
            return None, None
```

```
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    blurred = cv2.GaussianBlur(gray, (11, 11), 0)

    _, thresh = cv2.threshold(blurred, 150, 255, cv2.THRESH_BINARY)
    contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

    if not contours:
        logging.debug("No bright contours found")
        return None, None

    max_brightness = 0
    best_contour = None
    for contour in contours:
        mask = np.zeros_like(gray)
        cv2.drawContours(mask, [contour], -1, 255, -1)
        brightness = np.sum(gray[mask == 255])
        if brightness > max_brightness:
            max_brightness = brightness
            best_contour = contour

    if best_contour is None or max_brightness < 10000:
        logging.debug("No sufficiently bright contour found")
        return None, None

    M = cv2.moments(best_contour)
    if M["m00"] == 0:
        logging.debug("Invalid contour moment")
        return None, None
```

```
        cx = int(M["m10"] / M["m00"])
        cy = int(M["m01"] / M["m00"])
        logging.debug(f"Brightest region: cx={cx}, cy={cy},
brightness={max_brightness}")

        return cx, cy

    def release(self):
        if self.camera:
            try:
                self.camera.stop()
                self.camera.close()
            except Exception as e:
                logging.error(f"Camera release failed: {e}")
            finally:
                self.camera = None
```