«Автономна відеокамера»

**Прототип навчальної системи комп’ютерного зору**

Чернігів 2021

**Анотація**

В даній роботі був розроблений прототип системи комп’ютерного зору для навчальної лабораторії. Головне завдання полягало в отриманні відео потоку з модуля камери RaspberryPiCamerav2 та в з’ясуванні можливості керувати положенням камери відносно осей координатОX та ОY. Для вирішення завдання в якості керуючого елемента обрано мікрокомп’ютер RaspberryPi. В основі електромеханічної системи було використано кроковий двигун для повороту вздовж осі Х та серводвигун для повороту вздовж осі Y.

Розроблена система має унікальні можливості для застосування в навчальному процесі студентів із спеціальностей 171 Електроніка та 121 Інженерія програмного забезпечення.

Робота представленазвітом на 30с.; рисунків – 13,формул – 3,інформаційних джерел – 15, додатків – 2.

**Зміст**

[Вступ 4](#_Toc64819372)

[1 Огляд інформаційних джерел за темою дослідження 6](#_Toc64819373)

[1.1 Особливості відеокамер, які застосовуються для комп’ютерного зору 6](#_Toc64819374)

[1.2 Програмне забезпечення, використовуване для комп’ютерного зору 7](#_Toc64819375)

[Висновки до розділу 1 8](#_Toc64819376)

[2 Налаштування відеокамери RPicamera 9](#_Toc64819377)

[2.1 Отримання відео потоку з камери 9](#_Toc64819378)

[2.2 Методи відображення відео потоку 10](#_Toc64819379)

[Висновки до розділу 2 12](#_Toc64819380)

[3 Реалізація базових елементів комп’ютерного зору за допомогою RaspberryPi 13](#_Toc64819381)

[3.1 Апаратна реалізація комп’ютерного зору 13](#_Toc64819382)

[3.2 Графічний інтерфейс користувача 15](#_Toc64819383)

[3.3 Програмна реалізація комп’ютерного зору 18](#_Toc64819384)

[Висновки до розділу 3 20](#_Toc64819385)

[Висновки 21](#_Toc64819386)

[Список посилань 22](#_Toc64819387)

[Додатки 24](#_Toc64819388)

[Додаток А – Структура розробленої системи 24](#_Toc64819389)

[Додаток Б – Текст програми 25](#_Toc64819390)

# Вступ

Науково-технічний прогрес сьогодні визначають розробки в галузі штучного інтелекту, машинного навчання та інтернету речей. Як для практичного втілення наукових результатів, так і для подальшихдосліджень потрібні висококваліфікованіспеціалісти з широким світоглядом та належними практичними навичками.Підготовка таких спеціалістів – недешева річ, яка потребує створеннявідповідних апаратно-програмних комплексів для підтримки сучасного освітнього процесу. В цьому і полягає задача створення навчально-наукової лабораторії «Системи комп’ютерного зору та відображення інформації». Важливою частиною є апаратна платформа, яка лежатиме в основі майбутніх електронних тамехатроннихсистем, а також систем автоматизації та роботів. В якості такої апаратної платформи обрано мікрокомп’ютерRaspberryPi з наступним підключенням до нього різноманітних периферійних модулів. Для розробок в галузі комп’ютерного зору, насамперед, знадобиться камера та електромеханічні пристрої.

**Мета роботи** – захоплення відео потоку з камери RaspberryPiCamerav2 та відображення його на портативному дисплеї, який може бути підключеним до мікрокомп’ютера. В основі отримання відео потоку лежить використання бібліотеки pythonpicamera.Окремі наукові результати роботи опубліковані в [1, 2]

**Основні задачі**, які потрібно вирішити для досягнення поставленої мети:

* вибір моделі відеокамери, яку можна використовувати в галузі комп’ютерного зору;
* підключення відеокамери домікрокомп’ютера;
* програмне налаштування системи комп’ютерного зору;
* експериментальна перевірка та зняття характеристик розробленої системи.

**Наукова новизна** розробки полягає в тому, що вперше реалізовано двокоординатне позиціювання підключеної до мікрокомп’ютерамалогабаритної відеокамери, що дозволяє практично досліджувати як методи цифрової обробки зображень, так і віддаленого керування електроприводом.

**Практична цінність** роботи полягає в можливості її подальшого використання в освітньому процесів циклах лабораторних робіт з дисциплін, пов’язаних із автоматикою, робототехнікою та мехатронікою.

**Об’єктом дослідження** є сукупність програмно-апаратних засобів, які використовуються в системах комп’ютерного зору.

**Предмет дослідження** – організація та використання відеопотоку з відеокамери, яка взаємодіє з мікрокомп’ютером RaspberryPi.

**Апробація роботи.**Робота доповідалась на:

науково-практичній конференції«Новітні технології сучасного суспільства» (НТСС-2019), м. Чернігів, 12 грудня 2019 р.;

І міжнародній науково-практичній конференції«Новітні технології сучасного суспільства» (НТСС-2020), м. Чернігів, 17 грудня 2020 р.

# Огляд інформаційних джерел за темою дослідження

## Особливості відеокамер, які застосовуються для комп’ютерного зору

Відеокамера для комп’ютерного зору є одним із самих важливих периферійних компонентів у складіспеціалізованої комп’ютерної системи, яка може виконувати задачі в галузі автоматизації, робототехніки та мехатроніки. Саме тому відеокамері слідприділити найбільше уваги.

Модулі камери RaspberryPi є офіційними продуктами RaspberryPiFoundation[3]. Головною перевагоюцих модулівпорівняно зUSB камерамиє можливість підключення безпосередньо до відеочіпуVideoCoreодноплатних комп'ютерів,що економить системні ресурсиRaspberry.При цьому USB-порти залишаються вільними для іншої периферії.

Підключення камерного модуля до RaspberryPi здійснюється через FFC-шлейф в роз’єм CSI (CameraSerialInterface).

Прикладом такого модуля являється RaspberryPiCamera v2[4]. Вона є компактною камерою, якаобладнанавосьмимегапіксельним сенсором Sony IMX219 Exmor. Він дозволяє захоплювати, записувати і транслювати відео в форматах 1080p, 720p і VGA. Максимальна роздільна здатність для фотографій досягає 3280 × 2464 пікселів. Дана камера підходиться для запису при денному світлі або для місць з достатнім освітленням.

Версія NoIR виконана в чорному кольорі і відрізняється відCamera v2 матрицею. Це дозволяє знімати в темряві [4].

У випадку, коли якість фото та відео має ключове значення у підборі обладнання, використовують камеру RaspberryPiHighQuality[5]. Головною особливістю даної камери на 12 Мп є алюмінієвий байонет для установки змінних об'єктивів C / CS-mount і кріплення на штатив. Дану камеру використовують для відеоспостереження, зйомки дикої природи або фотографування панорамних пейзажів[4].

Увипадку, коли камера знаходиться на відстані від апаратної частини, використовують веб-камеруUSB. Дану камеру можна використовувати тільки за наявності драйверів для операційної системи Linux. Головним недоліком цієї камери є те, що її якість гірше, а налаштування важче порівняно зспеціальним модулем камери [6].

Поєднання достатньо потужного мікрокомп'ютера та відеокамеривідкриває широкі можливості застосування системи комп'ютерного зору врізноманітних пристроях з автономним живленням, зокрема, в роботах,дронах [2] та ін.

## Програмне забезпечення, використовуване для комп’ютерного зору

Головною задачею програмного забезпечення є обробка зображення, Воно обробляється для зменшення шумуабо для перетворення багатьох відтінків сірого у просте поєднання чорного та білого (бінаризація). Після початкової обробки програма зчитує, вимірює та / або визначає об'єкти, розміри, дефекти та інші властивості зображення.

Для отримання елементарних можливостей в обробці зображення використовують Pythonбібліотеку picamera[7]. Дана бібліотека дає можливість працювати з RaspberryPi та модулем камери. Основними можливостями є:

* захоплення зображення;
* запис відео;
* захоплення в масив numpy;
* Інтернет-трансляція[2].

В комп’ютерному зорі широкого застосування набула бібліотека з відкритим вихідним кодом OpenCV (OpenSourceComputerVision)[8]. Вона написана на мові високого рівня (C/C++) і містить алгоритми для інтерпретації зображень, калібрування камери за зразком, усунення оптичних спотворень, визначення подібності, аналізу переміщення об'єкта, визначення форми об'єкту та стеження за об'єктом, 3D-реконструкції, сегментації об'єкту, розпізнавання жестів тощо. Дана бібліотека має інтерфейси C++, Python, Javaта MATLAB, і підтримує операційні системи Windows, Linux, Android і Mac OS, що робить її більш універсальною та дозволяє працювати на різноманітнихапаратних платформах [8].

Для роботи з комп’ютерним зором використовують MATLAB. Це мова програмування, яка поєднує середовище робочого столу, налаштоване для інтерактивного аналізу і процесів проектування, з мовою програмування, яка безпосередньо виражає математику матриць. MATLABдає змогу розробляти рішення для зору за допомогою комплексного набору стандартних еталонних алгоритмів для обробки зображень, комп'ютерного зору і глибокого навчання. Головною перевагою MATLAB над OpenCV є зручність розробки та подання даних, хочашвидкодія алгоритмів може бути нижчою [9].

## Висновки до розділу 1

На основі проведеного огляду джерел інформації за темою дослідження встановлено:

1. Для отримання відео потоку за допомогою RaspberryPiдоцільніше використовувати модульні камери.
2. Для отримання та перетвореннявідео потоку з модульної камери найліпшевикористовувати бібліотекуPicameraдля RaspberryPi.
3. В сучасних розробках систем комп’ютерного зору найбільш поширеними єMATLAB та OpenCV.
4. OpenCV – найбільш повна та відкрита бібліотека, яказручна длястворення готових до роботи в реальному часіпроектів комп'ютерного зору.
5. MATLABнайліпше підходить для пошукових досліджень концепції комп’ютерного зору.

# Налаштування відеокамери RPicamera

## Отримання відеопотоку з камери

Перед тим, як отримати відеопотік з камери, потрібно налаштувати її.

• Підключити камеру до RaspberryPi в роз’єм CSI;

• Запустити RaspberryPi;

• Зайти в меню конфігурації параметрів Меню / Налаштування/ RaspberryPiConfiguration;

• Перейти на вкладку Interfaces та перемкнутирадіо кнопкуCamera в положення Enabled.

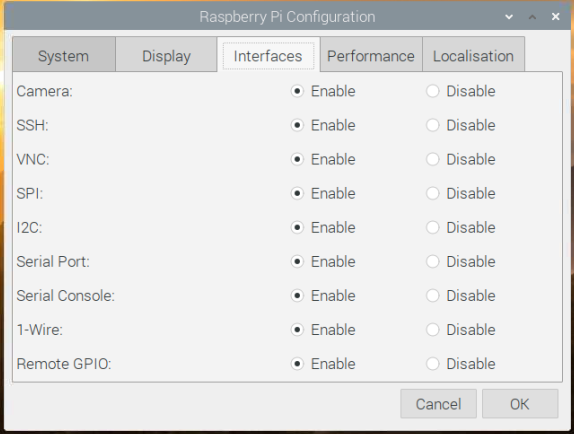


Рисунок 2.1 – Дозвіл доступу до інтерфейсів

Для перевірки працездатності камери в терміналі треба ввести наступну команду:



Рисунок 2.2 – Команда для захоплення зображення

Після успішного виконання команди протягом десяти секунд відбуватиметься трансляція з камери. Якщо підключення виконане через віддалений доступ, відео з камери отриматине вдасться.

Для того, щоб зберегти зображення з камери потрібно ввести команду, в якій задаємо шлях та назву для збереження зображення (рисунок 2.3). На рисунку 2.4. продемонстровано, що файл зберігся на робочому столі з назвою img111. Аналогічноіз збереженням зображення можна зберегти відео[3].

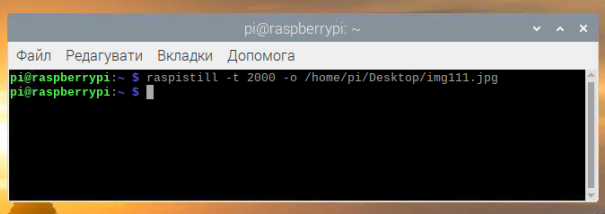


Рисунок 2.3 – Термінал RaspberryPi

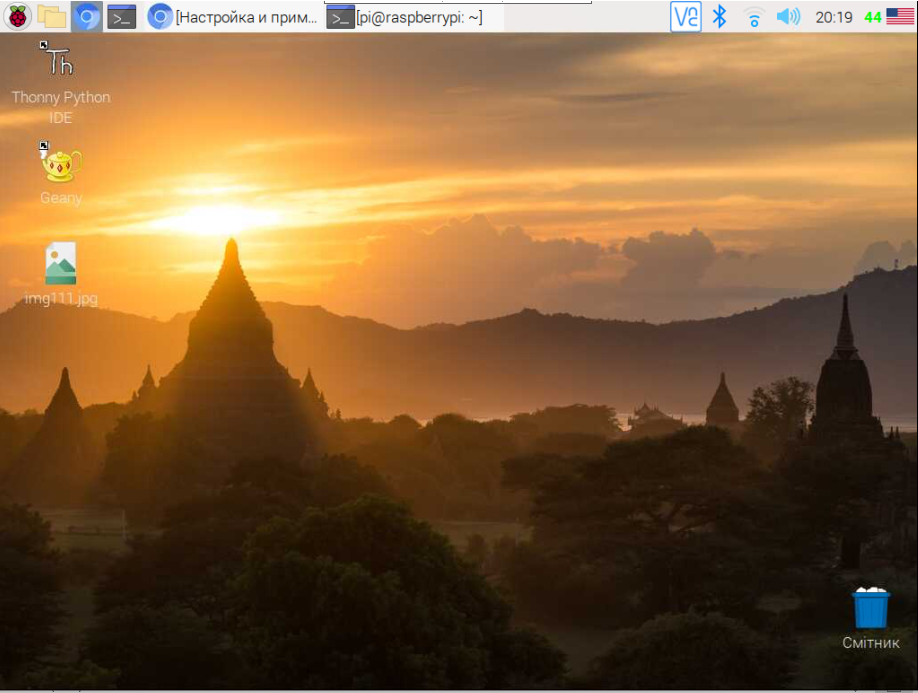


Рисунок 2.4 – Робочий стіл після збереження зображення

## Методи відображення відеопотоку

Отримати відеопотік з камери за допомогою програми можна декількома варіантами.

Варіант перший оснований на бібліотеці picamera. Функціонал цього метода аналогічний тому, як і під час запуску камериз консолі.

При запуску коду, приведеного нижче, отримаємо відображення відео на екрані поверх всіх віконпротягом 20 секунд. Головний недолік цього метода полягає в наступному: побачити відео з камери можна тільки, коли екран або монітор підключені[7].

Текст програми:

from time import sleep # підключаємо бібліотеку часу

from picamera import PiCamera # підключаемо бібліотеку камери

camera = PiCamera()

camera.start\_preview() # запуск камери

sleep(20)

camera.stop\_preview() # закінченя виконання програми

Другий варіант полягає в використанні стандартної бібліотеки Pythonhttp.server. Дана бібліотека дозволяє користувачеві створювати власний клас обробника HTTP-запитів для обробки запитів GET і POST за допомогою реалізаціїBaseHTTPRequestHandler класу. Головний недолікhttp.server – це неможливість обробки великого трафіку. В даному методі відео з камери отримується за допомогою бібліотеки Picamera. А далі, за допомогою бібліотеки http.server відео транслюється в браузер[10,11].

Щоб побачити відео, потрібно запустити скрипт та відкрити браузер, в якому в пошуковій стрічці треба написати IP-адресу RaspberryPi та добавити :8000. Наприклад, 192.168.1.4:8000 призведе до запуску HTML розмітки, яка прописана в скрипті.

Третій варіант заснований на використанні веб-фреймворкуPython під назвою Flask для того, щоб перетворити RaspberryPi в динамічний веб-сервер. Даний фреймворк дозволяє взяти існуючі скриптиPythonта додати шаблон HTML, щоб надати кінцевому користувачеві інтерфейс веб-сторінки.

Для використання Flask потрібно створити середовище веб-сервера. Наприклад, використовуючи наступну структуру(рисунок 2.5).

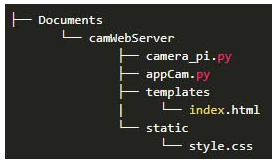


Рисунок 2.5 – Структура веб-сервера

appCam.py – це скрипт, в якому за допомогою Flask реалізується надання потокового відео:;

camera\_pi.py – скрипт, який отримує потік кадрів з камери.

В папці templates знаходиться файл розмітки веб сторінки index.html.

У файлі style.css прописані стилі для html-розмітки. Якщо стилів небагато, то можна їх написати в файлі index.html[1, 12,13].

## Висновки до розділу 2

1. Налаштування модуля камери виконано за допомогою терміналу операційної системи Raspbian, яка встановлена на RaspberryPi.
2. Було перевірено, всі вище перераховані методи отримання зображення відео потоку. Але в проекті реалізованометод який не передбачає використання веб сервісів, в зв’язку з тим, що даний проект не потребує віддалене керування системою.
3. Отримання відеопотоку з модуля камериRPicamera виконується за допомогою бібліотеки Picamera.

# Реалізація базових елементів комп’ютерного зору за допомогою RaspberryPi

## Апаратна реалізація комп’ютерного зору

Структурна схема керування камероюнаведена в додатку А. Схема складається з наступних блоків:

* ЛМІ – людино-машинний інтерфейс;
* ДЖ – джерело живлення;
* мікрокомп’ютер Raspberry Pi;
* модуль камери Raspberry Pi;
* серводвигун;
* кроковий двигун.

За основу системи керування було взято одноплатнийкомп’ютерRaspberryPi. Серед його переваг слід відзначити: невеликі габаритні розміри 85х56х17 мм, в основі − система-на-кристалі (SoC)BroadcomBCM2835, яка включає процесор ARM із тактовою частотою 700МГц, а встановлена OS Linux спрощує створення програмного забезпечення (ПЗ) та підвищує його рівень абстракції.

Для зручності взаємодії оператора з системою використовується дисплейTFT LCD для RaspberryPi(Waveshare)зтачскрином3.2" 320x240 і резистивною сенсорною накладкою. Він використовує високошвидкісний інтерфейс SPI на RaspberryPi.

В якості камери обрана RaspberryPiCamera v2. Підключення виконується за допомогою CSI (CameraSerialInterface).

Виходячи з того, що поворот камери вгору-вниз не перевищує 180 градусів, було обрано серводвигун,який підключений доGPIO19.

Повороткамери відносно осі Xвиконує кроковий двигун. В якості драйвера використана мікросхема ULN2003. Виходи мікросхеми IN1, IN2, IN3, IN4 під’єднанні до виводівRaspberryPiGPIO16, GPIO20, GPIO21, GPIO26,відповідно.

Для перевірки працездатності системи, а саме – виводів GPIO – використовуєтьсячервоний світлодіод з робочим струмом 20 мА та напругою 1,8…2В. Для правильної роботи світлодіоду потрібно витримувати робочий струм та напругу, а тому використовується баластний резистор. Номінал резистора розраховується, використовуючи закон Ома.

I = U/R;(3.1)

З цієї формули виражаємо опір R.

R = U/I; (3.2)

Розраховуємо опір резистора

R = 5-2/0.020 = 150Ом (3.3)

З розрахунків опір резистора світлодіодустановить 150Ом. В схемі використовується резистор більшого номіналу.

Електрична принципова схема приведена на рисунку 3.1[14].

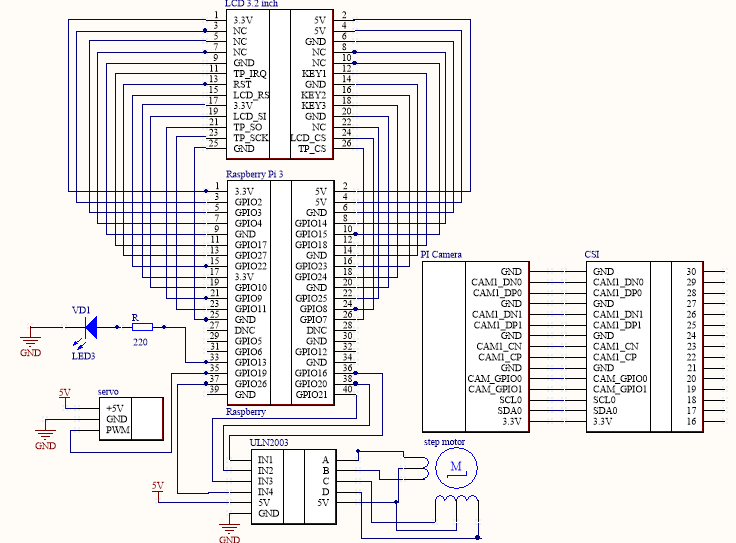


Рисунок 3.1 – Електрична принципова схема

## Графічний інтерфейс користувача

Графічний інтерфейс реалізується за допомогою Pythonбібліотеки tkinter[15](рисунки 3.1…3.6). Головний недолік даної бібліотеки полягає у неможливості прокрутки вікна з даними. У випадку, коли екран менший за вікно, воно обрізається, що не дає змоги його повноцінно використовувати. Даний інтерфейс розроблювався для екрану з діагоналлю 3,2 дюйми, тому він реалізований, як чотири вкладки,кожна з яких відповідає за певнупериферію.

Перша вкладка – «Лампа»(рисунок 3.2). При натисканні на кнопку «OFF» змінюється текст кнопки на «ON», та на виводі з’являється сигнал, що вмикає лампочку.

Друга вкладка – «Вгору/Вниз» (рисунок 3.3). Дана вкладка відповідає за керування серводвигуном, а саме – поворотом камери відносно осі Y. Кут повороту серводвигуна задається за допомогою слайдера, діапазон якого складає від 0 до 180°.

Третя вкладка – «Ліворуч/Праворуч»(рисунок 3.4) відповідає за керування кроковим двигуном відносно осі X, що дозволяє повертати камеру ліворуч або праворуч (рисунок 3.4). Для того, щоб користувач розумів, що він навів курсор миші на кнопку, був реалізований ховер на всіх кнопках проекту. На рисунку3.4 продемонстрована робота ховера при наведенні на кнопку «Ліворуч».

Четверта вкладка – «Камера»(рисунок 3.5). Вона відповідає за отримання відеопотоку з камери та перетворення зображення. На даній вкладці знаходяться дві кнопки, які вмикають і вимикають камеру: «Запустити камеру» та «Вимкнути камеру». Для перетворення зображення в графічному інтерфейсі знаходяться чотири слайдери: «Яскравість», «Контраст», «Насиченість», «Zoom», а такождва меню: «Баланс білого» та «Ефект зображення»,які відповідають за певні ефекти. Перевірка працездатності камери та ефектів наведена на рисунках3.6…3.7.

В програмі реалізовано функцію, яка дозволяє при закритті програми перепросити, чи потрібно закривати програму(рисунок 3.7).

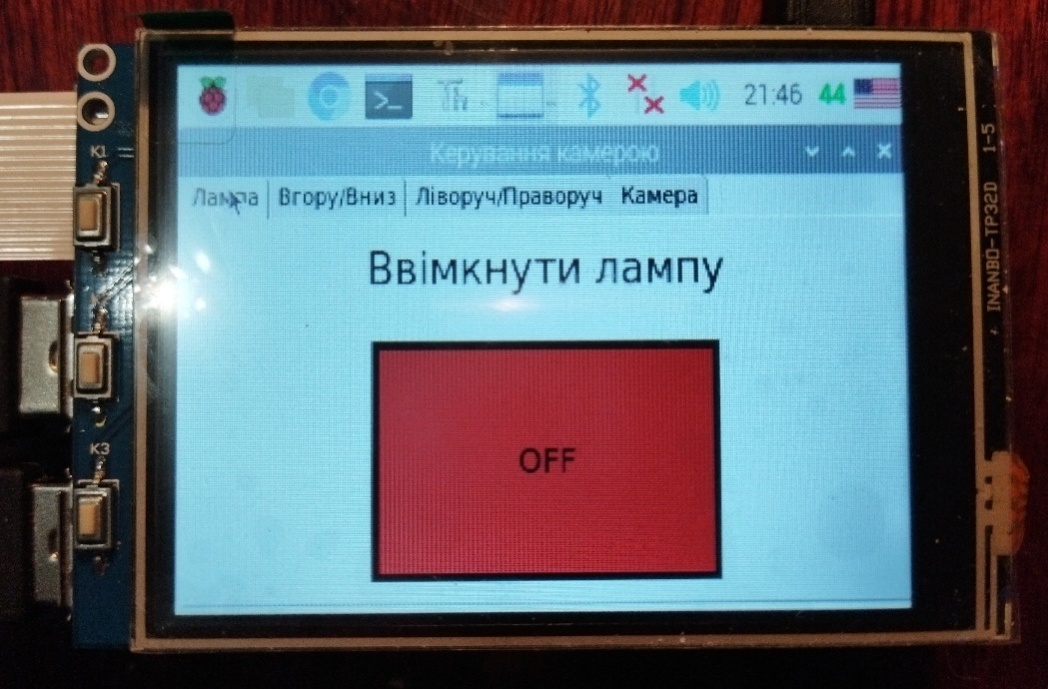


Рисунок 3.2 – Графічний інтерфейс. Перша вкладка

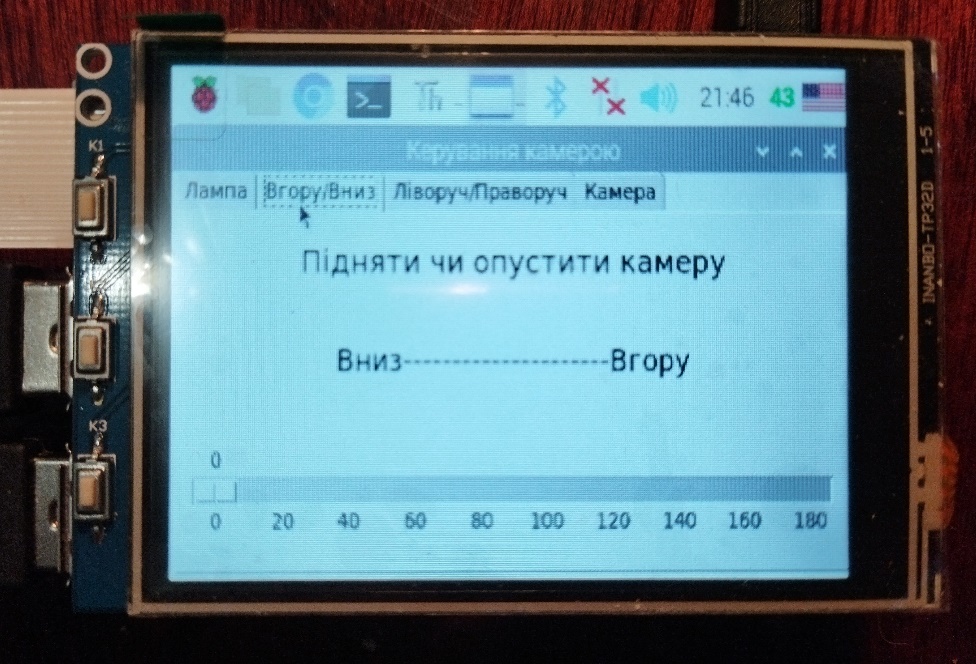


Рисунок 3.3 – Графічний інтерфейс. Друга вкладка

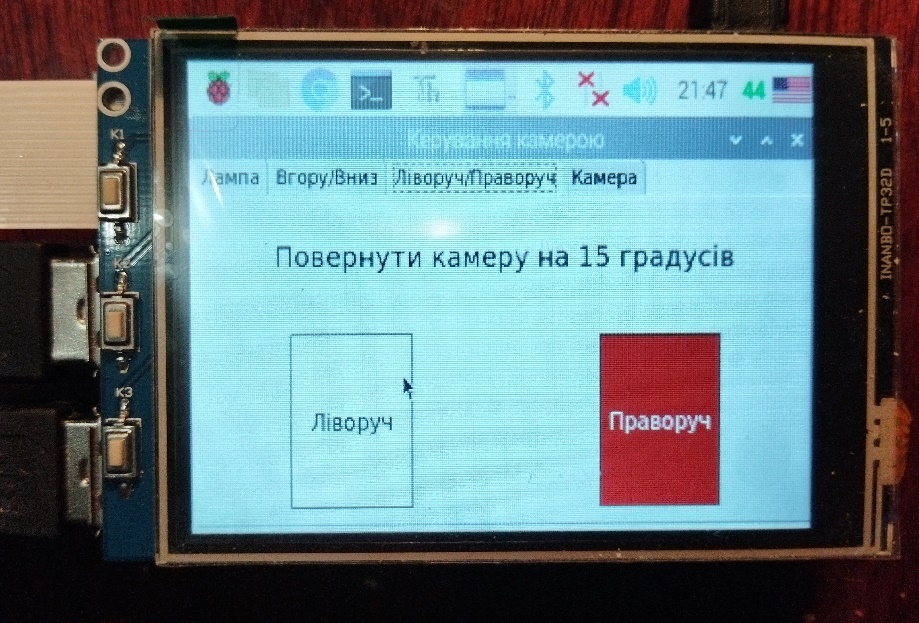


Рисунок 3.4 – Графічний інтерфейс. Третя вкладка

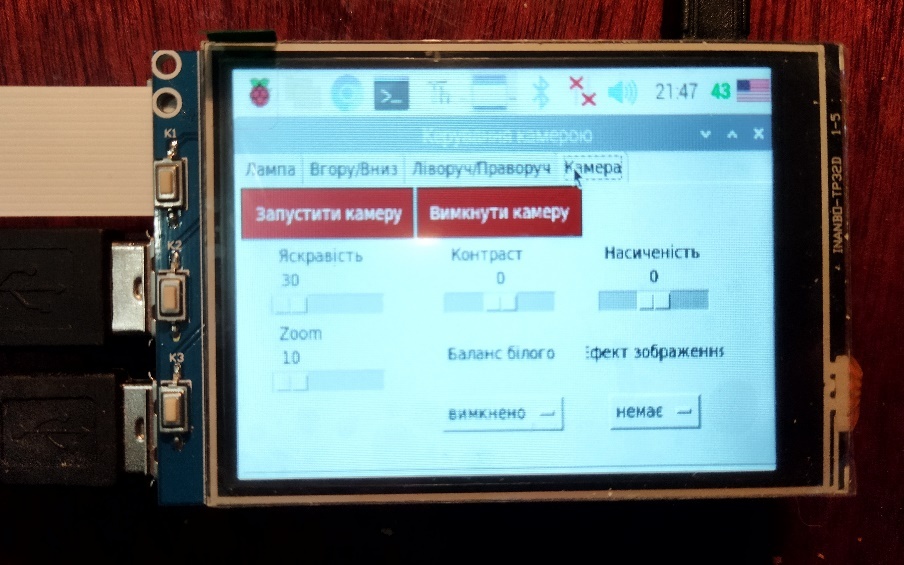


Рисунок 3.5 – Графічний інтерфейс. Четверта вкладка (1)

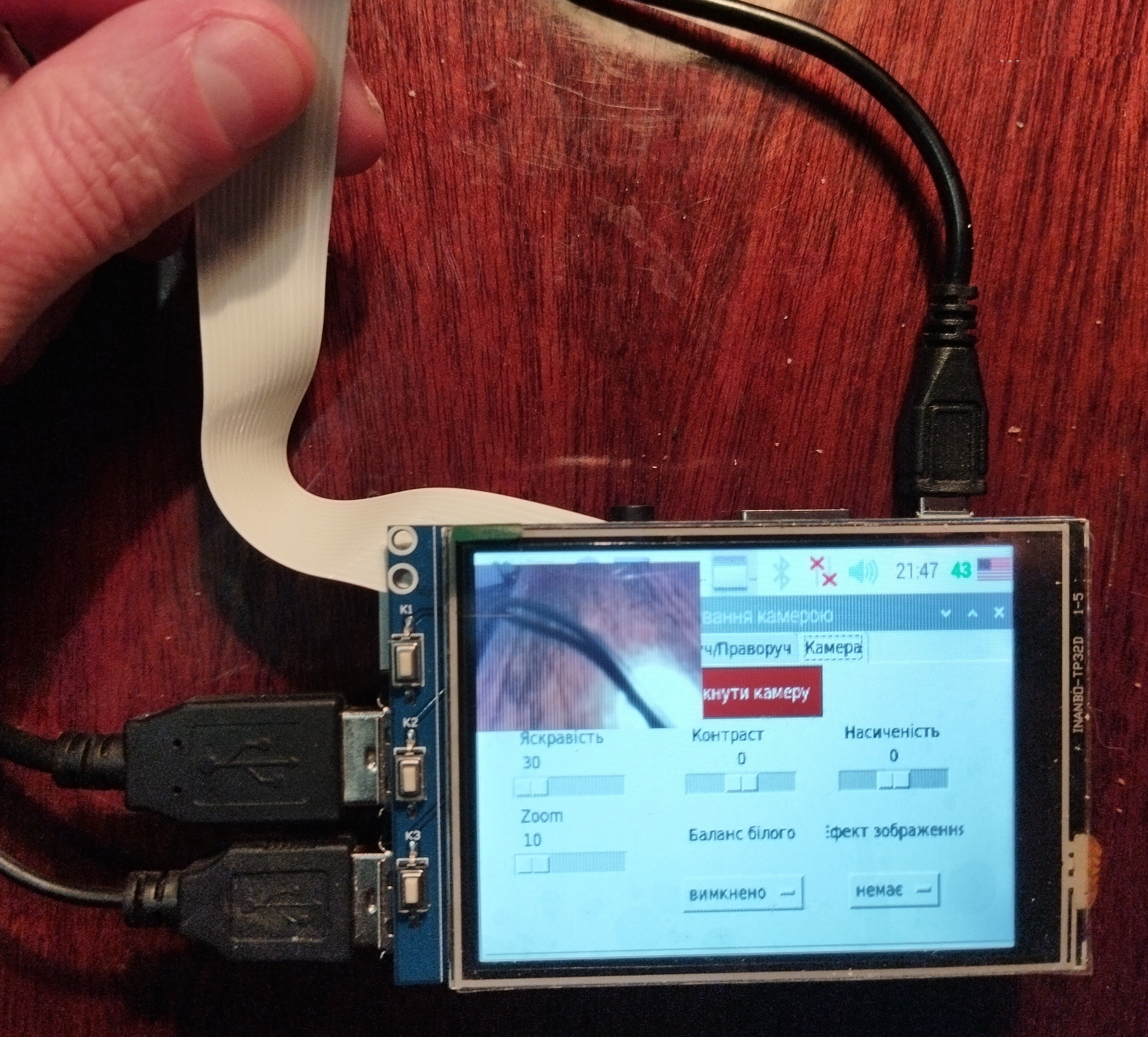


Рисунок 3.6 – Графічний інтерфейс. Четверта вкладка (2)



Рисунок 3.7 – Графічний інтерфейс. Четверта вкладка (3)

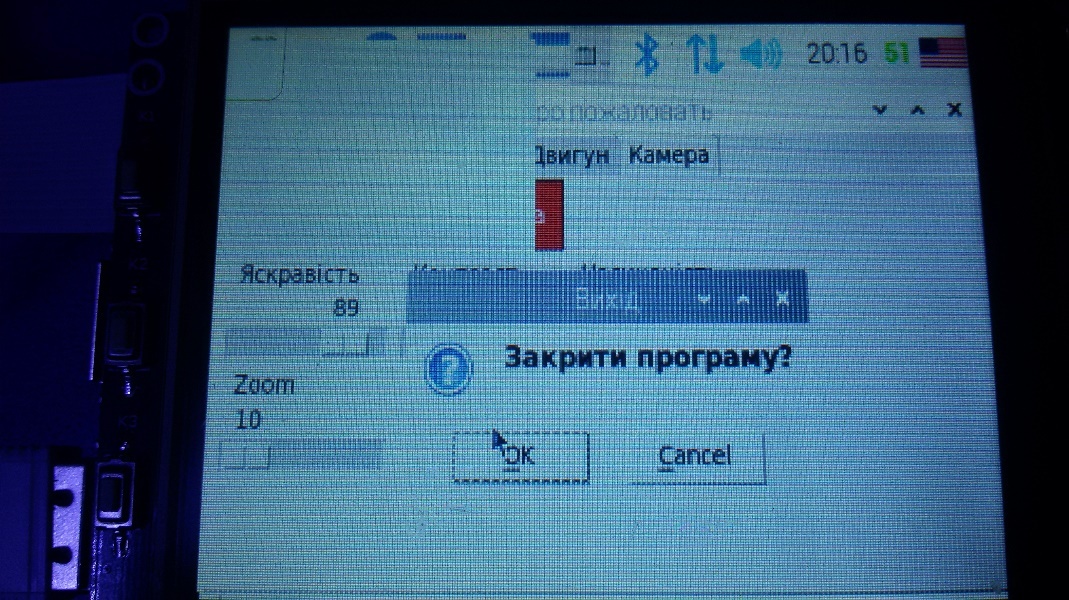


Рисунок 3.8 – Графічний інтерфейс. Закриття програми

## Програмна реалізація комп’ютерного зору

В проекті для отримання відео використовується функція Picamera з бібліотеки picamera(frompicameraimportPiCamera). Включення камери відбувається при натисканні кнопки «Ввімкнути камеру».

Нижче приведена функція, яка реалізує отримання відеопотоку з камери.

def CameraON(): # функція запуску камери

camera.preview\_fullscreen=False # відео з камери не на все вікно

camera.preview\_window=(0, 0, 200, 150) # розміщення вікна

camera.resolution=(300,200) # параметри вікна

camera.start\_preview() # запуск камери

Ефекти для відображення камери задаються, як масив.

choices = ['вимкнено', 'авто', 'сонячне світло','хмарно','тінь','спалах']

effects = ['немає','негативний','ескіз','олійна фарба','акварель','розмиття']

Вибір ефекту виконується наступним кодом

AWB\_Var = tk.StringVar(tab4)

AWB\_Var.set(choices[0]) #автоматично встановити значення #нульового елемента масиву.

AWB\_Option = tk.OptionMenu(tab4, AWB\_Var, \*choices, command=SetAWB)

def SetAWB(var): # функція яка застосовує вибраний ефект

camera.awb\_mode = var

Яскравість, контраст і насиченість зображення задається однаково. Нижче наведена функція, яка зчитує значення яскравості з слайдера, та записує йогодо параметру яскравості камери.

def UpdateBrightness(value): # функція яскравості камери

camera.brightness = int(value)

Керування серводвигуном виконується наступним чином: зі слайдера, шкала якого від 0 до 180°, зчитується значення та ділиться на 15. Ділення виконується для того, щоб робочий цикл був не більше 12.

Нижче приведений код реалізації повороту серводвигуна

def get\_val\_motion(event): # функція задання кута сервопривода

s1 = scal.get() # зчитування параметра з слайдера

sld = s1/15

pwm.ChangeDutyCycle(float(sld)) # Ця функція змінює робочий цикл #сигналу.

Для повороту двигуна використовується наступний код.

def left\_motor(): #функція повороту ліворуч крокового двигуна. #Викликається при натисканні на кнопку «Ліворуч»

for i in range(0,24): # крок двигуна 24 градуси. Щоб двигун #зробив один повний оберт, потрібно 15 кроків

GPIO.output(36, GPIO.HIGH)

GPIO.output(38, GPIO.LOW)

GPIO.output(40, GPIO.LOW)

GPIO.output(37, GPIO.HIGH)

time.sleep(delay)

# Крок 2.

GPIO.output(36, GPIO.HIGH)

GPIO.output(38, GPIO.HIGH)

GPIO.output(40, GPIO.LOW)

GPIO.output(37, GPIO.LOW)

time.sleep(delay)

# Крок 3.

GPIO.output(36, GPIO.LOW)

GPIO.output(38, GPIO.HIGH)

GPIO.output(40, GPIO.HIGH)

GPIO.output(37, GPIO.LOW)

time.sleep(delay)

# Крок 4.

GPIO.output(36, GPIO.LOW)

GPIO.output(38, GPIO.LOW)

GPIO.output(40, GPIO.HIGH)

GPIO.output(37, GPIO.HIGH)

time.sleep(delay)

## Висновки до розділу 3

1. В проекті використовується в якості регулюємого модуля RaspberryPi та чотири периферійних пристрої, а саме: дисплей TFTLCD, модуль камери, кроковий та серводвигун;
2. Отриманий відео потік є змога редагувати: змінивши яскравість, контраст, насиченість, віддалення чи приближення, ефект балансу білого,. Все це дозволяє зробити бібліотека Picamera;
3. Головною проблемою даного метода отримання відео є те, що відео працює тільки в тому випадку, коли підключений монітор або дисплей.

# Висновки

В результаті виконаних теоретичних та експериментальних досліджень отримані наступні результати:

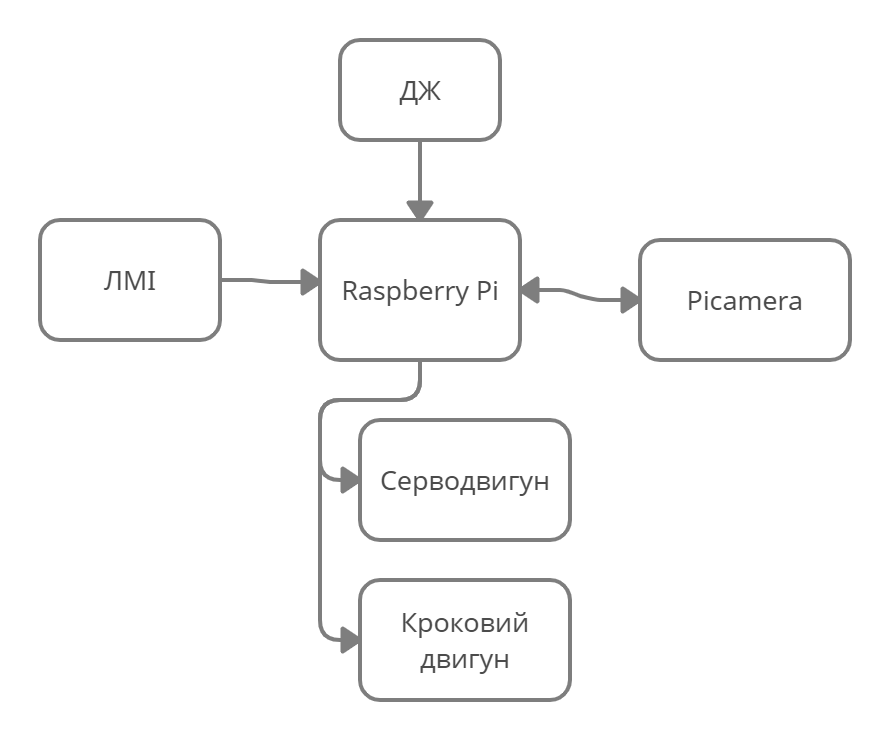
1. З’ясовано, що при використанні в якості керуючого елементу системи комп’ютерного зоруRaspberryPiтреба застосовувати модулі камер,оскількиїхнєпідключення безпосередньо до відеочіпуVideoCoreодноплатного комп'ютера економить системні ресурси.
2. Розроблена та експериментально перевірена система, яка отримує відео потік в реальному часі з модуля камериRaspberryPiCamerav2 з використанням бібліотеки Picamera. Керування камерою відносно осі X здійснює кроковий двигун, відносно осіY– серводвигун.
3. Графічний інтерфейс розробленої системи заснований на бібліотеціpythontkinter,головним недоліком якої є неможливість скролінгу віконменю.
4. Дана розробка може бути використана длянабуття навичок в галузі розробки систем комп’ютерного зору, а також віддаленого керування електроприводом.
5. В якості програмного забезпечення для реалізації алгоритмів комп’ютерного зору доцільно використання MATLAB та OpenCV, що розширює спектр можливостей для обробки зображення.

# Список посилань

1. Шифр «Автономна відеокамера»
2. Шифр «Автономна відеокамера»
3. Настройка и примерыработы камер для RaspberryPi [Електроний ресурс]. – Режим доступу: <http://wiki.amperka.ru/articles:raspberry-pi-camera-guide>
4. CameraModule [Електроний ресурс]. – Режим доступу: <https://www.raspberrypi.org/documentation/hardware/camera/>
5. RaspberryPiHighQualityCamera [Електроний ресурс]. – Режим доступу:<https://static.raspberrypi.org/files/productbriefs/Raspberry_Pi_HQ_Camera_Product_Brief.pdf>
6. Using a standard USB webcam [Електроний ресурс]. – Режим доступу: <https://www.raspberrypi.org/documentation/usage/webcams/>
7. Picamerainterface [Електроний ресурс]. – Режим доступу: <https://picamera.readthedocs.io/en/release-1.10/index.html>
8. LearningOpenCVbyGaryBradskiandAdrianKaehler, Printed in the United States of America. 571s, 2008
9. Компьютерное зрение в MATLAB для задач детектирования и слежения [Електронний ресурс]. – Режим доступу: <https://exponenta.ru/events/kompyuternoe-zrenie-v-matlab-dlya-zadach-detektirovaniya-i-slezheniya>
10. HowtocontrolRaspberryPi GPIO viahttpwebserver [Електроний ресурс]. – Режим доступу: <https://www.e-tinkers.com/2018/04/how-to-control-raspberry-pi-gpio-via-http-web-server/>
11. VideoStreamingwithRaspberryPiCamera [Електроний ресурс]. – Режим доступу: https://randomnerdtutorials.com/video-streaming-with-raspberry-pi-camera/
12. RaspberryPiFlask [Електроний ресурс]. – Режим доступу: <http://mattrichardson.com/Raspberry-Pi-Flask/>
13. Камера с механизмомповорота/наклона и управлением по локальной сети [Електроний ресурс]. – Режим доступу: <https://robototehnika.ru/content/article/kamera-s-mekhanizmom-povorota-naklona-i-upravleniem-po-lokalnoy-seti/>
14. ShematicRaspberryPi 3B [Електроний ресурс]. – Режим доступу: <https://www.raspberrypi.org/documentation/hardware/raspberrypi/schematics/rpi_SCH_3bplus_1p0_reduced.pdf>
15. Tkinter — Pythoninterface [Електроний ресурс]. – Режим доступу: <https://docs.python.org/3/library/tkinter.html>

# Додатки

## Додаток А – Структура розробленої системи



## Додаток Б – Текст програми

from tkinter import \*

import tkinter as tk

from tkinter import ttk, messagebox, Radiobutton, Scale

import RPi.GPIO as GPIO

from time import sleep

import time

import picamera

camera = picamera.PiCamera()

choices = ['вимкнено', 'авто', 'сонячне світло','хмарно','тінь','спалах']

effects = ['немає','негативний','ескіз','олійна фарба','акварель','розмиття']

def CameraON():

camera.preview\_fullscreen=False

camera.preview\_window=(0, 0, 200, 150)

camera.resolution=(300,200)

camera.start\_preview()

def CameraOFF():

camera.stop\_preview()

def UpdateBrightness(value):

camera.brightness = int(value)

def UpdateContrast(value):

camera.contrast = int(value)

def UpdateSaturation(value):

camera.saturation = int(value)

def SetAWB(var):

camera.awb\_mode = var

def SetEFFECTS(var):

camera.image\_effect = var

def Zoom(var):

x = float("0."+var)

camera.zoom = (0.5,0.5,x,x)

def get\_val\_motion(event):

s1 = scal.get()

sld = s1/15

pwm.ChangeDutyCycle(float(sld))

def left\_motor():

for i in range(0,24):

GPIO.output(36, GPIO.HIGH)

GPIO.output(38, GPIO.LOW)

GPIO.output(40, GPIO.LOW)

GPIO.output(37, GPIO.HIGH)

time.sleep(delay)

# Шаг 2.

GPIO.output(36, GPIO.HIGH)

GPIO.output(38, GPIO.HIGH)

GPIO.output(40, GPIO.LOW)

GPIO.output(37, GPIO.LOW)

time.sleep(delay)

# Шаг 3.

GPIO.output(36, GPIO.LOW)

GPIO.output(38, GPIO.HIGH)

GPIO.output(40, GPIO.HIGH)

GPIO.output(37, GPIO.LOW)

time.sleep(delay)

# Шаг 4.

GPIO.output(36, GPIO.LOW)

GPIO.output(38, GPIO.LOW)

GPIO.output(40, GPIO.HIGH)

GPIO.output(37, GPIO.HIGH)

time.sleep(delay)

def right\_motor():

for i in range(0,24):

GPIO.output(36, GPIO.LOW)

GPIO.output(38, GPIO.LOW)

GPIO.output(40, GPIO.HIGH)

GPIO.output(37, GPIO.HIGH)

time.sleep(delay)

# Шаг 2.

GPIO.output(36, GPIO.LOW)

GPIO.output(38, GPIO.HIGH)

GPIO.output(40, GPIO.HIGH)

GPIO.output(37, GPIO.LOW)

time.sleep(delay)

# Шаг 3.

GPIO.output(36, GPIO.HIGH)

GPIO.output(38, GPIO.HIGH)

GPIO.output(40, GPIO.LOW)

GPIO.output(37, GPIO.LOW)

time.sleep(delay)

# Шаг 4.

GPIO.output(36, GPIO.HIGH)

GPIO.output(38, GPIO.LOW)

GPIO.output(40, GPIO.LOW)

GPIO.output(37, GPIO.HIGH)

time.sleep(delay)

def but\_onoff(event, button):

if button == "OFF":

but["text"] = "ON"

but['bg'] = '#FFFFFF'

but['fg'] = '#C72828'

but['activeforeground'] = "red"

GPIO.setup(33, GPIO.HIGH)

else:

but["text"] = "OFF"

but['bg'] = '#C72828'

but['fg'] = '#FFFFFF'

but['activeforeground'] = "red"

GPIO.setup(33, GPIO.LOW)

def on\_closing():

if messagebox.askokcancel("Вихід", "Закрити програму?"):

window.destroy()

camera.stop\_preview()

camera.close()

GPIO.setmode(GPIO.BOARD)

GPIO.setup(35, GPIO.OUT)

pwm = GPIO.PWM (35, 50)

GPIO.setup(33, GPIO.OUT)

pwm.start (0)

in2 = 36

in1 = 37

in3 = 38

in4 = 40

delay = 0.002

GPIO.setup(36, GPIO.OUT)

GPIO.setup(37, GPIO.OUT)

GPIO.setup(38, GPIO.OUT)

GPIO.setup(40, GPIO.OUT)

window = Tk()

window.title("Керування камерою")

window.geometry('600x250')

window.configure(bg='#ffffff')

window.protocol("WM\_DELETE\_WINDOW", on\_closing)

tab\_control = ttk.Notebook(window)

tab1 = ttk.Frame(tab\_control)

tab2 = ttk.Frame(tab\_control)

tab3 = ttk.Frame(tab\_control)

tab4 = ttk.Frame(tab\_control)

tab\_control.add(tab1, text='Лампа')

tab\_control.add(tab2, text='Вгору/Донизу')

tab\_control.add(tab3, text='Ліворуч/Праворуч')

tab\_control.add(tab4, text='Камера')

lbl1 = Label(tab1, text='Ввімкнути лампу', font=("Arial Bold", 20))

but = Button(tab1, text="OFF", width=15, height=5, bg="#C72828", fg="#000000", borderwidth="4", relief="solid", font=("Arial Bold", 15))

but.bind("<Button-1>", lambda event: but\_onoff(event, but["text"]))

lbl1.pack(side="top", expand=1)

but.pack(side="bottom", expand=1)

lbl2 = Label(tab2, text='Підняти чи опустити камеру', font=("Arial Bold", 15))

lbl21 = Label(tab2, text='Вниз---------------------Вгору', font=("Arial Bold", 14))

lbl2.pack(side="top", expand=1)

lbl21.pack(side="top", expand=1)

scal = Scale(tab2,orient=HORIZONTAL, from\_=0, to=180, tickinterval=20)

scal.bind("<B1-Motion>",get\_val\_motion)

scal.pack(side="top", expand=1, fill=X, padx=10)

lbl3 = Label(tab3, text='Повернути камеру на 15 градусів', font=("Arial Bold", 15))

motor\_1 = Button(tab3, text="Ліворуч", width=7, height=5, bg="#C72828", fg="#FFFFFF", borderwidth="1", relief="solid", padx=10, pady=10, font=("Arial Bold", 12), command=left\_motor)

motor\_2 = Button(tab3, text="Праворуч", width=7, height=5, bg="#C72828", fg="#FFFFFF", borderwidth="1", relief="solid", padx=10, pady=10, font=("Arial Bold", 12), command=right\_motor)

lbl3.pack(side="top", expand=1)

motor\_1.pack(side="left", expand=1, pady=10)

motor\_2.pack(side="left", expand=1, pady=10)

on\_cam = Button(tab4, text='Запустити камеру', padx=10, pady=10, bg="#C72828", fg="#FFFFFF", borderwidth="1", relief="solid", command=CameraON)

off\_cam = Button(tab4, text='Вимкнути камеру', padx=10, pady=10, bg="#C72828", fg="#FFFFFF", borderwidth="1", relief="solid", command=CameraOFF)

brg\_scale = Scale(tab4, from\_=30, to=100, orient=tk.HORIZONTAL, label = "Яскравість", command=UpdateBrightness)

cntrst\_scale = Scale(tab4, from\_=-100, to=100, orient=tk.HORIZONTAL, label = "Контраст", command=UpdateContrast)

saturation\_scale = Scale(tab4, from\_=-100, to=100, orient=tk.HORIZONTAL, label = "Насиченість", command=UpdateSaturation)

zoom\_scale = Scale(tab4, from\_=10, to=99, orient=tk.HORIZONTAL, label = "Zoom", command=Zoom)

on\_cam.grid(row=1, column = 1)

off\_cam.grid(row=1, column = 2)

brg\_scale.grid(row=2,column=1)

cntrst\_scale.grid(row=2,column=2)

saturation\_scale.grid(row=2,column=3)

zoom\_scale.grid(row=3,column=1)

l3 = tk.Label(tab4, text='Баланс білого', width=15 )

l3.grid(row=3,column=2)

l4 = tk.Label(tab4, text='Ефект зображення', width=15 )

l4.grid(row=3,column=3)

AWB\_Var = tk.StringVar(tab4)

AWB\_Var.set(choices[0])

AWB\_Option = tk.OptionMenu(tab4, AWB\_Var, \*choices, command=SetAWB).grid(row=4,column=2)

EFFECT\_Var = tk.StringVar(tab4)

EFFECT\_Var.set(effects[0])

EFFECT\_Option = tk.OptionMenu(tab4, EFFECT\_Var, \*effects, command=SetEFFECTS).grid(row=4,column=3)

tab\_control.pack(expand=1, fill='both')

window.mainloop()