Міністерство освіти і науки України

## Харківський національний університет радіоелектроніки

|  |  |  |
| --- | --- | --- |
| Факультет | |  |
|  | | (повна назва) |
| Кафедра | електронних обчислювальних машин | |
|  | (повна назва) | |

### **АТЕСТАЦІЙНА РОБОТА (ПРОЕКТ)**

#### Пояснювальна записка

|  |
| --- |
|  |
| (освітньо-кваліфікаційний рівень) |
| ГЮІК.46XXXX.025 ПЗ |
| (позначення документа) |

|  |
| --- |
| Локальні комп’ютерні мережі |
| у системах інтелектуального управління |
|  |
| (тема) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Виконав: студент | | **V** | курсу, групи | КІз-12-3 |
| напряму підготовки (спеціальності) | | | |  |
| 6.050120 – Комп’ютерна інженерія | | | | |
|  | | | | |
| (шифр і назва напряму, спеціальності) | | | | |
| Лахно О.Г | | | | |
| (прізвище, ініціали) | | | | |
| Керівник: | Каргін А.О | | | |
|  | (прізвище, ініціали) | | | |
| Рецензент: |  | | | |
|  | (прізвище, ініціали) | | | |

##### Допускається до захисту

|  |  |  |  |
| --- | --- | --- | --- |
| Зав. кафедри ЕОМ |  |  | Міхаль О.П. |
|  | (підпис) |  | (прізвище, ініціали) |

2017 р.

## Харківський національний університет радіоелектроніки

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Факультет | Комп’ютерна інженерія та управління | | | | |
| Кафедра | електронних обчислювальних машин | | | | |
| Освітньо-кваліфікаційний рівень | | | | Бакалавр | |
| Напрям підготовки | | | 6.050102 – Комп’ютерна інженерія | | |
|  | | | | | (шифр і назва) |
| Спеціальність | |  | | | |
|  | | | | | (шифр і назва) |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ЗАТВЕРДЖУЮ: | | | | | |
| Зав. кафедри | | | |  | |
|  | | | | (підпис) | |
| “ |  | ” |  | | 2017 р. |

**ЗАВДАННЯ**

##### НА АТЕСТАЦІЙНУ РОБОТУ (ПРОЕКТ)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| студентові Лахно Олегу Геннадійовичу | | | | | | | | | | |
| (прізвище, ім’я, по батькові) | | | | | | | | | | |
| 1. Тема роботи (проекту) | Локальні комп’ютерні мережі у системах інтелектуального | | | | | | | | | |
| управління | | | | | | | | | | |
|  | | | | | | | | | | |
| затверджена наказом по університету від | | | “ | 31 | | ” | березня | 2017 р. | № | 72 Стз |
| 2. Термін подання студентом роботи (проекту) | | | | | 09 червня 2017 р. | | | | | |
| 3. Вихідні дані до роботи (проекту) | |  | | | | | | | | |
| The official website of Raspberry Pi: raspberrypi.org | | | | | | | | | | |
|  | | | | | | | | | | |
|  | | | | | | | | | | |
|  | | | | | | | | | | |
|  | | | | | | | | | | |
|  | | | | | | | | | | |
| 4. Зміст пояснювальної записки (перелік питань, що потрібно розробити) | | | | | | | | | | |
| Спроектувати і реалізувати бездротову систему управління мобільним роботом. | | | | | | | | | | |
| Обробити та продемонструвати інформацію, що отримується за допомогою | | | | | | | | | | |
| ультразвукового сповіщувача та камери. | | | | | | | | | | |
|  | | | | | | | | | | |
|  | | | | | | | | | | |
|  | | | | | | | | | | |
|  | | | | | | | | | | |

|  |
| --- |
| 5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслеників, плакатів) |
| Демонстраційні матеріали. Слайди – арк. ф. ф4 |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

6. Консультанти розділів роботи (проекту)

|  |  |  |  |
| --- | --- | --- | --- |
| Найменування розділу | Консультант  (посада, прізвище, ім’я, по батькові) | Позначка консультанта про виконання розділу | |
| (підпис) | (дата) |
| Основна частина | проф. Каргін А. О. |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

7. Дата видачі завдання 03 квітня 2017р.

КАЛЕНДАРНИЙ ПЛАН

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| №  п./п. | Назва етапів роботи (проекту) | | | | | Термін виконання етапів роботи (проекту) | | Примітка |
| 1 | Постановка задачі. | | | | | 02.04.2017 – 10.04.2017 | |  |
| 2 | Аналіз теоретичних відомостей. | | | | | 14.04.2017 – 01.05.2017 | |  |
| 3 | Проектування мобільное системи | | | | | 03.05.2017 – 07.05.2017 | |  |
| 4 | Збір мобільноі системи | | | | | 08.05.2017 – 11.05.2017 | |  |
| 5 | Розробка програми управління | | | | | 11.05.2017 – 26.05.2017 | |  |
| 6 | Тестування програми управління | | | | | 27.05.2017 | |  |
| 7 | Розробка інтерфейсу | | | | | 28.05.2017 – 29.05.2017 | |  |
| 8 | Розробка пояснювальної записки. | | | | | 30.05.2017 – 09.06.2017 | |  |
| Студент | |  | |  | | | | |
|  | | (підпис) | |  | | | | |
| Керівник роботи (проекту) | | |  | |  | | проф. Каргін А. О. | |
|  | | | (підпис) | |  | | (посада, прізвище, ініціали) | |

РЕФЕРАТ

Пояснювальна записка: 58с, 25 рисунків, 10 таблиць, 14 джерел, 2 додатка

Об’єкт дослідження – інтелектуальні автономні мобільні системи.

Мета роботи – розробка мобільної системи дослідження подій, що доповнює стаціонарні системи контролю та обробки інформації.

Результат роботи – розроблено систему дослідження подій у складі робота-розвідника на базі мікрокомп’ютера Raspberry Pi 2, шасі ZK-4WD, ультразвуковому сповіщувачу, бездротового USB адаптеру та камери. Розроблено скетч для бездротової передачі даних і руху роботу. Керування мобільним роботом і передача даних по бездротовій локальній мережі стандарту IEEE 802.11g. Розробка системи та візуалізації виконана повністю на мові програмування C, скомпонованою з бібліотекою wiringPi. Програма працює під управлінням операційних систем сімейства Unix.

Даний програмний комплекс можливо використовувати в автоматизованих охоронних системах, автоматизованих комплексах обслуговування небезпечних виробництв, побуті, автоматичних комплексах дослідження будівель, тунелів і т.д.

ДОСЛІДЖЕННЯ ПОДІЙ, ІНТЕЛЕКТУАЛЬНІ МАШИНИ, МОВА С, РОБОТ, RASPBERRY PI, IEEE 802.11g, БЕЗДРОТОВІ ЛОКАЛЬНІ МЕРЕЖІ

ABSTRACT

Explanatory note: 58 pages, 25 figures, 10 tables,14 sources,2 annexes.

The object of study - intelligent autonomous mobile systems.

The purpose of the work – development of analyzing mobile system for special events, which supplements stationary system of control and analysis.

Result of the work – developed analyzing mobile system as part of robot-scout based on microcomputer Raspberry Pi 2, chassis ZK-4WD, ultrasonic sensor, wireless adapter and cam. A sketch for wireless data transfer and robot movement. Robot control and data transfer in local network of IEEE 802.11g standart. System development and visualization fully completed in programming language C with wiringPi library. System operates under control of Unix operating systems.

This software package can be used in automated security systems, automated service complexes, hazardous industries, everyday life, automatic research complexes of buildings, tunnels, etc.

ANALYZING SITUATIONS, INTELLIGENT MACHINES, PROGRAMMING LANGUAGE C, ROBOT, RASPBERRY PI, IEEE 802.11g, WIRELESS LOCAL NETWORKS

ЗМІСТ

[ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ 8](#_Toc486107218)

[ВСТУП 9](#_Toc486107219)

[1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОГЛЯД ПІДХОДІВ ДО ВИРІШЕННЯ ЗАВДАННЯ 11](#_Toc486107220)

[1.1 Організація систем дослідження подій 11](#_Toc485260746)

[1.2 Обґрунтування і вибір мобільних установок «роботів» для аналізу подій. 11](#_Toc485260747)

[1.3 Огляд датчиків і систем контролю використовуваних в системах дослідження подій. 18](#_Toc485260748)

[1.4 Постановка задачі 23](#_Toc485260749)

[2 ДОСЛІДНИЦЬКА ЧАСТИНА.  ПРОЕКТУВАННЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ДОСЛІДЖЕННЯ ПОДІЙ НА ОСНОВІ МОБІЛЬНОГО РОБОТА 25](#_Toc486107225)

[2.1 Побудова мобільної платформи на базі мікрокомп’ютера Raspberry Pi 2. 25](#_Toc485260751)

[2.2 Детальний опис компонентів мобільної системи. 26](#_Toc485260752)

[2.2.1 Шасі ZK-4WD 26](#_Toc486107228)

[2.2.2 Мікрокомп'ютер Raspberry Pi 2 28](#_Toc486107229)

[2.2.3 Arduino MotorShield 29](#_Toc486107230)

[2.2.4 Ультразвуковий сповіщувач HC-SR04 31](#_Toc486107230)

[2.2.5 USB адаптер D-link DWA-110 33](#_Toc486107230)

[2.2.6 Raspberry Pi 5 MP Camera 34](#_Toc486107231)

[2.2.7 Power bank DCAE 36](#_Toc486107232)

[2.3 Вигляд зібраної мобільна система дослідження подій на базі мікрокомп’ютера Raspberry Pi 2 37](#_Toc485260760)

[3 ОПИС РЕАЛІЗАЦІЇ РІШЕННЯ 38](#_Toc486107234)

[3.1 Опис архітектури програми 38](#_Toc485260762)

[3.2 Опис даних 39](#_Toc485260763)

[3.3 Опис методів реалізації 40](#_Toc485260764)

[3.3.1 Реалізація графічного інтерфейсу 40](#_Toc486107238)

[3.3.2 Реалізація функції об’їзду перешкод. 42](#_Toc486107239)

[3.3.3 Реалізація функції вимірювання відстані до перешкоди. 43](#_Toc486107239)

[3.3.4 Реалізація обміну інформації мобільної системи з стаціонарною(ПК) 44](#_Toc486107240)

[3.4 Опис можливостей для еволюції додатку 44](#_Toc485260768)

[3.5 Опис вимог до ПЗ та АЗ 45](#_Toc485260769)

[3.6 Опис інструкцій користувачеві 45](#_Toc485260770)

[ВИСНОВКИ 46](#_Toc486107244)

[СПИСОК ПОСИЛАНЬ 47](#_Toc486107245)

[ДОДАТОК А 49](#_Toc486107246)

[ДОДАТОК Б 50](#_Toc486107248)

[Б.1 – Код файлу mrpiimain.c 50](#_Toc486107250)

[Б.2 – Код файлу MRPII.h 53](#_Toc486107251)

[Б.3 – Код файлу MRPII.c 54](#_Toc486107252)

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

IEEE - Institute of Electrical and Electronics Engineers

USB - Universal Serial Bus

RFB - Remote Framebuffer Protocol

GPIO - General-purpose input/output

GTK+ (Gimp Toolkit) - кроссплатформенна бібліотека елементів інтерфейсу

ДТЛ - датчик тепловий легкоплавкий

ВСТУП

Актуальність теми.Мобільні роботехнічні системи застосовуються сьогодні в самих різних галузях. Корпоративні замовники цікавляться багатофункціональними промисловими роботами, масовий покупець активно набуває інтелектуальні пилососи, служби безпеки та порятунку розраховують на автономні пристрої, здатні невтомно виконувати завдання стеження і пошуку. Однією з таких завдань є інтелектуальні системи дослідження подій.

Мета і завдання дослідження. Метою цієї роботи є розробка мобільної системи дослідження подій, що доповнює стаціонарні системи контролю та обробки інформації. Для досягнення мети сформульовані наступні завдання:

* проаналізувати функції, які необхідно додатково реалізувати у мобільній системі дослідження подій;
* створити мобільну систему на базі розвідника із модулів, пристроїв та датчиків;
* розробити скетч для бездротової передачі даних і руху роботу;
* розробити додаток для візуалізації і обробки інформації що отримується за допомогою ультразвукового сповіщувача та камери, за допомогою мови програмування C;
* організувати зручний інтерфейс взаємодії користувача з додатком;
* виконати тестування системи на тестових прикладах.

Об’єктом дослідження є інтелектуальні автономні системи.

Предметом дослідження є моніторинг події на підставі мобільних роботів-розвідників.

Методи дослідження*.* Управління автономними мобільними роботами. Обробка сповіщувальної інформації.

Особистий внесок студента.Студентом самостійно отримані наступні результати:

- обґрунтовані функції, які необхідні додатково реалізувати у мобільній системі дослідження подій;

- створена мобільна система на базі Raspberry Pi, пристроїв та датчиків;

- розроблено скетч для бездротової передачі даних і руху роботу;

- розроблено додаток для візуалізації і обробки інформації, що отримується за допомогою ультразвукового сповіщувача та камери за допомогою мови програмування C ;

- виконано тестування системи на тестових прикладах.

У першому розділівиконано огляд і аналіз підходів до вирішення задачі дослідження події. За результатами огляду та аналізу предметної області, зроблені висновки: по вибору мобільної платформи, датчиків і систем контролю.

У другому розділіописуються пропоновані моделі вирішення завдання – концептуальна, логічна та фізична.

У третьому розділі описана реалізація рішення поставленого завдання, яка включає опис архітектури додатку, опис даних, методів реалізації, опис можливостей для еволюції додатку. Також, наводяться сценарії роботи розробленого додатку, розширена інструкція для користувача та рекомендовані і мінімальні вимоги до ПЗ та АЗ.

У висновках роботи представлені теоретичні та практичні результати роботи.

У додатках роботинаводяться UML діаграми.

1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОГЛЯД ПІДХОДІВ ДО ВИРІШЕННЯ ЗАВДАННЯ

1.1 Організація систем дослідження подій

Система дослідження подій призначена для прибуття на місце події та її дослідження.

Система дослідження подій включає в себе наступні основні елементи:

* ультразвуковий сповіщувач, для омину можливих перешкод;
* відео камеру яка транслює подію у мережу;
* приймально-контрольну станцію, яка приймає данні з камери та сповіщувальну, і передає їх на стаціонарну систему контролю та обробки інформації;

1.2 Обґрунтування і вибір мобільних установок «роботів» для аналізу подій.

Із літературних джерел відомі наступні класи роботів широкого призначення:

Маніпуляційний робот - автоматична машина (стаціонарна або пересувна), що складається з виконавчого пристрою у вигляді маніпулятора, що має кілька ступенів рухливості, і пристрою програмного управління, який служить для виконання у виробничому процесі рухових і керуючих функцій. Такі роботи виготовляються в підлоговому, підвісному і портальному виконанні. Набули найбільшого поширення в машинобудівних і приладобудівних галузях [1];

Мобільний робот - автоматична машина, в якій є рухоме шасі з автоматично керованими приводами. Такі роботи можуть бути колісними, крокуючими і гусеничними (існують також плазуючі, плаваючі і літаючі мобільні робототехнічні системи) [1].

Компоненти роботів:

* приводи - це «м'язи» роботів. В даний час найпопулярнішими двигунами в приводах є електричні, але застосовуються і інші, що використовують хімічні речовини або стиснене повітря.

Двигуни постійного струму (рисунок 1.1):

- у даний момент більшість роботів використовують електродвигуни, які можуть бути декількох видів.



Рисунок 1.1 Двигун постійного струму.

Крокові електродвигуни (рисунок 1.2):

- крокові електродвигуни не обертаються вільно, подібно двигунів постійного струму. Вони повертаються покроково на певний кут під керуванням контролера. Це дозволяє обійтися без датчика положення, так як контролеру точно відомо, на скільки був зроблений поворот.



Рисунок 1.2 Крокові електродвигуни

Сучасної альтернативою двигунів постійного струму є п'єзодвигуни, також відомі як ультразвукові двигуни. Принцип їх роботи абсолютно відрізняється: малюсінькі п'єзоелектричні ніжки, вібруючі з частотою більше 1000 разів на секунду, змушують мотор рухатися по колу або прямій. Перевагами подібних двигунів є висока швидкість і потужність, непорівнянна з їх розмірами. П'єзодвигуни вже доступні на комерційній основі і також застосовуються на деяких роботах.

Повітряні м'язи (рисунок 1.3):

- повітряні м'язи простий, але потужний пристрій для забезпечення сили тяги. При накачуванні стисненим повітрям, м'язи здатні скорочуватися до 40% від своєї довжини. Причиною такої поведінки є плетіння, видиме з зовнішньої сторони, яке змушує м'язи бути або довгими і тонкими, або короткими і товстими. Так як спосіб їх роботи схожий з біологічними м'язами, їх можна використовувати для виробництва роботів з м'язами і скелетом, аналогічними м'язам і скелету тварин [2][3].

Електроактивні полімери – це вид пластмас, який змінює форму у відповідь на електричну стимуляцію. Вони можуть бути сконструйовані таким чином, що можуть гнутися, розтягуватися або скорочуватися. Однак, в наш час немає електроактивних полімерів, придатних для виробництва комерційних роботів, так як всі неефективні або неміцні.



Рисунок 1.3 Повітряні м’язи

[Еластичні нанотрубки](https://uk.wikipedia.org/w/index.php?title=%D0%95%D0%BB%D0%B0%D1%81%D1%82%D0%B8%D1%87%D0%BD%D1%96_%D0%BD%D0%B0%D0%BD%D0%BE%D1%82%D1%80%D1%83%D0%B1%D0%BA%D0%B8&action=edit&redlink=1): це багатообіцяюча експериментальна технологія, що знаходиться на ранній стадії розробки. Відсутність дефектів у нанотрубках дозволяє цьому волокну еластично деформуватися на кілька відсотків. Людський біцепс може бути замінений проводом з такого матеріалу діаметром 8 мм. Такі компактні «м'язи» можуть допомогти роботам в майбутньому обганяти і перестрибувати людей.

У мобільному роботі ZK-4WD використовуваному в даному проекті стоять 4 мотора редуктора.

Найбільш поширеними роботами даного класу є [4][5] чотириколісні і гусеничні роботи. Створюються також роботи, які мають інше число коліс - два або одне. Такого роду рішення дозволяють спростити конструкцію робота, а також надати роботу можливість працювати в просторах, де чотирьохколісна конструкція виявляється непрацездатна.

Двоколісні роботи, як правило, для визначення кута нахилу корпусу роботу використовують свої приводи з поданням відповідної керуючої напруги (з метою забезпечити утримання рівноваги і виконання необхідних переміщень) та ті чи інші гіроскопічні пристрої. Завдання утримання рівноваги двоколісного робота пов'язана з динамікою зворотного маятника. На даний момент, розроблено безліч подібних «балансуючих» пристроїв.

Одноколісні роботи багато в чому являють розвиток ідей, пов'язаних з двоколісними роботами. Для переміщення в 2D просторі в якості єдиного колеса може використовуватися куля, що приводиться в обертання декількома приводами. Кілька розробок подібних роботів вже існують. Роботи такого типу мають деякі переваги, пов'язані з їх витягнутою формою, які можуть дозволити їм краще інтегруватися в людське оточення, ніж це можливо для роботів деяких інших типів.

Існує кілька прототипів сферичних роботів. Деякі з них для організації переміщення використовують обертання внутрішньої маси.

Для переміщення по нерівних поверхнях, траві і кам'янистій місцевості розробляються шестиколісні роботи, які мають більше зчеплення, в порівнянні з чотириколісним. Ще більше зчеплення забезпечують гусениці. Багато сучасних бойових роботів, а також роботи, призначені для переміщення по грубим поверхням розробляються як гусеничні. Разом з тим, утруднене використання подібних роботів у приміщеннях, на гладких покриттях і килимах.

Крокуючі роботи: переміщення робота з використанням «ніг» являє собою складну задачу динаміки. Вже створено кілька роботів, що переміщаються на двох ногах, але ці роботи поки не можуть досягти такого стійкого руху, яке притаманне людині. Також створено безліч механізмів, які прямують на більш ніж двох кінцівках. Увага до подібних конструкцій обумовлено тим, що вони легше в проектуванні. Пропонуються також гібридні варіанти, здатні переміщатися на двох кінцівках під час ходьби і на чотирьох кінцівках під час бігу.

Роботи, що використовують дві ноги, як правило, добре переміщаються по підлозі, а деякі конструкції можуть переміщатися по сходах. Переміщення по пересіченій місцевості є складним завданням для роботів такого типу.

Літаючі роботи. Більшість сучасних літаків є літаючими роботами, керованими пілотами. Автопілот здатний контролювати політ на всіх стадіях - включаючи зліт і посадку. До літаючих роботів відносяться також безпілотні літальні апарати (важливий їх підклас становлять крилаті ракети). Подібні апарати мають, як правило, невелику вагу (за рахунок відсутності пілота) і можуть виконувати небезпечні місії. Крім методу руху, використовуваного літаками, літаючими роботами використовуються і інші методи руху - наприклад, подібні до тих, що використовують пінгвіни, скати, медузи.

Плазуючи роботи. Існує ряд розробок роботів, що переміщаються подібно зміям, хробакам, слимакам; при цьому для реалізації руху робот може використовувати сили тертя (при русі по опорній поверхні)[6][7] або зміни кривизни поверхні (в разі гладкої поверхні змінної кривизни). Передбачається, що подібний спосіб переміщення може надати їм можливість переміщатися у вузьких просторах; зокрема, передбачається використовувати подібних роботів для пошуку людей під уламками будівель. Розроблено також змієподібні роботи, здатні переміщатися в воді.

Роботи, що переміщаються по вертикальних поверхнях. При їх проектуванні використовують різні підходи. Перший підхід - проектування роботів, які переміщуються подібно до людини, яка підіймається на стіну, вкриту виступами. Інший підхід - проектування роботів, що переміщаються подібно геконам із забезпеченими вакуумними присосками [8].

Плаваючі роботи. Існує багато розробок роботів, які переміщуються у воді, наслідуючи рухи риб. За деякими підрахунками, ефективність подібного руху може на 80% перевершувати ефективність руху з використанням гребного гвинта. Крім того, подібні конструкції виробляють менше шуму, а також відрізняються підвищеною маневреністю. Це є причиною високого інтересу дослідників до роботів, які плавають подібно рибам.

Під управлінням робота розуміється рішення комплексу завдань, пов'язаних з адаптацією робота до кола розв'язуваних їм завдань, програмуванням рухів, синтезом системи управління і її програмного забезпечення [9].

За типом управління робототехнічні системи підрозділяються на:

1. Біотехнічні:

- командні (кнопкове і важельне управління окремими ланками робота);

- копіювальні (повтор руху людини, можлива реалізація зворотного зв'язку, що передає додаткове зусилля, екзоскелети);

- напівавтоматичні (управління одним командним органом, наприклад, рукояткою всій кінематичною схемою робота);

1. Автоматичні:

- програмні (функціонують за заздалегідь заданою програмою, в основному призначені для вирішення одноманітних завдань в незмінних умовах оточення);

- адаптивні (вирішують типові завдання, але адаптуються під умови функціонування);

- інтелектуальні (найбільш розвинені автоматичні системи);

1. Інтерактивні:

- автоматизовані (можливо чергування автоматичних і біотехнічних режимів);

- супервізорні (автоматичні системи, в яких людина виконує тільки вказівні функції);

- діалогові (робот бере участь в діалозі з людиною за вибором стратегії поведінки, при цьому як правило робот оснащується експертною системою, здатною прогнозувати результати маніпуляцій і дає поради щодо вибору мети).

На основі представленого вище аналізу для поставленого завдання ідентифікації ситуацій, пропонується вибрати варіант робота на мобільній платформі ZK-4WD, який за типом управління системи є адаптивним і має зв'язок з комп'ютером по бездротовому каналу зв’язку стандарту IEEE 802.11g. Також мобільна платформа є компактною і дозволяє розмістити всі необхідні датчики, елементи управління та елементи живлення.

1.3 Огляд датчиків і систем контролю використованих у системах дослідження подій.

Від того, як влаштована система залежать її можливості.

Теплові сповіщувачі діють при поширенні теплоти від вогнища пожежі. При пожежі нагріті маси повітря спрямовуються вгору, тому теплові сповіщувачі встановлюють на стелі приміщення, що охороняється. Чутливими елементами теплових сповіщувачів можуть бути біметалічні пластинки або спіралі, пружні пластинки зі спаяними легкоплавким припоєм кінців, електроконтактні термометри, термопари та ін. Датчики з плавкими або спалимими вставками не відновлюються, а газові, ртутні, рідинні, металеві та біметалічні самовідновлюються[10].

Тепловий пожежний сповіщувач типу ДТЛ (рисунок 1.4) одноразової дії працює на розрив електричної мережі при досягненні розрахункової температури плавлення використовуваного в сповіщувачі сплаву: 72,5 °С. Температуру спрацювання максимальних температурних сповіщувачів типу АТСМ (автоматичний тепловий сповіщувач максимального дії) і НТСМ (напівпровідниковий тепловий сповіщувач максимального дії) приймають на 20 °С вище нормальної робочої температури цього приміщення[11].

Пластини біметалевих сповіщувачів при нагріванні до критичної температури (60, 80 або 100 °С) деформуються, замикаючи або розмикаючи контакти сигнального кола.

Диференціальні температурні сповіщувачі налаштовують на спрацьовування при швидкості наростання температури навколишнього повітря 5...10 °С за 1 хв. Контрольована одним тепловим сповіщувачем площа становить 15...30м2, тривалість дії близько 60 с.

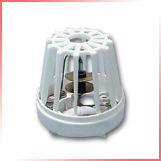


Рисунок 1.4 сповіщувач типу ДТЛ

Теплові сповіщувачі максимально-диференціальної дії (рисунок 1.5) працюють як при збільшенні швидкості підвищення температури до певного значення, так і при заданій критичній температурі. Час спрацювання їх не перевищує 50 с, контрольована площа близько 25 м2.



Рисунок 1.5 тепловий сповіщувач максимально-диференціальної дії

Димові сповіщувачі (рисунок 1.6) [12] в порівнянні з сповіщувачами інших типів значно швидше виявляють вогнище загоряння, так як пожежа може розвиватися дуже повільно, і дим в цьому випадку завжди супроводжує початковій стадії горіння. Способи виявлення диму засновані на застосуванні фотоелементів або іонізаційних камер з радіоактивними речовинами (наприклад, плутонієм-239). Дим, потрапляючи в іонізаційну камеру, знижує ступінь іонізації повітря, що в підсумку призводить до спрацьовування виконавчого реле приймальної станції, яке включає систему сигналізації. Час спрацювання димового сповіщувача при попаданні в нього диму не перевищує 5с.

Світлові сповіщувачі характеризуються безінерційністю і великою (до 600 м2) зоною контролю. У світлових сповіщувачів використовується явище фотоефекту. Встановлений в них фотоелемент реагує на ультрафіолетову або інфрачервону частину спектру полум'я. Так, автоматичний сповіщувач реагує на інфрачервоне випромінювання з довжиною хвилі 0,3 \* 10-6 ... 2 \* 10-6м, перетворюючи його в електричну енергію, яка надходить до приймальної станції і викликає подачу сигналу тривоги.



Рисунок 1.6 Димовий сповіщувач

Світловий сповіщувач (рисунок 1.7) складається з датчика - лічильника фотонів, електричної схеми та сигнального реле. Лічильник фотонів має високу чутливість і здатний виявити навіть невеликі осередки полум'я (наприклад, горіння сірника) практично миттєво. Незважаючи на високу чутливість, цей сповіщувач не спрацьовує від денного світла, що проходить через віконне скло, а також від електричного освітлення, так як ультрафіолетові промені поглинаються склом вікон і ламп. Він працює за принципом прямої видимості вогню. При відсутності прямої видимості для спрацьовування сигналізації може виявитися достатнім світла від вогню, відбитого будь-яким предметом, що знаходиться в приміщенні.

Однак світлові сповіщувачі можна застосовувати тільки в закритих приміщеннях, в яких відсутні джерела ультрафіолетових або інфрачервоних випромінювань, відкрите полум'я, працюючі зварювальні апарати, електричні іскри і т. д.

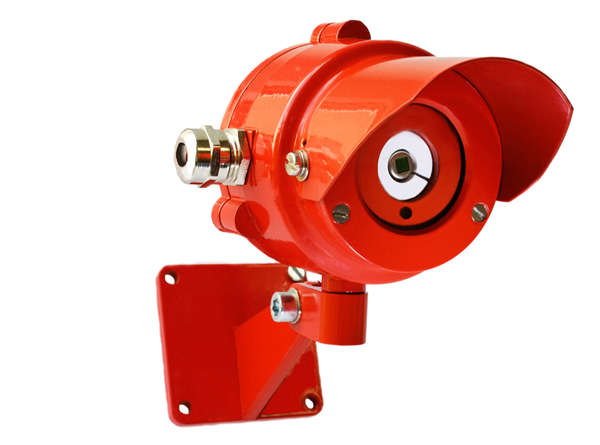


Рисунок 1.7 Світловий сповіщувач

Автоматичний комбінований сповіщувач (рисунок 1.8) реагує як на тепло, так і на дим. Він виконаний на базі димового сповіщувача з додаванням елементів електричної схеми, необхідних для виявлення вогнища пожежі за теплотою. Цей датчик спрацьовує при температурі 60...80 °С або попаданні в нього диму, створюваного розташованим безпосередньо під сповіщувачем тліючим гнітом діаметром 6мм. Контрольована площа становить 100 м2.



Рисунок 1.8 Автоматичний комбінований сповіщувач

Ультразвукові сповіщувачі (рисунок 1.9) реагують на нестійке полум'я і рухомі об'єкти. Робота датчиків цього типу заснована на ефекті Доплера, який полягає в тому, що відбиті від рухомих предметів ультразвукові коливання мають частоту, відмінну від випромінюваної. Дані сповіщувачі безінерційні і контролюють площу до 1000 м2 [13].

Для реєстрації рентгенівського випромінювання застосовують циліндричний лічильник Гейгера—Мюллера(рисунок 1.10). Він складається з металевої трубки або металізованої зсередини скляної трубки і тонкої металевої нитки, натягнутої по осі циліндра.



Рисунок 1.9 Ультразвуковий сповіщувач

Робота лічильника заснована на ударній іонізації. Електрони, рухаючись в газі і стикаючись з атомами газу, вибивають з атомів електрони і створюють позитивні іони і вільні електрони. Електричне поле між катодом і анодом прискорює електрони до енергій, при яких починається ударна іонізація.

У проекті використовується: мікрокомп’ютер Raspberri Pi 2, плата розширення Arduino MotorShield, ультразвуковий сенсор HC-Sr04, USB адаптер D-ling DW-110, камера Raspberry PI 5MP Camera.

1.4 Постановка задачі

На основі проведеного вище аналізу, можна сформулювати постановку задачі.

Метою даної роботи є розробка мобільної системи дослідження подій, що доповнює стаціонарні системи контролю та обробки інформації



Рисунок 1.10 Циліндричний лічильник Гейгера–Мюллера

Етапами вирішення поставленої задачі є:

- вивчити літературу з предметної області;

- вивчити літературу з Raspberri Pi 2, Arduino MotorShield, стандарту IEEE 802.11g, ультразвуковому сповіщувачу SR-hc04,камери Raspberry PI 5MP Camera;

- проаналізувати функції всіх пристроїв необхідних для реалізації програмного продукту;

- зібрати робот-розвідник на базі Raspberry Pi 2 і всіх модулів, пристроїв та датчиків;

- розробити скетч для бездротової передачі даних і руху робота;

- проаналізувати бібліотеку wiringPi, C для реалізації програмного продукту;

- розробити додаток для візуалізації і обробки інформації, що отримується за допомогою ультразвукового сповіщувача та камери, за допомогою мови програмування C, IDE Glade та GTK+;

- організувати зручний інтерфейс взаємодії користувача з додатком;

- виконати тестування системи на тестових прикладах.

2 ДОСЛІДНИЦЬКА ЧАСТИНА.  ПРОЕКТУВАННЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ДОСЛІДЖЕННЯ ПОДІЙ НА ОСНОВІ МОБІЛЬНОГО РОБОТА

2.1 Побудова мобільної платформи на базі мікрокомп’ютера Raspberry Pi 2.

Мікрокомп'ютер Raspberry Pi 2 поєднаний з Arduino MotorShield та ультразвуковим сповіщувачем HC-SR04 за допомогою інтерфейсу GPIO (Інтерфейс введення/виведення загального призначення), з камерою Raspberry PI 5MP Camera за допомогою коннектору CSI, з локальною бездротовою мережею через USB адаптер D-link DW-110, має Micro SD картку на 16 гб та під’єднаний до живлення ємністю 4400mAh.

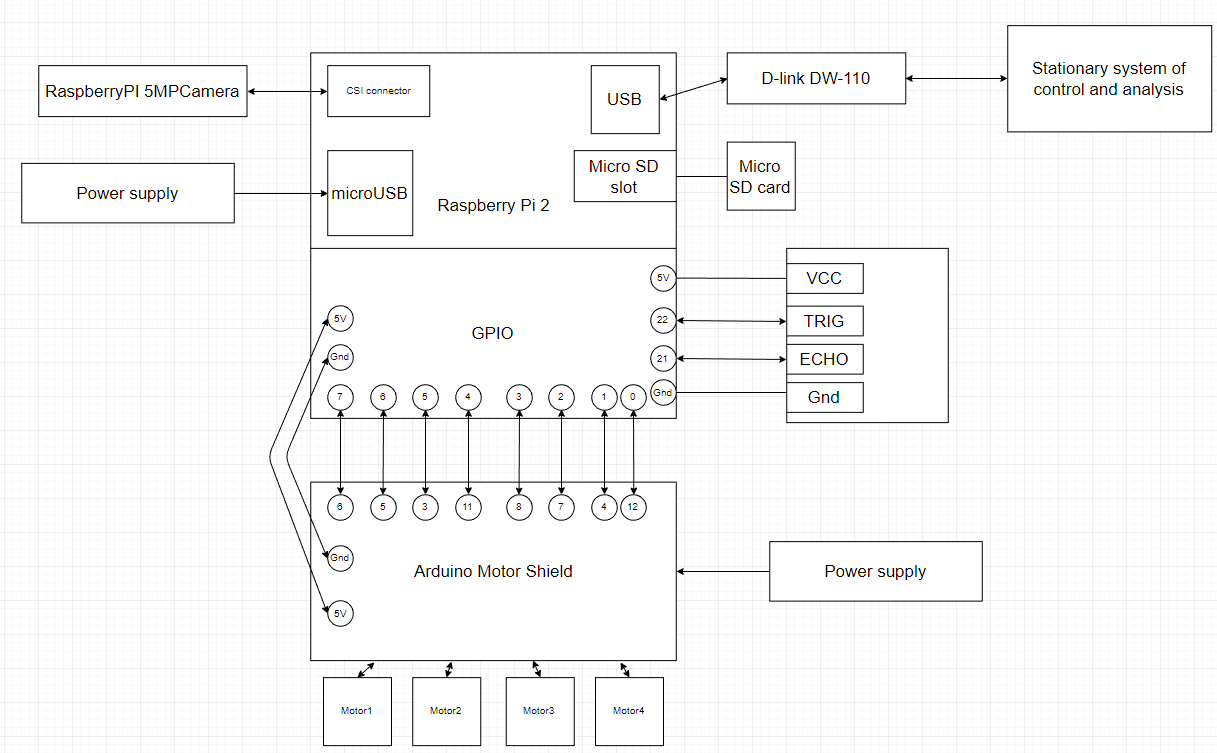


Рисунок 2.1 Схема побудови мобільного робота на базі Raspberry Pi 2 та поєднання його з стаціонарною системою контролю та обробки інформації

2.2 Детальний опис компонентів мобільної системи.

2.2.1 Шасі ZK-4WD



Рисунок 2.2 – шасі ZK-4WD

Вданій задачі використовується повнопривідне шасі ZK-4WD (рисунок 2.2) з чотирма колесами зі спеціальною гумовою ребристою фактурою протектора для кращої прохідності робота. Платформа має чотири мотори-редуктора (рисунок 2.3) з передавальним відношенням 48:1. Характеристика приводу мотора редуктора приведена в таблиці 2.1

Мотори працюють через процес, званий індукційним. Коли помістити електричний заряд через провід, створюється магнітне поле. Кручений дрот створить сильніше поле, та підвищить струм. У моторі постійного струму, намотаний дріт оточує вал двигуна. Сформоване магнітне поле витягується і відбивається магнітами усередині тіла мотора.

Мотор буде тягнути більш струму, коли вони починають роботу або мають навантаження. Коли мотор запущений і працює, він буде тягнути значно менше струму.



Рисунок 2.3 – мотор-редуктор

Таблиця 2.1 – Характеристика приводу мотора редуктора

|  |  |
| --- | --- |
| Номінальна напруга: | 4,5VDC |
| Швидкість, без навантаження: | 90±10 rpm |
| Струм навантаження: | 190 mA (250mA MAX) |
| Максимальний крутний момент: | 800 gf.cmmin |
| Розмір(мм): | 20 \* 20 \* 64 |

Номінальна напруга описує пікову робочу напругу для мотора, коли він працює з максимальною ефективністю. Якщо напруга більше або менше від номінальної напруги мотору, це може з часом скоротити термін служби мотору. Якщо ви надали менше номінальної напруги, то двигун буде обертатися повільніше. Як правило, двигун потрібно близько 1/2 його номінальної напруги для запуску. Якщо ви надасте менше ніж 1/2 його номінальної напруги, він ймовірно не почне рухатися.

2.2.2 Мікрокомп'ютер Raspberry Pi 2

Таблиця 2.2 – Характеристика Raspberry Pi 2

|  |  |
| --- | --- |
| Процессор | ARM Cortex-A7 |
| Частота | 0,9 ГГц |
| Количество ядер | 4 |
| Об'є́м оперативної пам'яті | 1 ГБ |
| Графіческий чіпсет | інтегрований VideoCoreIV 3D |
| Звуковий контроллер | інтегрований |
| Зовнішни порти | 4 USB, 1HDMI, 1Audio Jack (Mic in/Headphone out), 1LAN |
| Кардрідер | microSD |
| Необхідне живлення | 5 В, 1 А через microUSB/12 В |

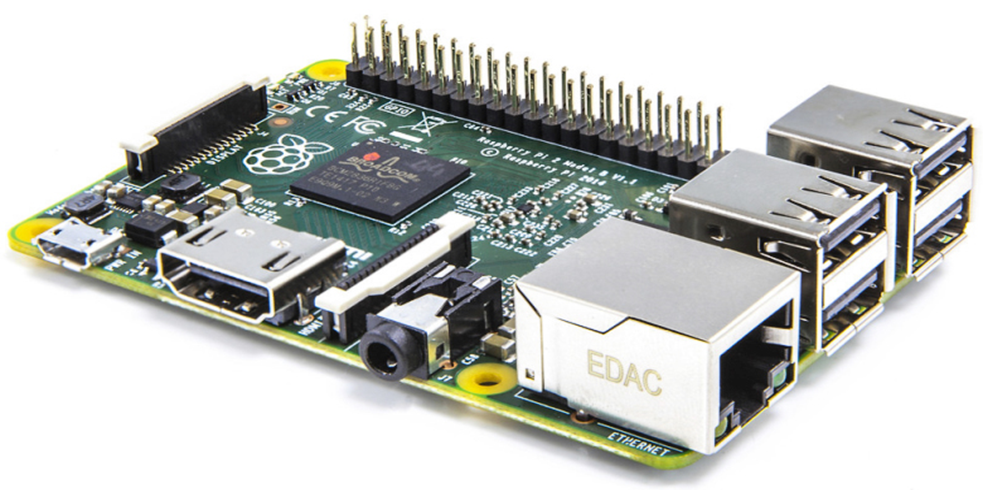


Рисунок – 2.4 Raspberry Pi 2

Зовні Raspberry Pi 2 являє собою невелику плату трохи більше банківської карти. Мозком є чотирьохядерний ARM Cortex-A7 з частотою 900 МГц. При бажанні можна і розігнати трохи за допомогою вбудованої утиліти. Крім цього, на платі розпаяний гігабайт пам'яті, з якого до 128 МБ можна виділити для роботи відеоядра. Відеовихід - HDMI. Від аналогового RCA, наявного в колишніх версіях, розробники відмовилися. Плата обладнана чотирма слотами USB. Кожен порт здатний видавати струм силою до 1,2 А. Правда, для цього Raspberry потрібно живити від блоку живлення 2 А. На Raspberry Pi 2 розташований слот для Micro SD, на яку була встановлена операційна система Raspbian сімейства Unix, та программа управління MRPII. Головне, що відрізняє Raspberry від звичайного комп'ютера, - наявність додаткових висновків. Їх багато. Завдяки окремим висновків з засувками можна підключити камеру (CSI) і дисплей (DSI). Обидва периферійних пристрої будуть працювати безпосередньо з відеоядром і процесором. А ще є 40 висновків GPIO: інтерфейсу введення-виведення загального призначення. З його допомогою можна підключати все і міняти призначення входів-виходів на льоту.

2.2.3 Arduino MotorShield

Arduino MotorShield (рисунок 2.5) являє собою модуль, що забезпечує функціональність для використання двигунів постійного струму, сервомоторів і крокових двигунів спільно з Raspberry Pi 2.

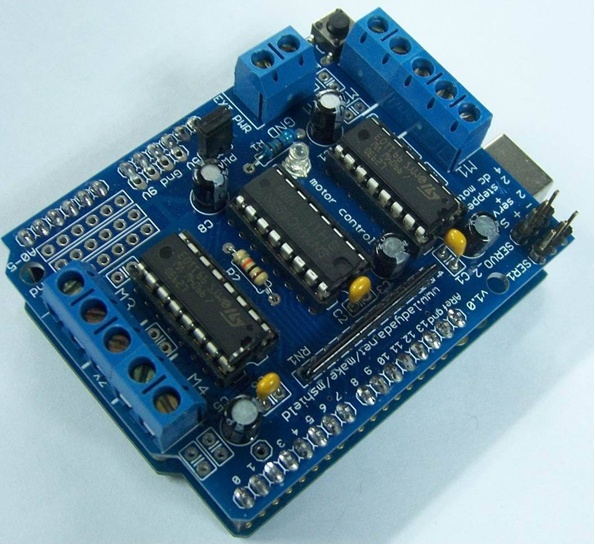


Рисунок 2.5 – Arduino MotorShield

Таблиця 2.3 – Характеристика Arduino MotorShield

|  |  |
| --- | --- |
| Напруга живлення | 6 – 24В |
| Кількість силових каналів | 4 |
| Максимально-тривалий струм кожного каналу | 0,6 А |
| Максимальний піковий (<100 мс) неповторяющийся ток в кожному каналі | 1,2 А |

Продовження таблиці 2.3

|  |  |
| --- | --- |
| Напряженіе сервоприводів | 5 В |
| Можливість реверсу кожного двигуна | є |
| Можливість незалежного управління кожним каналом | є |
| Захист від перегріву | є |
| Габарити | 69,3 х 53,1 х 20,0 мм |
| Вага | 34 г. |

2.2.4 Ультразвуковий сповіщувач HC-SR04

Ультразвуковий датчик вимірювання відстані HC-SR04 (рисунок 2.6).

Роз'єм сенсора - 4 звичайних піна, що дозволяє просто застромити його в беспаечное плату, припаяти до макетної платі або використовувати для підключення звичайних пінов (типу мама).



Рисунок 2.6 – Ультразвуковий сповіщувач HC-SR04

Таблиця 2.4 – Характеристика HC-SR04

|  |  |
| --- | --- |
| Напруга живлення: | 5V |
| Струм спокою: | <2mA |
| Ефективний кут | <15o |
| Діапазон вимірювання відстані: | 2-400 см |
| Дозвіл | 0,3см |

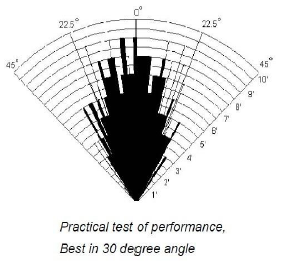


Рисунок 2.7 Діаграма спрямованості

Принцип роботи: сенсор випромінює короткий ультразвуковий імпульс (в момент часу 0), який відбивається від об'єкта і приймається сенсором. Відстань розраховується виходячи з часу до отримання луни і швидкості звуку в повітрі.т.ч., сенсор отримує сигнал луни, і видає відстань, яке кодується тривалістю електричного сигнал на виході датчика (Echo).

Наступний імпульс може бути ізлучён, тільки після зникнення луни від попереднього. Це час називається періодом циклу (cycle period). Рекомендований період між імпульсами повинен бути не менше 50 мс.

Якщо на сигнальний пін (Trig) подається імпульс тривалістю 10 мкс, то ультразвукової модуль буде випромінювати вісім пачок ультразвукового сигналу з частотою 40кГц і виявляти їх відлуння. Вимірювання відстаней до об'єкта пропорційно ширині луни (Echo) і може бути розрахована за формулою, наведеною на графіку вище.

Документація на сенсор, так само вказує, що якщо ніяких перешкод не виявлено, то на вих оді буде сигнал з тривалістю 38ms.

2.2.5 USB адаптер D-link DWA-110

Бездротовий USB-адаптер DWA-110 (рисунок. 2.8) працює на основі стандарту 802.11g і назад сумісний зі стандартом 802.11b, гарантуючи сумісність з широким діапазоном безпровідних пристроїв.

Проект стандарту IEEE 802.11g був затверджений в жовтні 2002 року. Цей стандарт передбачає використання діапазону частот 2,4 ГГц, забезпечуючи швидкість з'єднання до 54 Мбіт/с (брутто) і перевершуючи, таким чином, стандарт IEEE 802.11b, який забезпечує швидкість з'єднання до 11 Мбіт / с. Крім того, він гарантує сумісність зі стандартом 802.11b.



Рисунок 2.8 – D-link DWA-110

Зворотна сумісність стандарту IEEE 802.11g може бути реалізована в режимі модуляції DSSS, і тоді швидкість з'єднання буде обмежена одинадцятьма мегабітами в секунду або в режимі модуляції OFDM, при якому швидкість може досягати 54 Мбіт/с. Таким чином, даний стандарт є найбільш прийнятним при побудові бездротових мереж.

2.2.6 Raspberry Pi 5 MP Camera

Камера 5 магепікселей для Raspberry Pi від Waveshare (рисунок 2.10) на базі сенсора OV5647. Сумісна з усіма версіями Raspberry. Кріпиться шлейфом, який йде в комплекті, до гнізда CSI. Дозволяє робити фото з здатністю 2592 х 1944 пікселів, знімати відео в трьох режимах - 1080р @ 30fps, 720p @ 60 fps і 480p @ 90fps.

Для того чтоб транслювати відео з камери у локальну мережу я використовую програму MJPG-streamer.

MJPG-streamer являє собою додаток командного рядка, яка копіює JPEG кадрів з одного або декількох вхідних плагінів для декількох вихідних модулів. Він може бути використаний для потокової передачі файлів JPEG поверх IP-мережі від веб-камери до різних типів глядачів, таких як Chrome, Firefox, VLC, MPlayer, та іншого програмного забезпечення, яке здатне приймати MJPG потоки.

Спочатку він був написаний для вбудованих пристроїв з дуже обмеженими ресурсами в плані оперативної пам'яті і процесора. Його попередник «uvc\_streamer» був створений тому, що Linux-UVC сумісні камери безпосередньо виробляють JPEG-данні, що дозволяє швидкі та оптимізовані JPEG трянсляції відео навіть від вбудованого пристрою під керуванням OpenWRT. Вхідний модуль «input\_uvc.so» захоплює такі JPG кадри з підключеної веб-камери. MJPG-стримерний тепер підтримує безліч різних пристроїв введення.



Рисунок 2.10 – Raspberry PI 5MP Camera

Таблиця 2.5 – Характеристика Raspberry PI 5MP Camera

|  |  |
| --- | --- |
| 5-мегапіксельний датчик | OV5647 |
| Дозвіл фото: | 2592 х 1944 пікселів |
| Підтримувані формати відео: | 1080р @ 30fps, 720p @ 60 fps і 480p @ 90fps |
| Розмір матриці CCD: | ¼ дюйма |
| Апаратура (F): | 2.8 |
| Фокусна відстань: | 3.37 мм |
| Розміри: | 25 х 24 х 9 мм. |

2.2.7 Power bank DCAE

Для живлення Raspberry Pi використовується Power bank DCAE (рисунок 2.11).



Рисунок 2.11 – Power bank DCAE

Таблиця 2.6 – Характеристика Power bank DCAE

|  |  |
| --- | --- |
| Тип батареї: | Літій-полімерний Акумулятор |
| інтерфейс: | USB та Micro USB |
| Ємність батареї (мАг): | 5001-7000 мАг |
| Вхід: | 5V / 1А |
| Вихід: | 5V / 1A |
| Розмір: | 105 x 60 x 10 мм |

2.3 Вигляд зібраної мобільна система дослідження подій на базі мікрокомп’ютера Raspberry Pi 2 (рисунок 2.12)

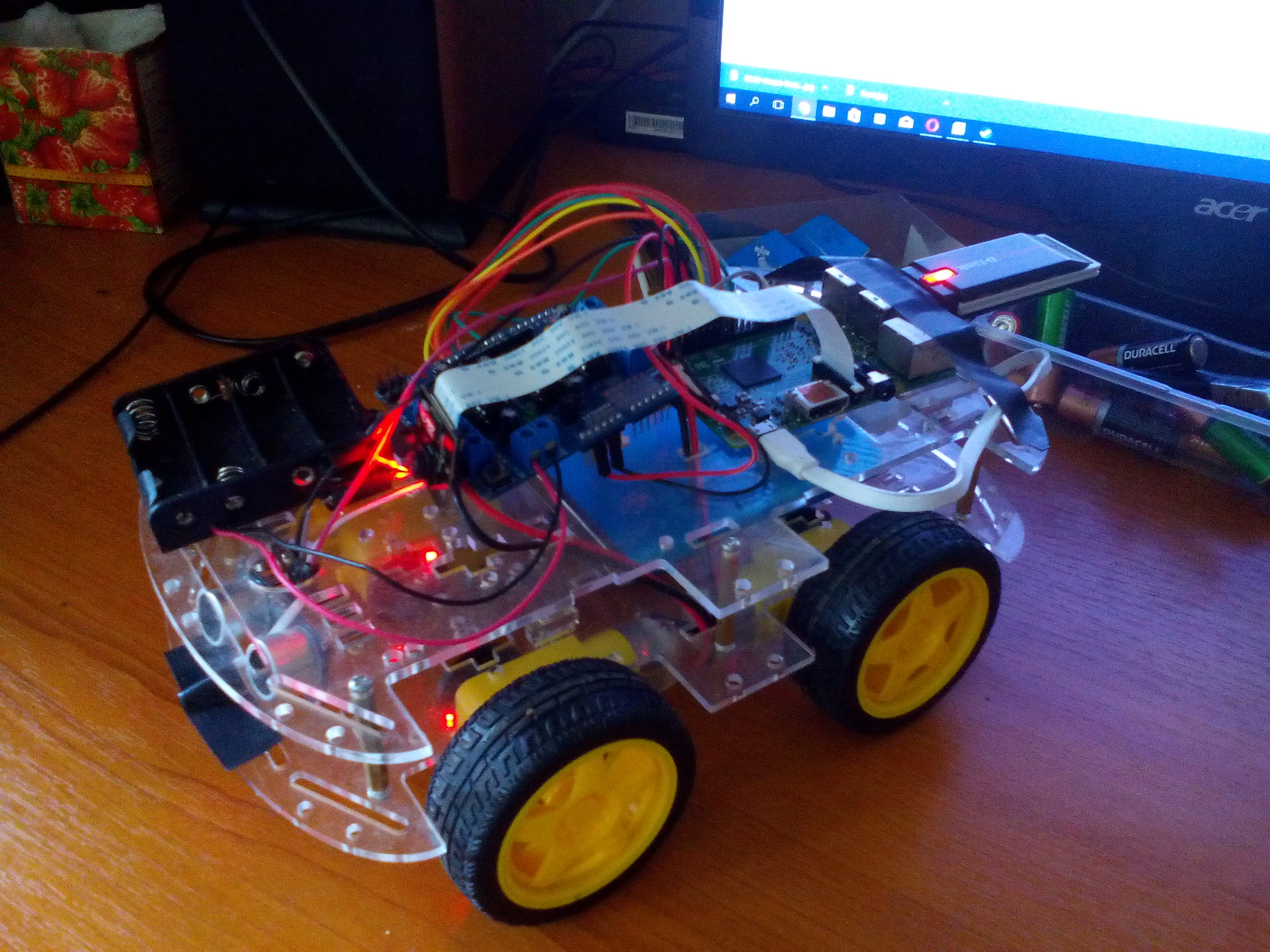


Рисунок 2.12 – повністю зібрана мобільна система на базі Raspberry Pi 2

3 ОПИС РЕАЛІЗАЦІЇ РІШЕННЯ

3.1 Опис архітектури програми

Додаток має 1 форму (рисунок 3.1) на якій розміщені: 6 кнопок, 2 текстових поля вводу і 4 ярлика.

1. Кнопки:

«btn\_left» - кнопка для руху роботу вліво;

«btn\_back» - кнопка для руху робота назад;

«btn\_right» - кнопка для руху робота вправо;

«btn\_forward» - кнопка для руху робота вперед;

«btn\_sonar» - кнопка-тумблер для активації ультразвукового сповіщувача;

«btn\_auto» - кнопка-тумблер для активації функція об'їзду перешкод.

1. Текстові поля вводу:

«entry\_block» - текстове поле вводу мінімальної дистанції до перешкоди;

«entry\_turn» - текстове поле вводу часу який потрібен роботу щоб здійснити повній поворот на 360°.

1. Ярлики:

«lbl\_sonar» елемент який відображає значення отримані від ультразвукового сповізувача;

«lbl\_distance» надпис “Distance”;

«lebel1» надпис “Distance to the obstacle”;

«lebel2» надпис “Time for a full turn”.

Модулі в даній системі взаємодіють, як частини проекту. Файл mrpiimain.c є файлом програми, з яким пов’язана бібліотека wiringPi [13], файл заголовку MRPII.h та файл-джерело MRPII.c .

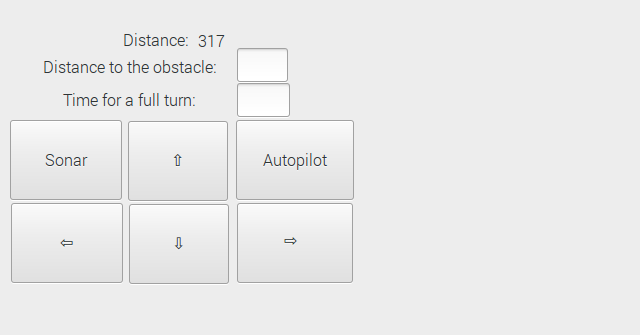


Рисунок 3.1 – Екранна форма

3.2 Опис даних

Таблиця 3.1 – Основні вхідні дані

|  |  |  |
| --- | --- | --- |
| Вхідні дані | Тип | Опис |
| Данні від ультразвукового сповіщувача | long | Мікросекунди які знадобилися для того щоб ультразвуковий пульс дійшов від перешкоди до сповіщувача. |

Таблиця 3.2 – Основні вихідні дані

| Вихідні дані | Тип | Опис |
| --- | --- | --- |
| distance | int | Відстань до перешкоди. |

3.3 Опис методів реалізації

3.3.1 Реалізація графічного інтерфейсу

Для розробки даного проекту була використана мова програмування С, інтегроване середовище розробки Glade (Рис 3.2)[14] та кроссплатформенная бібліотека GTK+[13]. Додана бібліотека wiringPi для роботі з Arduino MotorShield та ультразвуковим сповіщувачем HC-SR04.

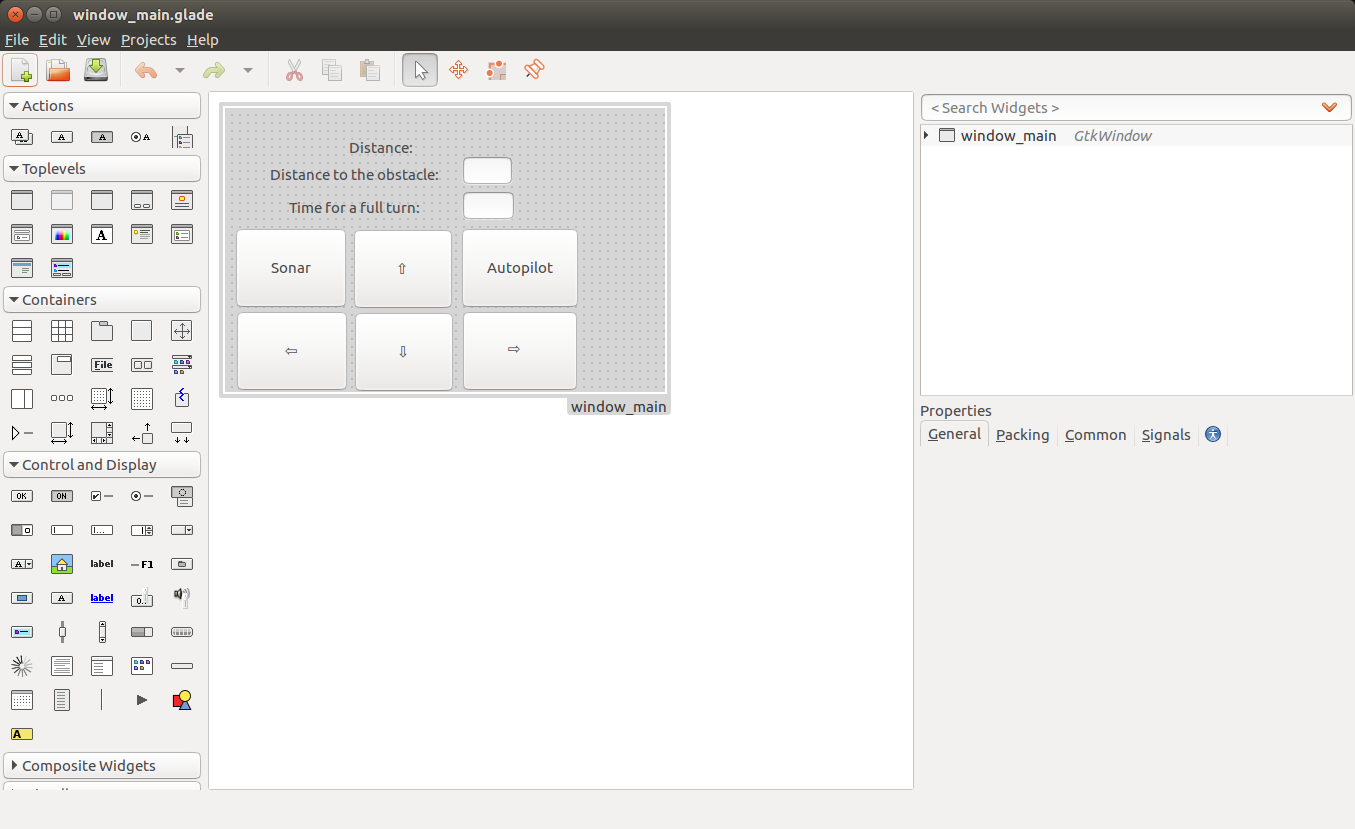


Рисунок 3.1 – інтегроване середовище розробки Glade.

Glade є RAD(Rapid Application Development) інструментом для швидкої та легкої розробки призначених для користувача інтерфейсів для GTK + і оточення робочого столу GNOME. Призначені для користувача інтерфейси, розроблені в Glade зберігаються у вигляді XML, а також за допомогою GtkBuilder GTK + об'єкту вони можуть бути завантажені додатками динамічно коли необхідно.

Опис основних методів показано в таблиці 3.3

Таблиця 3.3 – Основні методи.

|  |  |
| --- | --- |
| Метод | Опис |
| digitalWrite() | Встановлюе біт виходу |
| enable() | Встановлюе значення контактів |
| DCMotorInit (uint8\_t num) | Ініціалізація мотору |
| DCMotorRun(uint8\_t motornum, uint8\_t cmd) | Запуск мотору |
| getCM () | Зчитування даних з ультразвукового сповіщувача |
| lleft() | Повертає мобільну платформу вліво до сигналу зупинки моторів |
| rright() | Повертає мобільну платформу вправо до сигналу зупинки моторів |
| fforwd() | Рухає мобільну платформу вперед до сигналу зупинки моторів |
| backwd() | Рухає мобільну платформу назад до сигналу зупинки моторів |
| Stop() | Зупинка моторів |
| autopilot() | Функція об'їзду перешкод |

3.3.2 Реалізація функції об’їзду перешкод.

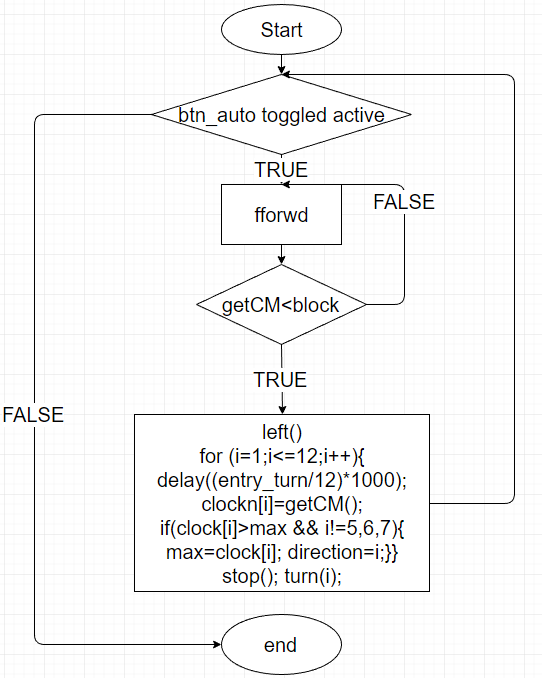


Рисунок 3.2 – Алгоритм об’їзду перешкод.

Код функції об’їзду перешкод (Приклад 3.1)

//autopilot

void autopilot(){

int clockn [12];

const char \*getentry;

getentry = gtk\_entry\_get\_text(GTK\_ENTRY(g\_entry\_block));

int block = atoi(getentry);

int max\_distance = 0;

int direction = 6;

int i;

const char \*getturn;

getturn = gtk\_entry\_get\_text(GTK\_ENTRY(g\_entry\_turn));

double fraction = atof(getturn);

fraction = fraction/12;

fraction = fraction\*1000;

int turn\_degree = 0;

while(gtk\_toggle\_button\_get\_active(g\_btn\_auto)){

fforwd();

//drive forward untill obsticle closer than block(cm)

while (getCM()>=block){}

stop();

delay(500);

lleft();

//turn left untill sonar gets 12 readings

for (i = 1;i<=12;i++) {

delay(fraction);

clockn[i]=getCM();

if (clockn[i] > max\_distance){

if (i!=5 && i!=6 && i!=7){

max\_distance = clockn[i];

direction = i;

}

}

}

stop();

delay(500);

if (direction <= 4){

turn\_degree = fraction \* direction;

left(turn\_degree);

}

else {

turn\_degree = fraction \* (12 - direction);

right(turn\_degree);

}

delay(500);

max\_distance = 0;

direction = 6;

//main iteration, checking for toggle button condition

while (gtk\_events\_pending()) {

gtk\_main\_iteration();

}

}

}

Приклад 3.1 – функція об’їзду перешкод (файл mrpiimain.c)

3.3.3 Реалізація функції вимірювання відстані до перешкоди (приклад 3.2).

int getCM() {

//delay(200);

//Send trig pulse

digitalWrite(TRIG, HIGH);

delayMicroseconds(60);

digitalWrite(TRIG, LOW);

//Wait for echo start

while(digitalRead(ECHO) == LOW);

//Wait for echo end

long startTime = micros();

while(digitalRead(ECHO) == HIGH);

long travelTime = micros() - startTime;

//Get distance in cm

int distance = travelTime / 58;

// printf("Distance: %dcm\n", distance);

return distance;

}

Приклад 3.2 – функція вимірювання відстані до перешкоди (файл MRPII.c)

3.3.4 Реалізація обміну інформації мобільної системи з стаціонарною(ПК)

Для того щоб налогодити спілкування мобільної системи з ПК я використовував додаток VNC Viewer.

VNC Viewer – це система віддаленого доступу до робочого столу комп'ютера, що використовує протокол RFB Управління здійснюється шляхом передачі натискань клавіш на клавіатурі і рухів миші з одного комп'ютера на інший і ретрансляції вмісту екрану через комп'ютерну мережу.

RFB (віддалений фреймбуфер) є відкритим простим протоколом для віддаленого доступу до графічного інтерфейсу користувача. Так як він працює на рівні фрейм буферу, він застосовується для всіх віконних систем і додатків, включаючи Microsoft Windows, MacOS і X Window System.

3.4 Опис можливостей для еволюції додатку

Подальший розвиток системи передбачае зміну ультразвукового оповіщувача на вимірювальний лазерний пристрий встановлений на сервопривід. Це покращить точність заміру дистанції до перешкоди та збільшить кут знаходження перешкоди з 15° до 360°.

3.5 Опис вимог до ПЗ та АЗ

Створювався проект на ПК з наступними характеристиками:

- операційною системою: Ubuntu 16.04.2 LTS;

- процесором 3,4 GHz;

- оперативною пам'яттю 8 Гб;

- відеоадаптером 2048 Мб;

При експлуатації програми необхідними мінімальними вимогами є:

-Raspberry Pi 2;

-Micro SD картка від 8гб;

3.6 Опис інструкцій користувачеві

Після того як живлення до Raspberry Pi і живлення на Arduino MotorShield подано, програма управління стане доступна відразу після загрузки операційної системи.

Роботом можна курувати за допомогою кнопок ⇦ ⇧ ⇩ ⇨ на екранний формі

Щоб увімкнути або вимкнуті ультразвуковий сповіщувач слід натиснути кнопку-тумблер Sensor.

Щоб запустити або відмінити функцію об’їзду перешкод слід натиснути кнопку-тумблер Autopilot.

Для того щоб програма управління автоматично запускалася разом с операційною системою слід редагувати файл /etc/rc.local та внести такий рядок : “sudo ./home/Desktop/MRPII/mrpii”.

ВИСНОВКИ

В результаті проведеної роботи було спроектовано і реалізовано систему дослідження подій на основі мікрокомп’ютера Raspberry Pi 2, та розроблено додаток який дозволяє керувати мобільним роботом по бездротовому зв'язку через локальну комп’ютерну мережу стандарту IEEE 802.11g, а так само передавати дані з ультразвукового сповіщувачу та налагоджена трансляція відео с камери.

Забезпечено живлення для Raspberry Pi 2 і двигунів постійного струму, а також реалізована зарядка батареї за допомогою mini usb.

Розроблено зручний інтерфейс користувача. Тестування програми на контрольних прикладах продемонструвало правильність роботи і відсутність збоїв. До програми написана інструкція для користувача по експлуатації.

Даний програмний комплекс можна використовувати в автоматизованих охоронних системах, автоматизованих комплексах обслуговування небезпечних виробництв, побуті, автоматичних комплексах дослідження будівель, тунелів і т.д.

СПИСОК ПОСИЛАНЬ

1. Попов Е. П., Письменный Г. В.  Основы робототехники: Введение в специальность. — М.: Высшая школа, 1990. — 224 с.
2. GoingFurther. [Електронний ресурс]. – Режим доступу: http://www.imagesco.com/articles/airmuscle/AirMuscleDescription06.html - Назва з екрану.
3. Тягунов О. А. Математические модели и алгоритмы управления промышленных транспортных роботов. Информационно-измерительные и управляющие системы. — 2007. — Т. 5, № 5.—С. 63 — 69.
4. Охоцимский Д. Е., Мартыненко Ю. Г. Новые задачи динамики и управления движением мобильных колёсных роботов. Успехи механики. — 2003. — Т. 2, № 1. — С. 3 — 47.
5. Черноусько Ф. Л. Волнообразные движения многозвенника по горизонтальной плоскости. Прикладная математика и механика. — 2000. — Т. 64, вып. 4. — С. 518 — 531.
6. Князьков M. M., Башкиров С. А. Плоское передвижение многозвенного робота по поверхности с сухим трением. Мехатроника, автоматизация, управление. — 2004. — № 3. — С. 28—32.
7. Градецкий В. Г., Вешников В. Б., Калиниченко С. В., Кравчук Л. Н. Управляемое движение мобильных роботов по произвольно ориентированным в пространстве поверхностям. — М.: Наука, 2001. — 360 с.
8. Зенкевич С. Л., Ющенко А. С. Основы управления манипуляционными роботами. 2-е изд. — М.: Изд-во МГТУ им. Н. Э. Баумана, 2004. — 480 с.
9. Тепловые пожарные извещатели: выбор типов и классов. [Електронний ресурс]. – Режим доступу: http://arton.com.ua/downloads/publications/ teplovye\_pozharnye\_izvewateli\_vybor\_tipov\_i\_klassov/- Назва публікації.
10. Как работают тепловые пожарные извещатели. [Електронний ресурс]. – Режим доступу: <http://ohranivdome.net/pozharnaya-signalizatsiya/kak-rabotayut-teplovye-pozharnye-izveshhateli.html>- Назва публікації.
11. Использование дымовых извещателей в помещении. [Електронний ресурс]. – Режим доступу: <http://protivpozhara.ru/signal/struktura/dymovye-izveshhateli> - Назва статті.
12. wiringPi [Електронний ресурс] – Режим доступу <http://www.wiringpi.com/> ShadowAirMuscles. [Електронний ресурс]. – Режим доступу: <http://www.shadowrobot.com/products/air-muscles/> - Назва з екрану.
13. GTK+ tutorial [Електронний ресурс]. - <https://developer.gnome.org/gtk3/stable/>
14. Glade tutorial [Електронний ресурс]. - <https://glade.gnome.org/>

ДОДАТОК А

UML ДІАГРАМИ

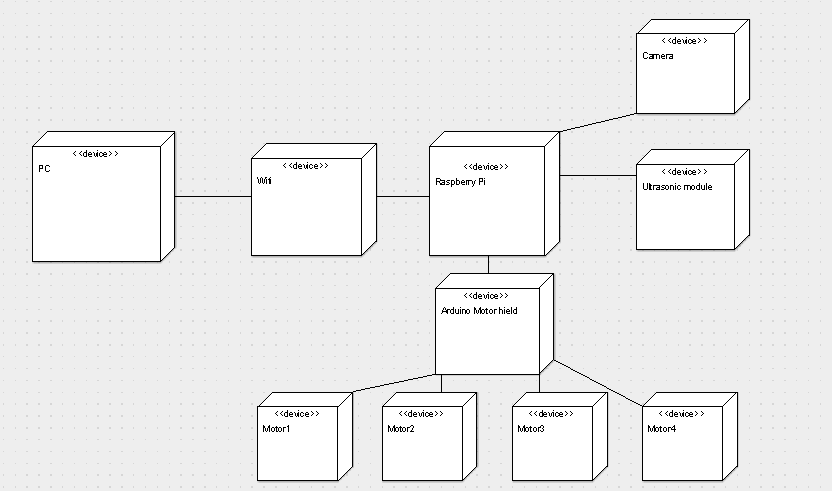


Рисунок А.1 – Діаграма розгортання



Рисунок А.2 – Діаграма компонентів

ДОДАТОК Б

ЛІСТІНГ ПРОГРАМИ mrpii

Б.1 – Код файлу mrpiimain.c

#include <gtk/gtk.h>

#include <stdio.h>

#include <errno.h>

#include <string.h>

#include <unistd.h>

#include <wiringPi.h>

#include <softPwm.h>

#include <stdlib.h>

#include "MRPII.h"

typedef unsigned char uint8\_t;

#define \_BV(bit) (1 << (bit))

#define TRIG 22

#define ECHO 21

#define MOTORLATCH 0

#define MOTORCLK 1

#define MOTORENABLE 2

#define MOTORDATA 3

#define MOTOR\_1\_PWM 4

#define MOTOR\_2\_PWM 5

#define MOTOR\_3\_PWM 6

#define MOTOR\_4\_PWM 7

#define MOTOR1\_A 2

#define MOTOR1\_B 3

#define MOTOR2\_A 1

#define MOTOR2\_B 4

#define MOTOR4\_A 0

#define MOTOR4\_B 6

#define MOTOR3\_A 5

#define MOTOR3\_B 7

#define FORWARD 1

#define BACKWARD 2

#define BRAKE 3

#define RELEASE 4

GtkWidget \*g\_lbl\_hello;

GtkWidget \*g\_lbl\_count;

GtkWidget \*g\_lbl\_sonar;

GtkToggleButton \*g\_btn\_sonar;

GtkToggleButton \*g\_btn\_auto;

GtkEntry \*g\_entry\_turn;

GtkEntry \*g\_entry\_block;

int main(int argc, char \*argv[])

{

if (wiringPiSetup () == -1)

{

fprintf (stdout, "oops: %s\n", strerror (errno)) ;

return 1 ;

}

pinMode(MOTOR\_1\_PWM, OUTPUT);

pinMode(MOTOR\_2\_PWM, OUTPUT);

pinMode(MOTOR\_3\_PWM, OUTPUT);

pinMode(MOTOR\_4\_PWM, OUTPUT);

digitalWrite(MOTOR\_1\_PWM, 0);

digitalWrite(MOTOR\_2\_PWM, 0);

digitalWrite(MOTOR\_3\_PWM, 0);

digitalWrite(MOTOR\_4\_PWM, 0);

enable();

DCMotorInit(1);

DCMotorInit(2);

DCMotorInit(3);

DCMotorInit(4);

GtkBuilder \*builder;

GtkWidget \*window;

gtk\_init(&argc, &argv);

builder = gtk\_builder\_new();

gtk\_builder\_add\_from\_file (builder, "window\_main.glade", NULL);

window = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "window\_main"));

gtk\_builder\_connect\_signals(builder, NULL);

// get pointers

g\_lbl\_hello = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "lbl\_hello"));

g\_lbl\_count = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "lbl\_count"));

g\_lbl\_sonar = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "lbl\_sonar"));

g\_btn\_sonar = GTK\_TOGGLE\_BUTTON(gtk\_builder\_get\_object(builder, "btn\_sonar"));

g\_btn\_auto = GTK\_TOGGLE\_BUTTON(gtk\_builder\_get\_object(builder, "btn\_auto"));

g\_entry\_block = GTK\_ENTRY(gtk\_builder\_get\_object(builder,"entry\_block"));

g\_entry\_turn = GTK\_ENTRY(gtk\_builder\_get\_object(builder,"entry\_turn"));

g\_object\_unref(builder);

gtk\_widget\_show(window);

gtk\_main();

return 0;

}

//manual drive

void btn\_right\_button\_press\_event\_cb(){

rright();

}

void btn\_right\_button\_release\_event\_cb(){

stop();

}

void btn\_back\_button\_press\_event\_cb(){

bbackwd();

}

void btn\_back\_button\_release\_event\_cb(){

stop();

}

void btn\_left\_button\_press\_event\_cb(){

lleft();

}

void btn\_left\_button\_release\_event\_cb(){

stop();

}

void btn\_forward\_button\_press\_event\_cb(){

fforwd();

}

void btn\_forward\_button\_release\_event\_cb(){

stop();

}

void on\_btn\_stopauto\_clicked(){

stop();

}

void updatesonarlabel(int distance){

gchar \*sonardistance;

sonardistance = g\_strdup\_printf("%d", distance);

gtk\_label\_set\_text (GTK\_LABEL(g\_lbl\_sonar), sonardistance);

g\_free(sonardistance);

}

//autopilot algorithm

void autopilot(){

int clockn [12];

const char \*getentry;

getentry = gtk\_entry\_get\_text(GTK\_ENTRY(g\_entry\_block));

int block = atoi(getentry);

int max\_distance = 0;

int direction = 6;

int i;

const char \*getturn;

getturn = gtk\_entry\_get\_text(GTK\_ENTRY(g\_entry\_turn));

double fraction = atof(getturn);

fraction = fraction/12;

fraction = fraction\*1000;

int turn\_degree = 0;

while(gtk\_toggle\_button\_get\_active(g\_btn\_auto)){

fforwd();

//drive forward untill obsticle closer than block(cm)

while (getCM()>=block){}

stop();

delay(500);

lleft();

//turn left untill sonar gets 12 readings

for (i = 1;i<=12;i++) {

delay(fraction);

clockn[i]=getCM();

if (clockn[i] > max\_distance){

if (i!=5 && i!=6 && i!=7){

max\_distance = clockn[i];

direction = i;

}

}

}

stop();

delay(500);

if (direction <= 4){

turn\_degree = fraction \* direction;

left(turn\_degree);

}

else {

turn\_degree = fraction \* (12 - direction);

right(turn\_degree);

}

delay(500);

max\_distance = 0;

direction = 6;

//main iteration, checking for toggle button condition

while (gtk\_events\_pending()) {

gtk\_main\_iteration();

}

}

}

//enable sonar

void on\_btn\_sonar\_toggled(){

int distance;

while(gtk\_toggle\_button\_get\_active(g\_btn\_sonar)){

distance=getCM();

updatesonarlabel(distance);

delay(500);

//main iteration, checking for toggle button condition

while (gtk\_events\_pending()) {

gtk\_main\_iteration();

}

}

}

//autopilot

void on\_btn\_auto\_toggled(){

autopilot();

}

// called when window is closed

void on\_window\_main\_destroy(){

gtk\_main\_quit();}

Б.2 – Код файлу MRPII.h

#ifndef MRPII\_H\_

#define MRPII\_H\_

typedef unsigned char uint8\_t;

int getCM();//activate sonar

void latch\_tx(void);

void enable(); //enable motors;

void DCMotorInit(uint8\_t num);

void DCMotorRun(uint8\_t motornum, uint8\_t cmd);

void left(unsigned int);//turn for set amout of seconds(unsigned int)

void lleft();//turn untill stop()

void right(unsigned int);

void rright();

void backwd(unsigned int);

void bbackwd();

void forwd(unsigned int);

void fforwd();

void stop();//stopping motors

#endif // MRPII\_H\_

Б.3 – Код файлу MRPII.c

#include <stdio.h>

#include <errno.h>

#include <string.h>

#include <unistd.h>

#include <wiringPi.h>

#include <softPwm.h>

#include <stdlib.h>

#include "MRPII.h"

typedef unsigned char uint8\_t;

#define \_BV(bit) (1 << (bit))

#define TRIG 22

#define ECHO 21

#define MOTORLATCH 0

#define MOTORCLK 1

#define MOTORENABLE 2

#define MOTORDATA 3

#define MOTOR\_1\_PWM 4

#define MOTOR\_2\_PWM 5

#define MOTOR\_3\_PWM 6

#define MOTOR\_4\_PWM 7

#define MOTOR1\_A 2

#define MOTOR1\_B 3

#define MOTOR2\_A 1

#define MOTOR2\_B 4

#define MOTOR4\_A 0

#define MOTOR4\_B 6

#define MOTOR3\_A 5

#define MOTOR3\_B 7

#define FORWARD 1

#define BACKWARD 2

#define BRAKE 3

#define RELEASE 4

static unsigned char latch\_state;

void latch\_tx(void)

{

unsigned char i;

digitalWrite(MOTORLATCH, LOW);

digitalWrite(MOTORDATA, LOW);

for (i=0; i<8; i++)

{

digitalWrite(MOTORCLK, LOW);

if (latch\_state & \_BV(7-i))

{

digitalWrite(MOTORDATA, HIGH);

}

else

{

digitalWrite(MOTORDATA, LOW);

}

digitalWrite(MOTORCLK, HIGH);

}

digitalWrite(MOTORLATCH, HIGH);

}

void enable(void)

{

wiringPiSetup();

pinMode(TRIG, OUTPUT);

pinMode(ECHO, INPUT);

pinMode(MOTORLATCH, OUTPUT);

pinMode(MOTORENABLE, OUTPUT);

pinMode(MOTORDATA, OUTPUT);

pinMode(MOTORCLK, OUTPUT);

//TRIG pin must start LOW

digitalWrite(TRIG, LOW);

delay(20);

latch\_state = 0;

latch\_tx();

digitalWrite(MOTORENABLE, LOW);

}

void DCMotorInit(uint8\_t num)

{

switch (num)

{

case 1:

latch\_state &= ~\_BV(MOTOR1\_A) & ~\_BV(MOTOR1\_B);

latch\_tx();

break;

case 2:

latch\_state &= ~\_BV(MOTOR2\_A) & ~\_BV(MOTOR2\_B);

latch\_tx();

break;

case 3:

latch\_state &= ~\_BV(MOTOR3\_A) & ~\_BV(MOTOR3\_B);

latch\_tx();

break;

case 4:

latch\_state &= ~\_BV(MOTOR4\_A) & ~\_BV(MOTOR4\_B);

latch\_tx();

break;

}

}

void DCMotorRun(uint8\_t motornum, uint8\_t cmd)

{

uint8\_t a, b;

switch (motornum)

{

case 1:

a = MOTOR1\_A; b = MOTOR1\_B;

break;

case 2:

a = MOTOR2\_A; b = MOTOR2\_B;

break;

case 3:

a = MOTOR3\_A; b = MOTOR3\_B;

break;

case 4:

a = MOTOR4\_A; b = MOTOR4\_B;

break;

default:

return;

}

switch (cmd)

{

case FORWARD:

latch\_state |= \_BV(a);

latch\_state &= ~\_BV(b);

latch\_tx();

break;

case BACKWARD:

latch\_state &= ~\_BV(a);

latch\_state |= \_BV(b);

latch\_tx();

break;

case RELEASE:

latch\_state &= ~\_BV(a);

latch\_state &= ~\_BV(b);

latch\_tx();

break;

}

}

int getCM() {

//delay(200);

//Send trig pulse

digitalWrite(TRIG, HIGH);

delayMicroseconds(60);

digitalWrite(TRIG, LOW);

//Wait for echo start

while(digitalRead(ECHO) == LOW);

//Wait for echo end

long startTime = micros();

while(digitalRead(ECHO) == HIGH);

long travelTime = micros() - startTime;

//Get distance in cm

int distance = travelTime / 58;

// printf("Distance: %dcm\n", distance);

return distance;

}

void left(unsigned int t)

{

DCMotorRun(2, BACKWARD);

digitalWrite(MOTOR\_2\_PWM, 1);

DCMotorRun(3, FORWARD);

digitalWrite(MOTOR\_3\_PWM, 1);

DCMotorRun(1, BACKWARD);

digitalWrite(MOTOR\_1\_PWM, 1);

DCMotorRun(4, FORWARD);

digitalWrite(MOTOR\_4\_PWM, 1);

delay(t);

DCMotorRun(1, RELEASE);

DCMotorRun(2, RELEASE);

DCMotorRun(3, RELEASE);

DCMotorRun(4, RELEASE);

}

void lleft()

{

DCMotorRun(2, BACKWARD);

digitalWrite(MOTOR\_2\_PWM, 1);

DCMotorRun(3, FORWARD);

digitalWrite(MOTOR\_3\_PWM, 1);

DCMotorRun(1, BACKWARD);

digitalWrite(MOTOR\_1\_PWM, 1);

DCMotorRun(4, FORWARD);

digitalWrite(MOTOR\_4\_PWM, 1);

}

void right(unsigned int t)

{

DCMotorRun(1, FORWARD);

digitalWrite(MOTOR\_1\_PWM, 1);

DCMotorRun(2, FORWARD);

digitalWrite(MOTOR\_2\_PWM, 1);

DCMotorRun(3, BACKWARD);

digitalWrite(MOTOR\_3\_PWM, 1);

DCMotorRun(4, BACKWARD);

digitalWrite(MOTOR\_4\_PWM, 1);

delay(t);

DCMotorRun(1, RELEASE);

DCMotorRun(2, RELEASE);

DCMotorRun(3, RELEASE);

DCMotorRun(4, RELEASE);

}

void rright()

{

DCMotorRun(1, FORWARD);

digitalWrite(MOTOR\_1\_PWM, 1);

DCMotorRun(2, FORWARD);

digitalWrite(MOTOR\_2\_PWM, 1);

DCMotorRun(3, BACKWARD);

digitalWrite(MOTOR\_3\_PWM, 1);

DCMotorRun(4, BACKWARD);

digitalWrite(MOTOR\_4\_PWM, 1);

}

void backwd(unsigned int t)

{

DCMotorRun(1, FORWARD);

digitalWrite(MOTOR\_1\_PWM, 1);

DCMotorRun(2, BACKWARD);

digitalWrite(MOTOR\_2\_PWM, 1);

DCMotorRun(3, FORWARD);

digitalWrite(MOTOR\_3\_PWM, 1);

DCMotorRun(4, BACKWARD);

digitalWrite(MOTOR\_4\_PWM, 1);

sleep(t);

DCMotorRun(1, RELEASE);

DCMotorRun(2, RELEASE);

DCMotorRun(3, RELEASE);

DCMotorRun(4, RELEASE);

sleep(1);

}

void bbackwd()

{

DCMotorRun(1, FORWARD);

digitalWrite(MOTOR\_1\_PWM, 1);

DCMotorRun(2, BACKWARD);

digitalWrite(MOTOR\_2\_PWM, 1);

DCMotorRun(3, FORWARD);

digitalWrite(MOTOR\_3\_PWM, 1);

DCMotorRun(4, BACKWARD);

digitalWrite(MOTOR\_4\_PWM, 1);

}

void forwd(unsigned int t)

{

DCMotorRun(1, BACKWARD);

digitalWrite(MOTOR\_1\_PWM, 1);

DCMotorRun(2, FORWARD);

digitalWrite(MOTOR\_2\_PWM, 1);

DCMotorRun(3, BACKWARD);

digitalWrite(MOTOR\_3\_PWM, 1);

DCMotorRun(4, FORWARD);

digitalWrite(MOTOR\_4\_PWM, 1);

sleep(t);

DCMotorRun(1, RELEASE);

DCMotorRun(2, RELEASE);

DCMotorRun(3, RELEASE);

DCMotorRun(4, RELEASE);

sleep(1);

}

void fforwd()

{

DCMotorRun(1, BACKWARD);

digitalWrite(MOTOR\_1\_PWM, 1);

DCMotorRun(2, FORWARD);

digitalWrite(MOTOR\_2\_PWM, 1);

DCMotorRun(3, BACKWARD);

digitalWrite(MOTOR\_3\_PWM, 1);

DCMotorRun(4, FORWARD);

digitalWrite(MOTOR\_4\_PWM, 1);

}

void stop()

{

DCMotorRun(1, RELEASE);

DCMotorRun(2, RELEASE);

DCMotorRun(3, RELEASE);

DCMotorRun(4, RELEASE);

}