Міністерство освіти і науки Україні
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет прикладних інформаційних технологій та електроінженерії
(назва факультету)
Кафедра автоматизації технологічних процесів та виробництв
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(освітній рівень)

на тему: Розробка роботизованої системи з дистанційним керуванням та можливістю обминання перешкод

Виконав: студент 4 курсу, групи КАс-41
Спеціальність *151*
"Автоматизазія та комп'ютерно-інтегровані технології"
(шифр і назва спеціальності)

(підпис) Антонишин В.В. (прізвище та ініціали)

Керівник (підпис) Козбур І.Р. (прізвище та ініціали)

Нормоконтроль (підпис) Козбур І.Р. (прізвище та ініціали)

Завідувач кафедри (підпис) Савків В.Б. (прізвище та ініціали)

Рецензент (підпис) Стухляк П.Д. (прізвище та ініціали)

м. Тернопіль – 2021

А н о т а ц і я

В дипломній роботі використані такі терміни: дистанційне керування, роботизовані системи, SSH , Технологія Bluetooth, ультразвуковий давач, інфрачервоний давач.

Об'єктом дослідження є процес автоматизації обминання перешкод при дистанційному керуванні.

Мета роботи – є розробка та тестування програми дистанційного керування з можливістю обминання перешкод.

Дистанціне керування з можливістю реагування на перешкоди в реальному часі сьогодні є життєво важливою темою. Проблемною, а також важливою сферою розробок є мінімізація зіткнень на підприємствах підчас транспортування серовини та готової продукції. Процедура переміщення за допомогою дистанційного керування. Зниження ризику виробничого травматизму - це альтернатива, що розглядається все більшою кількістю компаній. В роботі оцінено потенціал використання дистанційного керування та спсобів обминання перешкод для забезпечення надійного переміщення в просторі. Проведено аналіз робочого середовища щодо ергономіки з метою дослідити, як його можна вдосконалити. Два рішення, ультразвуковий та інфрачервоний методи, були оцінені як найбільш доцільні для використання у поєднанні з дистанційним керуванням для забезпечення без перешкодного переміщення.

A n o t a t i o n

The following terms are used in the diploma work: remote control, robotic systems, SSH, Bluetooth technology, ultrasonic sensor, infrared sensor.

The object of research is the process of automation of bypassing interference during remote control.

The purpose of the work is to develop and test a remote control program with the ability to bypass obstacles.

Remote control with the ability to respond to obstacles in real time today is a vital topic. A problematic as well as an important area of development is the minimization of collisions at enterprises during the transportation of raw materials and finished products. Remote control movement procedure. Reducing the risk of occupational injuries is an alternative that is being considered by a growing number of companies. The paper evaluates the potential of using remote control and methods of bypassing obstacles to ensure reliable movement in space. An analysis of the working environment in terms of ergonomics in order to explore how it can be improved. Two solutions, ultrasonic and infrared, were evaluated as the most appropriate for use in combination with remote control to ensure unimpeded movement.

ЗМІСТ

Вступ

Автоматизація машин, установок і виробничих процесів є в даний час одним з найважливіших напрямів технічного прогресу в усіх галузях народного господарства.

Дистанційно-керовано роботизована системи з можливістю обминання перешкод – це незамінна річ в арсеналі будь-якого підприємства, на якому люди мають безпосередній контакт з небезпечними речовинами. На сьогоднішній день, завдяки невисокій вартості розробки та великому попиту, на ринку представлений досить широкий спектр автоматизованих систем, тому, не розібравшись в цьому питанні більш детально, можна просто заплутатися.

За принципом дії роботизовані системи діляться на два типи: з безпосередньо втручанням людини в процес та з мінімальним втручанням людини в процес. Роботизовані системи з безпосереднім втручанням людини в процес є більш надійними так як працівники слідкують за процесом роботи. Головною ж перевагою роботизованої системи з мінімальним втручанням людини в процес це мінімальний обслуговуючий персонал так як майже всі процеси є абсолютно автоматичними.

У розроблюваному пристрої реалізована можливість обминання перешкод, а також дистанційне керування за допомогою смартфона. Управління є легким і зрозумілим на інтуїтивному рівні. Перевагою розроблюваного пристрою є його надійність в процесі роботи, дешеві деталі та простота конструкції самого пристрою. Програмний керуючий код можна доопрацювати для власних потреб. Також пристроєм можна керувати за допомогою ПК чи смартфона через мережу Wi – Fi у режимі онлайн.

1. Аналітична частина

1.1 Роботизовані системи та дистанційне керування

1.1.1 Роботизовані системи

Роботизовані системи – це метод розробки автоматизації виробництва при одночасній зміні кількості робочої сили, часу та виробничих витрат, звязаних з процесом виробництва. Дані системи застосовуються практично в кожній обробній галузі.

При цьому протягом довгого часу ручне виробництво домінувало, саме роботизовані системи запустили процес революції в процесах виробництва. На даний час виробники можуть виготовляти більше якісних виробів за коротший проміок часу завдяки роботизації підприємств.

На даний час існує три типи роботизованих систем – роботизована система збору та управління, роботизована система маніпуляцій та мобільна роботизована система.

Роботизована система маніпуляційних роботів є найбільш популярною у використанні, а саме у виробничих галузях. Дані ситеми мають в своєму складі множинну кількість робочих рук із 4-6 осями та з різним ступенем свободи. Дані системи можуть застосовуватися в декількох різних функціях, включаючи обробку матеріалів, зварювання та видавлення матеріалів.

Мобільні робототехнічні системи дещо відрізняються від попередніх. Дана система має можливість переміщення обєктів з однієї точки в іншу завдяки своїй автоматизованій платформі. Хоч ці системи використовуються на виробництві в більшій мірі для переміщення запасних частин та інструментів, також дані системи використовують в сільськогосподарській промисловості для транспортуваня продукції. Дані системи також

використовують в різних промислових галузях завдяки їх можливостям літати і плавати, а також рухатись по землі.

Роботизовані системи збору та управління даними застосовуються для збору, обробки та передачі інформації для різного виду сигналів. Також вони застосовуються в програмному устаткуванні для бізнесу та інженірингу. Велика кількість мобільних роботозованих систем мають можливість використовувати сигнали цих систем.

1.1.2 Дистанційне керування

Дистанційне керування – це один із методів передачі інформації сигналів з пристроєм над яким відбувається керування (на відмінно від механічного, кабельного або з допомогою оператора).

Найбільш часто використовувані способи дистанційного керування це за допомогою Wi-Fi, Bluetooth, інфрачервоної, GPS або ультразвукової передачі даних. Найпростіший приклад дистанційного керування це пульт від телевізора [2].

Дистанційне керування застосовується в тому випадку, якщо інший метод керувння складно або неможливо використати як приклад це ракети, безпілотники, квадрокоптери, космічні апарати та інші. Більшість способів використання це об'єкти які не призначені для знаходження на них людей або ж мають шкідливий вплив на організм людини. Дистанційне керування має великі перспективи в процесі виймання корисних копалин без втручання людини. В більшості випадках дистанційне керувння використовують при 20-30 об'єктах над якими здійснюватиметься керування та на відстані до 2-4 км. Для більших дистанціях використовується телемеханічні системи.

1.1.3 Технології для здійснення дистанційного керування

1.1.3.1 Технологія SSH

SSH (від англ. "Secure Shell") - це протокол для здійснення дистанційного адміністрування, створений для виконання віддаленого керування операційними ситемами і тунелювання TCP-з'єднання. Застосування цього протоколу дозволяє застосування різних алгоритмів шифрування, що забезпечує безпечну роботу фактично в будь-якому незахищеному середовищі: працювати з персональним комп'ютером через командну оболонку, використовувати шифрований канал для передачі будь-яких типів даних (наприклад аудіо та відео файли). Перша поява протоколу відбулась 1995 році, а в 1996 році вже була представлена його вдосконалена версія, яка стала основою для розвитку протоколу. На даний час для всі мережевих ОС доступні як SSH клієнти так і SSH сервери, а сам протокол SSH став одним з найпопулярніших способів для дистанційного керування системами і для передачі важливих даних. Протокол SSH використовує модель клієнт-серверну для підключення віддалених систем і забезпечує шифрування даних при обмінні якими відбувається при віддаленому керуванні [3].

По стандарту для використання протоколу застосовується порт TCP-22: на нього хост (сервер) виконує вхідне підключення і після проведення аутентифікації та отриманні команд, організовується запуск клієнта, включаючи виибрану користувачем оболочку. За потреби користувач може змінити порт підключення. Щоб створити SSH підключення клієнту потрібно створити зв'язок з сервером, надавши захищений зв'язок та підтвердити свої данні підключення (перевіряються певні ідентифікатори з минулими записами, що зберігаються в файлі RSA та особисті данні користувача, необхідні для авторизації).

Переваги SSH підключення:

- надійне і безпечне підключення до ПК з можливістю використання командної оболонки;
- можливість використання різних видів алгоритмів шифрування (асиметричного, симетричного і хеширования);
- надає можливість надійного використання будь-яких мережевих протоколів, що надає можливість захищеного обміну файлами будь-якого розміру [3].

1.1.3.2 Технологія Bluetooth

Bluetooth – технологія для бездротового підключення, розроблена групою яка складається з компаній: Ericsson, IBM, Intel, Nokia, Toshiba [4].

Технолгія Bluetooth дає можливість передачі як файлв так і звукових коливань (із можливістю передачі 64 Кбіт/с). Для пнадсилання данних використовуються як симетричний (зі швидкістю передачі 432,6 Кбіт/с) так і асиметричний (721 Кбіт/с в одномусторону так і в іншому зі швидкістю передачі 57,6 Кбіт/с в іншому) способи. Bluetooth-chip – це прийомопередавач який працює на частоті 2.4 ГГц надає можливість створення з'єднання у діапазоні відстані від 10 до 100 метрів. Хоча відстань доволі велика але підключення в межах 10 метрів надає можливість забезпечити мінімальне енергоспоживання як плюс до цього компактність та дешева ціна компонентів. Прийомопередавач малопотужний тому споживає лише 0.3 мА в режимі standby, а під час передачі данних приблизно 30 мА. В документації технології Bluetooth вказана можливість шифрування даних, які пересилаються задопомогою ключа ефективної довжини від 8 до 128 біт і має можливість вибору методу аутентифікації як односторонньої так і двосторонньої. Також окрім шифрування на рівні протоколу, додатково можливе і шифрування програмного рівня.

FHSS (англ. Frequency-hopping spread spectrum) являє собою принцип роботи технології Bluetooth. Попростому це дія при якій передавач перед

відправленням ділить данні на пакети та передає їх за допомогою псевдо-рандомного алгоритмому покрокової переформування частоти (кількістю 1600 разів на секунду), або за допомогою pattern тобто шаблоном, який будується з 79 підчастот. Обмін даними можливий лише між тими пристроями в яких одинаковий шаблон передачі в налаштуваннях – для інших пристроїв передані дані будуть лише шумом. Пікомережа (piconet) – є основним елементом мережі Bluetooth – це група пристроїв від 2 і до 8, що налаштовані на одни і тойже шаблон. В кожній такій мережі є один пристрій який працює як master (активний), а решта працює як slave (пасивні). Активний пристрій створює шаблон, за допомогою якого працюватимуть всі інші пристрої які належать його мережі і який в подальшому налаштує її роботу. Документація технології Bluetooth включає можливість з'єднання окремих і не налаштованих між собою мереж до 10 в так названу scatternet мережу. Для того щоб це було можливим всі пари такої мережі повинні матістити хочаб один спільний пристрій, який буде включений в одну пікомережі і пасивним для іншої. Завдяки цьому методом, у діапазоні scatternet з інтерфейсом Bluetooth можливо максимум одночасно пов'язано 71 пристрій, проте немає обмежинь для застосування пристроїв підключення, які використовує Internet за для з'єднання на виликих відстаннях [4].

1.2 Системи переміщення мобільного робота

1.2.1 Огляд існуючих розробок

1.2.1.1 Виклик програмованого робота

Основний виклик усієї робототехніки полягає в наступному: неможливо коли-небудь дізнатися справжній стан навколишнього середовища. Програмне устаткування для керування роботами може тільки здогадуватися про стан навколинього світу на базі вимірювань, надісланих

його давачами. Він може спробувати змінити стан реального світу лише за допомогою генерації контрольних сигналів (Рис.1.1).

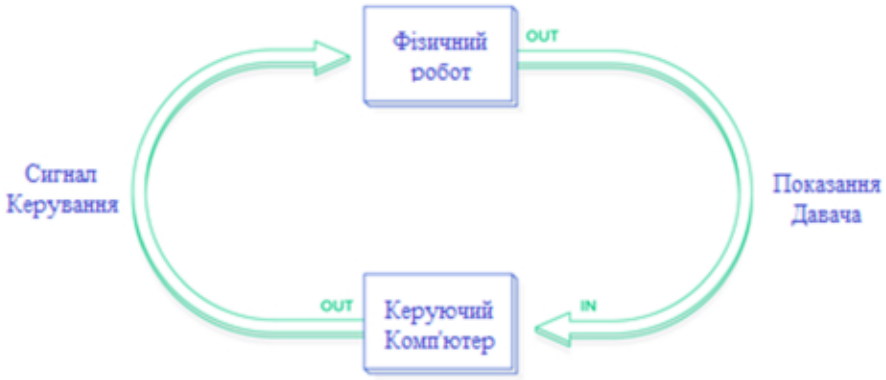

Рис.1.1 – діаграма принципу роботи розпізнавання світу

Програмне забезпечення для управління роботами може лише здогадуватися про стан реального світу на основі вимірювань, повернутих його датчиками. [6].

Таким чином, одним із перших кроків у проектуванні управління є розробка абстракції реального світу, відома як модель, за допомогою якої можна інтерпретувати показники з давачів і приймати рішення. Поки реальний світ поводиться згідно з припущеннями моделі, можна тільки гадати і здійснювати контроль. Як тільки реальний світ відхиляється від цих припущень і робот вже не зможе добре здогадуватися і контроль буде втрачений. Часто, втративши контроль, він більше не може бути повернутий. (Якщо якась доброзичлива зовнішня сила не відновить його).

Це одна з ключових причин того, що програмування робототехніки настільки складне. Часто в лабораторії можна побачити відео про останнього дослідницького робота, який виконує фантастичні подвиги спритності, навігації чи колективної роботи і виникає питання: "Чому це не використовується в реальному світі?" Що ж, в подальшому, коли побачите таке відео, погляньте, наскільки високо контролюється лабораторне

середовище. У більшості випадків ці роботи здатні виконувати ці вражаючі завдання до тих пір, поки екологічні умови залишаються у вузьких межах своєї внутрішньої моделі. Таким чином, одним із ключів до прогресу робототехніки є розробка більш складних, гнучких та надійних моделей - і зазначений прогрес підлягає обмеженню доступних обчислювальних ресурсів [6].

Одним із ключів прогресу робототехніки є розробка більш складних, гнучких і надійних моделей.

1.2.1.2 Програмований симулятор робота

Логіка управління роботом обмежена цими класами / файлами Python:

- models/supervisor.py—Цей клас відповідає за взаємодію між імітованим світом навколо робота і самим роботом. Він розвиває даного автоматизованого робота і запускає контролери для обчислення бажаної поведінки.
- models/supervisor_state_machine.py—Цей клас представляє різні стани, в яких може перебувати робот, залежно від його інтерпретації датчиків.
- Файли в models/controllerskаталозі - ці класи реалізують різну поведінку робота, враховуючи відомий стан навколишнього середовища. Зокрема, конкретний контролер вибирається залежно від автомата стану.

1.2.1.3 Програмований робот

Кожен робот має різні можливості та проблеми управління. Давайте познайомимося з даним змодельованим програмованим роботом.

Перше, на що слід звернути увагу, це те, що в цьому випадку робот буде автономним мобільним роботом. Це означає, що він буде вільно пересуватися в просторі і робити це буде під своїм контролем. Це на відміну від, скажімо, робота з дистанційним управлінням (який не є автономним) або

заводського плеча робота (яке не є мобільним). Даний робот повинен сам придумати, як досягти своїх цілей і розібратись в оточенні. Це виявляється досить важким викликом для початківців програмістів робототехніки [6].

1.2.1.4 Входи управління

Існує багато різних способів, як робот може бути обладнаний для спостереження за навколишнім середовищем. Вони можуть включати що завгодно: датчики наближення, датчики світла, бампери, камери тощо. Крім того, роботи можуть спілкуватися із зовнішніми давачами, які дають їм інформацію, яку вони самі не можуть безпосередньо спостерігати.

Даний робот оснащений дев’ятьма інфрачервоними датчиками - нова модель має вісім інфрачервоних та п’ять ультразвукових датчиків наближення, розташованих у нижній частині в будь-якому напрямку. Датчиків, спрямованих до передньої частини робота, більше, ніж до задньої, оскільки для робота зазвичай важливіше знати, що перед ним, ніж те, що позаду.

На додаток до давачів наближення, робот має пару колісних бірок, які відстежують рух колеса. Вони дозволяють відстежувати, скільки обертань робить кожне колесо, при цьому один повний поворот вперед колеса становить 2765 галочок. Повороти в зворотному напрямку відлічують назад, зменшуючи кількість кроків, замість того щоб їх збільшувати. Для цього не потрібно турбуватися про конкретні цифри в цьому випадку, оскільки програмне забезпечення, використовує пройдену відстань, виражену в метрах [6].

1.2.1.5 Виходи управління

Деякі роботи пересуваються на ногах. Деякі котяться, як м’яч. Деякі навіть ковзають, як змія.

Даний робот - робот з диференціальним приводом , тобто він котиться на двох колесах. Коли обидва колеса обертаються з однаковою швидкістю, робот рухається по прямій. Коли колеса рухаються з різною швидкістю, робот обертається. Таким чином, управління рухом цього робота зводиться до належного контролю швидкості, з якою обертається кожне з цих двох коліс.

1.2.1.6 API

У Sobot Rimulator розділення між «комп’ютером» робота та (змодельованим) фізичним світом втілено в файл robot_supervisor_interface.py, який визначає весь API для взаємодії з датчиками та двигунами «реального робота»:

- read_proximity_sensors () відправляє масив з дев'яти значень у власному форматі датчиків
- read_wheel_encoders () повертає масив із двох значень, що вказують загальну кількість галочок з початку
- set_wheel_drive_rates (v_l, v_r) приймає два значення (в радіанах на секунду) і встановлює ліву та праву швидкість коліс на ці два значення [6].

Цей інтерфейс внутрішньо використовує робот-об'єкт, який забезпечує дані від датчиків і можливість переміщення двигунів або коліс. Якщо потрібно зробити іншого робота, вам просто потрібно надати інший клас робота Python, який може використовуватися одним і тим же інтерфейсом, а решта коду (контролери, супервізор та симулятор) працюватиме нестандартно.

1.2.1.7 Симулятор

Файл world.py є класом Python, який представляє імітований світ із роботами та перешкодами всередині. Функція ступеня всередині цього класу піклується про розвиток простого шляху:

- Застосування фізичних правил до рухів робота
- Враховуючи зіткнення з перешкодами
- Надання нових значень для давачів робота

Функція кроку виконується в циклі таким чином, що robot.step_motion() рухає робота, використовуючи швидкість колеса, обчислену керівником на попередньому етапі моделювання.

Функція внутрішньо оновлює значення датчиків відстані робота, поставлені так що спостерігач матиме можливість оцінити навколишнє середовище на поточному кроці моделювання.

Проста модель

Для початку, у робота буде дуже проста модель.

Важливі примітки про навколишнє середовище:

- Місцевість повинна бути завжди рівна
- Перешкоди ніколи не повинні бувають круглими
- Колеса ніколи не повинні ковзають
- Ніхто не повинен штовхати робота
- Давачі ніколи не дають збоїв і не дають помилкових показників
- Колеса завжди повинні обертатись, коли їм це кажуть

Хоча більшість з цих припущень є обґрунтованими в домашньому середовищі, можуть бути присутні круглі перешкоди. Дане програмне забезпечення для уникнення перешкод має просту реалізацію і дотримується межі перешкод, щоб їх обійти [6].

1.2.1.8 Цикл управління

Тепер ввійдемо в ядро програмного забезпечення для управління та пояснення поведінки, яку потрібно запрограмувати всередині робота. До цього фреймворку можна додати додаткову поведінку. Програмне забезпечення для робототехніки на основі поведінки було запропоновано більше 20 років тому, і воно все ще є потужним інструментом для мобільної робототехніки. Як приклад, у 2007 році в DARPA Urban Challenge було використано набір поведінки - перший конкурс автономних автомобілів.

Робот - це динамічна система. Стан робота, показання його давачів та вплив сигналів управління постійно змінюються. Контроль за тим, як розгортаються події, включає такі три кроки:

1. Застосувати керуючі сигнали.
2. Вимірні результати.
3. Створення нових сигналів контролю, розраховані на наближення до вказаної мети.

Ці кроки повторюються знову і знову, поки не досягнуть своєї мети. Чим більше разів робот може робити це за секунду, тим кращий контроль буде над системою. Робот Sobot Rimulator повторює ці дії 20 разів на секунду (20 Гц), але багато роботів повинні робити це тисячі або мільйони разів на секунду, щоб мати більш точний контроль.

Загалом, щоразу, як даний робот проводить вимірювання за допомогою своїх давачів, він використовує ці вимірювання для оновлення своєї внутрішньої оцінки стану навколишнього світу - наприклад, відстані від своєї мети. Він порівнює цей стан із еталонним значенням того, яким він хоче, щоб стан був (для відстані він хоче, щоб стан був нульовим), і обчислює помилку між бажаним станом та фактичним станом. Як тільки ця інформація стане відомою, генерація нових керуючих сигналів може бути зведена до проблеми мінімізації помилки, яка врешті-решт перемістить робота до мети.

1.2.1.9 Спрощення моделі

Для того, щоб керувати роботом, який потрібно запрограмувати, для початку потрібно послати сигнал лівого колеса кажучи як швидко повернути, і окремий сигнал правому колесу говорити як швидко повернути. Давайте називати ці сигнали vL і vR. Однак постійно мислити з точки зору vL і vR дуже громіздко. Замість того, щоб запитувати: "Наскільки швидко потрібно повертати ліве колесо і як швидко має крутитись праве колесо?" Цілісніше запитати: "Наскільки швидко порібно, щоб робот рухався вперед, і як швидко потрібно, щоб він повернувся, або змінив свій напрямок?" Назвемо ці параметри швидкістю V і кутовою (обертовою) швидкістю ω. Виявляється, можна застосовувати дану модель на V і ω замість L і R, і тільки один раз визначити, для цього потрібно, щоб даний запрограмований робот міг рухатися, математично перетворити ці два значення в vL і vR це потрібно для фактичного керування колесами робота. Це відоме як одноколісна модель керування (Рис.1.).

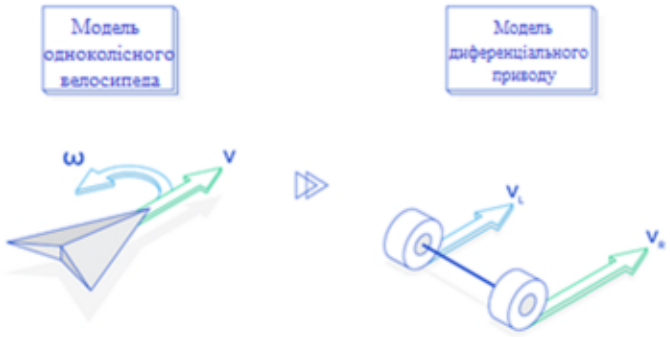

Рис.1.2 – моделі керування

Оцінка стану: Робота, розпізнавання

За допомогою своїх датчиків робот повинен спробувати оцінити стан навколишнього середовища, а також власний стан. Ці оцінки ніколи не будуть ідеальними, але вони повинні бути досить хорошими, оскільки робот

базуватиме всі свої рішення на цих оцінках. Використовуючи лише свої давачі наближення та програму, він повинен спробувати вгадати наступне:

Напрямок до перешкод

Відстань від перешкод

Положення робота

Заголовок робота

Перші дві властивості визначаються показаннями датчика наближення і є досить простими. Функція API read_proximity_sensors() повертає масив з дев'яти значень, по одному для кожного датчика [6].

Таким чином, якщо це значення показує показник, що відповідає відстані 0,1 метра, це означає що є перешкода за 0,1 метр, на 75 градусів ліворуч. Якщо перешкоди немає, датчик поверне показник максимального діапазону 0,2 метра. Таким чином, якщо робот прочитає 0,2 метра на давачі сім, можна припустити, що насправді немає перешкод у цьому напрямку.

Через те, як працюють інфрачервоні давачі (вимірювання інфрачервоного відбиття), повернені ними цифри є нелінійним перетворенням фактичної виявленої відстані. Таким чином, функція Python для визначення вказаної відстані повинна перетворювати ці показання у метри.

Всеж таки, є конкретна модель датчика в цій робототехнічній системі Python, тоді як у реальному світі датчики постачаються із супровідним програмним забезпеченням, яке повинно забезпечувати подібні функції перетворення з нелінійних значень у вимірювачі.

Визначення положення та напрямку роботи (разом відоме як поза у програмуванні робототехніки) є дещо складнішим. Даний робот використовує одометрію для оцінки своєї пози. Сюди входять колеса. Помірявши, скільки оберталося кожне колесо з часу останньої ітерації

контуру управління, можна отримати хорошу оцінку того, як змінилася поза робота, але лише якщо зміна невелика.

Це одна з причин, чому важливо дуже часто повторювати контур управління у реального робота, де двигуни, що рухають колеса, можуть бути не ідеальними. Якби робот чекавби занадто довго, щоб виміряти колесо, то обидва колеса могли б зробити досить багато, і неможливо буде оцінити, куди він потрапить.

Враховуючи випадковий програмний симулятор, можна дозволити запуск обчислення одометрії на частоті 20 Гц - на тій самій частоті, що і контролери. Але може бути гарною ідеєю мати окремий потік Python, який працює швидше, щоб зафіксувати менші рухи міток.

Нижче наведено повну функцію одометрії, supervisor.py яка оновлює оцінку пози робота. Зверніть увагу, що поза робота складається з координат x і y та курсу theta, який вимірюється в радіанах від позитивної осі X. Позитивне x- на схід, а позитивне y- на північ. Таким чином, заголовок 0 вказує, що робот повернений прямо на схід. Робот завжди приймає свою початкову позу (0, 0), 0.

Тепер, коли робот здатний сформувати хорошу оцінку реального світу, давайте використаємо цю інформацію для досягнення поставлених цілей.

Методи програмування на роботах Python: поведінка "цілі до мети"

Найвищою метою існування цього маленького робота є досягнення цілі. Почнемо з того, що трохи спростимо світогляд і припустимо, що на цьому шляху немає жодних перешкод [6].

Потім це стає простим завданням, яке можна легко запрограмувати на Python. Якщо робот будемо йти вперед, стоячи перед ціллю, він туди потрапить. Завдяки даній одометрії відомо, якими є поточні координати та напрямок. Також відомо, якими є координати цілі, оскільки вони були

попередньо запрограмовані. Тому, використовуючи невелику лінійну алгебру, можна визначити вектор від розташування до цілі.

Зверніть увагу, що отримання вектора до мети відбувається у системі відліку робота, а не у світових координатах. Якщо мета знаходиться на осі X в опорній системі робота, це означає, що об'єкт знаходиться прямо перед роботом. Таким чином, кут цього вектора від осі X - це різниця між бажаним курсом та курсом, на якому робот хоче знаходитись. Іншими словами, це помилка між поточним станом і тим, яким потрібно, щоб був поточний стан робота. Отже, потрібно відрегулювати швидкість повороту ω так, щоб кут між курсом і ціллю змінився на 0.

Тепер, коли робот має свою кутову швидкість ω, як визначити дану пряму швидкість v? Хорошим загальним емпіричним правилом є правило: Якщо робот не робить повороту то робот може рухатися вперед на повній швидкості, і тоді чим швидше відбувається повертання, тим більше роботу слід сповільнювати. Це, як правило, допомагає підтримувати стабільність даної системи та діяти в рамках цієї моделі. Таким чином, v є функцією від ω.

Рекомендацією детальніше розробити цю формулу є врахування того, що, як правило, сповільнюємось, коли знаходимось біля цілі, щоб досягти її з нульовою швидкістю. Ще декілька кроків і майже завершиний єдиний цикл управління. Єдине, що залишилось зробити, це перетворити ці два параметри моделі одноколісного велосипеда на диференціальні швидкості обертання коліс і відправити сигнали на колеса. Ось приклад траєкторії робота під контролером руху до мети, без перешкод (Рис.1.3):

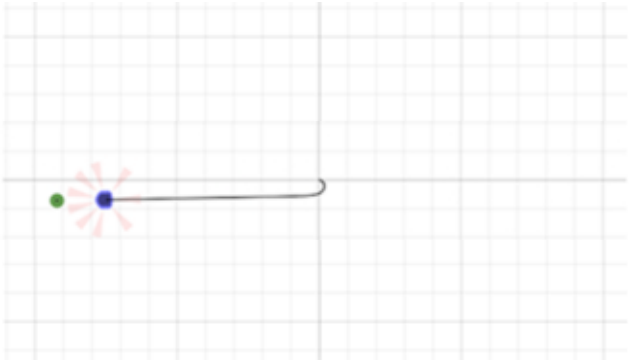

Рис.1.3 – траєкторія робота без перешкод

Як можна побачити, вектор до мети є ефективним посиланням для нас, щоб базувати контрольні розрахунки. Це внутрішнє уявлення про те, “куди робот хоче піти”. Як можна побачимо, єдина основна різниця між досягненням мети та іншими способами поведінки проявляється в тому, що інколи рух до мети є поганою ідеєю, тому потрібно обчислити інший контрольний вектор.

1.2.1.10 Методи програмування роботів на Python

Наочним прикладом є рух до мети, коли в цьому напрямку є перешкода. Замість того, щоб стрімголов натрапляти на речі, які у нас є, спробуємо запрограмувати закон управління, який змушує робота уникати їх.

Щоб спростити сценарій, давайте тепер повністю забудемо цільову точку і просто поставимо наступну мету:

Коли перед роботом немає перешкод, рухайся вперед. Коли зустрінете перешкоду, відверніться від неї, поки вона не зникне перед роботом.

Відповідно, коли перед роботом немає перешкоди, потрібно, щоб опорний вектор просто вказував вперед. Тоді ω буде дорівнювати нулю, а v буде максимальною швидкістю. Однак, як тільки виявляємо перешкоду за допомогою давачів наближення, потрібно, щоб контрольний вектор вказував у будь-якому напрямку, який знаходиться далеко від перешкоди. Це призведе

до того, що ω зміниться, щоб відвернути нас від перешкоди, а v знизиться, щоб переконатися, що випадково не натрапимо на перешкоду в процесі.

Акуратний спосіб створити бажаний контрольний вектор - це перетворити дев’ять показань відстані на вектори та взяти зважену суму. Коли перешкод не виявлено, вектори будуть підсумовувати симетрично, в результаті чого буде отриманий опорний вектор, який вказує прямо вперед за бажанням. Але якщо датчик, скажімо, на правій стороні виявляє перешкоду, це додасть менший вектор до суми, і результатом буде опорний вектор, який буде зміщений вліво (Рис.1.4).

Для загального робота з іншим розміщенням давачів може застосовуватися та сама ідея, але потрібно буде врахувати нюанси будови самого робота.

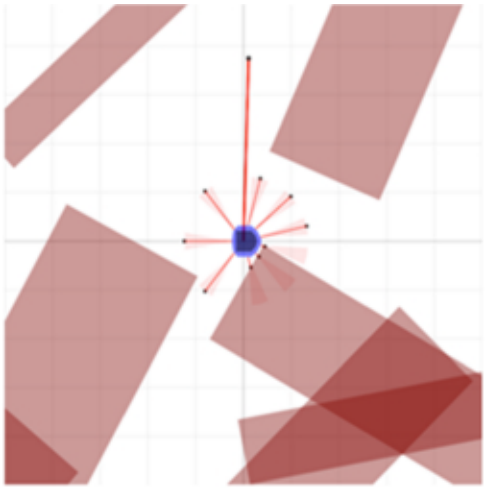

Рис.1.4 – зображення опорного вектора

Використовуючи отриманий результат ao_heading_vector як орієнтир для робота, який намагається відповідати, ось результати запуску програмного забезпечення робота в моделюванні, використовуючи лише контролер уникнення перешкод, повністю ігноруючи цільову точку. Робот безцільно стрибає навколо, але він ніколи не стикається з перешкодою і навіть встигає орієнтуватися в дуже вузьких місцях (Рис.1.5).

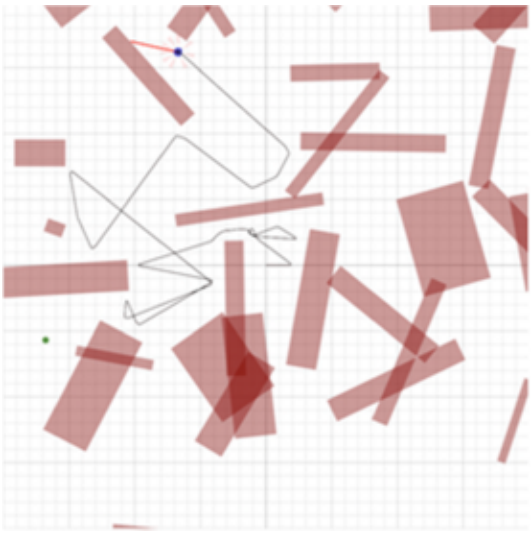

Рис.1.5 – орієнтування робота між перешкодами

Методи програмування робота на Python: поведінка наступних дій

Ось ідея: коли зустрічаємо перешкоду, візьміть два показання датчика, які є найближчими до перешкоди, і використовуйте їх для оцінки поверхні перешкоди. Потім просто встановіть опорний вектор паралельним цій поверхні. Продовжуйте слідувати за цією стіною до тих пір, поки (А) перешкода більше не буде між обєктом керування та ціллю і (Б) знаходження робота ближче до цілі, ніж коли починали. Тоді можна бути впевнені, що правильно подолали перешкоду.

Завдяки обмеженій інформації не можна точно сказати, чи буде швидше обійти перешкоду з ліва або з права. А би вирішити свою думку, потрібно обирати напрямок, який негайно наблизить нас до мети. Для розуміння, яким чином це є, для цього потрібно знати опорні вектори поведінки «йти до мети» та поведінки «уникнення перешкод», а також обидва можливі опорні вектори наступної стінки [6].

Ось ілюстрація того, як приймається остаточне рішення (у цьому випадку робот вирішить піти ліворуч) (Рис.1.6):

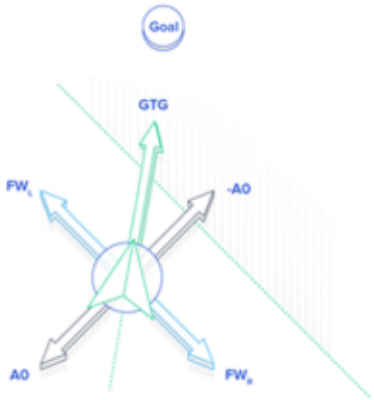

Рис.1.6 – прийняття остаточного рішення

Визначення опорних векторів наступної стінки виявляється дещо більш складним, ніж опорні вектори уникнення перешкоди або досягнення мети.

1.2.1.11 Дизайн остаточного контролю

Остаточний дизайн управління використовує поведінку наступних стін майже для всіх зустрічей з перешкодами. Однак, якщо робот опиниться в тісному місці, небезпечно близькому до зіткнення, він перейде в режим чистого уникнення перешкод, поки не буде на більш безпечній відстані, а потім повернеться до наступної стіни. Після успішного подолання перешкод робот переходить на мету. Ось остаточна діаграма стану (Рис.1.7):

Рис.1.7 – діаграма стану

Ось робот успішно орієнтується в переповненому середовищі, використовуючи цю схему управління (Рис.1.8):

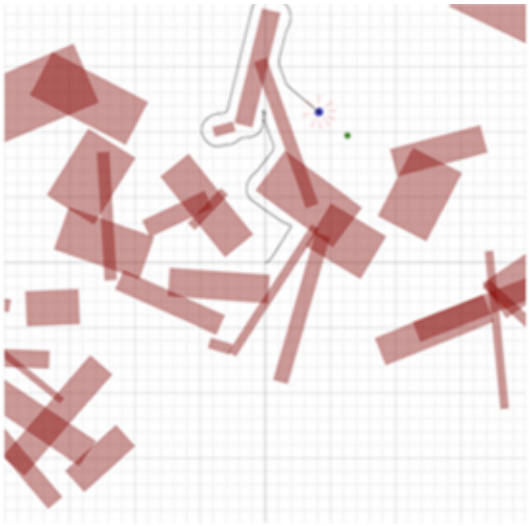

Рис.1.8 – орієнтування в переповненому середовищі

Додатковою особливістю робота, яку можна спробувати реалізувати, є спосіб уникнути кругових перешкод, переключившись на мету до мети як омога швидше, замість того, щоб слідувати за межею перешкод до кінця (якої для кругових об’єктів не існує) [6].

1.2.2 Принцип дії і призначення ультразвукового давача

Принцип роботи ультразвукових давачів побудований на вимірюваннях ультразвукових хвиль в навколишнім середовищем у якому відбувається взаємодія [10].

До таких належать коливання механічні, які перевищують діапазон частоти понад 20 000 Гц, яке є більше по значенні чим верхня межа сприйняття звуку вухом людини. Взаємодія коливань в різних середовищах залежить від самого середовища і властивостей в ньому. Для прикладу в різних газах розповсюдження коливань буде в діапазоні від 200 до 1300 м/с , в рідинних середовищах від 1100 до 2000 м/с, а в твердих середовищах (матеріалів) – від 1500 до 8000 м/с. Найбільш замітна залежність швидкості розповсюдження коливань в газах від тиску середовища.

При межі поділу кожне середовище має свої коефіцієнт відображення ультразвукових коливань, в такому випадку в різних середовищах буде і різна звукопоглинальна властивість. Тому для різних ультразвукових давачів дані про різних неелектричних величинах получаються за допомогою вимірювання параметрів коливань ультразвуку: загасання амплітуди даних коливань, зсуву по фазі даних коливань та часу їх розповсюдження.

Способи вимірювання ультразвуком належить до методів електричних, так як збурення і приймання коливань ультразвуку виконується електричним методом. В більшості для використання застосовують магнітострикційні і п’єзоелементи перетворювачі. П’єзоелектричні давачі перетворюю дані про тиски в електричний сигнал. Це перетворення використовує прямий п’єзоефект. Ця технологія застосовується в приймачах які випромінюють ультразвук. Зворотній п’єзоелектричний ефект базується на стисканні і навпаки розтягненні п’єзокристала, до якого застосовується змінна напруга. Для активації ультразвукових хвиль і застосовується даний ефект. За допомогою певних дій п’єзоелемент має можливість включатись позмінно з приймачем та з випромінювачем ультразвукових хвиль.

Магнітострикційні ультразвукові випромінювачі працюють за допомогою явища деформації так званих феромагнітів в непостійному магнітному полі.

Для прикладу роботу давача ультразвуку можна пояснити на прикладі ехолокатора – пристрій за допомогою якого вимірюють відстань до дна моря (рис. 1.9). Підчас надходження змінної напруги на п’єзоелемент 1 активуються коливання ультразвуку, які в свою чергу направлені прямо вниз. Відображений ультразвукова хвиля при надходжені сприймається п’єзоелементом 2. Вимір часу *t* між відправленими імпульсами і отриманими здійснює електро приладом 3. Глибина моря пропорційна проходженні цьому часу і швидкості переміщення звуку u у водяному середовищі:

$$h = \frac{ut}{2}. \quad (1.9)$$

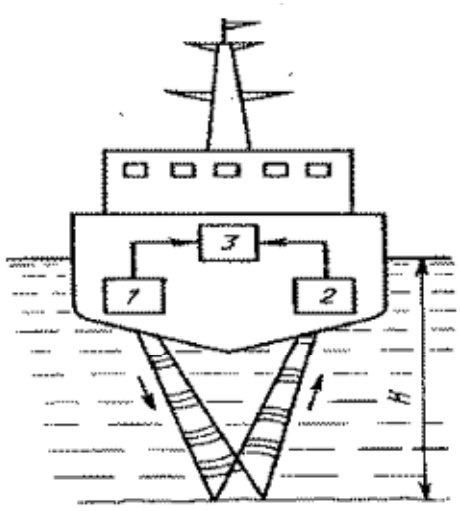

Рис.1.9 – ехолокатор з ультразвуковим давачем

Прилад вираховує відстань у метрах. За такоюж технологією працює і ультразвуковий локатор, працює в горизонтальному напрямку для визначення відстані до перешкоди. В деяких тварин є природні ехолокатори

наприклад у летючих мишей, він діє за таким принципом як і ультразвуковий локатор.

Ультразвукові хвилі містять значно більше енергії, а ніж звукові, так як квадрат частоти пропорційний енергії. Але в порівнянні просто здійснює напрям ультразвукових випромінювань [10].

Завдяки ультразвуковим давачам виявляють недоліки в деталях виготовлених з металу: в відливках порожнини; в виробах тріщини і тому поідбне. В виявленні дефектів та неруйнівних способах контролю дані давачі відіграють дуже важливу роль. Також дані давачі використовують в вимірюваннях витрачання, рівня, тиску.

1.2.3 Інфрачервоний датчик руху

Інфрачервоний датчик руху – це пристрій який характеризується здатністю реагувати на інтенсивність зміни фонового теплового випромінювання в зоні його спрацювання.

Тепловими випромінюваннями наділені не тільки люди, але і всі інші об’єкти які нас оточують. При попаданні об’єкта в зону дії інфрачервоного давача при певних розмірах і при певній швидкості, то відбудеться спрацювання давача та передача інформації про об’єкт на електронну схему керування для задіяння заздалегіть визначених дій, визначеними пристроями. Даним пристроєм може бути як будь який регулятор так і вимикач освітлення приміщення або навіть охоронна сигналізація та інше. [7].

Принцип роботи роботи інфрачервоного датчика зображений на рисунку 1.10. [8].

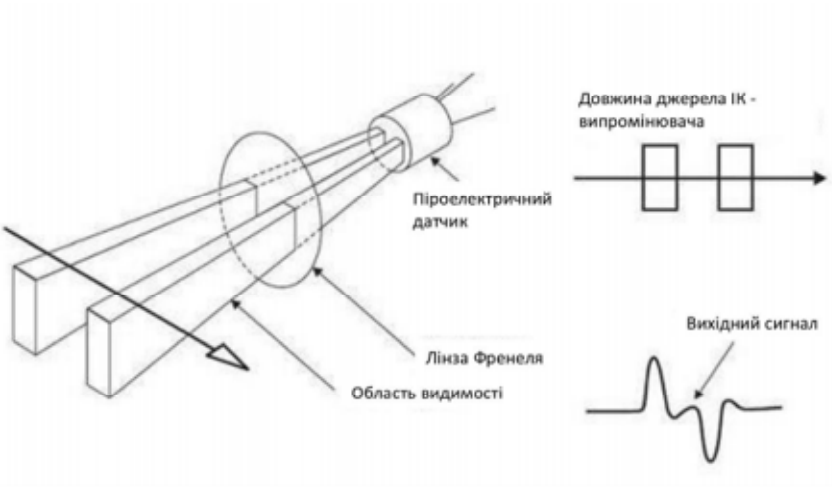

Рис.1.10. Схема роботи інфрачервоного датчика руху

Давачі даного типу використовуються у різних системах з можливістю застосування у різних цілях автоматизації, як на виробництві так і в житлових будинках та інших об’єктах де це необхідно. Так як застосування даних давачів немає обмежень.

В основі будови конструкції інфрачервоних давачів лежать піроприймачі, що застосовується для розпізнавання інфрачервоного випромінювання та мультилінза, яка в собі містить велику кількість маленьких лінз. Кожна мікролінза справцьовує на відповідний сигмент та працює зі мвоєю зоною об’єму робочого середовища. З цього робим висновки що чим більше мультилінза захоплює сигментів тим чутливішим буде давач [8].

В основі конструкції інфрачервоного давача застосовуються здвоєні або навіть декілька пар здвоєних піроелементів, це потрібно для того щоб робота давача була більш точною, для відсікання помилкових спрацювань застосовують світлові які викликаються зміною температури фону [8]. При застосування двох пар здвоєних піроелемнтів максимально виключаються можливості помилкових спрацювань. Такі конструкції використовують в нових моделях інфрачервоних давачів [8].

При використанні інфрачервоного давача потрібно дотримуватись певних правил. Для початку на давач не повинен попадати прямий потік світла від ламп або від інших джерел світла, при такому освітленні давач може працювати з похибками. Також якщо у зоні контролю давача не повинно бути перешкод, різні сторонні об’єкти, що що можуть заважати роботі пристрою. Також скляні перешкоди вони також будуть перешкоджати нормальній роботі давача, так як інфрачервоне світло не проходить через скло.

Якщо все ж така перешкода зявилась у зону дії датчика і її ніяк не позбутись то, це може утворити сліпу зону через яку небуде фіксуватись переміщення об’єктів, оскільки інфрачервоне світло, просто не зможе птрапити в лінзу давача.

Основною характеристикою інфрачервоного давача руху, являється радіус виявлення об’єкту який виконує переміщення. Зона переміщення об’єктів повинен обов’язково досягати кутів приміщення, а якщо давач не досягає то потрібно встановити ще декілька давачів [8-9]. Всі інфрачервоні давачі наділені властивістю кругової діаграми виявлення але якщо однієї діаграми недостатньо для того щоб перекрити ввесь простір, тоді необхідно вствновити ще декілька давачів, щоб їхні діаграми виявлення перекривали один одну, це надість більшу якість системи безпеки та автоматизації в цілому. Є велика кількість різних видів інфрачервоних давачів. Самі простіші це давачі які реагують на переміщення, але є і більш функціональні давачі, з більш розширеними можливостями та властивостями. Для прикладу є така модель ІС давачів, які можуть слідкувати за освітленням, при умові що люди є у зоні охоплення. Давач може як вмикати освітлення та і вимикати його, в задежності від інтенсивності навколишнього освітлення [8]..

1.2.4 Система управління поведінкою та переміщення

Система керування поведінкою розділяє дане поставлене завдання на послідовність підзадач і формує поведінку робота для виконання заданих йому завдань. Формує кінцеву точку, відповідні режими дій інформаційно–вимірювальної системи ідентифікації. Завдяки ній отримує інформацю про процес виконання який і передає їх пристрій користувача. Також надає користувачеві оброблені дані про ситуацію і результат, отриману від інформаційно–вимірювальної системи ідентифікації.

Система керування рухом, з огляду на невизначеність середовища динамічні властивості робота і динамічні властивості робота, визначається значення швидкості руху і напрямок для виконання поставленої цілі.

2.Проектна частина

2.1. Постановка задачі

Дистанціно керовані роботизовані системи які мають можливість обминати перешкоди мають ряд переваг перед звичайними роботизованими системами під час дистанційного керування та при винекненні неочікуваних перешкод підчас виконання певних завдань підчас дистанційного керування. Вони мають можливість виконувати роботу та реагувати на перешкоди, що полегшує роботу оператору та збільшує надійність при керуванні роботом в просторі.

Переваги використання дистанційно керованих роботизованих систем з можливістю обминання перешкод полягає в тому що:

1. Вони здатні швидко відреаговувати на перешкоди які вже знаходяться нашляху робота так і перешкоди які зявляються підчас самого керування роботом.

2. Дистанційне керування роботом надає перевагу оператору в обиранні найбільш оптимально швидкого проходження шляху роботом в реальному часу та для найбільш зручного і безпечного переміщення робота в просторі під час його експлоатації. Основою даної роботи є розробка оптимального алгоритму переміщення робота при проходженні ним по ділянці з обєктами які перешкоджають руху роботу підчас його використання.

2.2 Функціонально-елементна база

AlphaBot2 – Pi включають шасі (шасі AlphaBot2 – Base (Рис.2.1)) і адаптивну плату AlphaBot2 – Pi (Рис.2.2). Робот підтримує Raspberry Pi з адаптерною платою AlphaBot2 – Pi.

Завдяки високоінтегрованій модульній конструкції його досить легко зібрати за допомогою застібки, без пайки та без проводки. Через кілька хвилин, витрачених на збірку обладнання.

AlphaBot2-Base:

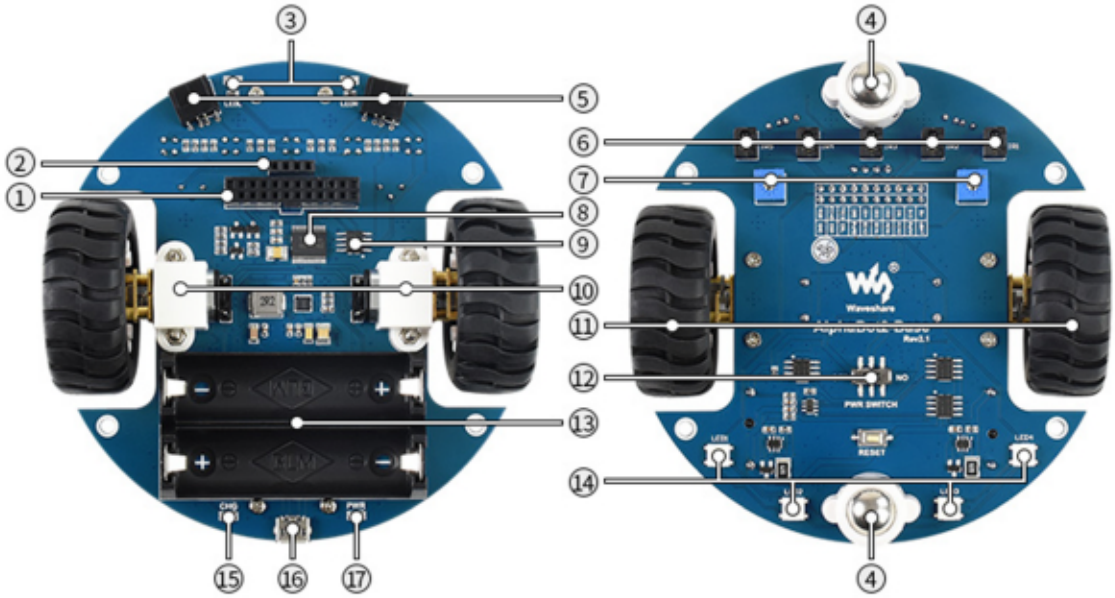

Рис.2.1 – шасі AlphaBot2 – Base

1. Інтерфейс управління: для підключення різних плат адаптерів контролера
2. Розєми для підключення ультразвукового давача
3. Індикатори уникнення об'єктів (перешкод)
4. Всебічне колесо
5. ST188: фотоелектричний інфрачервоний давач для обминання перешкод
6. ITR20001/T: фотоелектричний інфрачервоний давач для слідкування за лініями
7. Потенціометр для керування дальності уникнення перешкод
8. TB6612FNG подвійний гідравлічний мотор H-Bridge
9. Компаратор напруги LM393

10. Швидкість зменшення мікроредуктора N20 1:30, 6 В/ 600 об/хв
11. Гумові колеса шириною 19мм, діаметром 42мм
12. Виключатель живлення
13. Лоток акумулятора: використовує 14500 акумулятори
14. WS2812B: справжні кольорові RGB світлодіоди
15. Показник заряду акумулятора
16. 5В порт USB для зарядки акумулятора
17. Індикатор живлення

AlphaBot2-Pi:

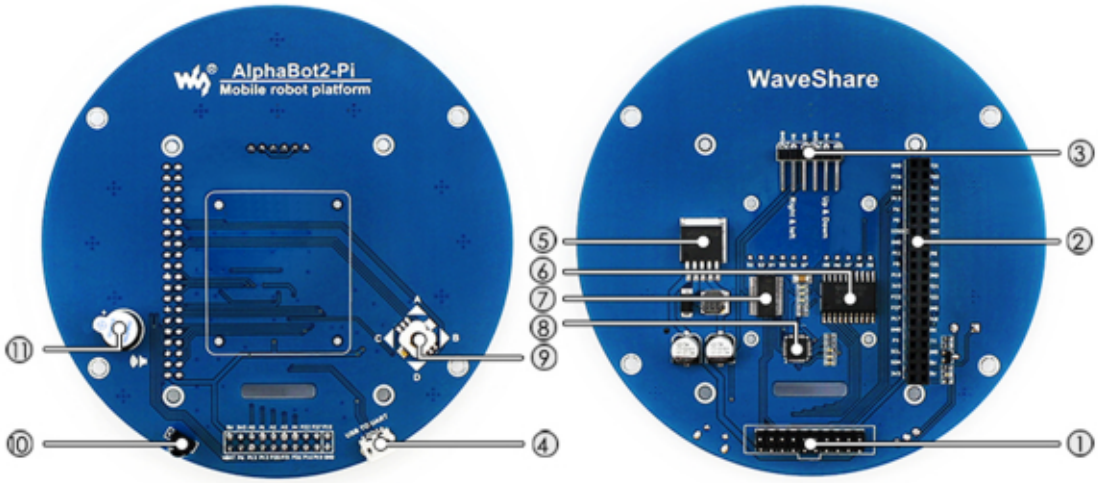

Рис.2.2 – адаптивна плата AlphaBot2 – Pi

1. Інтерфейс управління AlphaBot2: для зєднання з AlphaBot2 – Base
2. Інтерфейс Raspberry Pi: для зєднання з Raspberry Pi 3 Модель B
3. Сервоінтерфейс
4. USB TO UART: легко керувати Pi через UART
5. LM2596: 5В регулятор напруги
6. TLC1543: 10-розрядна мікросхема AD, дозволяє Pi використовувати аналогові датчики

7. PCA9685: сервоконтролер, робіть більш плавним обертання головки каструлі

8. CP2102: перетворювач USB TO UART

9. Джойстик

10. ІЧ-приймач

11. Зумер

2.3 Платформа Raspberry Pi 3 B

Raspberry Pi 3 Model B (Рис.2.3) - Повноцінне безшумний комп'ютер розміром з банківську карту, при цьому з 64-х бітовим чотирьохядерним процесором ARM Cortex-A53 на однокристальному чіпі Broadcom BCM2837.

Рис.2.3 – Raspberry Pi 3 Model B

Чіп BCM2837 (Рис.2.4):

Рис.2.4 – BCM2837

На Raspberry Pi 3 встановлений 64-х бітний процесор Broadcom BCM2837 на архітектурі ARM Cortex-A53 з тактовою частотою 1,2 ГГц і модулем оперативної пам'яті на 1 ГБ. Процесор і пам'ять розміщені за технологією «package-on-package» безпосередньо на процесорі. BCM2837 включає в себе також двоядерний графічний співпроцесор Video Core IV® Multimedia, який забезпечує Open GL ES 2.0, апаратне прискорення Open VG і 1080p30 H.264 декодування.

Розпіновка контактів Raspberry Pi (Рис.2.5):

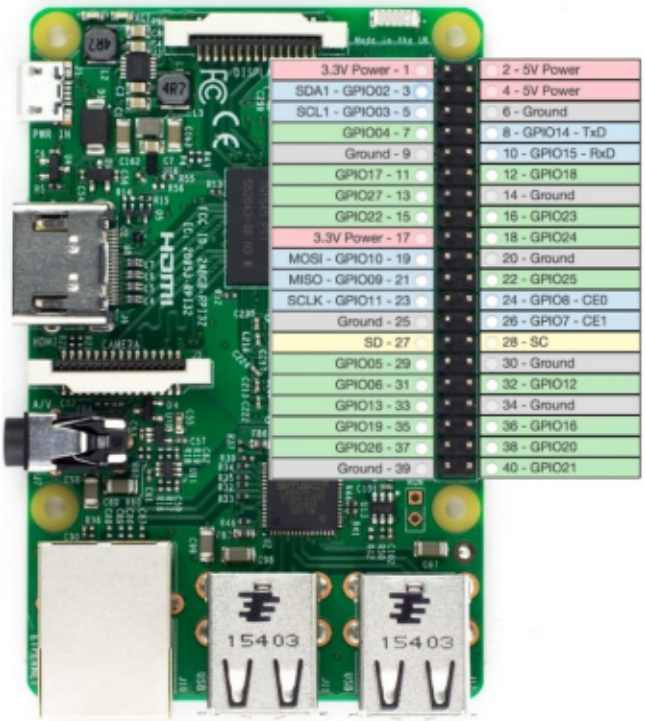

Рис.2.5 – розпіновка Raspberry Pi

Детальний опис розпіновки Raspberry Pi (Рис.2.6):

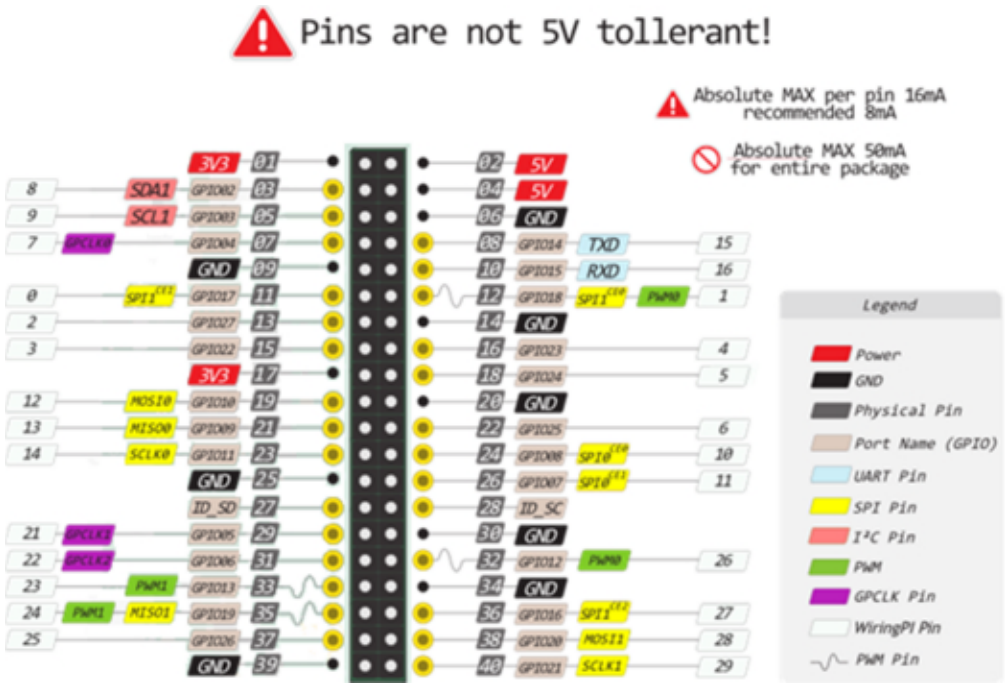

Рис.2.6 – розпіновка для WiringPi Pin

Піни живлення:

- **5V:** На вхід надходить напруга 5В при підключенні плати через USB.
- **3V3:** Пін від стабілізатора напруги з виходом 3,3 вольта і максимальних струмом 1 А. Регулятор забезпечує живлення процесора і інших елементів плати.
- **GND:** Вивід землі.

Порти введення / виводу:

- **Physical Pin:** нумерація (розпіновка), що відповідає за фізичне розташування контакту на гребінці.
- **GPIO:** нумерація (розпіновка) контактів процесора Broadcom. Може стати в нагоді при роботі з пакетом Rpi.GPIO.
- **WiringPi Pin:** нумерація (розпіновка) контактів для пакета Wiring Pi. Це Arduino-подібна бібліотека для роботи з GPIO-контактами.

- **ШИМ:** плата має на собі два канали ШИМ по два потоки в кожному:
  - PWM0 12, 18;
  - PWM1 13, 19.
- **I²C:** SDA12, SCL13. Для спілкування з периферією з синхронного протоколу, через два дроти.
- **SPI:** До SPI0 можна підключити два наявних пристрої, а до SPI1 - три. Вибір здійснюється сигналом на піні CEx.
  - SPI0:MOSI0 10, MISO0 9, SCLK0 11, CE0 8, CE1 7;
  - SPI1:MOSI1 20, MISO1 19, SCLK1 21, CE0 18, CE1 17, CE2 16.
- **UART:** 14 ,15. Асинхронний протокол послідовної передачі даних по двох кабелях RX і TX, який дозволяє обійтися безтактового сигналу.

Характеристики:

- Процесор: 64-розрядний 4-ядерний ARM Cortex-A53 на однокристальному чіпі Broadcom BCM2837;
- Тактова частота: 1,2 ГГц
- RAM: 1 ГБ LPDDR2 SDRAM;
- Цифровий аудіо / відео вихід: HDMI;
- Аналоговий аудіо / відео вихід: 3,5 мм (4 pin);
- USB порти: USB 2.0 × 4;
- Підключення до інтернет мережі:
  - WiFi: 802.11n;
  - Ethernet: 10/100 Мб RJ45;
- Bluetooth: Bluetooth 4.1, Bluetooth Low Energy;
- Послідовний роз'єм дисплея: Display Serial Interface (DSI);
- Послідовний роз'єм відеокамери: MIPI Camera Serial Interface (CSI-2);

- Слот карта пам'яті: MicroSD;
- Порти введення-виведення: 40;
- Розміри: 85 × 56 × 17 мм.

2.4 Налаштування Raspberry Pi

2.4.1 Встановлення бібліотек

Введіть команди нижче, щоб встановити бібліотеки

```
sudo apt-get update

sudo apt-get install ttf-wqy-zenhei

sudo apt-get install python-pip

sudo pip install RPi.GPIO

sudo pip install spidev

sudo apt-get install python-smbus

sudo apt-get install python-serial

sudo pip install rpi_ws281x
```

Опис основних команд які використовуються для налаштування Raspberry Pi:

Update – команда для обновлення списків пакетів та установки їхнього обновлення. (Команда не обновлює програмне забезпечення Raspberry).

ttf-wqy-zenhei – команда для вирішення проблем з помилками кодування.

python-pip – команда для системного керування пакетами, використовується для встановлення і керування пакетами які написані на Python.

RPi.GPIO – команда для установки пакета який дозволяє роботу з GPIO на Raspberry.

spidev – команда для установки пакета який дозволить роботу з різним SPI устаткуванням.

python-smbus – команда для установки пакета який налаштовує та надає можливість роботи з шиною I2C на Python в Raspberry Pi.

python-serial – команда для установки і обновлення пакетів функціоналу Python на Raspberry Pi

rpi_ws281x – команда для керування підсвіткою на Alphabot2.

2.4.2 Керування Alphabot2

Опис програми дистанційного керування завдяки спільній мережі Wi-Fi. Сама програма знаходиться в файлі main.py.

Першим кроком потрібно імпортувати всі необхідні пакети для роботи з сервоприводами та функціоналом безпровідної мережі. Дані дії виконуються в 2-8 рядках, в 9-11 рядках (Рис.2.7) прописуються файли з яких буде імпортуватись додаткова інформації для керування мисервоприводами і підключення їх до портів на Raspberry.

```
1  # імпортування необхідних пакетів
2  from bottle import get,post,run,route,request,template,static_file
3  from AlphaBot import AlphaBot
4  from PCA9685 import PCA9685
5  import threading
6  import socket #ip
7  import os
8
9  Ab = AlphaBot()
10 pwm = PCA9685(0x40)
11 pwm.setPWMFreq(50)
12
```

Рис.2.7 – імпорт пакетів та необхідних файлів

В рядках 13-19 (Рис.2.8) прописано з якою стартовою швидкістю при підключенні будуть рухатись сервприводи та їх можливість зміни швидкостей кожного поокремо при поворотах.

```
13  #Встановіть параметри сервоприводу
14  HPulse = 1500  #Встановлює початковий імпульс
15  HStep = 0      #Встановлює початкову довжину кроку
16  VPulse = 1500  #Встановлює початковий імпульс
17  VStep = 0      #Встановлює початкову довжину кроку
18  pwm.setServoPulse(1,VPulse)
19  pwm.setServoPulse(0,HPulse)
20
```

Рис.2.8 – стартова швидкість

Далі в 21-31 рядках (Рис.2.9) вказуються тимчасові файли в яких будуть зберігатись дані про підключення до даної безпровідної мережі та спільного пристрою з якого буде відбуватись керування.

```
21  @get("/")
22  def index():
23      return template("index")
24
25  @route('/<filename>')
26  def server_static(filename):
27      return static_file(filename, root='./')
28
29  @route('/fonts/<filename>')
30  def server_fonts(filename):
31      return static_file(filename, root='./fonts/')
32
```

Рис.2.9 – створення файлів

Наступним кроком (рядки 33-37 прописується що при підключенні пристрою з якого відбуватиметься керування активується частина коду який відповідає за відповідність кнопок керування до віповідного функціоналу який буде виконуватись при натисканні на них (рядки 38-72) (Рис.2.10) .

```
@post("/cmd")
def cmd():
    global HStep,VStep
    code = request.body.read().decode()
    speed = request.POST.get('speed')
    print(code)
    if(speed != None):
        Ab.setPWMA(float(speed))
        Ab.setPWMB(float(speed))
        print(speed)
    if code == "stop":
        HStep = 0
        VStep = 0
        Ab.stop()
        print("stop")
    elif code == "forward":
        Ab.forward()
        print("forward")
    elif code == "backward":
        Ab.backward()
        print("backward")
    elif code == "turnleft":
        Ab.left()
        print("turnleft")
    elif code == "turnright":
        Ab.right()
        print("turnright")
    elif code == "up":
        VStep = -5
        print("up")
    elif code == "down":
        VStep = 5
        print("down")
    elif code == "left":
        HStep = 5
        print("left")
    elif code == "right":
        HStep = -5
        print("right")
    return "OK"
```

Рис.2.10 – функціонал керування

Далі прописуємо затримку при натисканні на кнопки керування (рядки 74-76), та прописуємо створення ip-адреси для підключення пристрою керування (рядки 78-81) (Рис.2.11). Ip-адреса буде виведена на дисплей термінала.

```
    global t        #Примітка: використовуйте глобальну змінну!
    t = threading.Timer(0.02, timerfunc)
    t.start()

s = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
s.connect(('8.8.8.8',80))
localhost=s.getsockname()[0]
run(host = localhost, port = 8000)
```

Рис.2.11 – генерація Ip-адреси

Включення та додатковий функціонал програми:

Для зупинки останньої програми потрібно натиснути **Ctrl** + **C.**

Введіть команди нижче, щоб запустити програму веб-контролю

```
cd ~ / AlphaBot2 / Web-Control /

sudo python main.py
```

Відкрийте браузер на своєму смартфоні або ПК (радимо скористатися браузером Chrome), введіть IP-адресу Pi із суфіксом 8000. Наприклад: 192.168.0.120:8000 (Рис.2.12).

Увійдіть в Інтернет управління, тоді ви можете керувати двигуном та камерою за допомогою віртуальних кнопок.

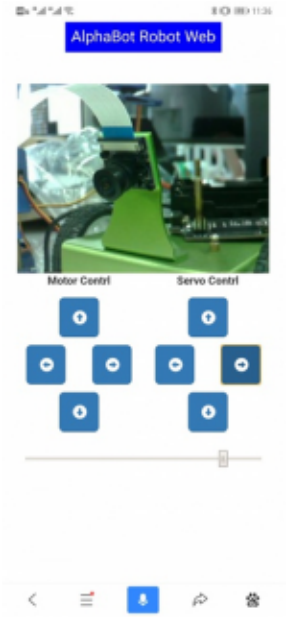

Рис.2.12 – зображення інтерфейсу в браузері

Є можливість встановити програму для включення під час завантаження. Відкрийте файл /etc/rc.local у своєму редакторі та додайте в нього наступний рядок.

```
sudo vi /etc/rc.local
```

Додайте команди після коментарів. Будь ласка, переконайтеся, що вихід 0 є останнім рядком. А потім збережіть модифікацію та вийдіть.

```
cd ~ / AlphaBot2 / mjpg-streamer / mjpg-streamer-Experimental /

sudo ./start.sh &

cd ~ / AlphaBot2 / Web-Control

sudo python main.py &
```

Оскільки Raspberry Pi використовує динамічний IP, який може змінюватися кожного разу при перезавантаженні пристрою, ви можете додати наступні рядки в каталог /etc/dhcpcd.comf, щоб встановити статичний IP:

```
interface wlan0

static ip_address = 192.168.6.234/24

static routers = 192.168.6.1
```

“192.168.6.1” - це IP-адреса маршрутизатора, а “192.168.6.234” - це IP-адреса, яку потрібно встановити, після чого “/ 24” не слід видаляти.

2.4.3 Уникнення інфрачервоних перешкод

Опис програми для обминання перешкод.

Для початку прописуємо імпортування всіх необхідних пакетів для роботи з ультразвуковим давачем (рядки 2-4)та файл з якого будуть

підтягуватись дані про роботу сервоприводів підчас руху робота (рядок 6). Також прописуємо зміну швидкості сервопроводів при виявленні перешкоди для того щоб відбулось уникнення перешкоди (рядки 8-9). В рядках 11-14 прописуємо функціонал роботи з бібліотекою RPi/GPIO (Рис.2.13).

```
1  # імпортування необхідних пакетів
2  import RPi.GPIO as GPIO
3  import time
4  from AlphaBot2 import AlphaBot2
5
6  Ab = AlphaBot2()
7
8  DR = 16
9  DL = 19
10
11 GPIO.setmode(GPIO.BCM)
12 GPIO.setwarnings(False)
13 GPIO.setup(DR,GPIO.IN,GPIO.PUD_UP)
14 GPIO.setup(DL,GPIO.IN,GPIO.PUD_UP)
15
```

Рис.2.13 – підключення пакетів

Далі прописуємо взаємодію давача з пакетом для справної роботи при зчитування даних з давачів (рядки 16-19). Наступним кроком прописуємо основні дані (рядок 21) та затримку (рядок 24) (Рис.2.14).

```
16 try:
17     while True:
18         DR_status = GPIO.input(DR)
19         DL_status = GPIO.input(DL)
20 #       print(DR_status,DL_status)
21         if((DL_status == 0) or (DR_status == 0)):
22             Ab.left()
23
24             time.sleep(0.002)
25             Ab.stop()
```

Рис.2.14 – налаштування роботи з давачем

В 27 рядку (Рис.2.15) прописується відключення переривання для клавіатури, а в 28 очищення даних пакета GPIO.

```
27 except KeyboardInterrupt:
28     GPIO.cleanup();
```

Рис.2.15 – відключення переривання

Інструкція для активації програми:

Включіть термінал і вводіть такі команди:

```
cd ~ / AlphaBot2 / python
sudo python Infrared_Obstacle_Avoidance.py
```

Якщо перед роботом немає перешкод, зелений світлодіод на роботі не світиться. Якщо виявлені перешкоди, загориться зелений світлодіод. Якщо світлодіод завжди вимкнений або постійно світиться, ви можете спробувати відрегулювати потенціометри внизу робота, щоб світлодіод працював належним чином.

Робот рухається прямо, коли сенсор зліва не виявляє перешкоди, і повертається праворуч, коли виявляється перешкода (Рис.2.16).

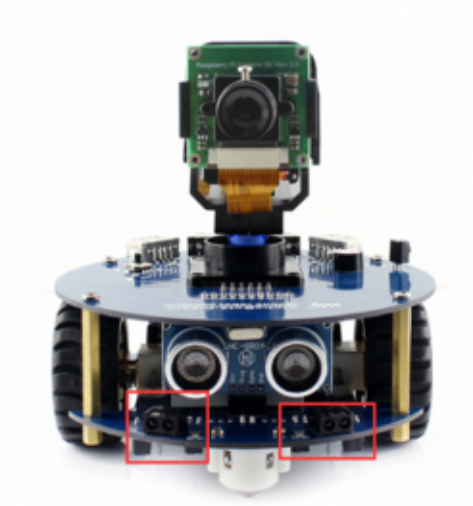

Рис.2.16 – розміщення інфрачервоних давачів

2.4.4 Ультразвукове вимірювання відстані

Окрім інфрачервоних давачів робот може також комплектуватися і ультразвуковим давачем (Рис.2.17).

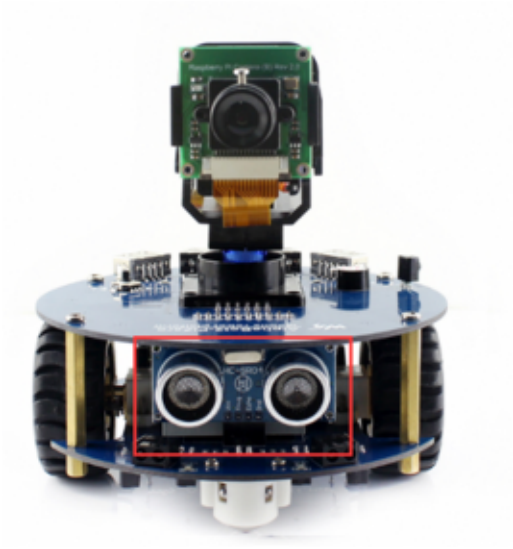

Рис.2.17 – розміщення ультразвукового давача

Включіть термінал і наступні команди:

```
cd ~ / AlphaBot2 / python
sudo python Ultrasonic_Ranging.py
```

Термінал друкує поточну виміряну відстань(Рис.2.18).

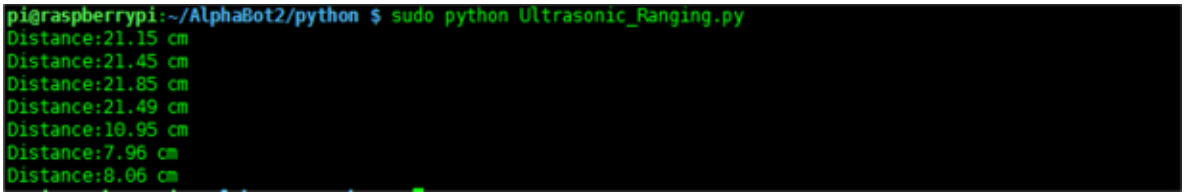

Рис.2.18 – результати вимірювання відстані

2.4.5 Розпізнавання обєктів

Частина 1: розпізнавання об'єктів в реальному часі – Опис коду програми.

Щоб зробити розпізнавання об'єктів в реальному часі, необхідно:

1. Потрібно отримати доступ до веб-камері/відео потоку.
2. Активувати розпізнавання об'єкта покадрово.

Щоб це зробити і переглянути як це робиться, відкриваєм новий файл, називаємо його **real_time_object_detection.py** і прописуємо наступний код:

```
1  # імпортування необхідних пакетів
2  from imutils.video import VideoStream
3  from imutils.video import FPS
4  import numpy as np
5  import argparse
6  import imutils
7  import time
8  import cv2
```

Рис.2.19 – імпортовані бібліотеки

Спочатку імпортуємо необхідні бібліотеки (на рядках 2-8) (Рис.2.19). Для цього необхідний imutils і OpenCV.

imutils - це пакет з набором зручних функцій для полегшення роботи з основним функціоналом обробки зображення, таких як переклад, повертання, зміна розмірів, скелетування, відображення зображень Matplotlib, сортування контурів, виявлення країв та спрощення роботи з OpenCV та Python 2.7 і Python 3.

OpenCV – це бібліотека машинного зору і машинного навчання з доступним вихідним кодом. В ній є понад 2500 алгоритмів, в яких є як стандартні, так і сучасні алгоритми машинного зору і машинного навчання. Дана бібліотека має інтерфейси з функціоналом різних мовах програмування, серед яких є Python, Java, C ++ і Matlab.

Прописуємо код для роботи з командним терміналом.

Наступним кроком проводи аналіз аргументів командного рядка (Рис.2.20).

```
9  # побудова аргументів синтаксичного і аналізу аргументів
10 ap = argparse.ArgumentParser()
11 ap.add_argument("-p", "--prototxt", required=True,
12     help="path to Caffe 'deploy' prototxt file")
13 ap.add_argument("-m", "--model", required=True,
14     help="path to Caffe pre-trained model")
15 ap.add_argument("-c", "--confidence", type=float, default=0.2,
16     help="minimum probability to filter weak detections")
17 args = vars(ap.parse_args())
```

Рис.2.20 – аналіз аргументів

- --prototxt: Шлях до файлу з іменем prototxt Caffe.

- --model: Шлях до попередньо підготовленої моделі.
- --confidence: Мінімальний поріг подібності для розпізнавання об'єкта (значення розпізнаваня за замовчуванням - 20%) .

Додаємо основні об'єкти розпізнавання.

Далі ініціалізуєм список класів і набір кольорів (Рис.2.21).

```
18  # ініціалізовуємо список міток класів, яким навчений MobileNet SSD
19  # розпізнаєм, а потім формуєм набір обмежувальних кольорів для кожного класу
20  CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
21      "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
22      "dog", "horse", "motorbike", "person", "pottedplant", "sheep",
23      "sofa", "train", "tvmonitor"]
24  COLORS = np.random.uniform(0, 255, size=(len(CLASSES), 3))
```

Рис.2.21 - ініціалізація списку класів і набору кольорів

На рядках 20-24 ініціалізовуємо мітки CLASS і відповідні випадкові кольору.

Тепер завантажуємо модель і налаштуємо відео потік (Рис.2.22).

```
25  # завантаження серіалізованої моделі з диска
26  print("[INFO] loading model...")
27  net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])
28  # ініціалізуєм відеопотік, даємо камері розігрітися,
29  # та ініціалізем лічильник FPS
30  print("[INFO] starting video stream...")
31  vs = VideoStream(src=0).start()
32  time.sleep(2.0)
33  fps = FPS().start()
```

Рис.2.22 – налаштування відео потоку

Завантажуємо серіалізовану модель, надаючи посилання на prototxt і моделі (**рядок 27**) - завдяки пакету OpenCV це все робиться дуже просто і швидко.

Далі ініціалізовуємо відео потік (це може бути відеофайл або трансляція за допомогою веб-камери). Спочатку запускаємо VideoStream (**рядок 31**), потрібно чекаємекати декілька секунд поки камера включиться (**рядок 32**) і коли вона активується починаємо підрахунок кадрів в секунду (**рядок 33**). Класи VideoStream і FPS є частиною пакета imutils.

Пишемо код для роботи з кадрами.

Тепер пройдемся по кожному кадру (щоб збільшити швидкість, можна пропускати кадри) (Рис.2.23).

```
34  # цикл для кадрів в відеопотоці
35  while True:
36      # Беремо кадр із потокового відеопотоку та змініть його розмір
37      # на максимальну ширину 400 пікселів
38      frame = vs.read()
39      frame = imutils.resize(frame, width=400)
40      # Беремо розміри рами і перетворіть її на крапку
41      (h, w) = frame.shape[:2]
42      blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)),
43          0.007843, (300, 300), 127.5)
44      # пропустити потік через мережу і отримати виявлення та
45      # прогнозування
46      net.setInput(blob)
47      detections = net.forward()
```

Рис.2.23 – цикл для кадрів відеопотоку

Перше, що треба зробити - це зчититати кадри (**рядок 35**) з потоку, потім замінюємо його розмір (**рядок 38**).

Оскільки в подальшому потрібн ширина і висота, отримаємо їх зараз (**рядок 41**). Далі слід зробити перетворення кадру в blob з модулем dnn (**рядки 42 і 43**).

Тепер переходимо до складного: встановлюємо blob як вхідні дані для даної нейромережу (**рядок 46**) і передаємо ці дані через net (**рядок 47**), яка розпізнає і виявляє об'єкти.

"Фільтруємо" об'єкти.

На даний етапі, дана програма виявиляє об'єкти в відео потоці. Тепер прийшов час подивитися на значення валідності і вирішити, чи повинна програма відобразити квадрат навколо об'єкта і відобразити мітку над ним (Рис.2.24).

```
48      # цикл для виявленнями
49      for i in np.arange(0, detections.shape[2]):
50          # вилучення довіри (тобто ймовірність), пов’язану з
51          # передбачення
52          confidence = detections[0, 0, i, 2]
53          # відфільтрування слабких виявлень, переконавшись, що "впевненість"
54          # перевищує мінімальну впевненість
55          if confidence > args["confidence"]:
56              # витягнутем індекс мітки класу з
57              # `detections`, потім обчислюється (x, y) -координати
58              # обмежувальної рамки для об’єкта
59              idx = int(detections[0, 0, i, 1])
60              box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
61              (startX, startY, endX, endY) = box.astype("int")
62              # відображення передбачення на кадрі
63              label = "{}: {:.2f}%".format(CLASSES[idx],
64                  confidence * 100)
65              cv2.rectangle(frame, (startX, startY), (endX, endY),
66                  COLORS[idx], 2)
67              y = startY - 15 if startY - 15 > 15 else startY + 15
68              cv2.putText(frame, label, (startX, y),
69              cv2.FONT_HERSHEY_SIMPLEX, 0.5, COLORS[idx], 2)
```

Рис.2.24 – цикл для розрахунку ймовірності

Дял початку починаємо проходитись циклами через елементи **detections**, пам'ятаючи, що при розпізнаванні кілька об'єктів можуть бути сприйняті програмою як єдине зображення. Також потрібно зробити перевірку на валідність (тобто ймовірність) для кожного виявлення і розпізнавання. Якщо валідність досить велика (тобто вище заданого порогу), відображаємо ймовірне розпізнавання в терміналі, а також малюємо на відео потоці дане розпізнавання об'єкт (обводимо об'єкт в кольоровий прямокутник і вішаємо мітку).

Тепер розберемо все по рядках:

Проходимо по **detections**, отримуємо значення валідності (**рядок 52**).

Якщо значення ймовірності вище заданого порогу (**рядок 55**), витягаємо індекс мітки в класі (**рядок 59**) і вираховуємо координати рамки навколо об'єкту який було виявлено (**рядок 60**).

Потім, витягаємо (x, y) - координати рамки (**рядок 61**), які в подальшому будемо використовувати для відображення прямокутника і тексту.

Робимо текстову мітку, що містить ім'я з CLASS і значення валідності розіпізнавання об'єкта (**рядки 63 і 64**).

Також, малюємо кольорової прямокутник навколо об'єкта, використовуючи кольори класу і раніше витягнуті (x; y) - координати (**рядки 65 і 66**).

В цілому, надалі потрібно, щоб мітка розташовувалась над кольоровим прямокутником, однак може виникнути і така ситуація, що зверху на розпізнаваємим об'єктом буде замало місця, тому в таких випадках виводимо мітку на верхній стороні прямокутника (**рядок 67**).

Нарешті, накладаємо кольоровий текст і рамку на кадр, використовуючи значення 'y', яке програма тільки що вирахувала (**рядки 68 і 69**).

Допоміжний функціонал (Рис.2.25):

1. Відображення кадру
2. Перевірка ключа виходу
3. Оновлення лічильника FPS

```
70  # показати вихідний кадр
71      cv2.imshow("Frame", frame)
72      key = cv2.waitKey(1) & 0xFF
73      # якщо натиснута клавіша `q`, цикл буде завершено
74      if key == ord("q"):
75          break
76      # оновлення лічильника FPS
77      fps.update()
```

Рис.2.25 – допоміжні функції

Короткий опис коду: по-перше, виводимо кадр (**рядок 71**). Далі фіксуємо натискання клавіші (**рядок 72**), перевіряючи, чи не натиснута клавіша "q" (quit). Якщо умова істинна то виходимо з циклу (**рядки 74 і 75**).

Та оновлюємо лічильник FPS (**рядок 77**).

При необхідності виходу з циклу (натискання клавіші "q" або завершення відео потоку), на дисплеї можна буде побачити деякі дані які належали даному відео потокую. (Рис.2.26)

```
78  # зупинка таймера і відображення інформації про FPS
79  fps.stop()
80  print("[INFO] elapsed time: {:.2f}".format(fps.elapsed()))
81  print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))
82  # мінімальне очищення
83  cv2.destroyAllWindows()
84  vs.stop()
```

Рис.2.26 – реалізація виходу з циклу

При виході з циклу, зупиняємо лічильник FPS (**рядок 79**) і виводимо інформацію про кінцевий значенні FPS в термінал (**рядки 80 і 81**).

Закриваємо вікно програми (**рядок 83**), зупиняючи програму та припиняючи відео потік (**рядок 84**).

Тепер після того як всі необхідні елементи коду було прописано можна спробувати запустити програму на Raspberry Pi для того щоб удостовіретись що все було зроблено правильно.

Частина 2: тестуємо розпізнавання об'єктів в реальному часі.

Щоб побачити детектор об'єктів в реальному часі в дії, переконайтеся, що на Raspberry було завантажено файли з вихідними кодами і попередньо підготовлений файл Convolutional Neural Network. Звідти відкриваєте термінал і прописуємо наступні команди (Рис.2.27):

$ python real_time_object_detection.py \

--prototxt MobileNetSSD_deploy.prototxt.txt \

--model MobileNetSSD_deploy.caffemodel

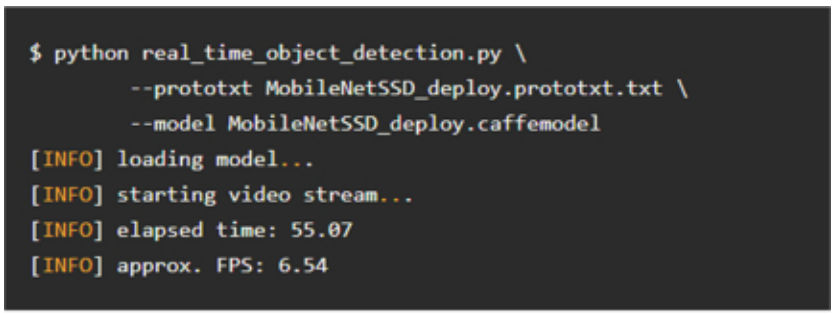

Рис.2.27 – команди для активації програми

За умови, що OpenCV може отримати доступ до вашої веб-камері, це умова при якій повинні побачити вихідний кадр з будь-якими виявленими об'єктами.

3 Спеціальна частина

3.1 Підключення до Raspberry Pi

Для роботи з Raspberry Pi є декілька методів підключення.

В даній роботі розглянемо два найбільш надійній з них способи підключення:

1. За допомогою виходу HDMI (Рис.3.1) який знаходиться на Raspberry Pi.

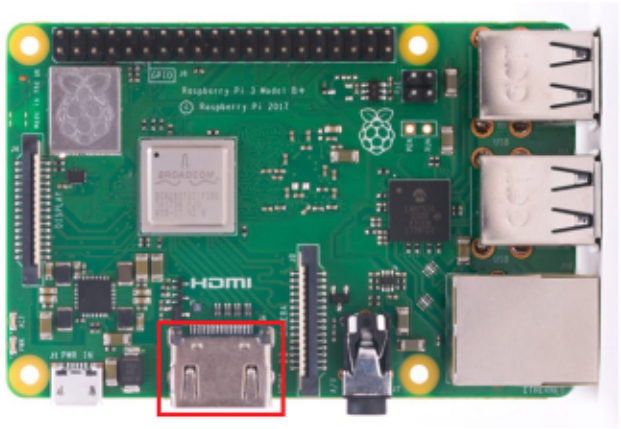

Рис.3.1 – розташування HDMI

Для роботи з клавіатурою і мишю використовуєм підключення через USB порти (Рис.3.2) які знаходяться на пережній частині Raspberry Pi.

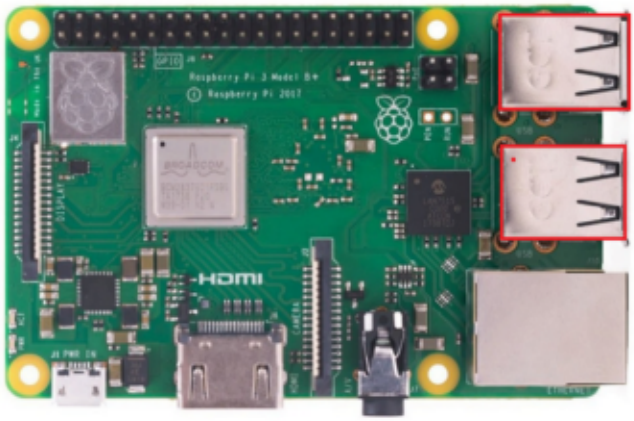

Рис.3.2 – розташування USB портів

Також на Raspberry Pi є роз’єм LAN за допомогою якого є можливість підключитись до інтернет мережі Ethernet (Рис.3.3).

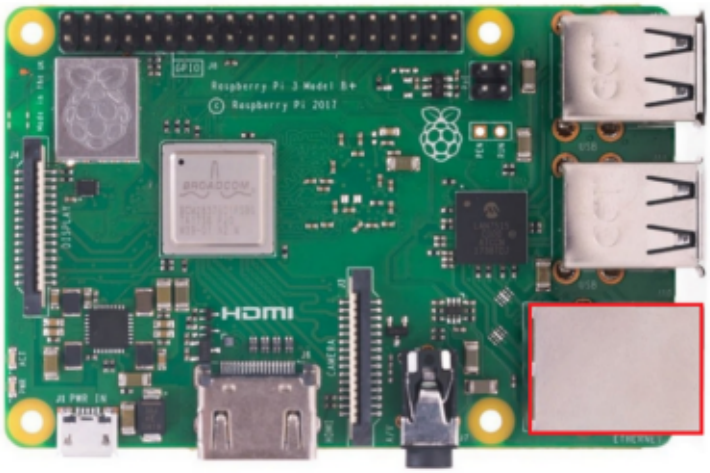

Рис.3.3 – роз’єм LAN

2. За допомогою спільної мережі WI-FI підключення.

Для того щоб підключитись до Raspberry Pi потрібно спочатку дізнатись IP – адресу який належить Raspberry до якої буде відбуватись підключення. Для цього потрібно ввійти в налаштування WI-FI – роутера і знайти IP – адресу до якого буде відбуватись підключення. Якщо перше налаштування Raspberry буде відбуватись через HDMI то використаємо команду sudo ifconfig (в даному випадку ім’я пристрою 192.168.0.120 (Рис.3.4)).

```
pi@raspberrypi:~ $ sudo ifconfig
eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether b8:27:eb:74:8b:52  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.120  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::8cb6:95d1:d7f0:2f2  prefixlen 64  scopeid 0x20<link>
        ether b8:27:eb:21:de:07  txqueuelen 1000  (Ethernet)
        RX packets 1831  bytes 315036 (307.6 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 249  bytes 44257 (43.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

pi@raspberrypi:~ $
```

Рис.3.4 – знаходження IP-адреси за допомогою коанди sudo ifconfig

Після того як знайшли IP-адресу Raspberry Pi на пристрій з якого буде відбуватись віддалене підключення потрібно встановити на свій персональний комп’ютер програмну для роботи з SSH підключенням яка називається “ ***PuTTY***” (Рис.3.5).

Рис.3.5 – логотип “ ***PuTTY***”.

Після встановлення “ ***PuTTY***” на комп’ютер відкриваєм програму. Далі коли програма запуститься активується інтерфейс даної програми (Рис.3.6).

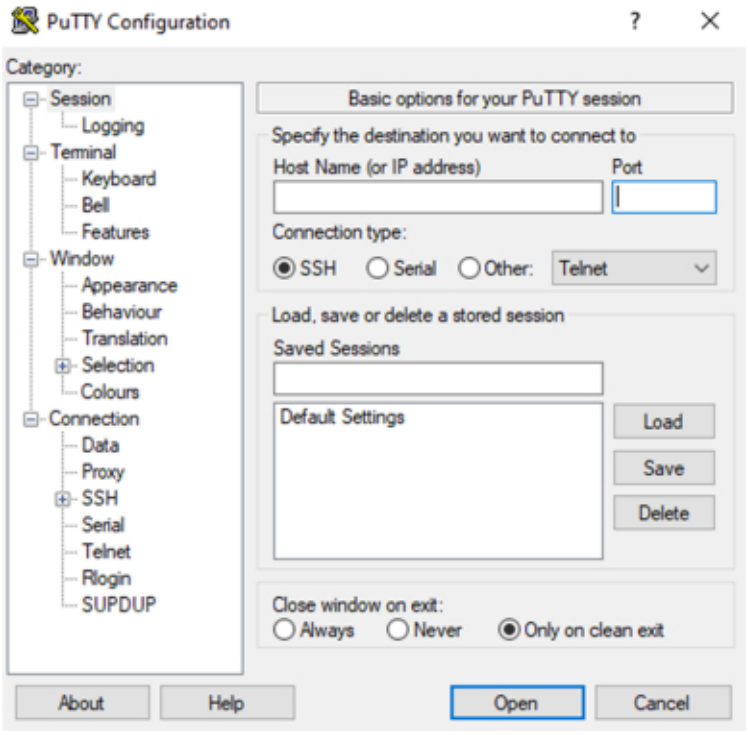

Рис.3.6 – інтерфейс “ ***PuTTY***”.

Після включення інтерфейсу програми в розділі Host Name (or IP address)

(Рис.3.7) прописуємо IP-адрес Raspberry Pi який було знайдено в налаштуваннях WI-FI-роутера.

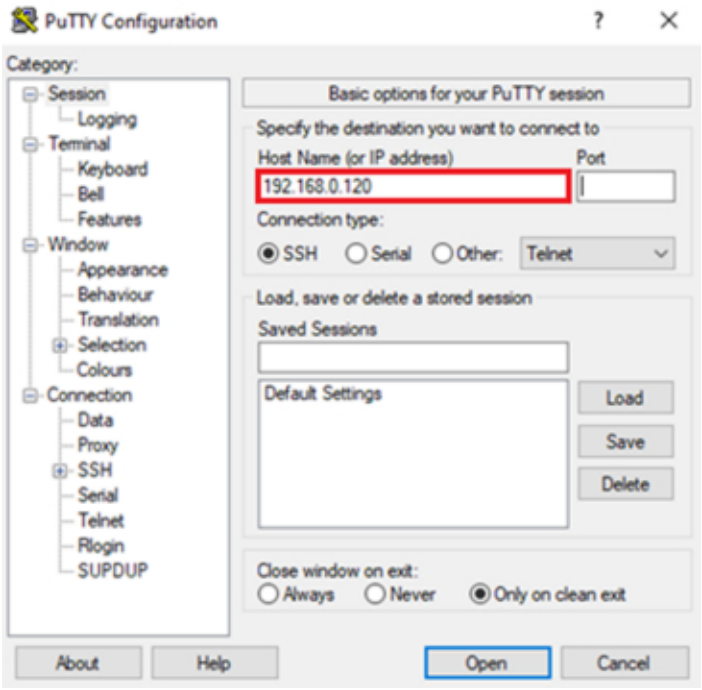

Рис.3.7 – розділі Host Name (or IP address)

Якщо в розділі Port (Рис.3.8) автоматично нічого не прописано то прописуємо вручну “ ***22***”.

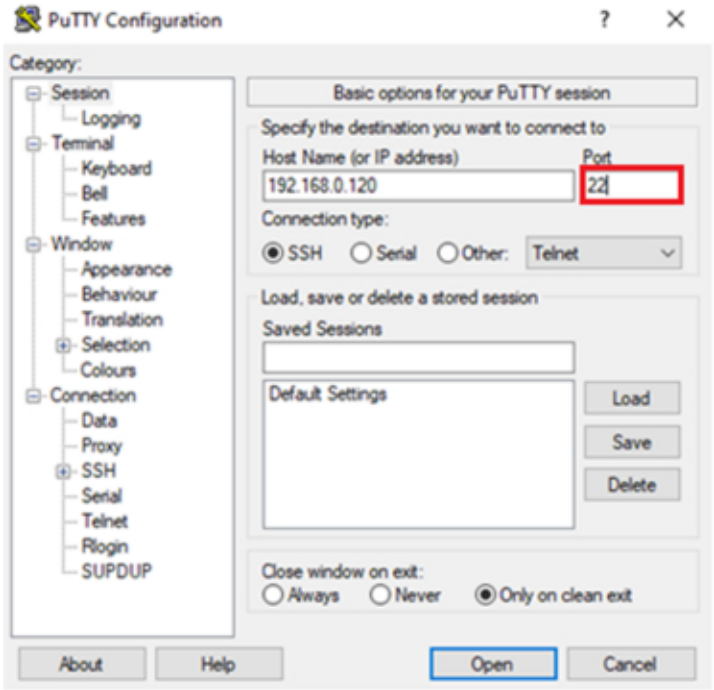

Рис.3.8 – розділі Port

Наступним кроком в пункті Connection type (Рис.3.9) потрібно обрати той тип підключення який потрібний нам для роботи. В нашому випадку нам потрібний тип підключення SSH.

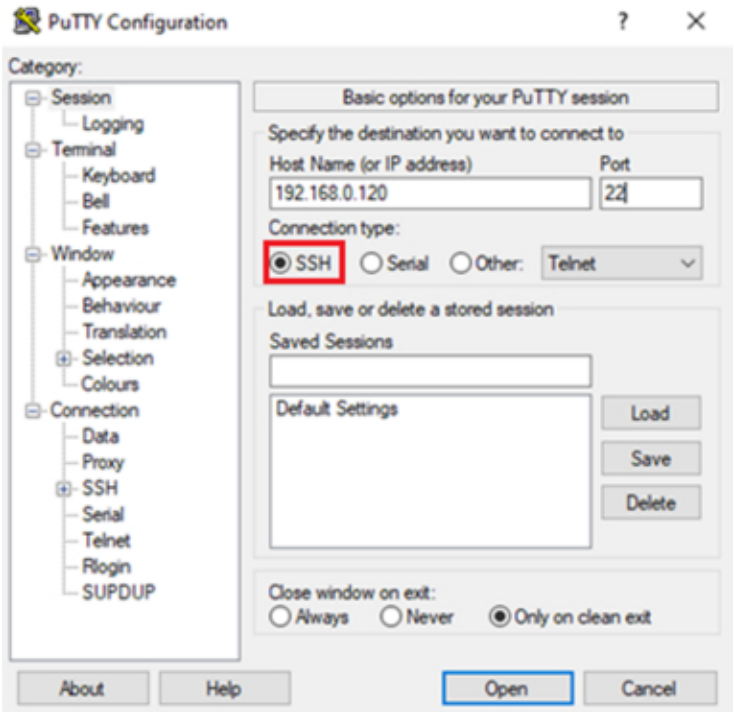

Рис.3.9 – пункті Connection type

Далі в пункті Saved Sessions (Рис.3.10) прописуємо назву підключення (назва підключення ні нащо не впливає вона використовується для зручного використання та упорядкування підключення які використовуватимуться користувачем в даній програмі). Після того як користувачем було надано назву підключення натискаємо Save для збереження підключення.

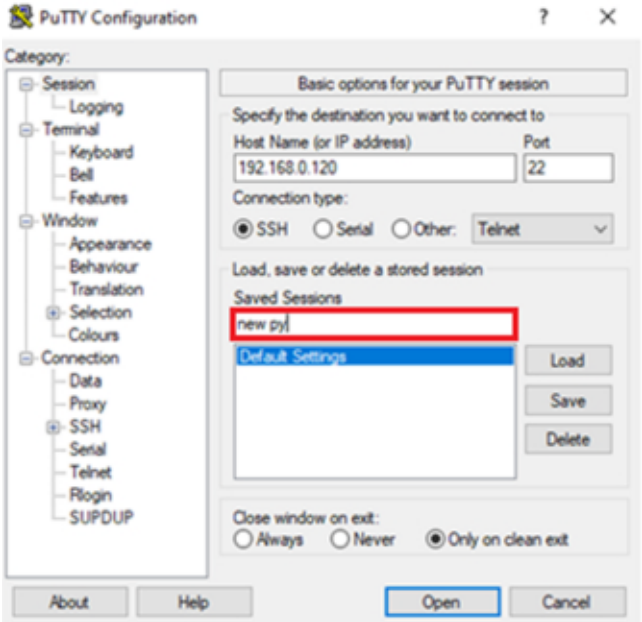

Рис.3.10 – пункті Saved Sessions

Після виконання всіх приготування для роботи з “ ***PuTTY***” натискаємо кнопку Open (Рис.3.11).

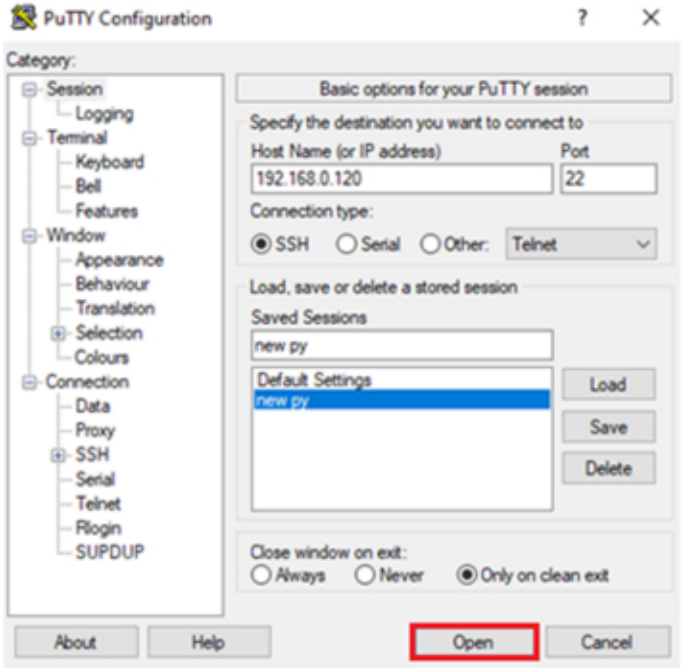

Рис.3.11– завершення налаштувань

При правильному введенні даних і правильно вибраних налаштувань повинно з’явитись командний термінал. В якому потрібно ввести логін pi і пароль 12345678 (Рис.3.12) при введенні паролю символи з яких складається пароль на робочій панелі термінала не відображатимуться.

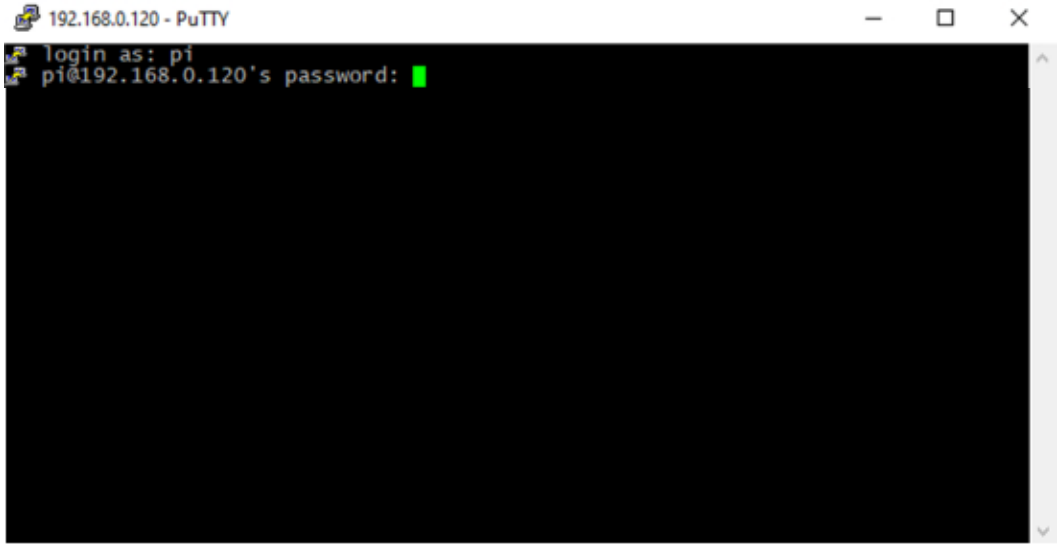

Рис.3.12 – ввід даних для авторизації

3.1.1 Увімкнення інтерфейсу

1. Введіть команду (Рис.3.13), щоб відкрити raspi-config (Рис.3.14).

```
sudo raspi-config
```

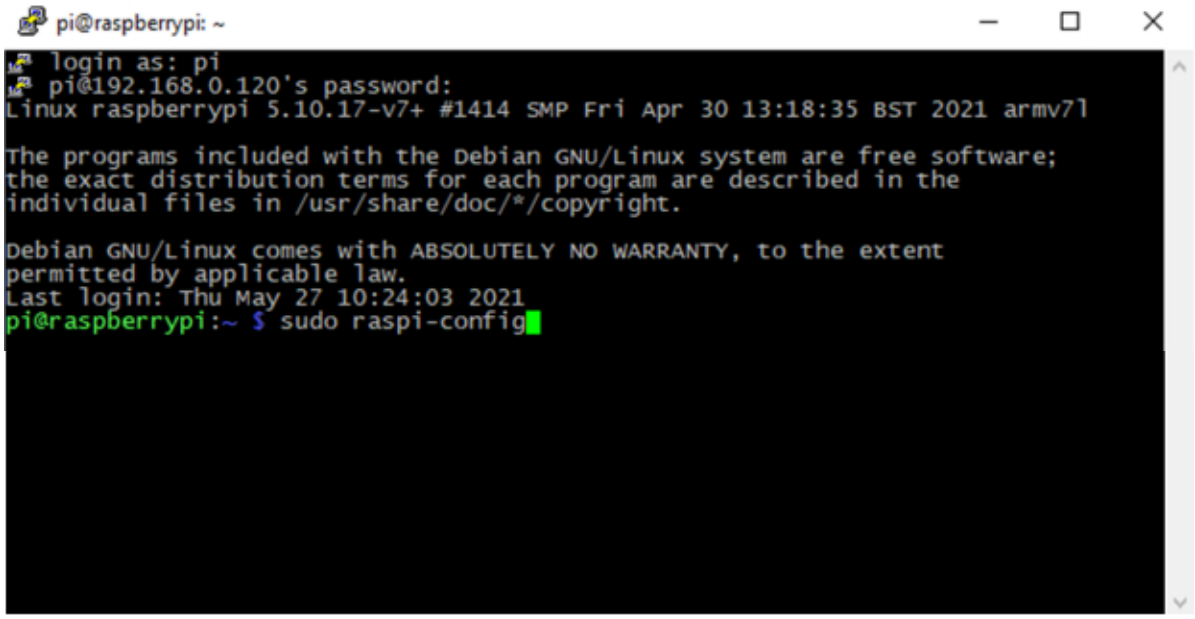

Рис.3.13 – введення команди

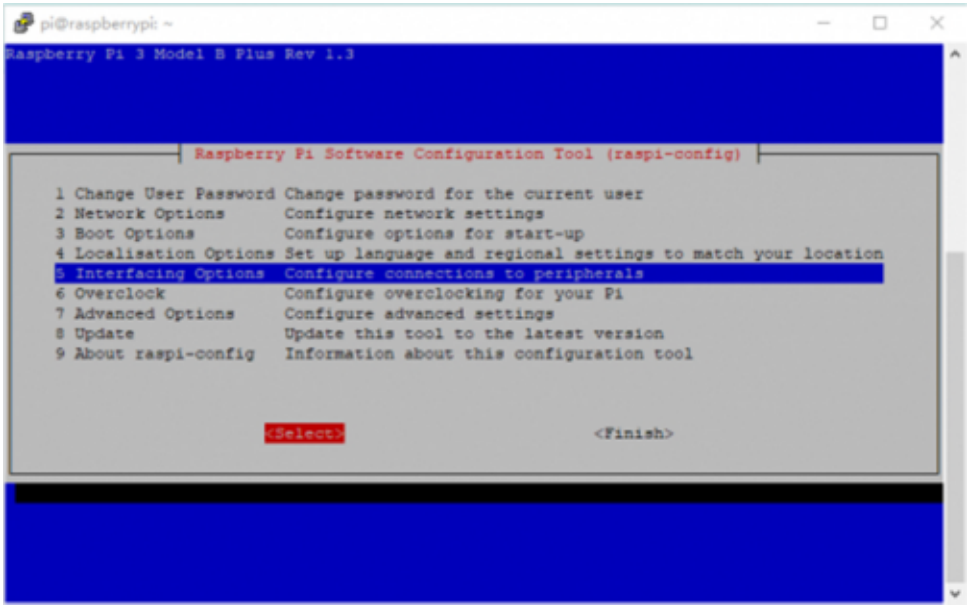

Рис.3.14 – налаштування raspi-config

2. Виберіть Параметри взаємодії-> Camera -> Yes -> Yes -> OK (Рис.3.15).

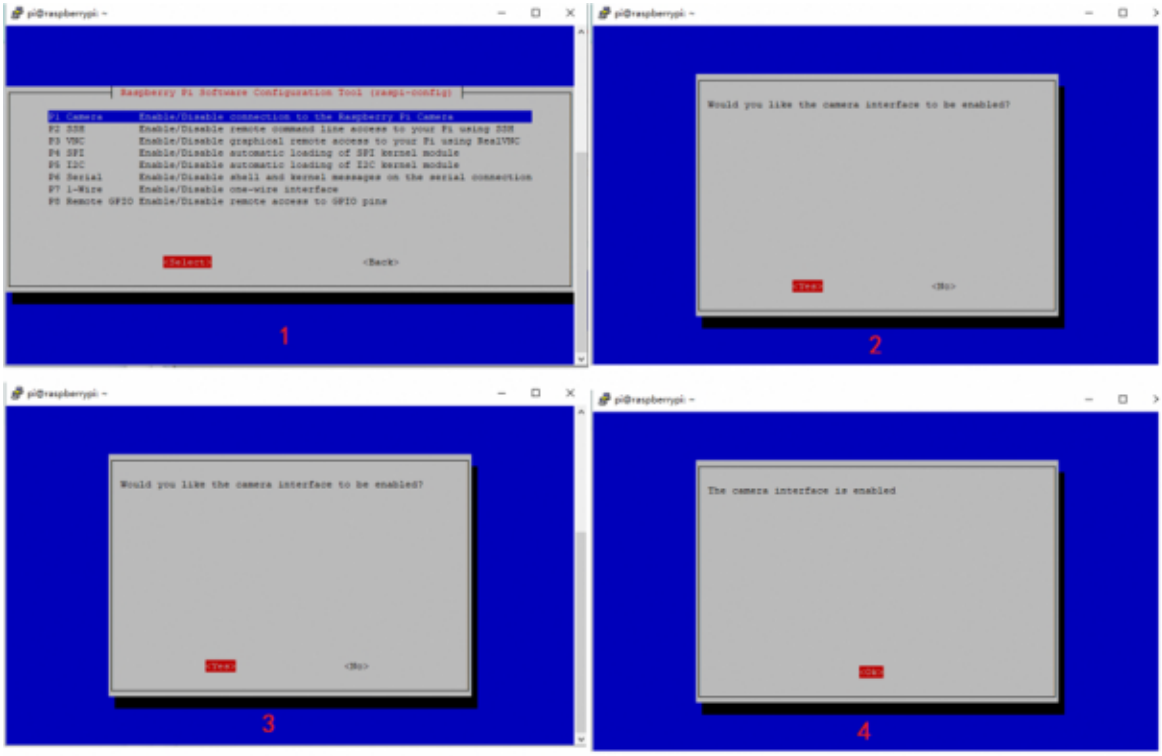

Рис.3.15 – вибір параметрів Camera

Виберіть Параметри взаємодії-> SPI -> Yes -> OK (Рис.3.16).

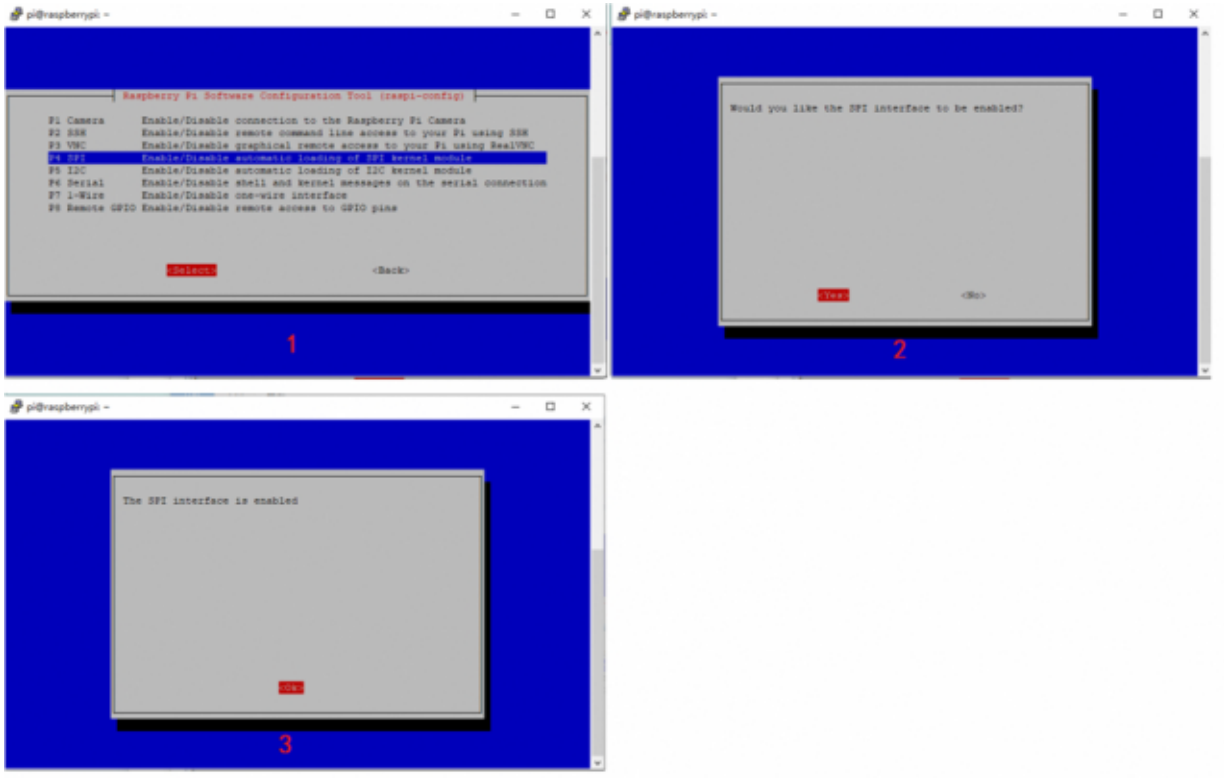

Рис.3.16 – вибір параметрів SPI

4. Виберіть Параметри взаємодії -> I2C -> Yes -> OK (Рис.3.17).

Рис.3.17– вибір параметрів I2C

5. Виберіть Параметри взаємодії-> UART -> No -> Yes -> OK (Рис.3.18)

Рис.3.18 – вибір параметрів UART

6. Потім вибераєм Finish, щоб вийти. Він може попросити вас перезавантажитись, будь ласка, виберіть Yes або ви можете ввести команду (Рис.3.19) для перезавантаження вручну.

```
sudo reboot
```

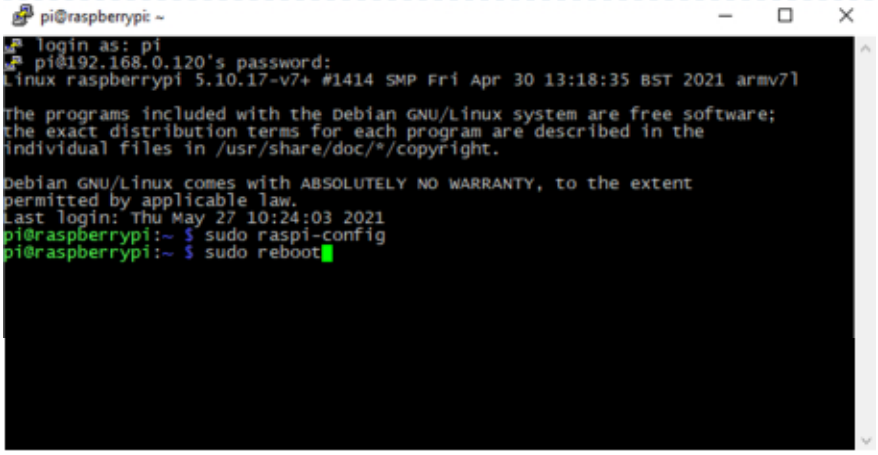

Рис.3.19 – команда для перезавантаження вручну

3.1.2 Програма WinSCP

Для початку встановлюємо WinSCP на свій персональний комп’ютер (рис.3.20)

Рис.3.20 – логотип WinSCP

Вигляд інтерфейсу WinSCP (Рис.3.21)

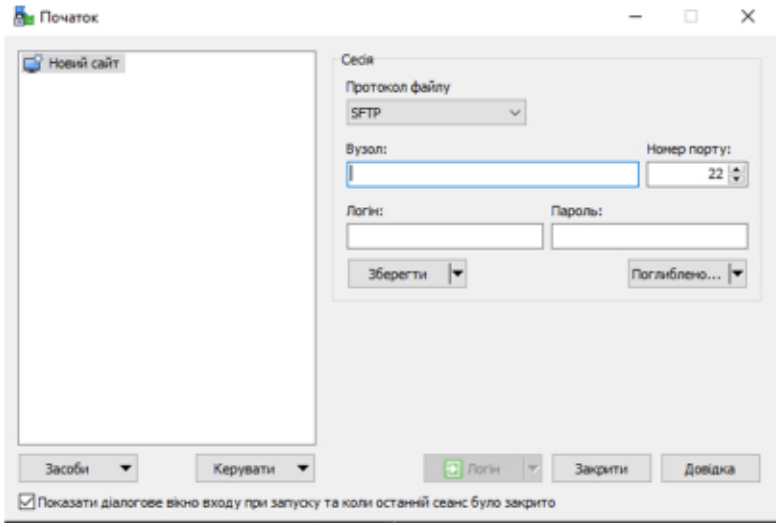

Рис.3.21 – інтерфейс WinSCP

В полі «Вузол» прописуємо IP-адресу даної

Raspberry Pi (Рис.3.21)

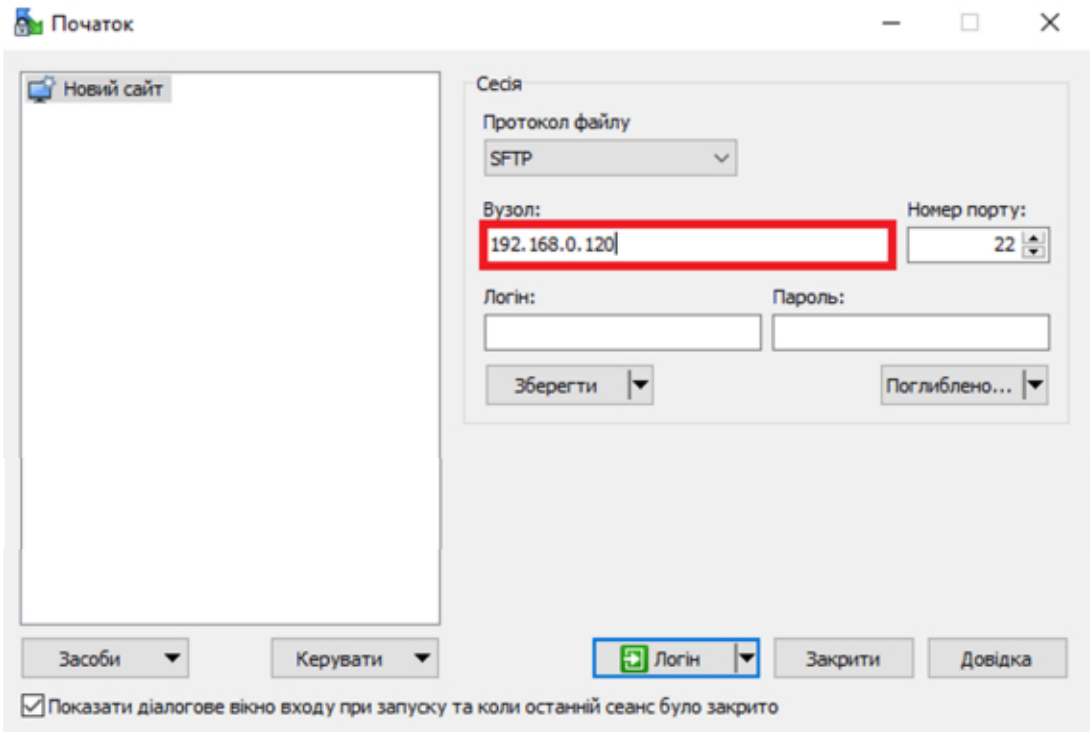

Рис.3.21 – розташування поля «Вузол»

Далі прописуємо логін і пароль для підключення до Raspberry Pi (Логін: pi, пароль: 12345678) і нажимаємо зберегти (Рис.3.22).

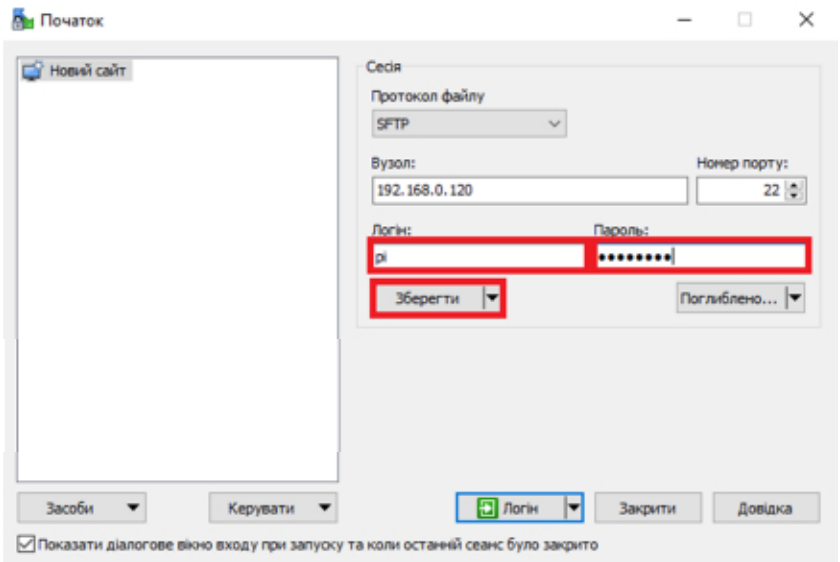

Рис.3.22 – поля авторизації

Зберігання даних для підключення потрібне для подальшого, зручнішого користування даною програмою. Звіряємо IP-адресу в полі «Зберегти сесію як:» з IP-адресою Raspberry якщо вони збігаються натискаємо «Ок» (Рис.3.23)

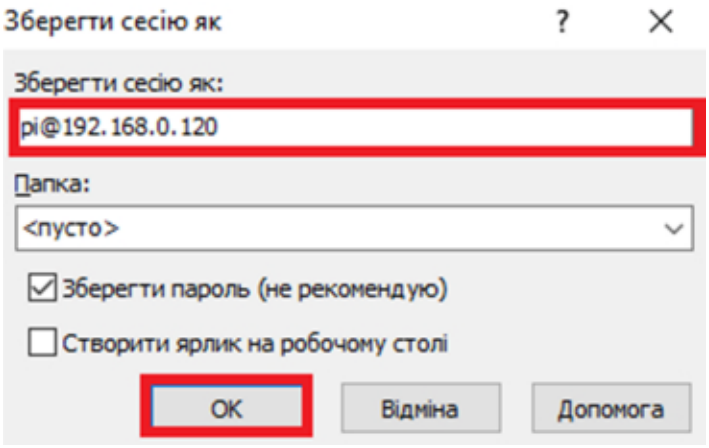

Рис.3.23 – вікно збереження данних

Тепер після збереження нашої сесії натискаємо на неї і натискаєм на кнопку «Логін» для активації з’єднання WinSCP з Raspberry Pi (Рис.3.24).

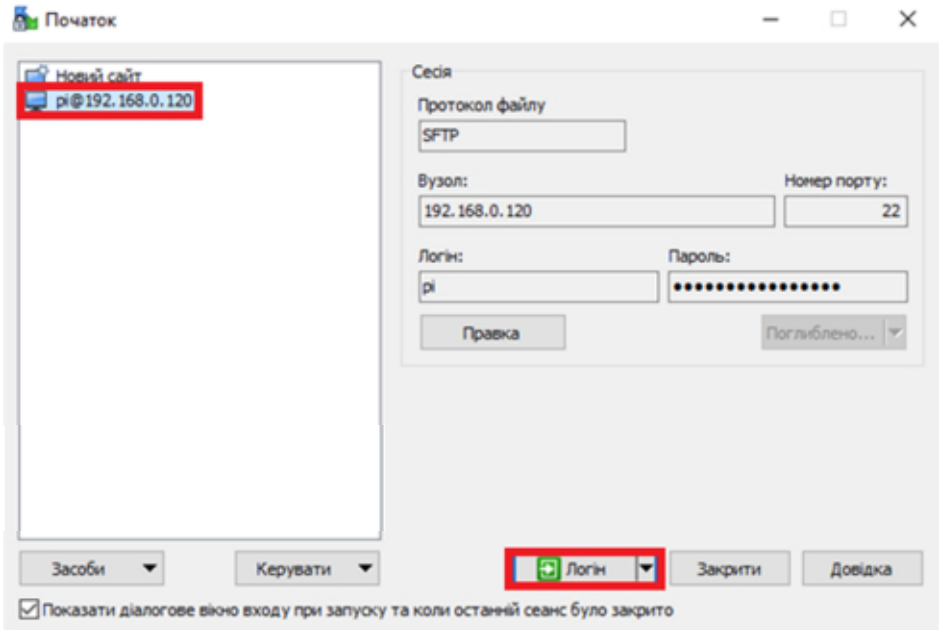

Рис.3.24 – збереженні підключення

Після з’явиться вікно в якому буде показана покрокова перевірка програмою виконання всіх необхідних умов для підключення через безпровідну мережу Wi-Fi (Рис.3.25)

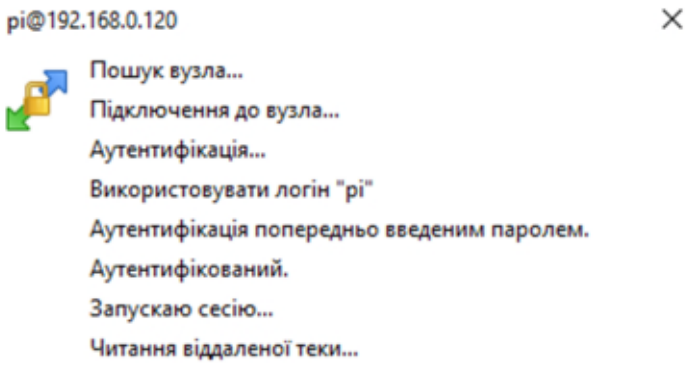

Рис.3.25 – виконання підключення

Якщо все було зроблено правильно і підключення пройшло вдало то повинен з’явитись даний інтерфейс який зображений на рисунку 3.26.

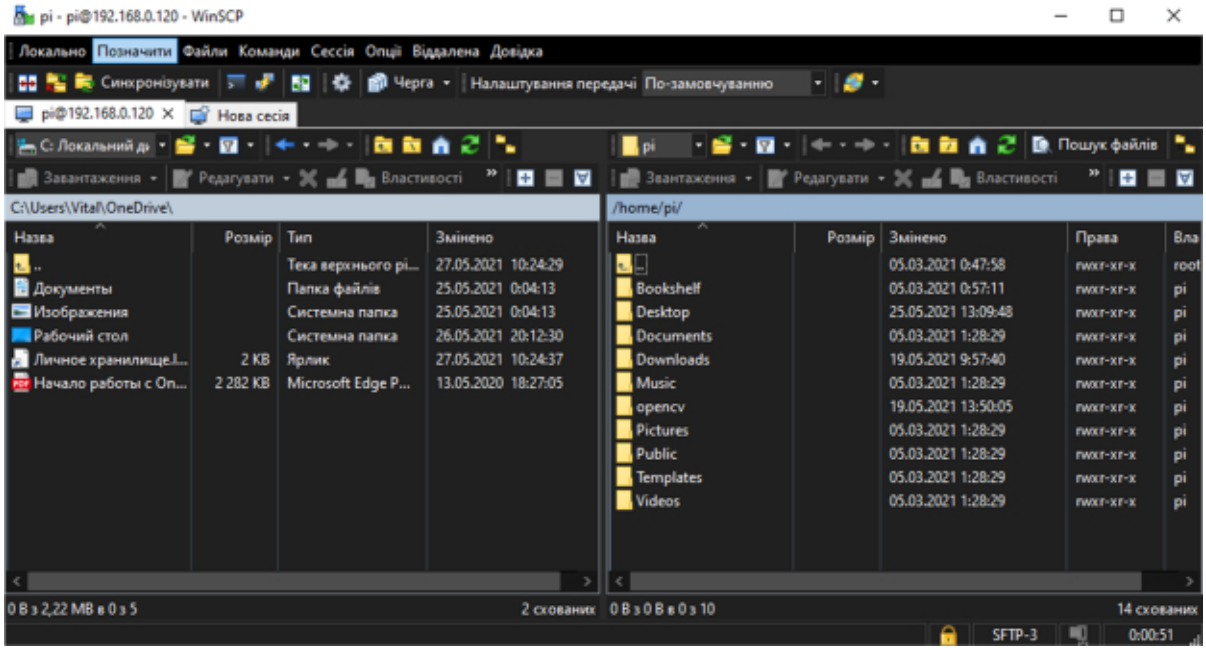

Рис.3.26 – робочий інтерфейс

3.1.3 Програма VNC Viewer

Підключення дистанційного дисплею для Raspberry Pi за допомогою спільною мережі Wi-Fi.

Підключення до Raspberry буде відбуватись за допомогою програми VNC Viewer (Рис.3.27).

Рис.3.27 – логотип VNC Viewer

Для того щоб почати роботу з VNC Viewer потрібно встановити на нашу Raspberry «tightvncserver» (Рис.3.28).

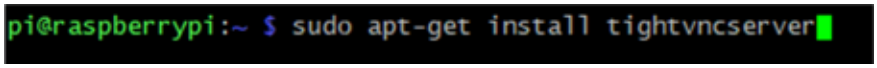

Рис.3.28 – введення команди

Це потрібно для того щоб на Raspberry Pi можна було створити сервер для подальшого підключення.

Далі за допомогою команди vncserver активуємо сервер для підключення до Raspberry, в кінці команди прописуємо «:1» (Рис.3.29). (При зміні безпровідної мережі Wi-Fi і створенні нового підключення при вводі команди потрібно буде обрати інший номер підключення тобто замість «:1» потрібно буде обрати любу іншу цифру на вибір для підключення так як після «:» вказується номер запам’ятовування програмою VNC Viewer IP-адреси Raspberry в певній мережі підключення. При не виконанні цієї умови підключення не відбудеться).

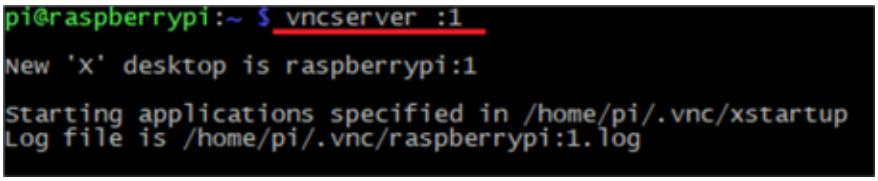

Рис.3.29 – команда створення сервера

Інтерфейс програми VNC Viewer. Вибираємо пункт File для підключення до сервера (Рис.3.30).

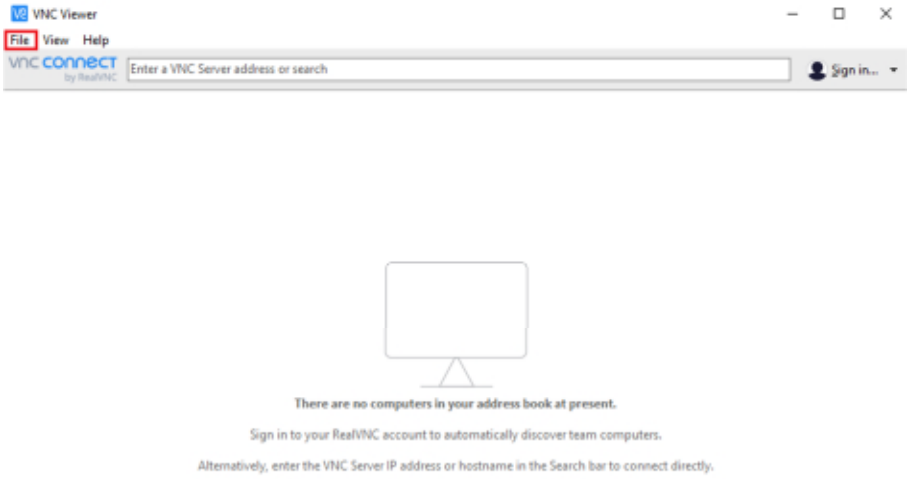

Рис.3.30 – інтерфейс VNC Viewer

В полі VNC Server прописуємо адресу сервера який було створено за допомогою команди vncserver в даному випадку це 192.168.0.120:1. Після цього натискаємо «Ок»(Рис.3.31).

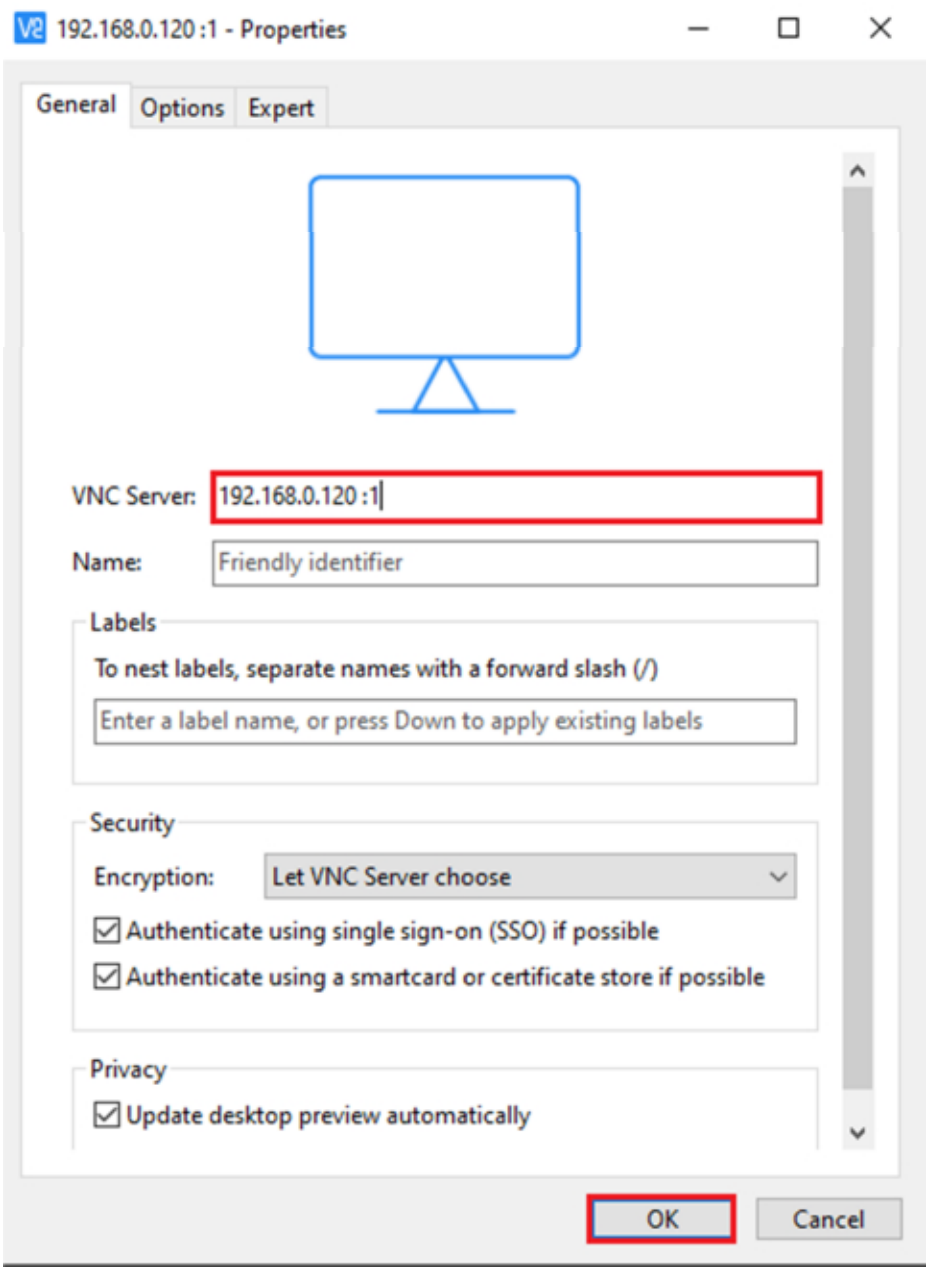

Рис.3.31 – введення даних для підключення

Всі інші налаштування залишаємо без змін.

Після створення підключення при збереженні на дисплеї програми зявиться іконка при натисканні на яку можна підключитись до потрібної нам Raspberry Pi (Рис.3.32).

Рис.3.32 – створене підключення

Підчас підключення програма з’являється вікно з повідомленням про інформацію що з’єднання з сервером не буде зашифроване і чи ми бажаємо продовжити підключення. Для продовження підключення в даному вікні натискаємо на кнопку «Continue» (Рис.3.33) і підключення продовжиться.

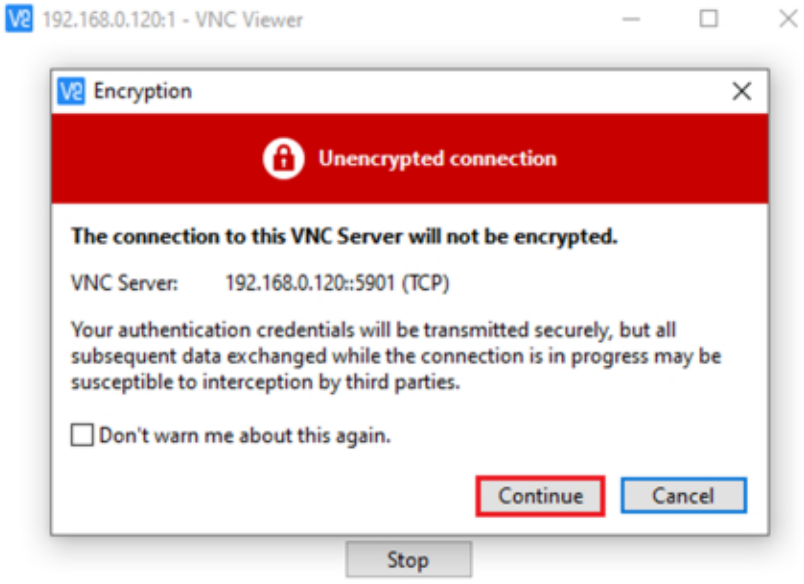

Рис.3.33 – запит на підключення

При продовженні підключення програма висвітить іще одне вікно де треба ввести пароль який в подальшому буде використовуватись для підключення VNC Server до Raspberry (Рис.3.34).

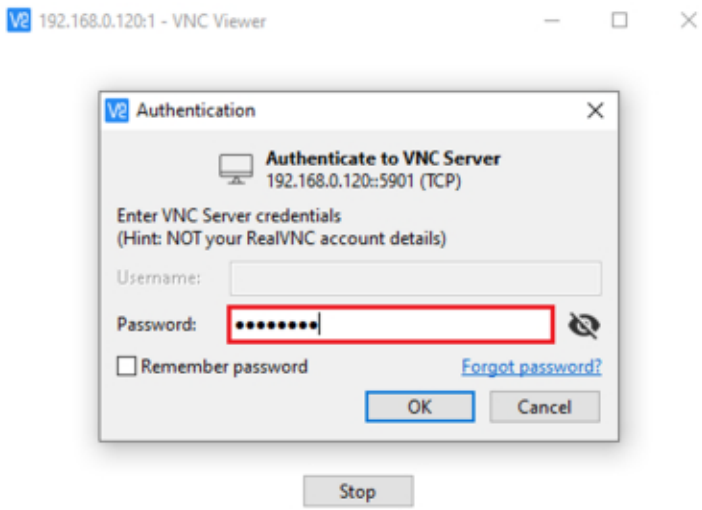

Рис.3.34 – вікно авторизації

При правильному налаштуванні і підключенні на інтерфейсі VNC Server можна буде побачити робочий стіл нашої Raspberry Pi (Рис.3.35).

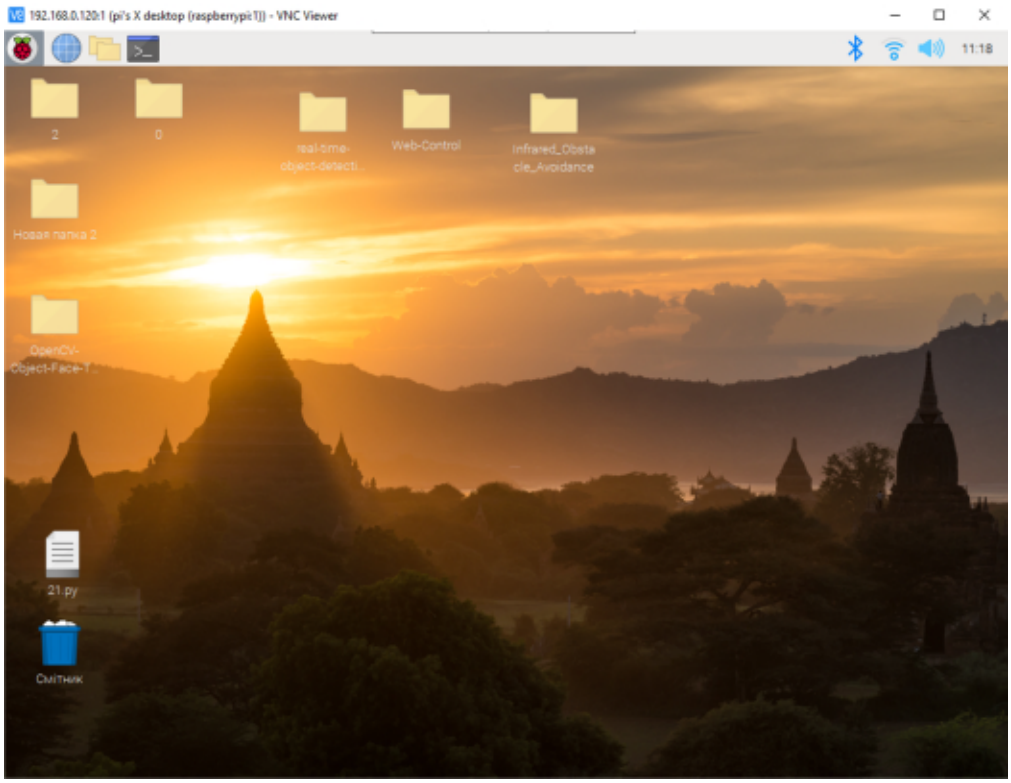

Рис.3.35 – інтерфейс Raspberry в VNC Server

3.2 Визначення похибки ультразвукового далекоміра на основі прямих багаторазових вимірювань відстані

При оцінці помилок при багаторазових вимірювань потрібно прийняти наступниі для виконання операції.

По-перше проводять вимірювання заданого фізичного параметра 20 разів за однакових умовах і результати записуються в таблицю.

Якщо результати деяких вимірювань різко відрізняються за своїм значенням від решти вимірів, то вони як промахи відкидаються, якщо після перевірки не підтверджуються.

Обчислюють середнє арифметичне $\bar{x}$ із 20 однакових вимірювань. Яке приймається за найбільш ймовірне значення вимірюваної величини;

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad (3.1)$$

Знаходять абсолютні похибки за допомогою окремих вимірювань

$$\Delta x_i = \left| x_i - \bar{x} \right|; \qquad (3.2)$$

Обчислюються квадрати абсолютних помилки окремих вимірювань $(\Delta x_i)^2$.

Визначається середня квадратична помилка середнього арифметичного за формулою:

$$S_{\bar{x}} = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n(n-1)}} \qquad (3.3)$$

За допомогою формули визначаємо, $S_{\bar{x}} = 0{,}0059$ м.

Задають значення надійної ймовірності Р. В лабораторіях практикуму прийнято ставити Р = 0,95.

Знаходять коефіцієнт Стюдента $t_{\alpha,n}$ для заданої довірюваної ймовірності α і кількості проведених вимірювань.

Згідно табличного значення коефіцієнт Стюдента для двадцяти вимірювань для кращої імовірності 0,95 буде рівний 2,086. $t_{\alpha,n}$ =2,086

Таблиця 3.1 – Оцінка похибки прямих багаторазових вимірювань

n	P				N	P			
	M_n	$\overline{M}$	ΔM_i	$(\Delta M_i)^2$		M_n	$\overline{M}$	ΔM_i	$(\Delta M_i)^2$
1	2,05	2,012	0,038	0,00144	11	2,00	2,012	-0,012	0,000144
2	1,98	2,012	-0,032	0,00102	12	2,02	2,012	0,008	0,000064
3	2,03	2,012	0,018	0,00032	13	1,98	2,012	-0,032	0,00102
4	2,05	2,012	0,038	0,00144	14	2,01	2,012	-0,002	0,000004
5	1,99	2,012	-0,022	0,000484	15	2,05	2,012	0,038	0,00144
6	2,00	2,012	-0,012	0,000144	16	1,97	2,012	-0,042	0,001764
7	2,02	2,012	0,008	0,000064	17	2,00	2,012	-0,012	0,000144
8	1,97	2,012	-0,042	0,001764	18	2,01	2,012	-0,002	0,000004
9	2,05	2,012	0,038	0,00144	19	2,03	2,012	0,018	0,00032
10	2,01	2,012	-0,002	0,000004	20	2,02	2,012	0,008	0,000064

Визначають випадкову похибку за формулою:

$$\Delta x = t_{\alpha,n} \cdot S_{\bar{x}}$$

(3.4)

$\Delta x = t_{\alpha,n} \cdot S_{\bar{x}}$ =2,086·0,0059=0,0123м

Оцінюють відносну похибку результату вимірювань за формулою:

$$\delta = \frac{\Delta x}{\bar{x}} \cdot 100\%$$

(3.5)

$$\delta = \frac{\Delta x}{\bar{x}} \cdot 100\% = \frac{0,0123}{2,012} \cdot 100\% = 0,61\%$$

Записують остаточний результат у вигляді:

δ = 0,61%.

4 Безпека життєдіяльності, основи охорони праці

4.1 Значення охорони праці і навколишнього середовища в забезпечені безпечних і здорових умов праці

Значення охорони праці полягає в сприянні росту ефективності суспільного виробництва шляхом безперервного вдосконалення і поліпшення умов праці, підвищення їх безпеки, зниження виробничого травматизму і профзахворювань.

Соціальне значення охорони праці проявляється в зростанні продуктивності праці, збереженні трудових ресурсів і збільшенні сукупного національного продукту.

Зростання продуктивності праці відбувається в результаті збільшення фонду робочого часу завдяки скороченню внутрішньо-змінних простоїв шляхом ліквідації мікротравм або зниження їх кількості, а також завдяки запобіганню передчасного стомлення шляхом раціоналізації і покращення умов праці та введенню оптимальних режимів праці і відпочинку та інших заходів, які сприяють підвищенню ефективності використання робочого часу. Небезпечні і шкідливі виробничі фактори виробничого середовища впливають на здоров'я і працездатність людини. Вони можуть бути причиною травм за певних умов.

Основним принципом політики в галузі ОП є визнання пріоритету життя і здоров’я працівників. Тому, на всіх підприємствах, в установах, організаціях створюються безпечні і нешкідливі умови праці.

Причини нещасних випадків поділяються на організаційні, технічні та санітарно-гігієнічні.

Організаційні причини:

– порушення режиму роботи і відпочинку;

– незадовільна організація, розташування і утримання робочих місць,

проходів та проїздів;

– використання невідповідного інструмента, обладнання, пристроїв;

– незадовільна якість або відсутність індивідуальних захисних засобів;

– неправильна організація праці, нераціональний режим роботи;

– тривале вимушене одноманітне або ненормальне положення тіла чи окремих його частин та їх перенапруження та інші.

Технічні причини:

– недосконаліть технологічних процесів;

– недосконаліть обладнання і пристроїв;

– незадовільний стан обладнання, інструмента і пристроїв.

Санітарно-гігієнічні причини:

– недостатність об'єму і площі виробничих приміщень;

– ненормальні метеорологічні умови (температура, вологість, швидкість руху і тиск повітря);

– освітлення не відповідає нормам;

– шкідливі випромінювання.

До витрат на охорону навколишнього природного середовища належать усі види витрат, спрямовані на запобігання, зменшення чи ліквідацію забруднення, інших видів шкідливого впливу господарської та іншої діяльності на навколишнє природне середовище, при наданні послуг чи використанні продукції.

Такі напрями витрат, як економія ресурсів та енергозбереження, ураховуються тільки в тому випадку, коли вони спрямовані передусім на захист охорони навколишнього природного середовища, наприклад утилізацію відходів, яка здійснюється з метою охорони навколишнього природного середовища.

4.2 Аналіз потенційно небезпечних та шкідливих виробничих факторів

Тривале сидяче положення приводить до напруги м'язів шиї, голови, рук і плечей, остеохондрозу. Перевантаження суглобів кистей рук приводить головним чином до такого явища, як синдром зап'ястного каналу.

До основних шкідливих факторів при роботі з комп'ютером відносять: тривале сидяче положення, електромагнітне випромінювання, навантаження на зір, перевантаження кистьових суглобів, можливість захворювань органів дихання, алергії, підвищена температура, відсутність або недостатність природного світла на робочому місці, електричний струм, статична електрика, розумове перенапруження, монотонність роботи.

Основним джерелом проблем, пов'язаних з охороною здоров'я людей, що використовують у своїй роботі ПК, є дисплеї з електронно-променевими трубками. Вони є джерелами найбільш шкідливих випромінювань, що несприятливо впливають на здоров'я людини. Існує два типи випромінювань, які виникають при роботі монітора: статичне і електромагнітне. Перше виникає при опроміненні екрану потоком заряджених частинок. Неприємності, викликані ним, пов'язані з пилом, що накопичується на електростатичних заряджених екранах, яка летить на людину під час його роботи за дисплеєм. Результати медичних досліджень показали, що така наелектризована пил може викликати запалення шкіри.

При роботі з ПК можуть виникнути потенційно небезпечні та шкідливі фактори, вплив яких на організм людини може принести йому шкоди і призвести до травматизму.

ПК встановлюються і розміщуються відповідно до вимог технічних умов заводів-виготовлювачів. Вплив шкідливих електромагнітних випромінювань зменшується за рахунок видалення їх джерел від оператора і установкою захисного екрана на монітор ПК. Вплив загазованості, запиленості і шкідливих парів, що виділяються ізоляцією установки

усувається за рахунок правильного розміщення обладнання, що забезпечує хорошу природну вентиляцію. Індекс ізоляції повітряного шуму між залом для глядачів і апаратної звукового забезпечення (при закритих оглядових вікнах) повинен бути не гірше 50 дБ. Стіни апаратної звукового забезпечення та стеля повинні оброблятися звукопоглинальними матеріалами з коефіцієнтом звукопоглинання не менше 0,6 в діапазоні частот 500 - 2000 Гц.

При тривалій роботі за екраном дисплея в операторів відзначається виражена напруга зорового апарату з появою скарг на незадоволеність роботою, порушення сну, головну біль, дратівливість і хворобливі відчуття в очах, у попереку, руках і ін.

Небезпечні фактори і їх допустимі значення наведено в таблиці 4.1.

Таблиця 4.1 – Аналіз потенційних небезпек виробничих факторів

Джерело небезпек	**Характеристика потенційно-небезпечних виробничих факторів та їх допустимі значення**
- електричний струм - електромагнітне поле (ЕМП)	Фактичні (середні) дані вимірів: напруга 220-230 В, струм 25 А, частота 50 Гц. Можливість ураження електричним струмом. Діюче значення напруженості ЕМП: Е=30 А/м в діапазоні частот 50 Гц- 100кГц ГДР: Ен=50 В/м в діапазоні частот 60 Гц- 3МГц Н=30 А/м в діапазоні частот 50 Гц- 100кГц ГДР: Нн=5 А/м в діапазоні частот 60 Гц- 3МГц
- ультрафіолетове випромінювання - рентгенівське випромінювання	Фактичні дані вимірів: 0,2Вт/м^2 Допустима інтенсивність: 0,01 Вт/м^2 Фактичні дані вимірів: 10 мкР/год. ГДД: 75мкр.год.

- електростатичне поле - ІЧ – випромінювання - видимий діапазон - яскравість	Фактичні дані: 13 кВ/м (0 Гц) Допустима напруженість поля 20-60 кВ/м. Фактичні дані вимірів: 0,06-7 Вт/м2 (в діапазоні 700 нм-1мм). Допустима інтенсивність: 100Вт/м^2 Фактичні дані: 9,5 Вт/м^2 (в діапазоні 4- 700 нм). Допустима інтенсивність: 10 Вт/м^2. Фактичні дані: 65кД/м^2 Допустиме значення: 35 кд/м^2

4.3 Забезпечення нормальних умов праці

Одним з основних завдань охорони праці є забезпечення таких умов праці, які б вилучали можливість дії на працівників небезпечних виробничих факторів.

Згідно зі статтею 153 Кодексу закону про працю, власник підприємства зобов'язаний забезпечити належне технічне об ладнання всіх робочих місць і створювати на них умови праці відповідно до нормативних актів з охорони праці.

Організація робочих місць проводиться згідно з ГОСТ 12.2.032-78 «Робоче місце при виконанні робіт сидячі. Загальні ергономічні вимоги».

Згідно ГОСТ 12.2.032-78 конструкція робочого місця і взаємне розташування всіх його елементів повинне відповідати антропометричним, фізичним і психологічним вимогам. Велике значення має також характер роботи. Зокрема, при організації робочого місця користувача ПК дотримано наступні основні умови:

– достатній робочий простір, що дозволяє здійснювати всі необхідні рухи і

переміщення;

– оптимальне розміщення устаткування;

– природне і штучне освітлення.

Для збереження працездатності і попередження розвитку захворювань опорно-рухового апарату користувачів ПК організовано для них робочі місця, що відповідають цим вимогам.

Приміщення являє собою кімнату площею 12м^2 та об’ємом 31,2м^3.

Кількість робочих місць у кімнаті – одне.

На одного працівника припадає 31,2м^3 об’єму приміщення та 12м^2 площі, що задовольняє вимогам “Державних санітарних правил та норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПІН 3.3.2.007-98”, п.2 “Вимоги до виробничих приміщень для експлуатації ВДТ ЕОМ та ПЕОМ”, згідно з якими площа на одне робоче місце має становити не менше ніж 6м^2, а об'єм не менше ніж 20м^3.

Роботи відносяться до легкого характеру роботи категорії Іа, що виконуються сидячи й не потребують фізичного напруження. При нормуванні умов для різних галузей промисловості виходять із загальних міжгалузевих норм ГОСТ 12.1.005-88 “Загальні санітарно-гігієнічні вимоги до повітря робочої зони”. На сьогодні основним нормативним документом, що визначає параметри мікроклімату виробничих приміщень є санітарні норми ДСН 3.3.6.042-99.

Для підтримання в приміщеннях нормальних параметрів повітряного середовища, яке відповідає санітарно-гігієнічним і технологічним вимогам, влаштовують вентиляцію.

Оптимальні параметри мікроклімату у робочій зоні виробничого приміщення в теплий та холодний періоди року наведені в таблиці 4.2.

Таблиця 4.2 – Оптимальні параметри мікроклімату

Назва приміщення	**Категорія важкості**	**Період Року**	**Темпера – тура оС**	**Відносна вологість**	**Швидкість руху повітря**

Приміщення з ЕОМ	Легка-1а	Холодний	20-22	40-50	0,1
	Легка-1а	Теплий	21-25	40-50	0,1

4.4 Розрахунок освітлення в приміщенні

4.4.1 Алгоритм розрахунку рівня освітлення

В основі алгоритму розрахунку рівня освітлення використовується метод люмену та формула визначення рівня КПО, блок-схема алгоритму наведена на рис. 10.

Після початку роботи алгоритму, з програмного модуля введення даних отримуються вхідні дані, які необхідні для проведення перевірки рівня освітлення в приміщенні, який задано згідно з умовами задачі. Перевірка рівня освітлення виконується за формулами (1.1) та (2.3) для штучного та природного відповідно.

За результатами перевірки виконується умова: якщо рівень освітлення відповідає нормам ДБН В.2.5-28:2018, то на екран про це виводиться повідомлення, якщо ні – виводиться повідомлення про низький рівень освітлення і запускається алгоритм проведення оптимізації розміщення джерел світла.

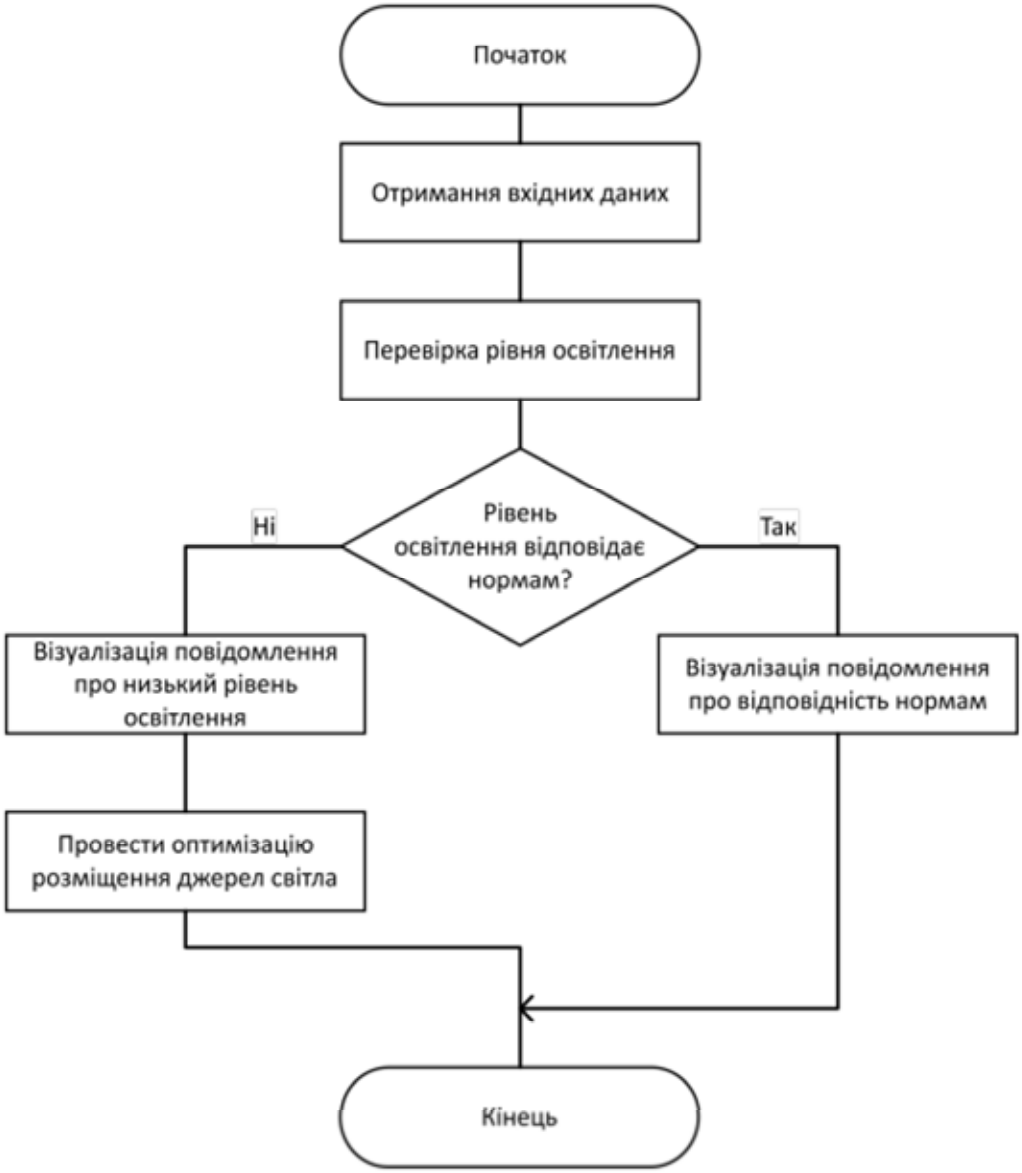

Рис. 10. Блок-схема алгоритму розрахунку рівня освітлення

4.4.2 Алгоритм оптимального розміщення джерел світла

Кроки даного алгоритму формалізовано подано на рис. 11.

Алгоритм оптимального розташування джерел світла запускається за умови, коли перевіркою встановлено, заданий рівень освітлення приміщення є низьким. Після старту роботи алгоритму виконується перевірка вхідних даних на наявність у приміщенні внутрішніх перегородок. Якщо в приміщенні перегородки відсутні, площа приміщення

розбивається на квадрати. Кожний виділений квадрат має площу поверхні, яка може бути освітлена джерелом світла вибраної потужності (в даному розміщення джерел світла випадку маються на увазі світильники).

Рис. 11. Блок-схема алгоритму оптимального

Тобто кожне джерело світла у моделі приміщення має освітлювати квадратну ділянку. Таким чином, приміщення буде рівномірно освітлено, а рівень освітлення відповідатиме нормам стандарту. Далі програма розташовує джерела світла по таких квадратах. Вибір потужності джерела світла виконується автоматично.

Якщо в приміщенні присутні перегородки, то приміщення ділиться

на частини та для кожної з цих частин виконуються операції з розбивки площі частини приміщення на квадрати з подальшим розташуванням джерел світла по квадратах.

4.5 Розрахунок звукового тиску

У виробничому приміщенні з рівнем звукового тиску від зовнішніх джерел L1 =72дБ планується зробити звукопоглинальне облицювання стелі та стін. Площа стін – 300 м^2, площа стелі – 250 м^2, площа підлоги – 250 м^2. Середній коефіцієнт звукопоглинання в приміщенні до облицювання дорівнює 0,1, коефіцієнт звукопоглинання використаного облицювання – 0,9. Необхідно визначити зниження шуму після використання облицювання у виробничому приміщенні та можливість улаштування в ньому конструкторського бюро.

Зниження рівня шуму в приміщенні як наслідок використання облицювання з більш високим коефіцієнтом звукопоглинання, дБ, можна визначити за формулою:

$$\Delta \text{Lобл} = 10\lg\frac{B2}{B1},$$

де В1 та В2 – сталі приміщення відповідно до та після облицювання.

У загальному випадку:

$$B = \frac{\text{A}}{1 - \alpha\text{ср}},$$

де А – еквівалентна площа звукопоглинання, $A = \alpha\text{ср} \cdot S\text{пов}$;

αср – середній коефіцієнт звукопоглинання внутрішніх поверхонь приміщення площею Sпов;

$$S\text{пов} = S\text{підл} + S\text{ст} + S\text{стелі},$$

де Sпідл – площа підлоги, м2;

Sст – площа стін, м2;

Sстелі – площа стелі, м2.

За таких означень стала приміщення до облицювання складає:

$$B1 = \frac{\text{αсер} \cdot S\text{пом}}{1 - \text{αсер1}}.$$

При визначенні сталої приміщення після облицювання треба звернути увагу на зміну середнього коефіцієнта звукопоглинання. До облицювання він складав для всіх внутрішніх поверхонь за умовами задачі αсер1 = 0,1. Після облицювання підлога залишилась з попереднім коефіцієнтом звукопоглинання (αсер = 0,1), а у стелі та стін він буде дорівнювати α2 = 0,9. Середній коефіцієнт звукопоглинання після облицювання можна визначити як середньозважену величину від площі внутрішніх поверхонь, що мають різні коефіцієнти звукопоглинання:

$$\alpha\text{сер} = \frac{(\text{Sст} + \text{Sстелі}) \cdot \alpha 2 + \text{Sпідл} \cdot \text{αсер1}}{\text{Sпідл} + \text{Sст} + \text{Sстелі}}.$$

Рівень шуму в приміщенні після облицювання, дБА, визначається за формулою:

$$L2 = L1 - DL\text{обл.}$$

Порівнюючи рівень шуму в приміщенні після облицювання з допустимим за ГОСТ 12.1.003-83*, який становить для конструкторського бюро 50дБА можна зробити висновок, що навіть після зазначеного облицювання в даному приміщенні влаштовувати конструкторське бюро недоцільно.

Висновки

В дипломному проекті проаналізовано сучасні системи автоматичного керування дистанційно-керованої роботизованої системи з можливістю обминання перешкод. Розглянуто структурну схему системи керування роботизованої системи. У роботі розглянуто принцип дії як всього пристрою, так і його окремих структурних блоків.

У дипломному проекті також розроблено схему електричну принципову системи дистанційно-керованої роботизованої системи з можливістю обминання перешкод, розроблено алгоритм роботи системи та програмне забезпечення для платформи Raspberry Pi для реалізації системи автоматичного керування дистанційно-керованої роботизованої системи з можливістю обминання перешкод та здійснення дистанційного керування його вихідними параметрами з Android пристрою по спільній мережі Wi – Fi.

З економічної точки зору пристрій не є фінансово затратним і являється непоганим стартапом через свою простоту та низьку затратність на купівлю всіх необхідних елементів для складання пристрою.

Перелік посилань

1) Пристрій давача руху [Електронний ресурс]. – Режим доступу: https://rutlib5.com/book/18999/p/6

2) Навчальний посібник автоматизація технологічних процесів і систем автоматичного керування (частина 1) [Електронний ресурс].

http://kyrator.com.ua/index.php?option=com_content&view=article&id=696:titulna1&catid=23&Itemid=130&limitstart=1

3) Принцип роботи мережевого протокола SSH. [Електронний ресурс]. https://freehost.com.ua/faq/wiki/chto-takoe-ssh/

4) Опис роботи технології Bluetooth. [Електронний ресурс].

https://wiki.cuspu.edu.ua/index.php/%D0%A2%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D1%96%D1%8F_Bluetooth

5) Принцип работи давачів руху [Електронний ресурс]. – Режим доступу: https://progress.online/tehnologii/694-princip-rabotydatchikovdvizheniya

6) Створення алгоритму для обминання перешкод [Електронний ресурс]. https://www.toptal.com/robotics/programming-a-robot-an-introductory-tutorial

7) Кашкаров, А.П. Давачі в електронних схемах: від простого до складного: навч. посіб. / А. П. Кашкаров — К. : ДМК, 2017. — 200 с.

8) Повний А. Как устроены инфракрасные датчики [Електронний ресурс]. – Режим доступу: http://electrik.info/main/automation/917-kak-ustroeny-irabotayut-infrakrasnye-datchiki-dvizheniya.html

9) Ультразвуковий давач руху для включення світла, принцип работи [Електронний ресурс].

Режим доступу: https://elektronchic.ru/avtomatika/ultrazvukovoj-datchik-dvizheniya.html

10) Побудова принципу роботи ультразвукових давачів. [Електронний ресурс]. https://opticstoday.com/katalog-statej/stati-na-ukrainskom/elementi-ta-pristroi-sistem-upravlinnya-avtomatiki/ultrazvukovi-davachi/ultrazvukovi-davachi-princip-dii-i-priznachennya.html

11) Путівник мовою програмування Python. [Електронний ресурс] https://pythonguide.rozh2sch.org.ua/