Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет прикладних інформаційних технологій та електроінженерії
(повна назва факультету)
Кафедра комп'ютерно-інтегрованих технологій
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розроблення роботизованого транспортного засобу з Bluetooth керуванням

Виконав: студент IV курсу, групи КТс-41

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

(підпис) Косенко В.О. (прізвище та ініціали)

Керівник (підпис) Тотосько О.В. (прізвище та ініціали)

Нормоконтроль (підпис) (прізвище та ініціали)

Завідувач кафедри (підпис) Микитишин А.Г. (прізвище та ініціали)

Рецензент (підпис) (прізвище та ініціали)

Тернопіль
2024

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет Факультет прикладних інформаційних технологій та електроінженерії
(повна назва факультету)

Кафедра комп'ютерно-інтегрованих технологій
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Микитишин А.Г.
(підпис) (прізвище та ініціали)
«____» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавр
(назва освітнього ступеня)

за спеціальністю 151 Автоматизація та комп'ютерно-інтегровані технології
(шифр і назва спеціальності)

Студенту Косенку Владиславу Олеговичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення роботизованого транспортного засобу з Bluetooth керуванням

Керівник роботи Тотосько Олег Васильович, к.т.н., доцент кафедри КТ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «17» травня 2024 року № 4/7-515

2. Термін подання студентом завершеної роботи 21 червня 2024р.

3. Вихідні дані до роботи Технічні вимоги щодо роботизованого транспортного засобу

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ
Аналітична частина
Проєктна частина
Спеціальна частина
Безпека життєдіяльності, основи охорони праці
Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Титульний слайд
Актуальність роботи
Завдання роботи
Основна частина
Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Сенчишин В.С., доцент кафедри МТ		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	20.05.2024	*Виконано*
2.	Підбір джерел по темі роботи	22.05.2024-24.05.2024	*Виконано*
3.	Опрацювання публікацій та збір даних по	24.05.2024-27.05.2024	*Виконано*
	темі роботи		
4.	Виконання роботи згідно мети	27.05.2024-29.05.2024	*Виконано*
5.	Оформлення першого та другого розділів	29.05.2024-31.05.2024	*Виконано*
6.	Оформлення третього розділу	31.05.2024-11.06.2024	*Виконано*
7.	Оформлення четвертого розділу	11.06.2024-13.06.2024	*Виконано*
8.		14.06.2024-15.06.2024	*Виконано*
9.		16.06.2024-17.06.2024	*Виконано*
10.	Оформлення кваліфікаційної роботи	18.06.2024-19.06.2024	*Виконано*
11.	Нормоконтроль	19.06.2024-20.06.2024	*Виконано*
12.	Перевірка на плагіат	21.06.2024	*Виконано*
13.	Попередній захист кваліфікаційної роботи	21.06.2024	*Виконано*
14.	Захист кваліфікаційної роботи		

Студент _____ Косенко В.О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Тотосько О.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота магістра складається з пояснювальної записки та графічної частини (ілюстративний матеріал – слайди).

Об’єм графічної частини дипломної роботи становить _____.

Об’єм пояснювальної записки складає ___ друкованих сторінок формату А4.

В роботі використано ___ літературних джерел.

У роботі було проведено розробку автоматизованої системи керування транспортним засобом з блютуз керування.

Підсумовуючи, створено основу роботизованої платформи для підключених і автоматизованих транспортних засобів. Платформа наразі може використовувати 14 функціональних автомобілів-роботів, здатних переміщатися по ланцюгах, дотримуючись чорної лінії на білій землі, вимірювати відстань до автомобіля попереду та контролювати їх швидкість до 50 см/с. Ними також можна дистанційно керувати з комп’ютера, реалізуючи зв’язок по блютузу.

Для створення різних схем на платформі є велика біла складна вінілова стрічка та чорна вінілова стрічка, тому її можна згортати та розгортати в різних місцях.

Ключові слова: КОНТРОЛЕР, АВТОМОБІЛЬ, БЛЮТУЗ, АДРУЇНО, АВТОМАТИЧНИЙ КОНТРОЛЬ.

ЗМІСТ

ВСТУП

Автономні автомобілі, також відомі як безпілотні, самокеровані або роботизовані автомобілі, — це транспортні засоби, здатні виконувати транспортні можливості традиційного автомобіля без участі людини.

Багато автомобілів на дорогах сьогодні мають електронні системи, які допомагають водіям. Починаючи від досить старих, як-от «Антиблокувальна гальмівна система» (ABS) для оптимізації гальмівного шляху в разі надзвичайної ситуації, круїз-контроль тощо. До більш сучасних, як-от виявлення зміни колії, допоміжний круїз-контроль або автоматичне екстрене гальмування під час виявлення перешкоди. Навіть Tesla нещодавно оновила свою систему «Автопілот», щоб дозволити своїм автомобілям автономно їздити по магістралях, включаючи зміну смуги руху, коли це необхідно.

В роботі було розроблено прототип автономної системи водіння з можливістю корекції руху та ручного керування за допомогою блютуз керування.

Однак усі ці системи потребують уважності водія за кермом, щоб у разі потреби вжити заходів. Є багато несподіваних речей, які можуть раптово піти не так, і навіть якщо вони дуже малоймовірні, вони стануться в певний момент, враховуючи кількість автомобілів, що їздять навколо. Автомобіль без водія повинен бути в змозі впоратися з усіма цими ситуаціями швидко та оптимальним чином.

Саме тому безпілотні автомобілі не стануть модернізацією існуючих систем допоміжного водіння, а стануть цілком новою концепцією. Google, наприклад, роками працює над автономними автомобілями, навіть проїжджаючи тисячі миль реальними дорогами майже без жодних аварій; і їхня концепція автомобіля без водія навіть не має керма, що більше нагадує концепцію транспортного засобу «в стилі ліфта», який доставляє вас із пункту А в пункт Б, ніж звичайний автомобіль, який просто керує собою.

1. АНАЛІТИЧНА ЧАСТИНА

1.1 Роботизовані транспортні засоби без водія

Окрім звичайних датчиків, які вже є у багатьох автомобілях, ці нові концептуальні транспортні засоби потребуватимуть технології, здатної повністю контролювати оточення в режимі реального часу, адаптуючи їхні карти та місцезнаходження на ходу. Основна технологія, що використовується для цього, називається LIDAR, яка заснована на лазерному світлі, яке використовується як радар, щоб дуже точно складати карту навколишнього середовища. Завдяки цьому та комп'ютерному зору для розпізнавання світлофорів і сигналів можна запрограмувати програмне забезпечення, яке керуватиме автомобілем ефективніше та безпечніше, ніж люди-водії.

Важливо зазначити, що це програмне забезпечення не буде типовим програмним забезпеченням із умовами «якщо» та «поки», які зустрічаються на сучасних комп'ютерах. Оскільки йому потрібно реагувати на багато різних обставин, деякі з яких є новими для платформи, використаний підхід базується на машинному навчанні. Програмне забезпечення автомобілів навчатиметься за імовірнісними правилами під час водіння, наприклад, уникати перешкоди на дорозі так чи інакше, залежно від того, чи це м'яч, собака чи людина, перетин; або зниження швидкості при виявленні певних моделей, які зазвичай пов'язані з дітьми, які граються навколо. Однак таке програмне забезпечення потребує тривалого навчання та тестування в максимально реальних умовах. Ось де роботизовані платформи, подібні до тієї, що побудована в цій дипломній роботі, можуть допомогти отримати дані, спробувати функції та виявити можливі проблеми.

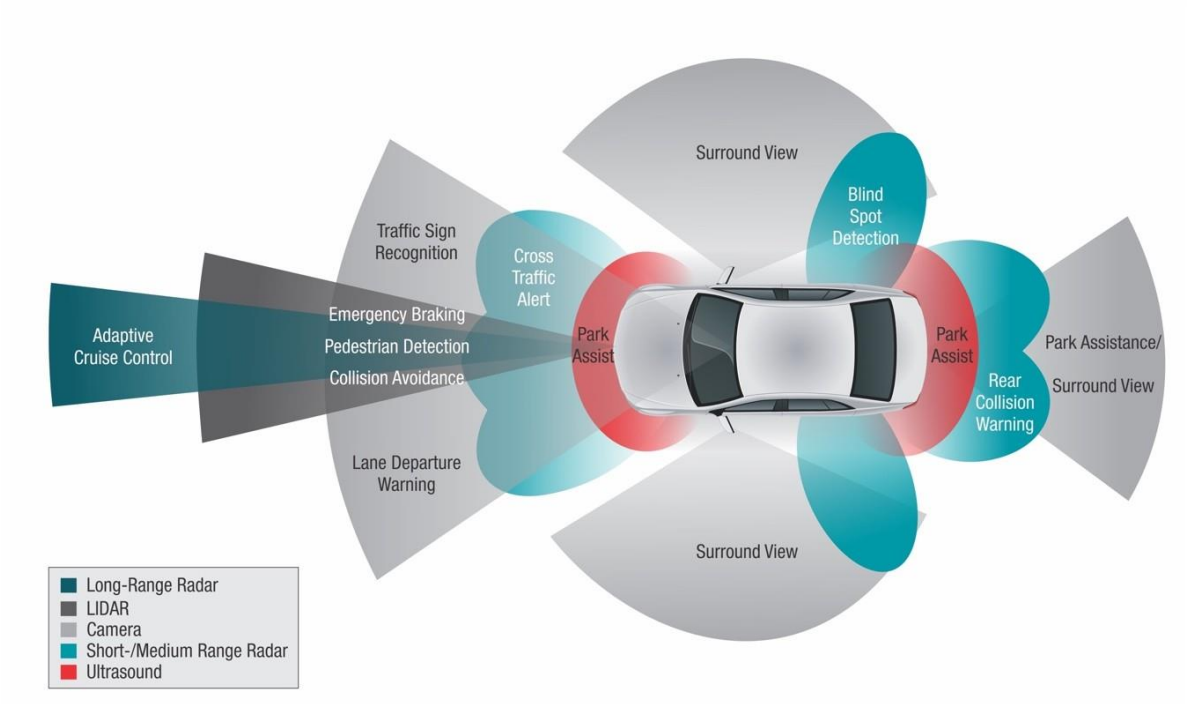


Рисунок 1.1 - Сучасні технології для допомоги водієві.

Наявність великої кількості автомобілів, які автономно керуються бортовими комп’ютерами, також відкриває можливість оптимізувати весь транспортний потік за допомогою зв’язку між транспортним засобом (V2V) і транспортним засобом з інфраструктурою (V2I). Завдяки тому, що всі автомобілі об’єднані в однорангову мережу, транспортні засоби можуть використовувати інформацію, зібрану іншими водіями, і краще адаптуватися до раптових подій, таких як затори, аварії, блокування доріг тощо, одночасно координуючи рух для оптимізації транспортного потоку замість кожного окремого транспортного засобу. діючи кожен сам по собі. Крім того, велика мережа транспортних засобів дозволяє передавати повідомлення далеко за межі початкового діапазону випромінювача, перекидаючи його з одного автомобіля на інший.

Основна мета цієї роботи полягає в тому, щоб створити основу тестової платформи з невеликими роботизованими автомобілями, які можуть імітувати основні характеристики справжніх автономних автомобілів; і перевірка надійності і функціональності, щоб випробувати на ньому алгоритми керування трафіком.

1.2 Приклади платформ для тестування автономних автомобілів у світі

Випробувальний комплекс M City

20 липня 2015 року Мічиганський університет відсвяткував урочисте відкриття M City, унікального випробувального центру, де автономні транспортні засоби можна випробувати за багатьох різних обставин, перш ніж вивести їх на справжні дороги. Він складається з 32 акрів місцевості, де вони відтворили ціле міське середовище з перехрестями, світлофорами, сигналами, будівлями тощо. Він містить усілякі елементи та перешкоди, які можуть бути присутніми в реальному світі, від бордюрів, лавок чи гідрантів, до будівельних бар'єрів, різних дорожніх покриттів, змінних умов освітлення або навіть роботів, які представляють людей у цьому районі.

Ця випробувальна установка може бути дуже корисною для автомобілів, щоб безпечно вивчати різні можливі ситуації та небезпеки, але вона набагато більша та складніша, ніж запланований випробувальний стенд для нашого проекту.

Рисунок 1.2 - Тестовий майданчик MCity для автомобілів без водіїв

«Роботизоване міське середовище» від Бостонського університету

Найбільш схоже на платформу, яку планували побудувати, знайшли в Бостонському університеті. Це була платформа, створена кілька років тому, яку

вони назвали «Robotic Urban Like Environment» (RULE). Він складався з ланцюга вулиць, перехресть і світлофорів, по якому рухалася група роботів. Вони використовували його в основному для випробування алгоритмів для оптимізації пошуку паркувальних місць, коли роботи ходили навколо в пошуках вільного місця для паркування.

Вони пояснили, що програмне забезпечення було написано власноруч, на базі Windows, без використання будь-якого спеціального середовища, такого як ROS (операційна система роботів). Роботи мають унікальну комбінацію трьох кольорових кіл у верхній частині, яка використовується основним програмним забезпеченням для отримання інформації про роботів (наприклад, ID, положення та напрямок) через 2 веб-камери, встановлені на стелі.

Світлофори реалізовані за допомогою інфрачервоного світлодіода, який вмикається, коли горить червоне світло. Роботи виявляють це світло своїми інфрачервоними датчиками.

Світлофори спілкуються з головним ПК через XBee, а роботи через WiFi або Bluetooth. Різні елементи можуть спілкуватися безпосередньо між собою, але вони сказали, що зазвичай направляють усі комунікації через ПК.

Нарешті, вони також сказали, що платформа змінювалася протягом багатьох років, і вони щоразу встановлювали різні конфігурації залежно від цікавого додатка. Дошка платформи, яку вони використовують зараз, має розміри 11x8,5 кв. футів (3,4x2,6 м2).

Рисунок 1.3 - Комплекс в Бостонському університеті

Роботи, які використовувала дослідницька група Бостонського університету, являли собою вже створеного робота під назвою Khepera 3, який

мав багато датчиків і можливостей, але був занадто великим і дорогим для того, що передбачалося в нашому проекті.

1.3 Системи автономного керування танспортними засобами

Автономне водіння

Ідея автоматизації транспортних засобів є старою. Люди намагалися автоматизувати автомобілі та літаки з 1930 року, але ажіотаж щодо безпілотних автомобілів різко зріс між 2004 і 2013 роками. Щоб заохочувати технології автономних автомобілів, дослідницький відділ Міністерства оборони США, Агентство перспективних оборонних досліджень (DARPA), створив виклик під назвою Grand DARPA Challenge у 2004 році. Метою завдання було розробити транспортний засіб, який міг би автономно їздити в реальному середовищі. Переможцем змагання стала команда Team Taran Racing з Університету Карнегі-Меллона, а друге місце посіла команда Stanford Racing зі Стенфордського університету.

Виклики DARPA популяризували автономне водіння та підштовхнули автомобільні компанії до роботи над впровадженням можливостей автономного водіння у своїх автомобілях. В даний час майже всі автомобільні компанії намагаються розробити власну модель безпілотного автомобіля. У 2009 році Google почав розробляти власний безпілотний автомобіль, тепер відомий як Waymo, що значно вплинуло на інші компанії, які почали самостійно розробляти автономні автомобілі .

Концепція автономного водіння

Автомобіль, який здатний автономно керувати автомобілем, повинен мати можливість керувати автомобілем без участі людини (Autonomous car, nd). Щоб досягти цього, автономний автомобіль має відчувати навколишнє середовище, орієнтуватися та реагувати без участі людини (Autonomous car, nd). Широкий спектр датчиків, таких як LIDAR, RADAR, GPS, датчики одометрії коліс і камери, використовуються в безпілотних автомобілях для сприйняття навколишнього середовища. Крім того, автономний автомобіль повинен мати

систему керування, яка здатна розуміти дані, отримані від датчиків, і розрізняти дорожні знаки, перешкоди, пішоходів та інші очікувані та несподівані речі на дорозі.

Щоб автомобіль працював автономно, кілька систем реального часу повинні тісно працювати разом. Ці системи реального часу, як визначено Levinson, включають картографування та розуміння середовища, локалізацію, планування маршруту та контроль руху. Щоб ці системи реального часу мали платформу для роботи, сам безпілотний автомобіль має бути оснащений відповідними датчиками, обчислювальним апаратним забезпеченням, мережею та інфраструктурою ПЗ.

Автономне водіння має багато переваг для людства. Захаренко передбачає, що автономний транспорт зменшить вартість подорожей, дозволить дітям подорожувати без присутності дорослих і звільнить людей від тягаря водіння, що все призведе до покращення досвіду подорожі. Захаренко додає, що безпілотні автомобілі будуть безпечнішими, обиратимуть більш оптимальні маршрути та збільшуватимуть пропускну здатність шосе. Захаренко також додає, що автомобілі зможуть самостійно припаркуватися, якщо потрібно, подалі від своїх власників, що зменшило б витрати на паркування. Нарешті, Захаренко зазначає, що інтерес до розробників автономного водіння зростає разом із зростанням індустрії автономного водіння. Однією з цінних навичок, які могли б мати у своїх кишенях розробники автономного водіння, є знання програмування ROS.

Щоб машину можна було назвати роботом, вона повинна задовольняти щонайменше трьом важливим можливостям: бути здатною відчувати, планувати та діяти. Щоб автомобіль називався автономним автомобілем, він має відповідати тим же вимогам. Безпілотні автомобілі — це, по суті, машини-роботи, які можуть приймати рішення про те, як дістатися з пункту А в пункт Б. Пасажиру потрібно лише вказати пункт призначення, і автономний автомобіль повинен мати можливість безпечно доставити його туди. Щоб перетворити звичайний автомобіль на автономний, потрібно додати датчики та бортовий комп’ютер.

Рівні автономності автомобіля

Джозеф виділяє шість рівнів автономії автомобіля:

Рівень автономії 0. Транспортні засоби з рівнем автономності 0 повністю ручні, з людиною-водієм. Більшість старих автомобілів належать до цієї категорії.

Рівень автономії 1. Транспортні засоби з рівнем автономності 1 мають людину-водія, але вони також мають систему допомоги водієві, яка може автоматично керувати двигуном або системами рульового керування, використовуючи інформацію про навколишнє середовище.

Рівень автономії 2. Цей рівень кваліфікується як часткова автоматизація, оскільки на цьому рівні транспортний засіб може виконувати як керування двигуном, так і функції рульового керування, тоді як усі інші завдання контролює водій.

Рівень автономії 3. Цей рівень називається умовною автоматизацією, оскільки на цьому рівні очікується, що всі завдання виконуються автомобілем автономно, хоча також очікується, що людина буде втручатися, коли це буде потрібно.

Рівень автономії 4. Джозеф стверджує, що на цьому рівні немає потреби в людині-водії, оскільки все обробляється автоматизованою системою. Цей рівень називається високою автоматизацією, і такий вид автономної системи працюватиме в заданому типі області за певних погодних умов.

Рівень автономії 5. Цей рівень автономності називається повною автоматизацією, оскільки на цьому рівні все повністю автоматизовано, і автомобіль може працювати автономно на будь-якій дорозі за будь-якої погоди. Джозеф зазначає, що на цьому рівні автоматизації взагалі немає потреби в людині-водії.

Згідно з Hughes, найвищий рівень автономності, який зараз досягають сучасні автомобілі, – це рівень 3. Це означає, що поточна технологія не досягає рівня повної автономності чи високої автономності, хоча є перспективи від великих гравців в автомобільній промисловості, що це зміниться в найближчому майбутньому.

Датчики безпілотних автомобілів

За словами Джозефа, у всіх безпілотних автомобілях повинні бути присутні такі датчики:

Глобальна система позиціонування (GPS). Глобальна система позиціонування використовується для визначення положення безпілотного автомобіля шляхом тріангуляції сигналів, отриманих від супутників GPS. Він часто використовується в поєднанні з даними, зібраними з IMU та одометричного кодера коліс, для більш точного позиціонування та стану автомобіля за допомогою алгоритмів об'єднання датчиків.

Виявлення та визначення дальності світла (LIDAR).Основний датчик безпілотного автомобіля, який вимірює відстань до об'єкта, надсилаючи лазерний сигнал і приймаючи його відображення. Він може надавати точні тривимірні дані навколишнього середовища, обчислені на основі кожного отриманого лазерного сигналу. Безпілотні транспортні засоби використовують LIDAR для картографування навколишнього середовища, виявлення та уникнення перешкод.

Камера.Камера на борту безпілотного автомобіля використовується для виявлення дорожніх знаків, світлофорів, пішоходів тощо за допомогою алгоритмів обробки зображень.

РАДАР.RADAR використовується для тих же цілей, що й LIDAR. Переваги RADAR перед LIDAR полягають у тому, що він легший і має можливість працювати в різних умовах. Хоча він має більшу дальність, усі категорії RADAR мають обмежене поле зору.

Ультразвукові датчики.Ультразвукові датчики відіграють важливу роль у паркуванні безпілотних транспортних засобів, уникненні та виявленні перешкод у сліпих зонах, оскільки їх радіус дії зазвичай становить до 10 метрів.

Датчик одометрії коліс.Енкодери коліс надають дані про обертання коліс автомобіля за секунду. Одометрія використовує ці дані, розраховує швидкість і оцінює положення та швидкість автомобіля на їх основі (Odometry and encoders, 2010). Одометрія часто використовується з даними інших датчиків, щоб точніше визначити положення автомобіля.

Інерційна вимірювальна одиниця (IMU).IMU складається з гіроскопів і акселерометрів, одна пара яких орієнтована на кожну з осей. Ці датчики надають дані про обертальний і лінійний рух автомобіля, які потім використовуються для обчислення руху і положення транспортного засобу незалежно від швидкості чи будь-якого типу перешкод сигналу.

Бортовий комп'ютер.Це основна частина будь-якого безпілотного автомобіля. Як і будь-який комп'ютер, він може мати різну потужність, залежно від того, скільки даних датчиків йому потрібно обробити та наскільки він повинен бути ефективним. Усі датчики підключаються до цього комп'ютера, який має використовувати дані датчиків, розуміючи їх, плануючи маршрут і керуючи виконавчими механізмами автомобіля. Управління здійснюється шляхом надсилання команд керування, таких як кут повороту керма, дросельна заслінка та гальмування, на колеса, двигуни та сервопривід автономного автомобіля.

Блок-схема SW безпілотних транспортних засобів

Рисунок 1.4 ілюструє блок-схему програмного забезпечення стандартного безпілотного автомобіля, представленого Джозефом.

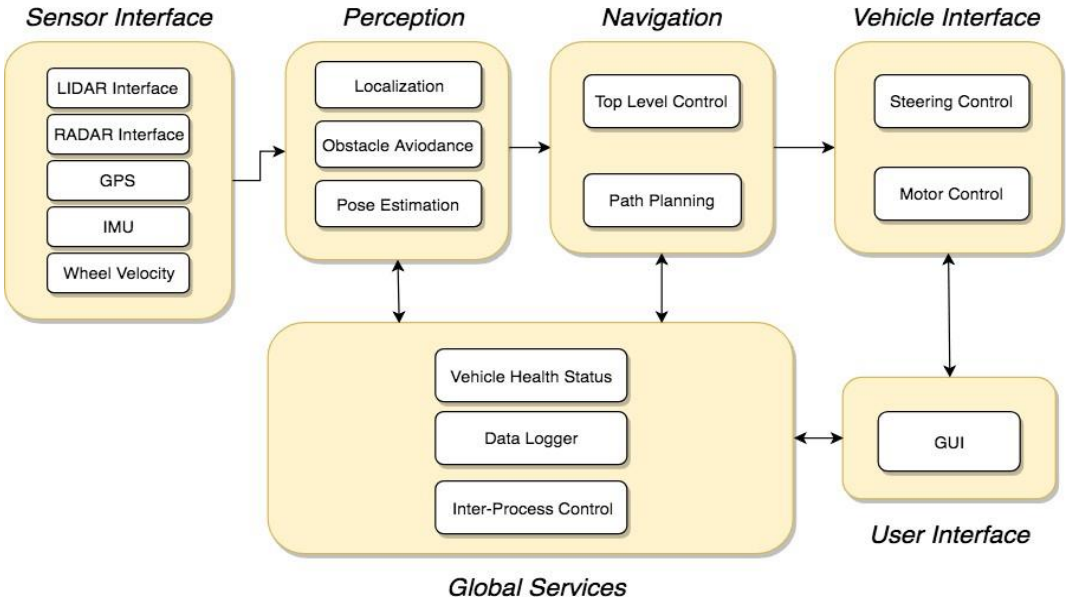


Рисунок 1.4 - SW блок-схема самокерованого автомобіля.

Кожен блок, показаний на рис. 1.4, може взаємодіяти з іншими блоками за допомогою міжпроцесного зв'язку (IPC) або спільної пам'яті, тому система

обміну повідомленнями ROS тут працює дуже добре. Джозеф визначив такі блоки для блок-схеми SW типового безпілотного автомобіля:

Інтерфейсні модулі датчиків. Весь зв'язок між датчиками та автомобілем здійснюється в цьому блоці, оскільки він дає змогу обмінюватися даними, отриманими від датчиків, з іншими блоками. Основними датчиками, з яких збираються дані в цьому блоці, є: LIDAR, RADAR, GPS, IMU, камери та колісні кодери.

Модулі сприйняття. Ці модулі обробляють дані сприйняття від датчиків, таких як LIDAR, RADAR і камери, а потім сегментують оброблені дані, щоб визначити місцезнаходження різних об'єктів, які залишаються нерухомими або рухаються. Ці модулі також допомагають локалізувати самокерований автомобіль відносно створеної карти навколишнього середовища.

Навігаційні модулі. Навігаційні модулі визначають поведінку безпілотного автомобіля, оскільки мають планувальники маршруту та руху, а також станковий автомат поведінки автомобіля. Щоб створити найоптимальніший маршрут для автомобіля з пункту А в пункт Б, навігаційні модулі спілкуються з модулями сприйняття.

Інтерфейс автомобіля. Метою цього інтерфейсу є надсилання керуючих команд, таких як кермо, дросель і гальмування, до автомобіля після того, як шлях буде нанесено на навігаційний модуль.

Інтерфейс користувача. Інтерфейс користувача надає елементи керування, які можуть бути пов'язані з установкою пункту призначення або переглядом карти. Як правило, кнопка аварійної зупинки також має бути доступною для користувача.

Глобальні послуги. Модуль глобальних послуг контролює надійність програмного забезпечення автомобіля, дозволяючи реєструвати дані датчиків автомобіля та відмітки часу.

Операційна система робота (ROS)

ROS — це не справжня операційна система, а скоріше метаопераційна система. Простіше кажучи, ROS працює поверх іншої операційної системиі дозволяє різним процесам спілкуватися один з одним під час виконання. Як

метаопераційна система, ROS пропонує структурований комунікаційний рівень, який працює поверх операційних систем хост-комп'ютерів. Використання ROS не обмежується лише робототехнікою, оскільки більшість інструментів ROS сумісні з периферійним обладнанням і можуть використовуватися для різних цілей. Ядро ROS складається з більш ніж двох тисяч пакетів, де кожен пакет має свою особливу функціональність.

Нарешті, ROS — це набір інструментів, який забезпечує функціональність і послуги операційної системи на одному чи кількох комп'ютерах. Ці послуги включають абстракцію апаратного забезпечення, обмін повідомленнями між процесами, керування пакетами тощо. Ademovic стверджує, що найбільшою перевагою ROS є кількість доступних інструментів ROS.

Існує багато різних областей програмування робототехніки, і для однієї людини чи команди практично неможливо знати, як вирішити всі проблеми програмування робототехніки. Через це групам програмістів робототехніки та лабораторіям необхідно співпрацювати одна з одною для створення складних рішень для робототехніки. ROS є зручним, тому що створення надійного програмного забезпечення робота є надто вимогливим для одного інженера. Основна ідея ROS полягає в тому, щоб розробити метаопераційну систему, яка дозволить людям співпрацювати в розробці рішень робототехніки, використовуючи стандартизований спосіб програмування, обмінюватися стандартизованими повідомленнями, використовувати повторно використовувані та масштабовані модулі програмування. Таким чином, ROS було створено з метою, щоб різні групи інженерів могли співпрацювати, співпрацювати та створювати свої рішення на основі роботи один одного.

Одна дуже важлива характеристика ROS полягає в тому, що він повністю відкритий. ROS було розроблено з метою повторного використання роботизованого ПЗ. Про це стверджує Quigley, що писати ПЗ для роботів важко, особливо в міру того, як зростає масштаб і сфера застосування робототехніки. Різні типи роботів можуть мати дуже різне апаратне забезпечення, що робить повторне використання коду надзвичайно складним. Крім того, програмне

забезпечення робота тісно пов’язане, і видобування багаторазового коду може бути дуже складним, і ROS створено для подолання цих проблем.

Щоб пояснити ROS, Quigley узагальнив філософські цілі ROS, використовуючи такі п’ять тверджень:

ROS є рівноправним.Вузли ROS — це одиниці виконання, які взаємодіють один з одним безпосередньо або через механізми публікації/підписки.

ROS базується на інструментах.Він використовує дизайн мікроядра та кілька невеликих інструментів і модулів.

ROS є багатомовним.Він підтримує C++, Python, Octave та LISP.

ROS – це «тонкий».Він використовує систему побудови catkin для забезпечення сегментації коду за пакетами та бібліотеками.

ROS є безкоштовним і відкритим кодом.ROS є загальнодоступним за ліцензією BSD.

2 ПРОЄКТНА ЧАСТИНА

2.1 Проєктування платформи для реалізації автомобілів-роботов

Реалізація функцій автоматизованих транспортних засобів на платформі робототехніки

Першим етапом створення випробувальної лабораторії було обдумати різні технології та платформи для використання, щоб відтворити найцікавіші функції реальних автомобілів на невеликій площі.

Роботи Arduino

Arduino було обрано як найкращу платформу для основного керування роботами. Arduino є дуже популярною платформою з пристойною обчислювальною потужністю, хорошою ціною та багатьма можливостями оновлення через кількість апаратного забезпечення, сумісного з нею. Були знайдені різні варіанти:

а) Офіційний робот Arduino: дуже простий, але з великим простором для додавання датчиків і опцій. Не дуже хороші відгуки. [1]

Ціна: 200$ + комплект датчиків 60$

б) Parallax BoeBot шилд робота з Arduino: сумісна з Arduino версія дуже популярного комплекту роботів. Багато аксесуарів від основної компанії-виробника. [2]

Ціна: 150$ + 80$ (сенсори для вимірювання відстані та стеження за лінією) + 60$ (Bluetooth модуль).

в) Зуморобот для Arduino: Добре побудований робот із хорошими двигунами та багатьма датчиками (датчик відбиття для визначення лінії, 3-осьовий акселерометр, гіроскоп, компас…). Можливість додавання датчика відстані та іншого обладнання. [3]

Ціна: 100$ + 25$ (Arduino Leonardo) + 30$ (Bluetooth модуль)

г) Rovera 2WD ArduinoРоботизований комплект: Набір робота Arduino, який поставляється з уже вбудованим датчиком відстані та датчиком відбиття для відстеження лінії. Разом із ІЧ-датчиком для керування ним з будь-якого пульта дистанційного керування телевізором. [4]

Ціна: 170$

д) Спаркі: новий робот із великою кількістю вбудованих датчиків. Він оснащений вимірюванням відстані, відстеженням лінії та визначенням країв, датчиком світла, акселерометром, магнітометром, дистанційним керуванням і навіть модулем Bluetooth. Однак це здається дещо складнішим для оновлення, а програмне середовище не таке стандартне. [5]

Ціна: 160$

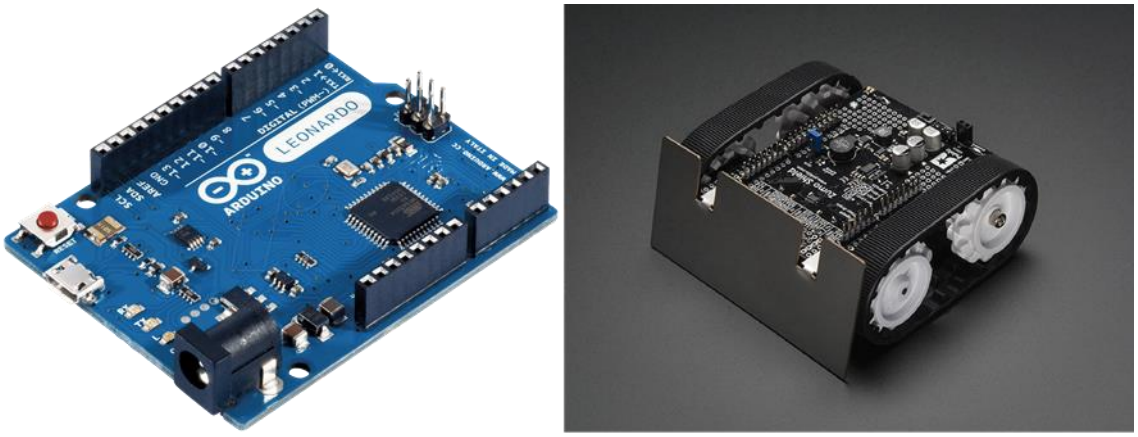


Рисунок 2.1 - Плата Arduino Leonardo та робот Zumo для Arduino shield.

Датчики та технології виявлення

Впровадження системи LIDAR у невеликих роботах для моніторингу навколишнього середовища було б дуже дорогим і складним. Крім того, основною метою платформи для тестування роботизованих автомобілів було проведення досліджень управління дорожнім рухом, тому не було необхідності контролювати оточення в режимі реального часу, шукаючи пішоходів або інші раптові проблеми. Таким чином, було вирішено обмежити кількість необхідних технологій, зменшивши платформу до «двовимірного» середовища. Таким чином, роботизовані автомобілі можуть

слідувати по доріжці та взаємодіяти з іншими роботами, сигналами тощо на цій доріжці.

Щоб створити доріжку, слідування за лінією здавалося найкращим варіантом, оскільки за допомогою простого набору світлових датчиків можна запрограмувати робота слідувати за чорною лінією, намальованою на білій поверхні.

Щоб реалізувати функцію відстеження автомобіля та виявлення сигналу, було розглянуто багато варіантів. Ультразвукові або інфрачервоні датчики відстані досить точні, коли намагаються виявити об’єкти та стежити за ними, але їм бракує можливості розрізнити тип об’єкта, який видно. Оскільки LIDAR також було відкинуто через те, що він надто дорогий і складний у реалізації, комп’ютерний зір став збалансованим варіантом. Точніше, було знайдено датчик під назвою Піху, який, здавалося, забезпечує всі необхідні функції.

CMUCam5 (Піху) [6] — це камера, інтегрована з чіпом, який може розпізнавати та відстежувати різні об’єкти за їх кольорами. Він також виконує всі обчислення та лише надсилає корисну інформацію (кількість і тип об’єктів, розташування тощо) на плату Arduino. Це робиться дуже швидко (кажуть, що аналізується 50 кадрів за секунду), тому його можна використовувати для відстеження об'єктів. Його також можна використовувати для виявлення об’єктів за допомогою кольорових кодів, тому його можна реалізувати для виявлення світлофорів або інших елементів руху на робочому столі.

Цю саму камеру або подібну можна використовувати для впровадження системи комп’ютерного бачення на стелі лабораторії, щоб контролювати, позиціонувати та контролювати роботизовані транспортні засоби. Як і створення внутрішньої системи позиціонування.

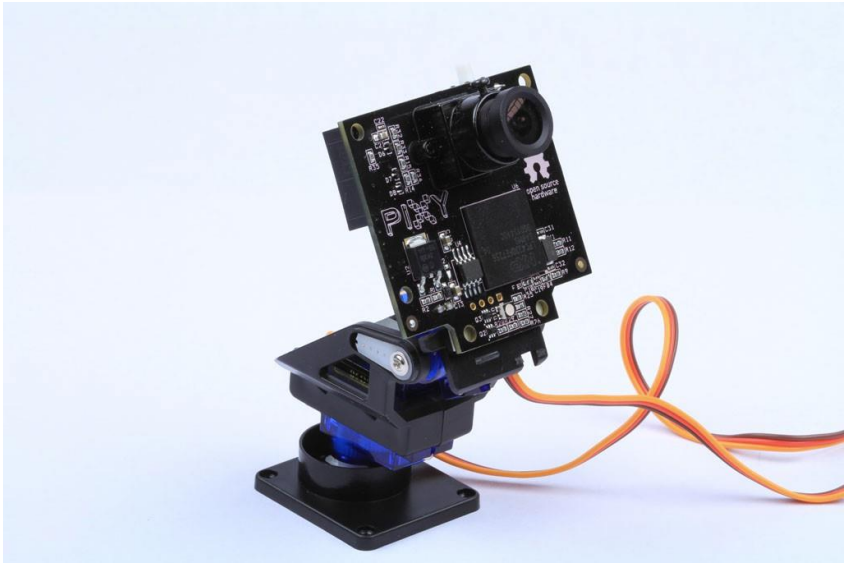


Рисунок 2.2 - Pixy Cam з набором повороту/нахилу.

Бездротовий зв'язок

Щодо бездротового зв’язку, основними необхідними характеристиками були висока швидкість і частота оновлення та низьке енергоспоживання, але кількох метрів діапазону вже було достатньо. Найпоширеніша бездротова технологія, WiFi, здавалася надто потужною для розміру роботів і цільового використання. Bluetooth здавався кращим варіантом із зменшеним розміром і діапазоном, але все ще з досить високим споживанням енергії. Нарешті, технологія під назвою XBee була визнана дуже придатною для роботизованих додатків, оскільки вона споживає менше енергії, ніж Bluetooth, з аналогічним радіусом дії та кращими функціями зв’язку між кількома пристроями (дозволяє використовувати більше вузлів у мережі).

Існує два типи пристроїв XBee, Series 1 і Series 2. Модулі Series 1 працюють із коробки в конфігурації точка-багато точок; Серія 2 потребує додаткових налаштувань, але вони працюють безпосередньо в конфігурації «сітки» (дозволяючи досягати пристроїв поза зоною дії, пропускаючи повідомлення через інші вузли в мережі). Після деяких міркувань для платформи роботизованих автомобілів було обрано XBee Series 1, оскільки його асортимент охоплює майже всю настройку, а за допомогою таких

пристроїв базове керування та зв’язок від комп’ютера до всіх роботів можна здійснювати дуже легко. Якщо в майбутньому знадобиться більш складний зв’язок (наприклад, розмова з певним автомобілем з комп’ютера або розмова безпосередньо від одного конкретного транспортного засобу до іншого), програмне забезпечення модулів можна змінити на платформу під назвою ZigBee, яка надає їм можливості, подібні до модуля XBee Series 2.

Щоб приєднати модуль XBee до робота Zumo, потрібно використовувати XBee Arduino shield плюс роз’єми. [7] [8]

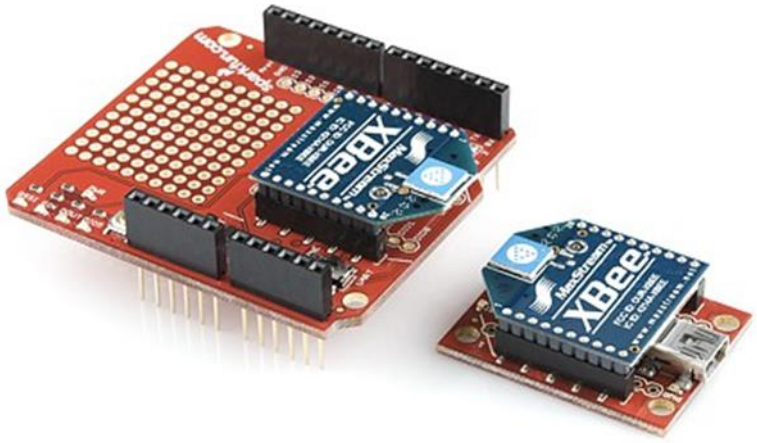


Рисунок 2.3 - Модуль XBee з екраном XBee для Arduino та USB-адаптером для комп’ютера.

2.2 Обрана платформа

Після всіх попередніх міркувань було обрано платформу Zumo Robot для Arduino + Pixy CMUCam-5 + XBee Series 1. Загальна ціна кожного робота з Pixy Cam і бездротовим модулем буде близько 270 доларів.

Основні переваги робота Zumo для роботизованої платформи Arduino + Pixy Cam:

+ Можна використовувати з акумуляторними батареями, які можна заряджати безпосередньо зсередини робота.

+ вбудований 3-осьовий акселерометр + магнітометр + гіроскоп.

+ Хороша адаптивна швидкість.

+ Хороші можливості стеження за лінією.

+ Можливість додати ІЧ або ультразвуковий датчик для виявлення перешкод.

+ Може відстежувати об'єкти за допомогою Pixy Cam.

+ Можна оцінити відстань до об'єкта за допомогою Pixy Cam (за зміною розміру).

+ Можна зберігати та завантажувати параметри Pixy (щоб мати однакові у всіх роботів).

+ Легке впровадження дистанційного керування та зв'язку V2I з XBee Series 1.

+ Гарна ціна.

Основні недоліки робота Zumo для роботизованої платформи Arduino + Pixy Cam:

- Відстань до об'єкта та пройдену відстань вимірюють не дуже точно.
- Відсутність енкодерів в колесах для отримання зворотного зв'язку швидкості.
- Потрібна деяка збірка (Pixy Cam і адаптер бездротового зв’язку).
- Потрібна зміна мікропрограми в модулі XBee для реалізації зв’язку V2V.
- Ескалація до реальних транспортних засобів

Щоб порівняти швидкість і дані, зібрані роботами, з даними реальних транспортних засобів, необхідно визначити коефіцієнт ескалації. Цей коефіцієнт було вибрано рівним 50, оскільки довжина роботів Zumo становить 10 см, що еквівалентно 50-метровому автомобілю, що є досить середньою довжиною для сучасних легкових автомобілів.

Використовуючи цей коефіцієнт ескалації, ми можемо конвертувати швидкості роботів у реальні швидкості автомобіля. Теоретична максимальна

швидкість у роботів становить близько 60 см/с, але під час слідування по лінії та в наших підлогових умовах ми досягли максимальної швидкості близько 45 см/с. Застосувавши коефіцієнт ескалації 50, це еквівалентно 81 км/год (50 миль/год), пристойна швидкість для міського середовища.

Цей коефіцієнт ескалації, однак, не можна безпосередньо застосувати до маси автомобіля. Наші роботи з аксесуарами та акумуляторами важать близько 400 грамів, що, помножене на 50, дасть 20-кілограмовий автомобіль у реальному розмірі. Це означає, що ми не можемо розраховувати на інерцію транспортного засобу, щоб імітувати умови прискорення та гальмування реальних автомобілів, оскільки ці маленькі роботи можуть зупинятися або застосовувати значні зміни у своїй швидкості майже в реальному часі без дрейфу. Таким чином, потрібно буде впроваджувати моделі програмного забезпечення, які імітують умови, які ми хочемо відтворити. Основною моделлю, яка буде використовуватися для відтворення водіння людиною, є так звана «модель оптимальної швидкості».

2.3 Модель оптимальної швидкості

Щоб відтворити поведінку реальних водіїв, модель оптимальної швидкості використовується для розрахунку бажаної швидкості транспортного засобу в кожен момент на основі відстані з попереднім транспортним засобом.

Ця модель спочатку розраховує швидкість, необхідну для досягнення попереднього транспортного засобу за певний час (константа Тау, яка на практиці визначатиме відстань, яку залишив попередній транспортний засіб). Потім він обчислює прискорення, необхідне для досягнення цієї швидкості за інший заданий час (постійна Т, яка становить близько 1,2 секунди в поведінці водія). Нарешті, він розраховує швидкість, яку потрібно застосувати через

певний час (типовий час реакції людей під час водіння), враховуючи поточну швидкість і розраховане прискорення.

Змінюючи параметр Tau, можна дозволити більші або менші відстані між транспортними засобами, щоб імітувати справжнє водіння людини або збільшити щільність транспортних засобів завдяки швидкій реакції електроніки та роботів. У таблиці 1 показано деякі відстані з різною швидкістю залежно від використовуваного Tau:

Таблиця 2.1 - Відстань із попереднім транспортним засобом для різних Tau та швидкості [довжина автомобіля].

		швидкість		
		25 км/год (16 миль/год)	**50 км/год (31 миль/год)**	**80 км/год (50 миль/год)**
Tau	**0,5**	0,7	1.4	2.3
	1	1.4	2.8	4.5
	1.6	2.2	4.5	7.1
	2	2.8	5.5	8.9

Параметр T також можна змінювати, впливаючи на те, як швидко автомобіль-робот розганяється або гальмує. Отже, чим менше значення, тим швидше робот зможе адаптуватися до мінливих умов і тим точніше буде імітувати рухи попереднього автомобіля.

Використовуючи Matlab, можна виконати деякі моделювання поведінки роботів з різними значеннями T. Зі стандартним T 1,2 для імітації поведінки людини-водія роботам-транспортним засобам потрібно приблизно

5 секунд, щоб досягти максимальної швидкості 45 см/с (еквівалентно 80 км/год із попередньо визначеною ескалацією). Змінюючи значення T, ми можемо отримати різні швидкості прискорення, як це видно на наступних рисунках.

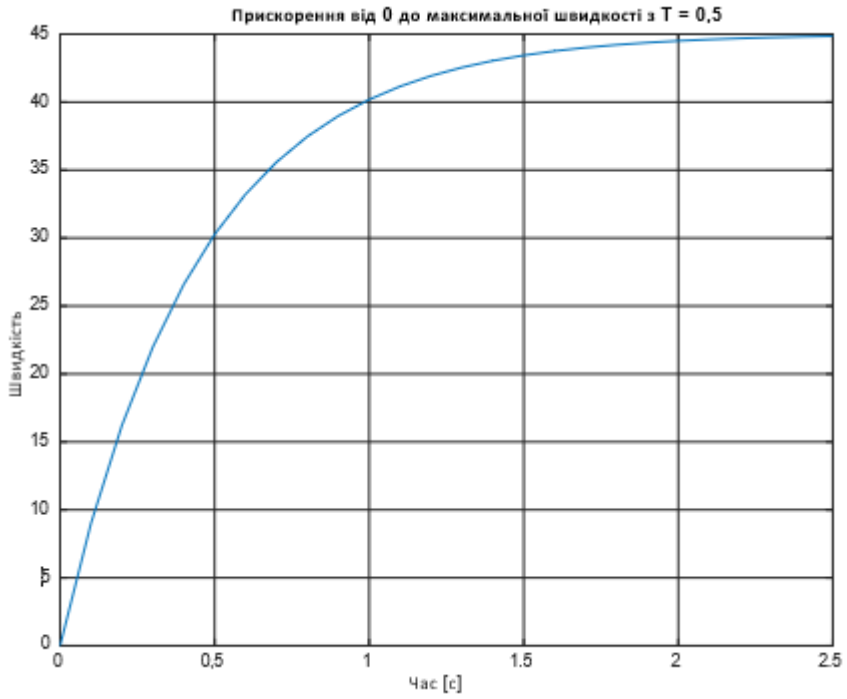


Рисунок 2.4 - Швидкість прискорення для T = 0,5.

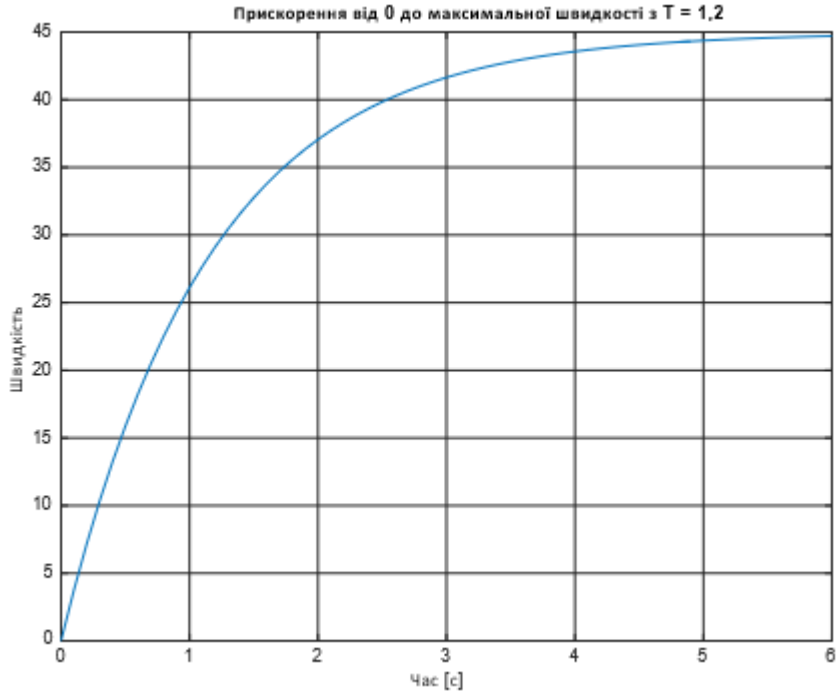


Рисунок 2.5 - Швидкість прискорення для T = 1,2.

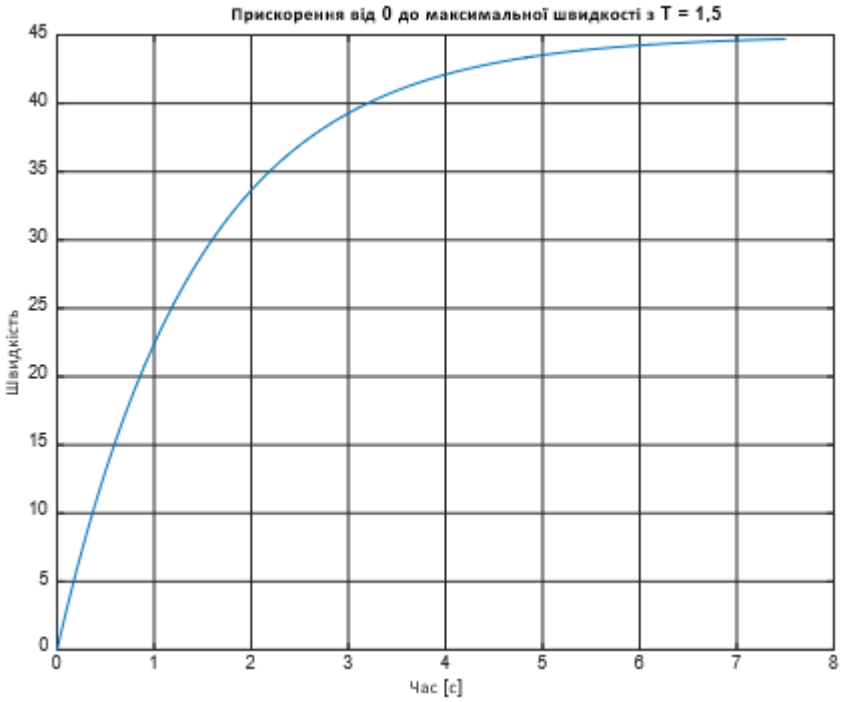


Рисунок 2.7 - Швидкість прискорення для T = 1,5.

2.4 Складання роботизованих автомобілів

Транспортні засоби-роботи базуються на проекті Pixy Pet Robot від Adafruit [9], який додає деякі можливості бездротового зв’язку та адаптує їх до конкретних потреб платформи. Вони складаються з 4 основних частин:

- Плата Arduino Leonardo виступає в якості основного контролера робота.
- Щит Zumo Robot для Arduino, який дає базову структуру робота, включаючи колеса, основні датчики, кнопки, корпус батареї та електронну плату для підключення всього до Arduino.
- Камера Pixy CMU5 і комплект Mini Pan/Tilt зібрані з мікросервоприводами, які забезпечують роботу комп’ютерним баченням, обробляючи зображення та надсилаючи дані на Arduino.
- Бездротові модулі XBee Series 1 із екраном Arduino, щоб надати роботизованому транспортному засобу можливості бездротового зв’язку.

Щоб зібрати роботизовані транспортні засоби, потрібно виконати наступні кроки:

1) Зберіть Pixy Cam за допомогою комплекту повороту/нахилу

Як це дуже детально описано на веб-сторінці проекту Adafruit Pixy Pet Robot [9], перше, що потрібно зробити, це вирізати всі виступи, які виступають із монтажного кронштейна комплекту повороту/нахилу, щоб отримати плоску поверхню, куди прикріпити Pixy Cam. Одну зі сторін також потрібно трохи обрізати, щоб звільнити місце для кабелів, як це видно на наступних рисунках.

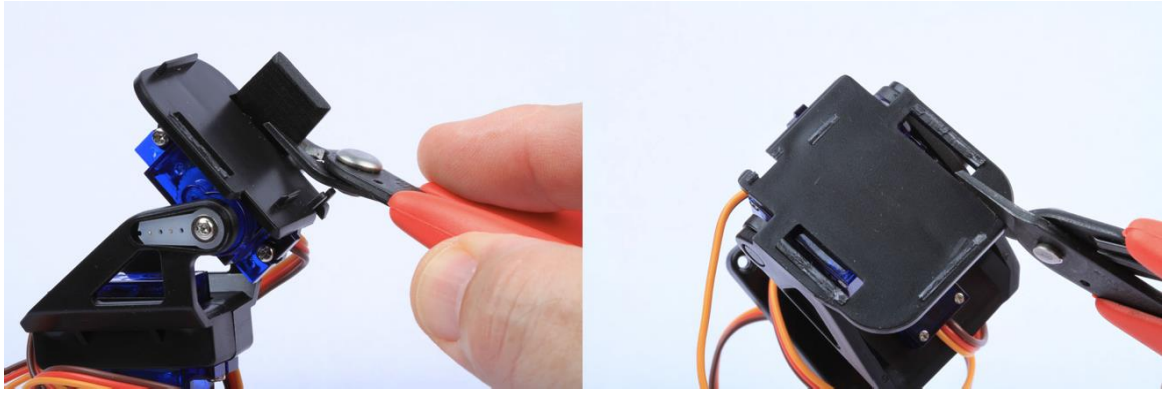


Рисунок 2.8 - Як обрізати комплект повороту/нахилу для Pixy Cam.

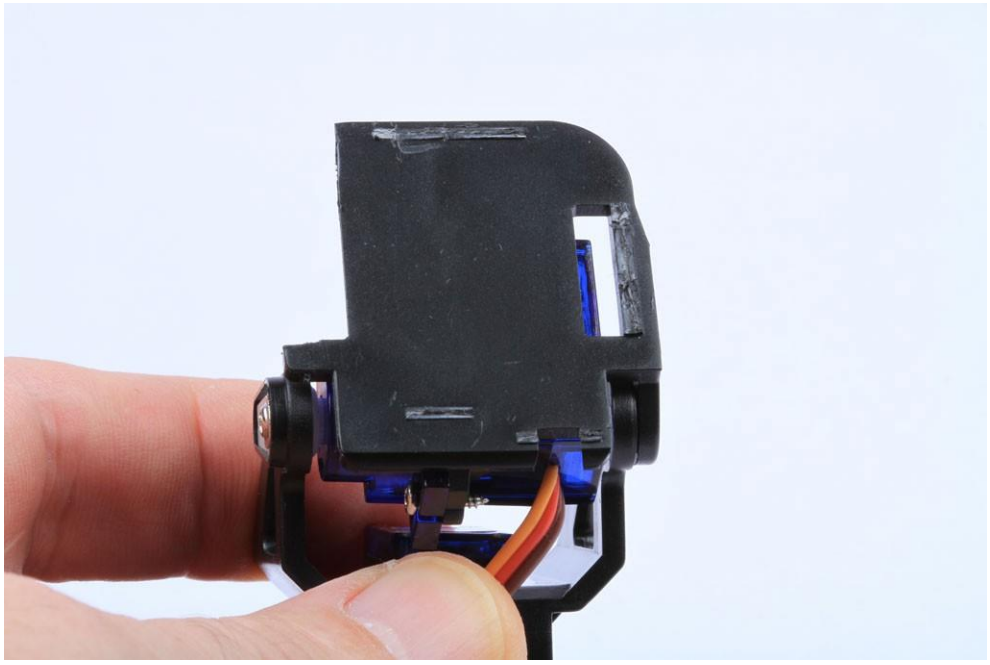
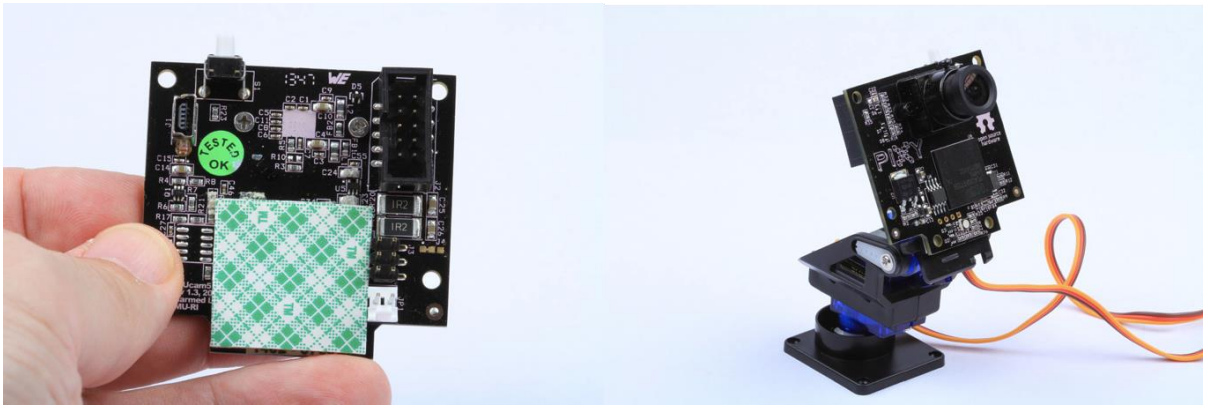


Рисунок 2.9 - Набір повороту/нахилу, готовий для приєднання Pixy Cam.

Пізніше модуль Pixy Cam потрібно прикріпити до плоскої поверхні комплекту повороту/нахилу двостороннім скотчем.



Риснуок 2.10 - Pixy Cam із двостороннім скотчем позаду та нарешті складений до комплекту повороту/нахилу

Коли камеру прикріплено до рухомої опори, потрібно під’єднати кабелі, як показано на рисунках нижче, дотримуючись цих 3 інструкцій:

- Коричневий кабель завжди внизу / жовтий кабель завжди зверху.
- Кабель сервоприводу панорамування (нижній, який йде ззаду) йде ліворуч.
- Кабель сервоприводу нахилу (верхній, що йде спереду) йде вправо.

Потім знову використовуючи двосторонній скотч, опору для повороту/нахилу з камерою прикріплюють у верхній частині плати Arduino, намагаючись тримати її на одній лінії з пунктирною лінією над «Відповідно RoHS» і передньою частиною, яка починається між «C». » і «E» у «FC CE».

Нарешті, сірий стрічковий кабель, який постачався з Pixy Cam, потрібно підключити від камери до Arduino. Однак до цього маленький чорний пластик трохи закриває 6-контактний порт Arduino потрібно вийняти, щоб основа кабелю не вилізла занадто сильно.

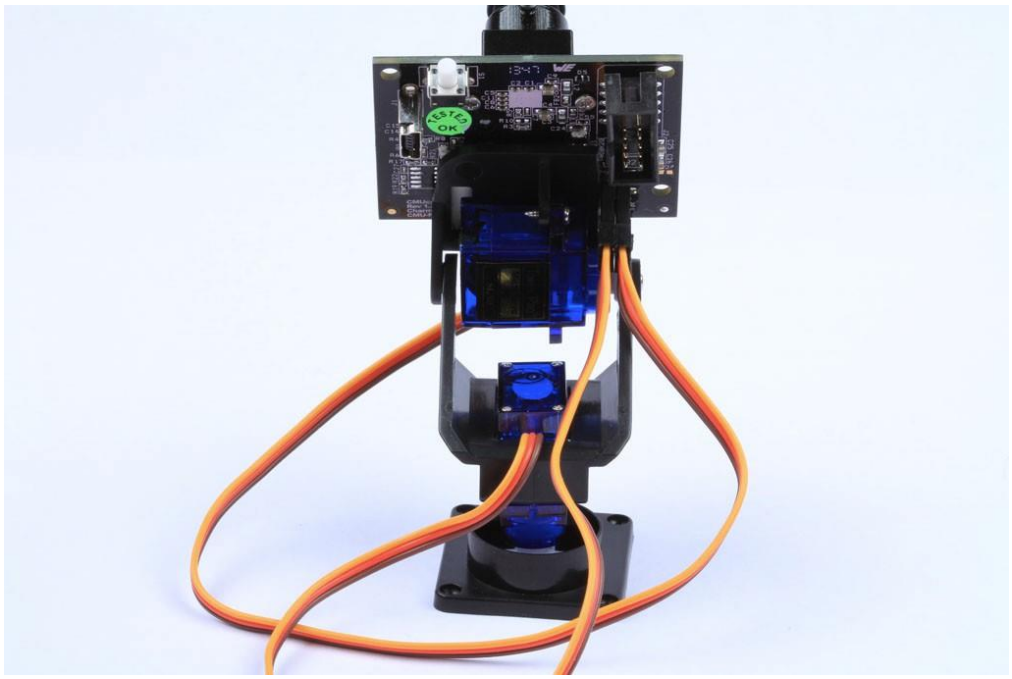


Рисунок 2.11 – Підєднання кабелю між камерою Pixy і комплектом повороту/нахилу.

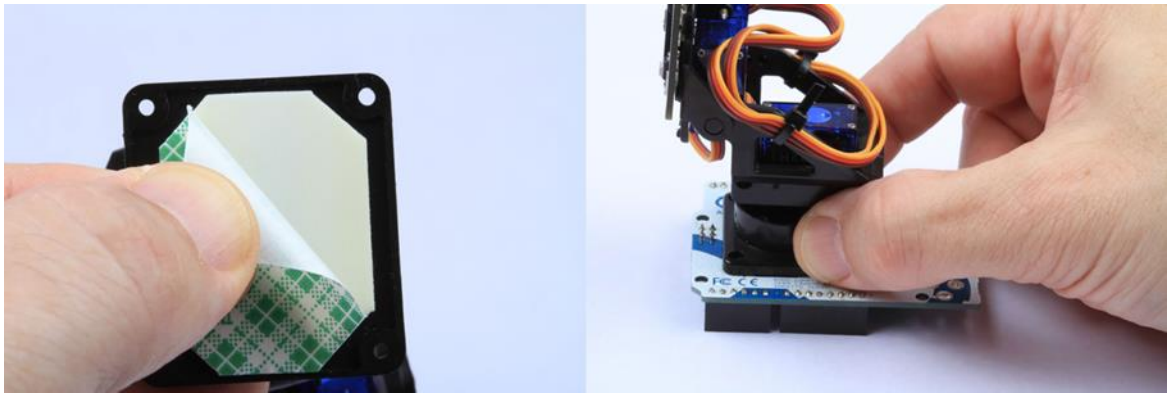


Рисунок 2.12 - Кріплення камери Pixy Cam із комплектом Pan/Tilt до верхньої частини плати Arduino.

Тепер модуль із Arduino та Pixy Cam готовий, і його можна приєднати безпосередньо до щита Zumo. Однак модуль XBee потрібно приєднати посередині, щоб робот мав бездротові можливості.

2) Припаяйте екран XBee

Екран XBee для Arduino поставляється без роз'ємів, тому набір роз'ємів потрібно припаяти, щоб його можна було приєднати до екрану Zumo

та Arduino. Чорна частина заголовків має бути на тій самій стороні, що й задня частина, де буде прикріплено модуль XBee, як це видно на наступних рисунках.

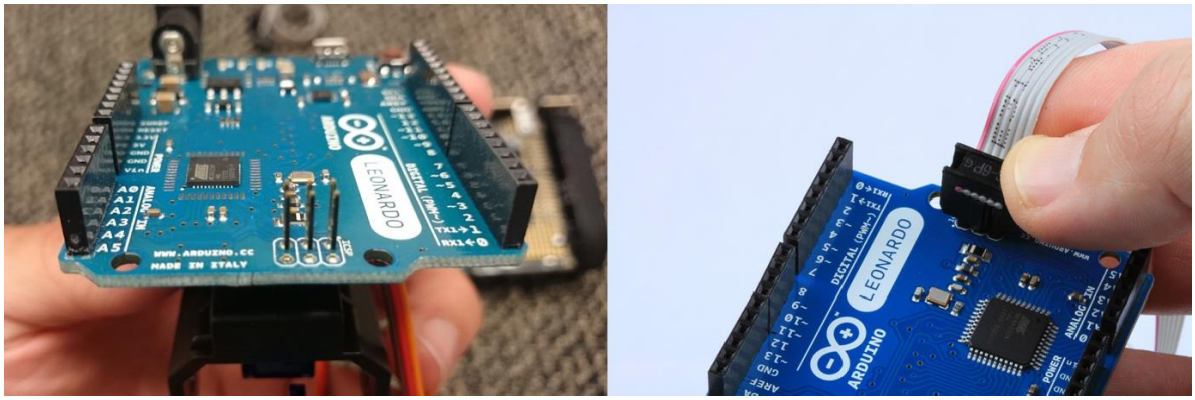


Рисунок 2.13 - Підключення Pixy Cam до плати Arduino за допомогою сірого кабелю.

Рисунок 2.14 - Припаювання роз’ємів до екрану XBee для Arduino.

Після того, як роз’єми припаяні, модуль XBee можна приєднати до екрану (дотримуючись напрямних ліній, накреслених для кутової частини, щоб отримати правильну орієнтацію). Нарешті, дуже важливо встановити маленький перемикач на екрані в режим «UART» замість «DLINE», щоб XBee міг спілкуватися через порт «Serial1» Arduino Leonardo.

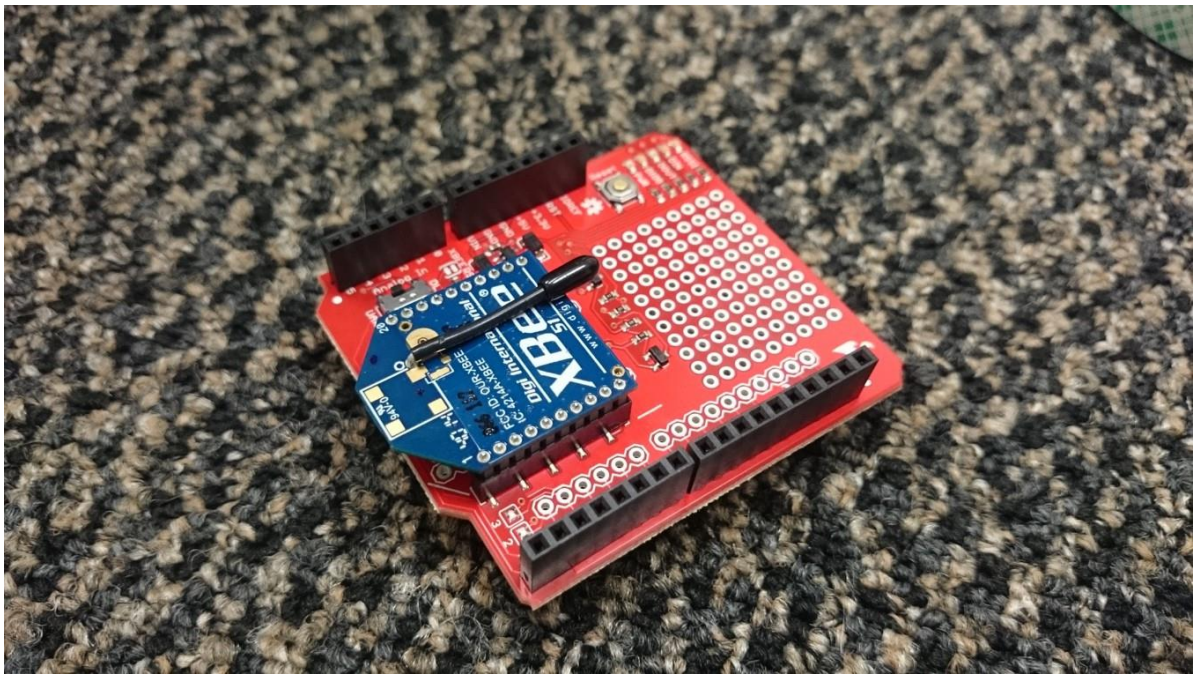


Рисунок 2.15 - Модуль XBee в щиті XBee із заголовками для підключення до плати Arduino Leonardo.

3) З’єднання системи

Перш ніж збирати все разом, потрібно виконати останнє налаштування в щиті робота Zumo. Синю перемичку, що дозволяє використовувати дві різні конфігурації, потрібно встановити так, щоб вона охоплювала два контакти, розташовані ближче до передньої частини, щоб вибрати «32u4», тип мікроконтролера, наявного в Arduino Leonardo, як це можна побачити на рисунку 18.

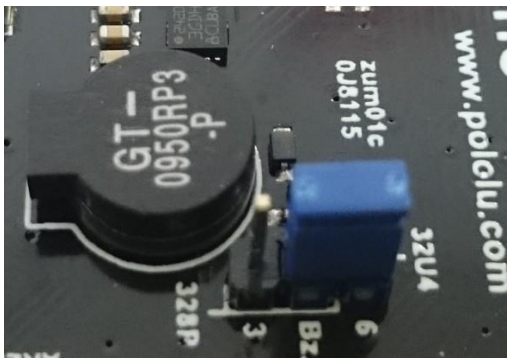


Рисунок 2.16 - Синя перемичка, яка вибирає конфігурацію "32u4" у щиті робота Zumo.

Тепер робота нарешті можна зібрати, приєднавши 3 отримані основні модулі (щит робота Zumo внизу, червоний щит XBee догори дном посередині та Arduino з Pixy Cam поверх усього). Ці три частини тепер легко монтувати та демонтувати на випадок, якщо потрібно щось змінити або потрібно приєднати більше шарів, щоб додати функції.

Нарешті, кабелі, що з'єднують сервоприводи та камеру, можна організувати за допомогою кабельних стяжок, щоб вони не дратували й не збивалися з чимось, коли робот рухається. При цьому важливо перевірити, чи достатньо провисання модуля камери для повороту та нахилу в усіх напрямках.

Варто також відзначити, що камера поставляється з чорною пластиковою кришкою перед об'єктивом, щоб захистити її від пилу та подряпин. Цю кришку можна вийняти, просто потягнувши за неї (і, логічно, це потрібно зробити, перш ніж спробувати нею скористатися).

4) Прикріпіть етикетку для розпізнавання Pixy Cam

Щоб Pixy Cam розпізнала машину попереду, до задньої частини роботів потрібно прикріпити шматок картону із зеленою крапкою. Шматок картону, який використовувався досі, був передньою частиною коробки щита робота Zumo, оскільки він має відповідний розмір і білий, тому можна легко малювати поверх нього. Використана зелена крапка мала діаметр 3,5 см, щоб роботизовані автомобілі могли вгадати правильну відстань відповідно до набору калібрування. Його було намальовано з верхнім краєм на відстані 0,5 см від вершини та по центру (з 0,9 см до країв з обох боків кола), щоб він розташовувався приблизно в місці Pixy Cam.

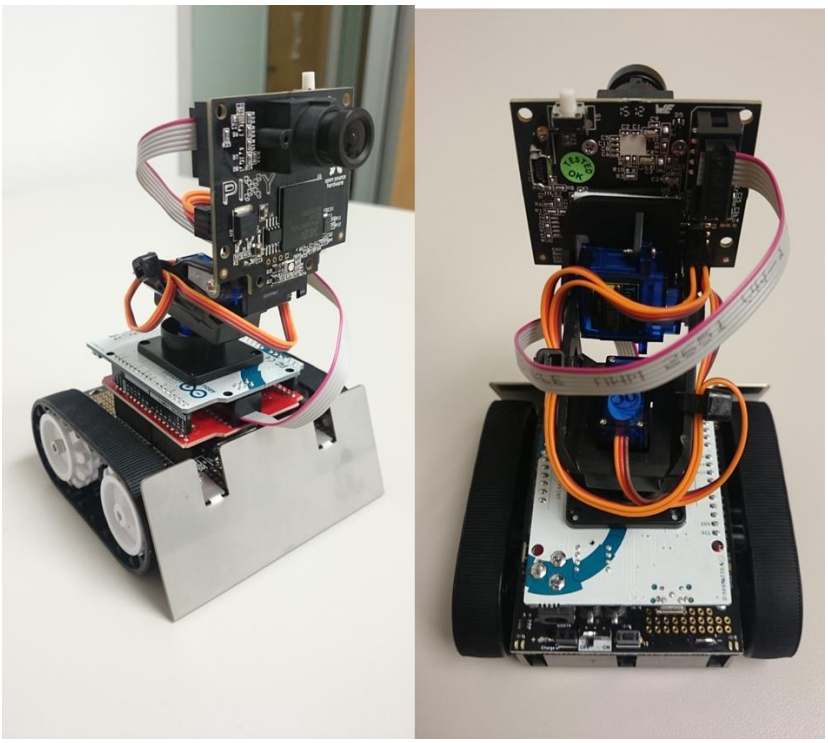


Рисунок 2.17 - Основні частини роботизованого автомобіля зібрані разом із кабельними стяжками, щоб кабелі не рухалися.

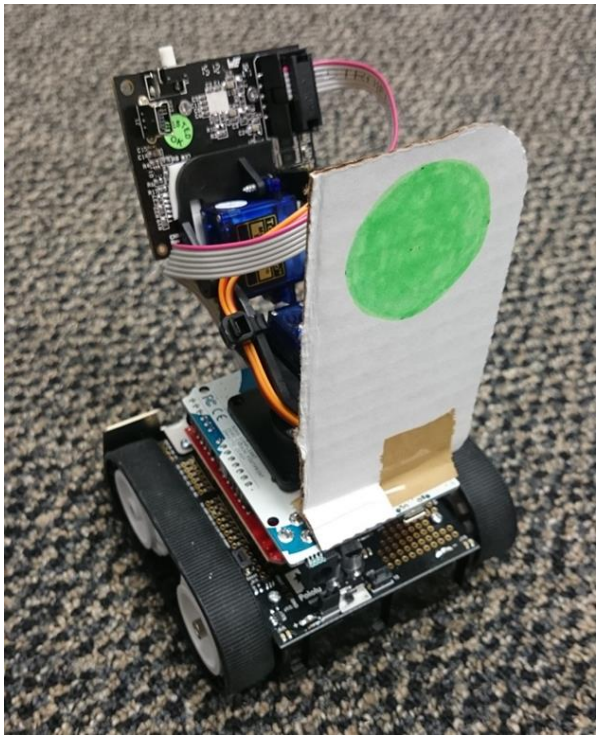


Рисунок 2.18 - Роботизований автомобіль із зеленою крапковою етикеткою, прикріпленою ззаду для вимірювання відстані за допомогою Pixy Cam.

Підсумовуючи цю частину, на наступних Рисуноках можна побачити 14 готових роботизованих транспортних засобів, які зараз знаходяться в лабораторії.

Рисунок 2.19 - 14 готових автомобілів-роботів присутні на платформі, вид спереду.

Рисунок 2.20 - 14 готових автомобілів-роботів присутні на платформі, вид ззаду.

3 СПЕЦІАЛЬНА ЧАСТИНА

3.1 Програмування автоматизованих роботизованих транспортних засобів

Є 3 основні речі, які потрібно запрограмувати в автомобілях-роботах:

- Модуль XBee, щоб він міг правильно спілкуватися з іншими XBee.
- Pixy Cam, щоб він міг правильно розпізнавати об’єкти
- Arduino Leonardo, щоб контролювати те, що роблять транспортні засоби.

Далі пояснюється, як програмувати кожну з цих частин.

3.2 Модуль XBee

Модулі XBee серії 1 (ті, що реалізовані в автомобілях-роботах) мають бути налаштовані за допомогою програми під назвою «XCTU», щоб вони могли спілкуватися один з одним. У Sparkfun є гарний підручник про те, як налаштувати та використовувати ці XBee з XCTU [10], але тут буде підсумовано базову конфігурацію, необхідну для транспортних засобів-роботів:

Перш за все, модуль XBee підключається до комп'ютера через USB-адаптер XBee Explorer. Це можна використовувати для зміни параметрів у модулі XBee або для спілкування з іншими XBee з комп’ютера.

Щоб XBees могли спілкуватися між собою, вони повинні мати однакові параметри, зокрема «Швидкість передачі даних» (як швидко вони спілкуються), «Канал» (частота, яку вони використовують для спілкування) та «PAN ID» (особиста зона). Ідентифікатор мережі, номер, який ідентифікує мережу, в якій розмовляє XBee).

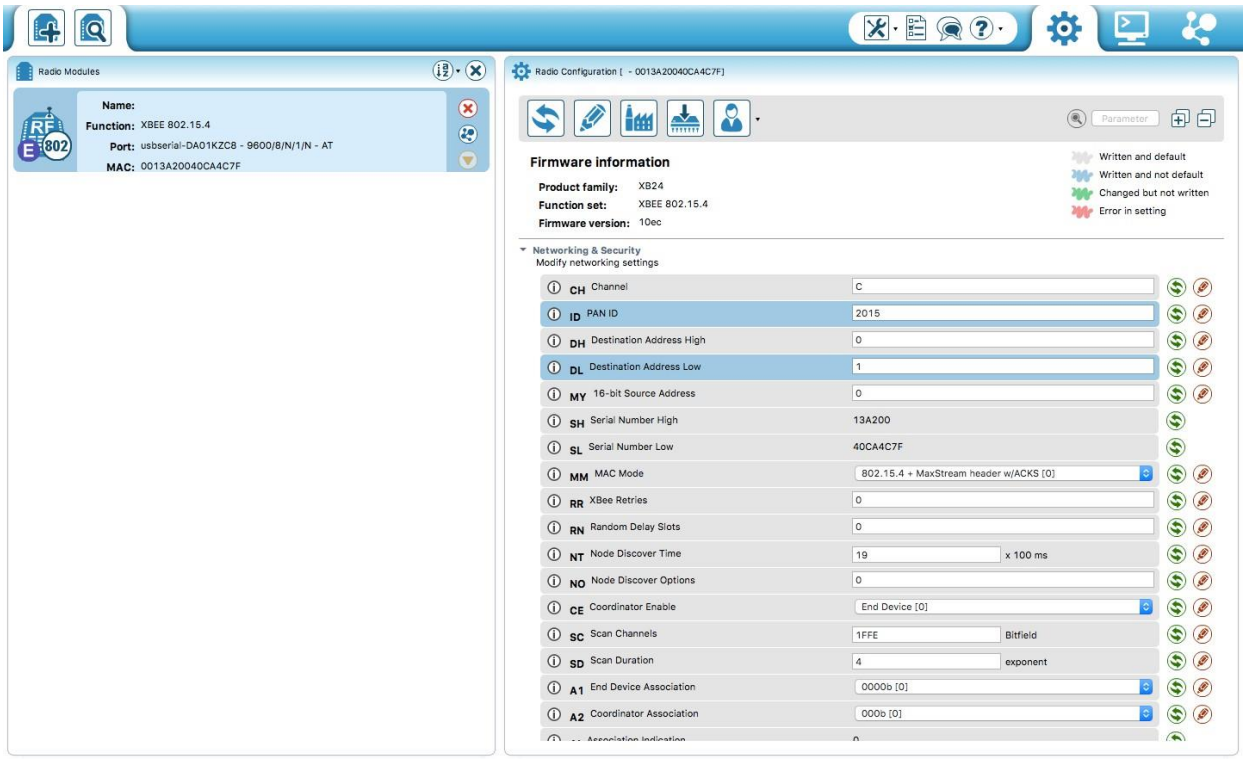


Рисунок 3.1 - Головний екран програмного забезпечення XCTU, що використовується для налаштування модулів XBee та зв’язку з роботами.

Єдиним із цих параметрів, зміненим із заводської конфігурації, є «PAN ID», для якого було вирішено встановити значення «2015». Щоб надсилати та отримувати повідомлення на платформі, для всіх XBees потрібно встановити «PAN ID» на «2015».

Іншими важливими параметрами є адреса «MY» і адреси призначення («DH» і «DL»). Кожен XBee має адресу («МОЯ» адреса) і надсилає повідомлення на адресу призначення, яка буде отримана лише модулями XBee, які мають цю адресу призначення як «МОЮ» адресу.

Щоб контролювати та надсилати повідомлення всім роботам одночасно з комп’ютера, один із

XBeesбуло призначено «МОЮ» адресу «0» та адресу призначення «1» («DH» встановлено на 0 і «DL» встановлено на 1. Цей модуль має антену, висунуту вгору, і буде підключатися до комп’ютера. Усім іншим модулям, які будуть прикріплені до роботів, присвоєно «1» у полі адреси «MY» і «0» у

полях адрес призначення «DH» і «DL». Таким чином, усі машини-роботи отримують повідомлення від комп’ютера, а комп’ютер отримує повідомлення від усіх роботів.

3.3 Pixy Camera

Камеру Піху необхідно налаштувати, щоб розпізнавати певний колірний візерунок як об’єкт. Це можна зробити, безпосередньо натиснувши кнопку в камері, або за допомогою PixyMon (програмне забезпечення від виробника), як це пояснюється на їх веб-сайті [11].

Для роботи з транспортними засобами найзручнішим способом є підключення камери до комп’ютера за допомогою кабелю Mini USB-USB і програмування за допомогою програмного забезпечення PixyMon, оскільки це дозволяє користувачеві бачити, як камера обробляє зображення. , а також зберігати та завантажувати параметри в кожну камеру без необхідності повторного навчання об’єкта. Однак для кожної камери може знадобитися виконати деякі налаштування, оскільки були враховані деякі відмінності в способі обробки зображень.

Піху Cam може зберігати до 7 колірних підписів (це означає, що вона може розпізнавати до 7 різних кольорів) і багато інших об’єктів за допомогою функції колірного коду (різні кольорові підписи, розміщені поруч). Щоб встановити ці кольорові підписи, об’єкт, який потрібно розпізнати, потрібно розташувати перед камерою, виберіть «Установити підпис №» у «Дії» меню, а кольоровий візерунок потрібно вибрати за допомогою миші. Тепер камера повинна виділяти об’єкти, щойно виявляє цей колірний візерунок.

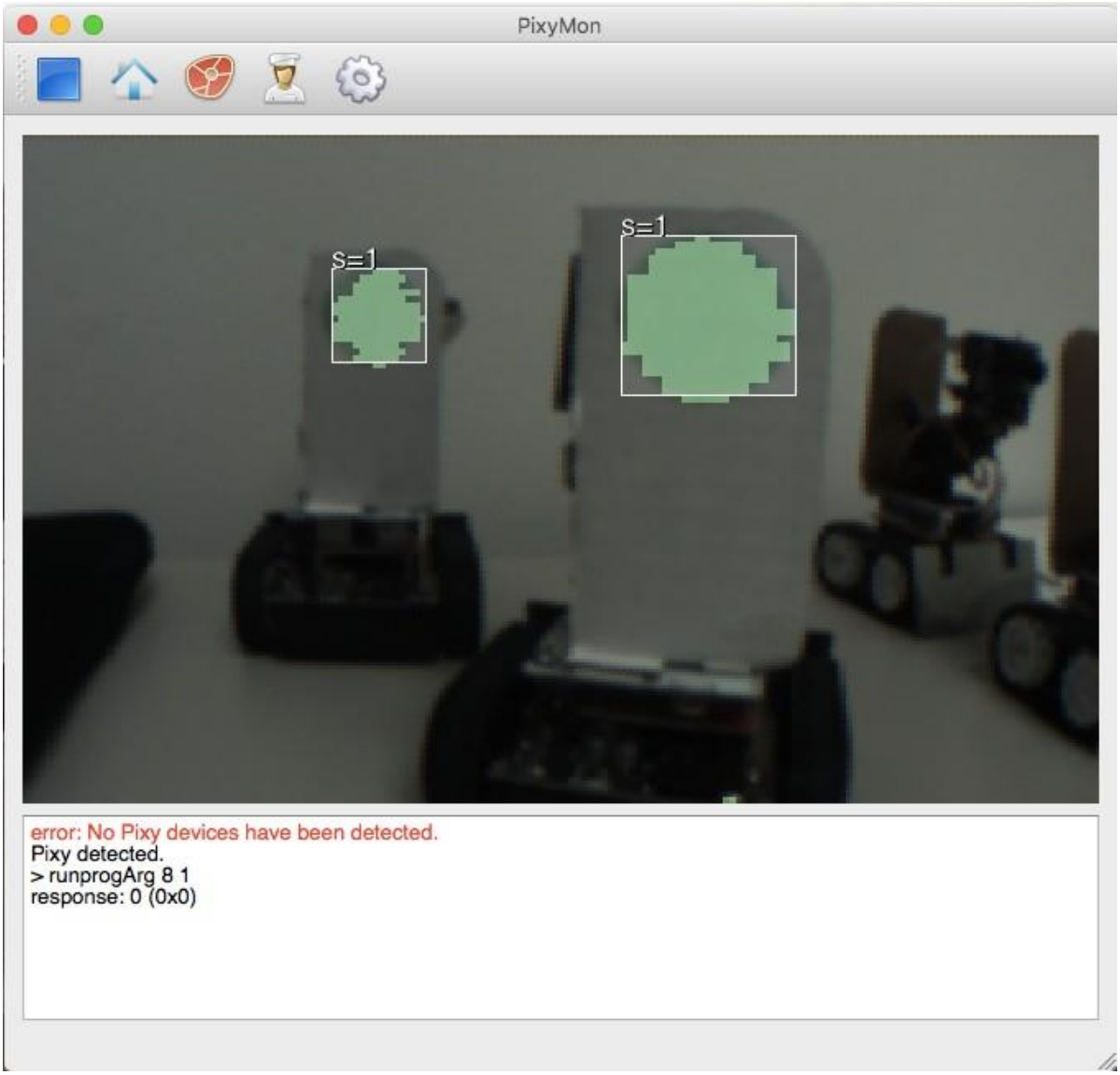


Рисунок 3.2 - Головний екран програмного забезпечення PixyMon для встановлення колірних підписів Pixy Cam.

3.3 Arduino

Плата Arduino Leonardo є мозком роботизованого автомобіля. Саме через нього можна контролювати все, що робить робот, і отримувати дані з датчиків. Його можна запрограмувати за допомогою програмного забезпечення Arduino від розробників плати.

Мова, що використовується, заснована на «С», з багатьма бібліотеками та ресурсами, доступними в Інтернеті, щоб зробити програмування досить простим і легким для вивчення. Далі в цьому документі детально описано та пояснено різні коди, які використовуються в роботах.

Для того, щоб завантажити код у кожен автомобіль-робот, плату Arduino потрібно підключити до комп'ютера за допомогою кабелю Micro USB-USB.

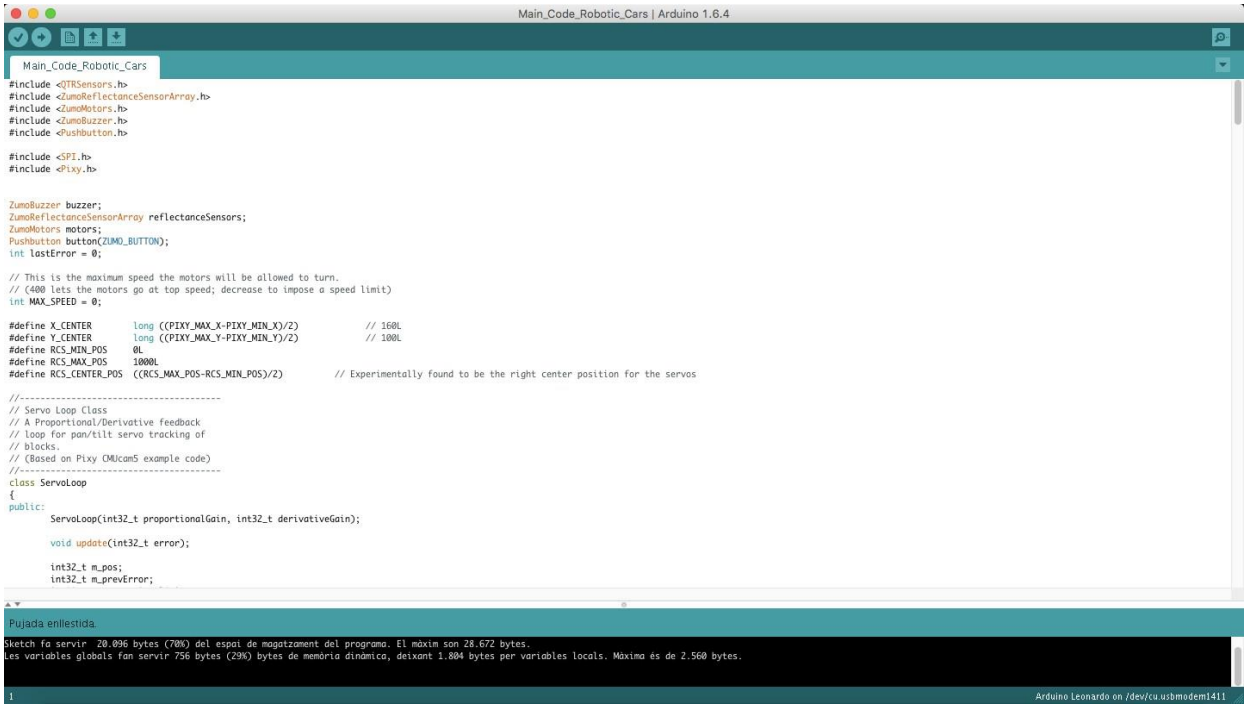


Рисунок 3.3 - Головний екран програмного забезпечення Arduino для програмування кодів, що визначають поведінку автомобілів-роботів.

3.4 Зв'язок і управління роботизованими автомобілями

Завдяки модулям XBee, встановленим у роботах, вони можуть спілкуватися з комп'ютером, а також ними керувати з цього останнього. XBees використовує послідовний зв'язок, це означає, що набір символів надсилається від випромінювача до приймача, і завданням приймача є їх

інтерпретація та дія відповідно. Ця система зв'язку досить проста, але має масу можливостей.

За допомогою програмного забезпечення «Serial Monitor» у комп’ютері (наприклад, інтегрованого в XCTU) можна отримувати дані від транспортних засобів і надсилати їм інші символи. Одне з перших застосувань цієї технології – отримання інформації від транспортних засобів. Вводячи певний рядок коду в програму Arduino, робот може друкувати деякі змінні або значення через XBee, щоб їх можна було побачити на комп’ютері.

Іншим основним використанням бездротового зв’язку є дистанційне керування робототехнікою. Це можна зробити, додавши фрагмент коду до Arduino, який зчитує символи, надіслані з комп’ютера, і реагує по-різному відповідно до отриманого символу чи рядка. Точні команди керування робототехнікою пояснюються пізніше під час детального опису кодів Arduino.

У майбутньому на комп’ютері можна буде запускати програму, яка автоматично взаємодітиме з транспортними засобами. Це може бути дуже цікаво, якщо додати світлофори, сигнали, перехрестя тощо.

3.5 Калібрування автомобілів-роботів

Вимірювання відстані за допомогою Ріху Cam

Щоб оцінити відстань до об’єкта за допомогою Ріху Cam, використовувана техніка базується на зміні розміру відстежуваного об’єкта. Ріху Cam надає значення висоти та ширини об’єктів, які вона відстежує, у пікселях. Отже, після певного калібрування, специфічного для його об’єкта, можна вгадати відстань за значенням висоти чи ширини.

Для перевірки точності використовувався предмет розміром 5,5x4,5 см. Об’єкт було встановлено на висоті піксі-камери, і в роботі було запущено код, щоб камера наводила на нього та записувала значення висоти та ширини

через серійний монітор (Додаток 1, сторінка 67). Робота рухали по прямій лінії, дивлячись на об’єкт, щоб проводити вимірювання кожні 5 см у діапазоні від найближчої відстані 5 см до найдальшої відстані 50 см. Щоб зробити вимірювання більш стабільними та легкими для зчитування на послідовному моніторі, Arduino прочитав значення 100 разів і видав їх середнє значення.

Після збору даних їх можна побудувати в Excel, щоб отримати тенденцію залежності відстані від висоти чи ширини зазначеного об’єкта. Це рівняння реалізовано в коді Arduino, щоб робот міг вгадати відстань до об’єкта. У двовимірному середовищі, подібному до того, у якому рухатимуться транспортні засоби-роботи, використання значення висоти замість ширини здається кращим варіантом, оскільки воно менш чутливе до змін кута огляду. Код для друку відстані, виміряної через послідовний монітор, схожий на той, який використовувався раніше, але з деякими відмінностями в налаштуванні та основному циклі.

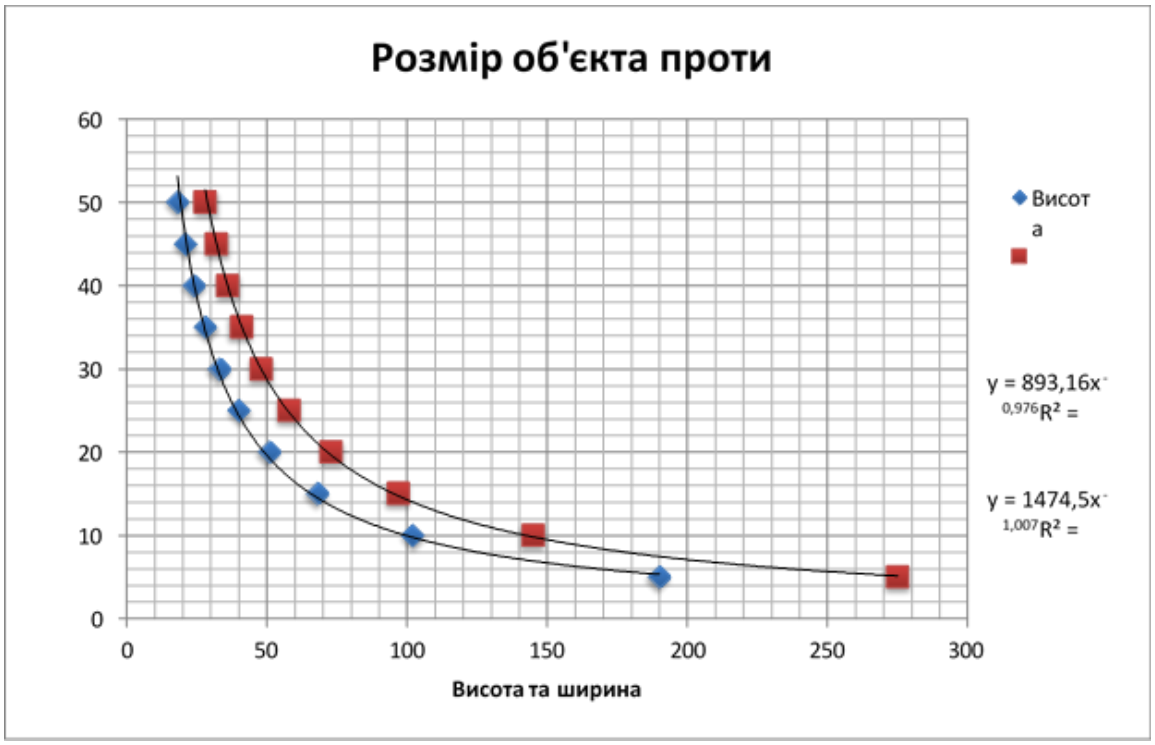


Рисунок 3.4 - Графік залежності відстані від розміру об’єкта для калібрування вимірювання відстані.

Ця техніка багато в чому залежить від калібрування кожного конкретного об’єкта, який потрібно відстежувати, але після калібрування вона показала досить хорошу точність.

Для зазначеного об'єкта, використаного для тесту, отримано наступний графік і лінії тенденцій.

Контроль швидкості робота Zumo

Тест прямолінійної швидкості

Щоб контролювати швидкість робота, бібліотека ZumoMotors.h дозволяє встановити змінну від 0 до 400, пропорційну швидкості лівого або правого двигуна. Отже, щоб знайти еквівалентність цієї змінної реальній швидкості робота, була написана програма, яка рухає робота вперед, а потім назад протягом певного часу з певною швидкістю (Додаток 1, сторінка 70). Змінюючи значення змінної швидкості та щоразу вимірюючи відстані, пройдені роботом, можна отримати графік із співвідношенням змінної швидкості та реальної швидкості.

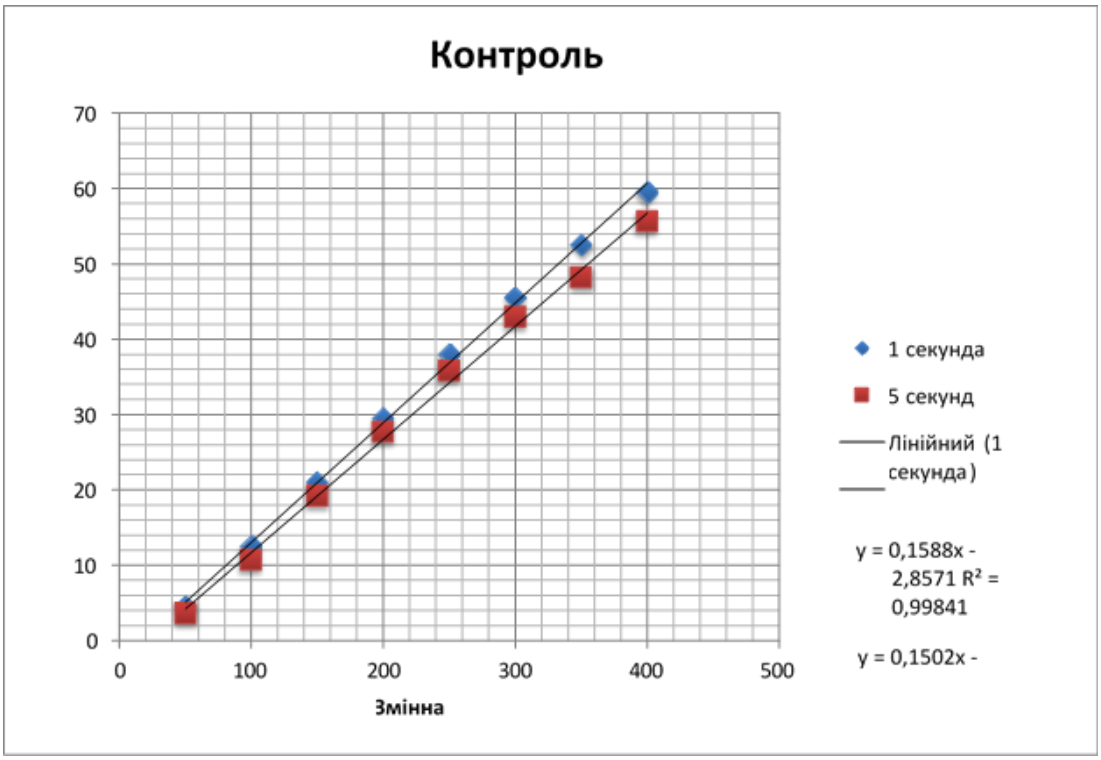


Рисунок 3.5 - Графік, що пов’язує реальну швидкість роботизованого автомобіля, що рухається вперед, зі змінною швидкості.

Як бачимо, є деякі відмінності між подорожжю протягом 1 або 5 секунд. Це могло бути частково через нерівності на килимовій підлозі, де проводився експеримент. Однак дані досить узгоджуються зі специфікаціями виробника.

Деякі відмінності також можна знайти під час руху вперед або назад, зазвичай останній повільніший. Однак не було плану їздити роботизованими автомобілями назад, тому увага буде зосереджена на швидкості вперед.

Також була оцінена схильність робота повертатися вліво, ймовірно, через деякий рух коліс і нерівності на землі. Така поведінка була більш важливою під час руху назад, але, як було сказано раніше, немає інтересу до точного переміщення автомобілів-роботів назад. Однак це також було трохи помітно під час руху вперед, тому, щоб компенсувати це та змусити робота їхати прямо, швидкість лівого двигуна було збільшено на значення від 3% до 5% відносно швидкості правого двигуна. У будь-якому випадку, оскільки в схемі роботи будуть слідувати лінії, ця асиметрія буде автоматично компенсована алгоритмом слідування лінії.

Беручи до уваги всі попередні дані, здається, що швидкість роботів можна досить точно контролювати, змінюючи змінну Speed у бібліотеці ZumoMotors.h. Однак через велику залежність від зовнішніх факторів (таких як нерівність рельєфу, рух коліс, вага робота тощо) він потребує спеціального калібрування в умовах, максимально схожих на ті, що використовуються в кожній схемі. Це означає почати з того, що дані такого ж типу від робота слідують рядку.

Тест швидкості за рядком

Для перевірки швидкості робота під час проходження лінії була використана установка, подібна до попередньої. Наведений нижче алгоритм рядка з прикладів робота Zumo був змінений, щоб він працював протягом заданого часу, зупинявся на 5 секунд і повторював цю послідовність знову і

знову. Щоб отримати дані для різних швидкостей, змінна MAX_SPEED кожного разу змінювалася.

Пряма чорна лінія була намальована та встановлена на тому самому килимовому покритті, що й у попередніх тестах. Провівши по ньому робота, вимірявши пройдену відстань і проаналізувавши його, був отриманий такий графік:

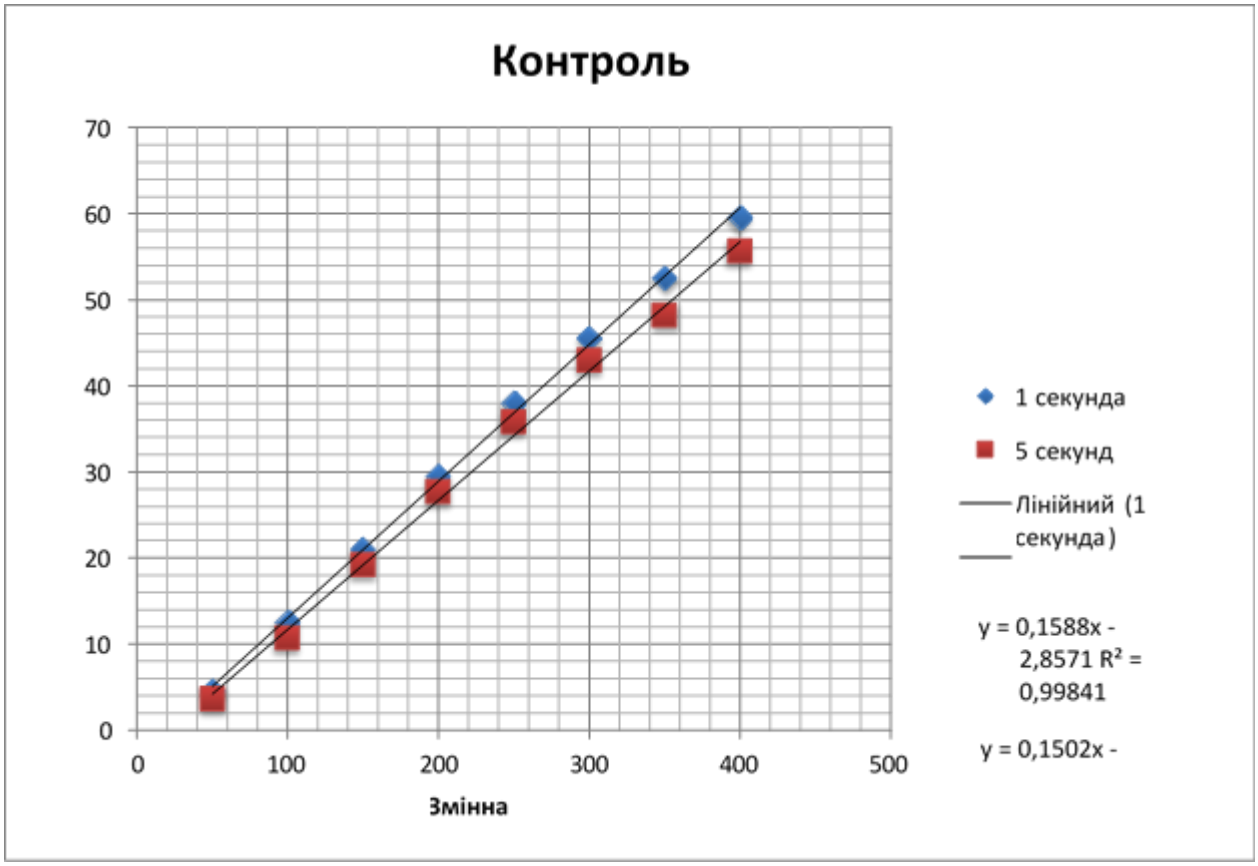


Рисунок 3.6 - Графік, що пов’язує змінну швидкості з фактичною швидкістю роботизованого автомобіля під час руху по лінії.

Як видно, зв’язок між змінною швидкістю та фактичною швидкістю робота під час слідування по лінії є лінійним і, здається, є досить стабільним незалежно від часу роботи робота. Також можна побачити, як і очікувалося, що робот, який слідує за лінією, трохи повільніший, ніж рухається прямо, але досягнута швидкість у будь-якому випадку досить хороша.

Найважливішими параметрами, що впливають на швидкість автомобіля, є фактична вага самого робота та тип поверхні, по якій рухається. Таким чином, щоб отримати рівняння для реалізації в кодах, що дозволяє встановити швидкість для робота, аналогічне калібрування повинно бути виконано для кожної зміни в конструкції транспортного засобу або поверхні схеми.

Основна функція: слідування по лінії, адаптація швидкості до попереднього автомобіля.

Базовий код, який запускатиме роботизовані автомобілі по ланцюгу, базується на двох основних функціях: слідування лінії та відстеження об’єктів.

Для того, щоб роботизованими автомобілями можна було керувати та направляти їх по ланцюгу, буде використано функцію наступної лінії. Таким чином, будь-яку схему можна намалювати чорною лінією на білому тлі, щоб транспортні засоби могли слідувати за нею. Pololu надає наступний рядок коду для робота Zumo Arduino серед прикладів кодів у своїй бібліотеці.

Щоб адаптувати швидкість відповідно до відстані з попереднім транспортним засобом, буде використана функція відстеження об’єктів Pixy Cam, прикріплюючи кольорову крапку на задній частині кожного робота, щоб камера автомобіля-роботу, що наближається, могла бачити її. Adafruit надає код Zumo Robot для Arduino для відстеження об’єктів за допомогою Pixy Cam.

На конкретній платформі, що будується, чорні лінії позначають дороги, якими продовжуватимуть рухатися роботизовані автомобілі, перевіряючи відстань до попередніх транспортних засобів за допомогою механізму відстеження об’єктів. Для цього частини двох основних зазначених кодів потрібно було об’єднати разом із новими реалізаціями, щоб уся система працювала. Нижче пояснюється основна структура цього коду:

Оголошення змінних

Перша частина коду запускається один раз під час запуску робота та оголошує використані бібліотеки та змінні разом із «Класом циклу сервоприводу», пропорційним/похідним циклом зворотного зв'язку, призначеним для переміщення сервоприводу панорамування та нахилу Pixy Cam. щоб відстежуваний об'єкт завжди залишався в центрі огляду. Більше інформації про те, як саме це робиться, можна знайти на веб-сайті проекту Pixy Pet від Adafruit [12].

```
#include <QTRSensors.h>
#include <ZumoReflectanceSensorArray.h>
#include <ZumoMotors.h>
#include <ZumoBuzzer.h>
#include <Pushbutton.h>

#include <SPI.h>
#include <Pixy.h>


ZumoBuzzer buzzer;
ZumoReflectanceSensorArray reflectanceSensors;
ZumoMotors motors;
Pushbutton button(ZUMO_BUTTON);
int lastError = 0;

// This is the maximum speed the motors will be allowed to turn.
// (400 lets the motors go at top speed; decrease to impose a speed limit)
int MAX_SPEED = 0;

#define X_CENTER        long ((PIXY_MAX_X-PIXY_MIN_X)/2)             // 160L
#define Y_CENTER        long ((PIXY_MAX_Y-PIXY_MIN_Y)/2)             // 100L
#define RCS_MIN_POS     0L
#define RCS_MAX_POS     1000L
#define RCS_CENTER_POS ((RCS_MAX_POS-RCS_MIN_POS)/2)            // Experimentally found to be the
right center position for the servos

//---------------------------------------
// Servo Loop Class
// A Proportional/Derivative feedback
// loop for pan/tilt servo tracking of
// blocks.
// (Based on Pixy CMUcam5 example code)
//---------------------------------------
class ServoLoop
{
public:
        ServoLoop(int32_t proportionalGain, int32_t derivativeGain);

        void update(int32_t error);

        int32_t m_pos;
        int32_t m_prevError;
        int32_t m_proportionalGain;
        int32_t m_derivativeGain;
};

// ServoLoop Constructor
ServoLoop::ServoLoop(int32_t proportionalGain, int32_t derivativeGain)
{
        m_pos = RCS_CENTER_POS;
```

```
        m_proportionalGain = proportionalGain;
        m_derivativeGain = derivativeGain;
        m_prevError = 0x80000000L;
}

// ServoLoop Update
// Calculates new output based on the measured
// error and the current state.
void ServoLoop::update(int32_t error)
{
        long int velocity;
        char buf[32];
        if (m_prevError!=0x80000000)
        {
                velocity = (error*m_proportionalGain + (error - m_prevError)*m_derivativeGain)>>10;

                m_pos += velocity;
                if (m_pos>RCS_MAX_POS)
                {
                        m_pos = RCS_MAX_POS;
                }
                else if (m_pos<RCS_MIN_POS)
                {
                        m_pos = RCS_MIN_POS;
                }
        }
        m_prevError = error;
}
// End Servo Loop Class
//---------------------------------------

Pixy pixy;  // Declare the camera object

ServoLoop panLoop(200, 200);  // Servo loop for pan
ServoLoop tiltLoop(150, 200); // Servo loop for tilt
```

Налаштування

Це базова структура кодів Arduino, заснована на дії під назвою «налаштування», яка виконується один раз на початку програми, щоб підготувати робота до роботи. У нашому випадку він використовується для ініціалізації датчиків відбиття, послідовних портів і деяких інших змінних. Він також запускає фрагмент коду, який дозволяє користувачеві вирішувати за допомогою дистанційного керування або натисканням кнопки, коли починати калібрування датчиків відбиття. Нарешті, він чекає натискання кнопки в машині-роботі або надсилання будь-якого символу через пульт дистанційного керування, щоб запустити основну програму:

```
void setup()
{

  // Initialize the reflectance sensors module
  reflectanceSensors.init();

  // Set up both ports at 9600 baud. This value is most important
  // for the XBee. Make sure the baud rate matches the config
  // setting of your XBee.

  Serial1.begin(9600);  //XBee/UART1/pins 0 and 1
  Serial.begin(9600);   //USB

  // Wait for wireless command or for the user button to be pressed and released
  // button.waitForButton();

  Serial1.println(F("Calibrate? Press 'Y' to calibrate."));
  while (Serial1.available() < 1 && !button.getSingleDebouncedRelease())
  ;                                                                    // Wait for button or
answer to be pressed
  char temp = Serial1.read();                                           // Read the pressed
key.

  switch (temp) {
    case 'Y':
    case 'y':
    CalibrateLineSensors();
    break;
    default:
    CalibrateLineSensors();
    break;
  }
  pixy.init();                                                        // Initialize Pixy Cam

  // Wait for wireless command or for the user button to be pressed and released
  // button.waitForButton();
  Serial1.println(F("Calibrated!"));
  Serial1.println(F("Press any key to start"));
  while (Serial1.available() < 1 && !button.getSingleDebouncedRelease())
  ;                                                                    // Wait for any key to
be pressed
  temp = Serial1.read();      // Read the pressed key.
}


uint16_t blocks;
uint32_t lastBlockTime = 0;
uint32_t lastSpeedUpdate = 0;
int32_t BlockHeight = 0;
float DesiredSpeed = 0;           // Starting speed of 21 cm/s (equivalent to around 200 in speed
variable, maximum speed of robot)

int SafetyDistance = 10;         // Minimum Safety Distance with the previous vehicle [cm] (1/3 -
2/3 of vehicle length)
float VarT = 0.1;                // Delay in drivers reaction [s]
float Tau = 0.3;                  // Constant of time (time we want to stay to reach the previous
vehicle) [s]
float T = 0.5;                   // Time to reach the desired velocity [s]

float SpeedVariable = 0;
float Distance = 100;            // Distance to previous vehicles [cm]
```

3.6 Основний цикл

Основний цикл є найважливішою структурою коду Arduino. Це набір інструкцій, які виконуватимуться неодноразово назавжди після одного запуску інсталяції. У нашому випадку цей фрагмент коду відповідає за

утримання роботизованого автомобіля на трасі, одночасно перевіряючи дистанцію та адаптуючи швидкість до попереднього автомобіля.

Щоб відстежувати лінію, Arduino зчитує значення датчиків відбиття під роботом під час кожного проходу циклу та, використовуючи параметри, відкалібровані в частині налаштування, обчислює значення помилки, що відображає, наскільки далеко робот знаходиться від центру лінії. . Потім це значення помилки використовується в ПІД-контролері для розрахунку різниці швидкостей між правим і лівим двигунами, щоб робот рухався до центру лінії. Термін Integer зазвичай не дуже корисний у наступному рядку, тому насправді реалізовано контролер PD. На даний момент пропорційна постійна використовується дорівнює ¼, а похідна 6, але їх можна змінити, якщо потрібно, щоб адаптувати реакцію роботів до кожної конкретної доріжки:

```
int speedDifference = error/4 + 6 * (error - lastError)
```

Інша важлива функція, реалізована в основному циклі, - контроль дистанції з попереднім транспортним засобом. Це робиться шляхом зміни «Максимальної швидкості», яку можна застосовувати до двигунів, у рядку, що відповідає набору інструкцій. Для цього Arduino зчитує параметри з Піху Cam, щоб отримати кількість відстежуваних об’єктів у полі зору та висоту найбільшого з них. За цією висотою відстань обчислюється за допомогою рівнянь, отриманих під час калібрування визначення відстані. Він також виконує деякі допоміжні функції для сервоприводів, щоб навести камеру на найбільший відстежуваний об’єкт.

Потім він застосовує модель оптимальної швидкості, щоб обчислити бажану швидкість відповідно до оціненої відстані, і встановлює це значення параметру MAX_SPEED. Ця остання частина виконується лише кожен

певний час (наразі кожні 0,1 секунди), щоб імітувати реальний час реакції людей. Якщо блоків не видно або їх висота менша за певний поріг, відстань встановлюється на значення, достатньо велике, щоб моделі могли обчислити параметри швидкості так, ніби транспортного засобу в полі зору немає.

Важливо враховувати час оновлення різних пристроїв. Arduino проходитиме код із найшвидшою швидкістю в системі (16 МГц). Pixy Cam, однак, надсилатиме інформацію повільніше (приблизно кожні 10 циклів плати Arduino). Тому важливо оновлювати швидкість ще повільніше, щоб переконатися, що відстань оновлюється на основі показань камери. Ця повільна швидкість не є проблемою, оскільки вона все ще швидша, ніж можна оцінити. Крім того, це дозволяє застосовувати корекції, що слідують за лінією, на набагато вищій швидкості, ніж будь-що інше, забезпечуючи таким чином плавний рух роботизованого автомобіля по трасі.

Нарешті, на початку коду також є рядки, що відповідають за дистанційне керування роботами. Під час кожного циклу Arduino перевіряє наявність отриманих символів, і якщо вводиться буква «s», він зупиняє транспортний засіб і чекає, поки буде введено будь-який інший символ, щоб відновити рух. Якщо натиснути символ «r», роботизовані автомобілі зупиняються та запускають дію RemoteControl, щоб оновити деякі параметри автомобіля без необхідності перепрограмувати його.

```
void loop()
{

  // Wireless Communication features: Stop program when letter 's' or 'S' is pressed, stop and
enter remote control mode when 'r' or 'R', and listen for communication when getting 'c' or 'C'

  if (Serial1.available() >= 1) {
    char WirelessCommand = Serial1.read();


  switch (WirelessCommand) {
    case 's':
    case 'S':
    motors.setSpeeds(0, 0);
    Serial1.println(F(" Stopped!"));
    Serial1.println(F("Press any key to start again"));
    while (Serial1.available() < 1)
    ;                                                    // Wait for a key to be pressed
    WirelessCommand = Serial1.read();
    break;
```

```
      case 'r':
      case 'R':
      motors.setSpeeds(0, 0);
      Serial1.println(F(" Stopped!"));
      Serial1.println(F("Entered remote Control mode"));
      RemoteControl();
      break;
    }
  }
  // Follow line and every 'VarT' seconds update MAX_SPEED according to: 1) Object Tracking.  2)
Distance Detection.  3) Optimal Velocity Model

  blocks = pixy.getBlocks();
  float Speed = 0.13 * MAX_SPEED - 5;                                                    // Current
speed [cm/s]


  // If we have blocks in sight, track and follow them
  if (blocks)
  {
    int trackedBlock = TrackBlock(blocks);
    BlockHeight = pixy.blocks[trackedBlock].height;
    lastBlockTime = millis();

    Serial.print("Block Height [pixels] = ");
    Serial.println(BlockHeight);
  }
  else if (millis() - lastBlockTime > 100) {
      panLoop.m_pos = RCS_CENTER_POS;
      tiltLoop.m_pos = RCS_CENTER_POS;
      pixy.setServos(panLoop.m_pos, tiltLoop.m_pos);
  }

  Serial.print("Run ");

  if (millis() - lastSpeedUpdate > VarT*1000) {
    if (BlockHeight >= 5) {
      Distance = 782.66 * pow(BlockHeight,-0.981);                 // Distance to previous vehicle
[cm]
    }
    SpeedVariable = 7.5 * DesiredSpeed + 40;
    MAX_SPEED = (int) SpeedVariable;
    if (MAX_SPEED > 300) {
      MAX_SPEED = 300;
    }

    if (Distance < 8) {
        DesiredSpeed = 0;
      }
      else {
        DesiredSpeed = OptimalVelocityModel(Distance, Speed, VarT);      // Desired speed [cm/s]
      }


       Print_Values_Serial(BlockHeight, Distance, Speed, DesiredSpeed, MAX_SPEED);
       // Print_Values_Serial1(BlockHeight, Distance, Speed, DesiredSpeed, MAX_SPEED);

       BlockHeight = 0;
       Distance = 100;                                                                   // What we
 consider "Infinite Distance": no vehicle on sight.
       lastSpeedUpdate = millis();
     }


 unsigned int sensors[6];

 // Get the position of the line.  Note that we *must* provide the "sensors"
 // argument to readLine() here, even though we are not interested in the
 // individual sensor readings
 int position = reflectanceSensors.readLine(sensors);

 // Our "error" is how far we are away from the center of the line, which
 // corresponds to position 2500.
 int error = position - 2500;
```

```
  // Get motor speed difference using proportional and derivative PID terms
  // (the integral term is generally not very useful for line following).
  // Here we are using a proportional constant of 1/4 and a derivative
  // constant of 6, which should work decently for many Zumo motor choices.
  // You probably want to use trial and error to tune these constants for
  // your particular Zumo and line course.
  int speedDifference = error / 4 + 6 * (error - lastError);

  lastError = error;

  // Get individual motor speeds.  The sign of speedDifference
  // determines if the robot turns left or right.
  int m1Speed = MAX_SPEED + speedDifference;
  int m2Speed = MAX_SPEED - speedDifference;

  // Here we constrain our motor speeds to be between 0 and MAX_SPEED.
  // Generally speaking, one motor will always be turning at MAX_SPEED
  // and the other will be at MAX_SPEED-|speedDifference| if that is positive,
  // else it will be stationary.  For some applications, you might want to
  // allow the motor speed to go negative so that it can spin in reverse.
  if (m1Speed < 0)
    m1Speed = 0;
  if (m2Speed < 0)
    m2Speed = 0;
  if (m1Speed > MAX_SPEED)
    m1Speed = MAX_SPEED;
  if (m2Speed > MAX_SPEED)
    m2Speed = MAX_SPEED;

  motors.setSpeeds(m1Speed, m2Speed);
}
```

3.7 Допоміжні функції

Це частини коду, викликані з основного циклу або циклу налаштування для певних цілей. Вони можуть бути функціями або діями та записуються в кінці коду поза циклами. Їх можна викликати більше одного разу в кожному циклі з різними параметрами, таким чином уникаючи написання тих самих рядків коду кілька разів у основній структурі.

Використання цих допоміжних функцій є найпростішим способом реалізувати більше функцій для роботів у майбутньому, як-от бездротовий зв’язок, виявлення дорожніх знаків або іншу логіку поведінки.

На даний момент в коді присутні такі допоміжні функції:

1) Трекблок функція

Це фрагмент коду, який відповідає за пошук найбільшого блоку, який Піху Cam має в полі зору, і переміщення сервоприводів для камери, щоб навести на нього:

```
int oldX, oldY, oldSignature;

//---------------------------------------
// Track blocks via the Pixy pan/tilt mech
// (based in part on Pixy CMUcam5 pantilt example)
//---------------------------------------
int TrackBlock(int blockCount)
{
        int trackedBlock = 0;
        long maxSize = 0;

        Serial.print("blocks =");
        Serial.println(blockCount);

        for (int i = 0; i < blockCount; i++)
        {
                if ((oldSignature == 0) || (pixy.blocks[i].signature == oldSignature))
                {
                        long newSize = pixy.blocks[i].height * pixy.blocks[i].width;
                        if (newSize > maxSize)
                        {
                                trackedBlock = i;
                                maxSize = newSize;
                        }
                }
        }

        int32_t panError = X_CENTER - pixy.blocks[trackedBlock].x;
        int32_t tiltError = pixy.blocks[trackedBlock].y - Y_CENTER;


        panLoop.update(panError);
        tiltLoop.update(tiltError);

        pixy.setServos(panLoop.m_pos, tiltLoop.m_pos);

        oldX = pixy.blocks[trackedBlock].x;
        oldY = pixy.blocks[trackedBlock].y;
        oldSignature = pixy.blocks[trackedBlock].signature;
        return trackedBlock;
}
```

2) Оптимальна модель швидкості функція

Він розраховує бажану швидкість для робота, враховуючи відстань до попереднього транспортного засобу, поточну швидкість і час, встановлений для відображення затримки реакції:

```
float OptimalVelocityModel(float d, float v, float VarT)
{
  float Vd = (d - SafetyDistance) / Tau;            // Desired velocity [cm/s]

  float a = (Vd - v) / T;                            // Acceleration to reach desired velocity in
desired time [cm/s^2]

  float vf = v + a*VarT;                            // Velocity to apply after delay time [cm/s]

  return vf;
}
```

3) Віддалене керування

Дія, що використовується для зміни параметрів програми без її перепрограмування. На даний момент реалізовано зміну безпечної відстані, VarT, Tau і T.

```
void RemoteControl()                  // Modify parameters remotely.
{
  while (Serial1.available() < 1)
  ;
  char WirelessCommand = Serial1.read();
  switch (WirelessCommand) {
    case 's':                                                    // Modify Safety Distance
of Optimal Velocity Model.
    case 'S':
    Serial1.println(F("Enter new 'Safety Distance' as ## cm:"));
    while (Serial1.available() < 2)
    ;                                                          // Wait for all the expected
values to be entered
    SafetyDistance = int(Serial1.read()) * 10;                // Convert next 2 values
    SafetyDistance += int(Serial1.read());                    // into a number.
    break;


    case 'a':                                                   // Modify 'Tau' of Optimal
Velocity Model.
    case 'A':
    Serial1.println(F("Enter new delay in drivers reaction 'VarT' as #.# seconds (type only the #
numbers, not the '.'):"));
    while (Serial1.available() < 2)
    ;                                                          // Wait for all the expected
values to be entered
    VarT = int(Serial1.read());                               // Convert next 2 values
    VarT += int(Serial1.read()) * 0.1;                        // into a number.
    break;

    case 'w':                                                   // Modify 'VarT' of Optimal
Velocity Model.
    case 'W':
    Serial1.println(F("Enter new 'Tau' parameter for OVM as #.# seconds (type only the # numbers,
not the '.'):"));
    while (Serial1.available() < 2)
    ;                                                          // Wait for all the expected
values to be entered
    Tau = int(Serial1.read());                                // Convert next 2 values
    Tau += int(Serial1.read()) * 0.1;                         // into a number.
    break;

    case 't':                                                   // Modify 'Tau' of Optimal
Velocity Model.
    case 'T':
    Serial1.println(F("Enter new 'T' parameter for OVM as #.# seconds (type only the # numbers,
not the '.'):"));
    while (Serial1.available() < 2)
    ;                                                          // Wait for all the expected
values to be entered
    T = int(Serial1.read());                                  // Convert next 2 values
    T += int(Serial1.read()) * 0.1;                           // into a number.
    break;
  }
}
```

3.8 Тест енергетичної автономності автомобілів-роботів

Дуже важливою для платформи є автономність і продуктивність роботизованих автомобілів, які працюють від 4 батарейок типу AA. Для такої платформи оптимальним варіантом були акумуляторні батареї, тому не потрібно постійно купувати нові. Спочатку було куплено та випробувано два різних типи батарей:

- Акумуляторні батареї Amazon Basics NiMH AA (2000 мАг) [13]

- Акумуляторні батареї NiMH AA великої ємності Amazon Basics (2400 мАг)[14]

Щоб перевірити їх, двох роботів із повністю зарядженими батареями було запущено на ¾ їхньої максимальної швидкості (конфігурація, яка зазвичай використовується), рухаючись по кільцевій лінії довжиною трохи більше 4 метрів. Кожні 15 хвилин вимірювався та фіксувався час, який вони витрачають на проходження кола, доки батареї більше не зможуть рухати їх.

Також було цікаво спостерігати за ефектом обробки зображень Піху Cam і руху на розряд батареї. Щоб отримати це, той самий експеримент повторили, але роботизовані автомобілі також вимірювали відстань до іншого автомобіля на кільці та адаптували швидкість до нього.

Після усереднення даних, отриманих у різних циклах, можна побудувати графік на рис. 3.7.

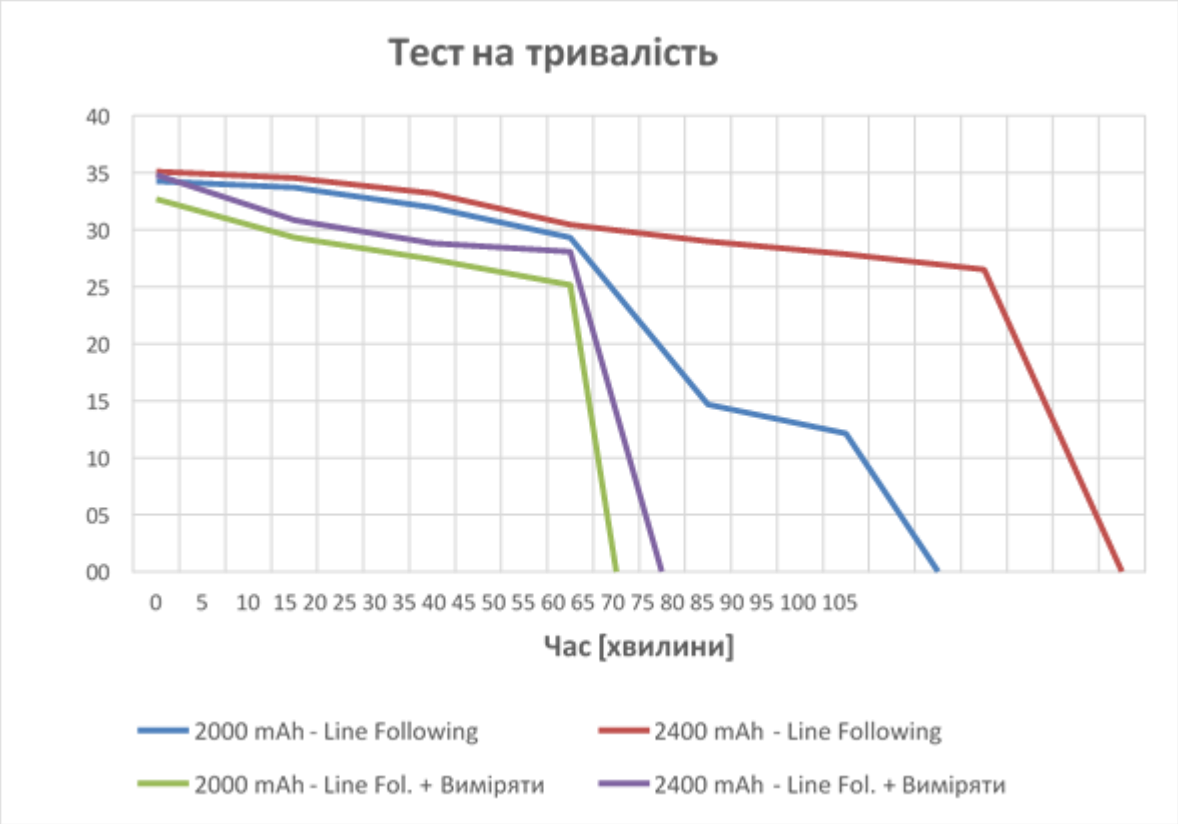


Рисунок 3.7 - Графік тесту автономності автомобілів-роботів

Як і очікувалося, батареї на 2000 мАг розрядилися раніше і дали трохи меншу продуктивність в цілому, ніж 2400 мАг. З графіка також чітко видно, що ввімкнення Pixy Cam значно скорочує час роботи батареї приблизно на 50–60%.

Загалом і враховуючи, що в реальних умовах роботизовані автомобілі не рухатимуться завжди на постійній високій швидкості, можна очікувати, що вони отримають середню автономність від 45 футів до 1 години з хорошою продуктивністю. Також було б доцільно встановлювати повністю заряджені батареї однакового типу для всіх автомобілів-роботів, що використовуються одночасно (і, якщо можливо, з максимальною ємністю), щоб мінімізувати різницю в продуктивності між транспортними засобами.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Вимоги охорони праці під час роботи з електроустаткуванням

Загальні положення

Інструкція з охорони праці для електрика при виконанні робіт з ремонту та обслуговування електроустаткування розроблена відповідно до Закону України «Про охорону праці» (Постанова ВР України від 14.10.1992 № 2694-XII) в редакції від 20.01.2018 р, на основі «Положення про розробку інструкцій з охорони праці», затвердженого Наказом Комітету по нагляду за охороною праці Міністерства праці та соціальної політики України від 29 січня 1998 року № 9 в редакції від 01 вересня 2017 року, з урахуванням «Правил технічної експлуатації електроустановок споживачів», затвердженими наказом Міністерства палива та енергетики 25.07.2006 р. № 258 (у редакції наказу Міністерства енергетики та вугільної промисловості України 13.02.2012 р. №91, «Правил безпечної експлуатації електроустановок споживачів», затверджених наказом Держнаглядохоронпраці України 09.01.1998 р. № 4.

Всі положення даної інструкції з охорони праці поширюються на електриків освітньої установи, які виконують роботи з ремонту та обслуговування електроустаткування.

До самостійного виконання робіт з ремонту та обслуговування електричного обладнання допускаються особи не молодше 18 років, які пройшли навчання за фахом, а також:

медичний огляд і не мають протипоказань за станом здоров'я до виконання даної роботи;

вступний і первинний на робочому місці інструктажі з охорони праці;

навчання безпечним методам і прийомам праці;

перевірку знань правил улаштування електроустановок, правил безпеки при експлуатації електроустановок, вимог охорони праці;

при ремонті і обслуговуванні електрообладнання напругою до 1000В мають групу з електробезпеки не нижче III, а понад 1000В - не нижче IV.

Електрику необхідно знати і виконувати вимоги інструкції з охорони праці при виконанні робіт з ремонту та обслуговування електроустаткування, інструкцій по роботі з ручним інструментом, електричним інструментом і драбинами.

Електрику при виконанні робіт з ремонту та обслуговування електроустаткування слід дотримуватися вимог Правил безпечної експлуатації електричних установок споживачів і Правил технічної експлуатації електричних установок споживачів, і мати відповідну групу з електробезпеки згідно з вимогами цих Правил.

Виконуючи роботи з ремонту та обслуговування електричного обладнання, може спостерігатися вплив нижчеперелічених шкідливих і небезпечних виробничих факторів:

падіння з висоти;

ураження електричним струмом;

підвищена напруженість електричного поля;

підвищена запиленість повітря робочої зони;

підвищений рівень вібрації;

недостатня освітленість робочої зони;

фізичні перевантаження;

нервово-психічні перевантаження.

Електрику при виконанні ремонту і обслуговування електроустаткування необхідно використовувати наведені нижче ЗІЗ:

напівкомбінезон бавовняний - на 12 місяців;

рукавички на - 3 місяці;

черевики шкіряні на - 24 місяці;

калоші діелектричні - чергові;

рукавиці діелектричні - чергові;

килимки діелектричні - чергові.

Електрик при ремонті і обслуговуванні електрообладнання зобов'язаний:

тримати у чистоті і порядку своє робоче місце;

дотримуватися Правил внутрішнього трудового розпорядку;

вміти застосовувати засоби індивідуального, колективного захисту, засоби пожежогасіння;

вміти надавати першу допомогу потерпілим від нещасних випадків;

знати і виконувати всі вимоги нормативних актів з охорони праці, правил протипожежного захисту та виробничої санітарії.

негайно повідомляти своєму безпосередньому керівнику про будь-який нещасний випадок, що трапився на виробництві, про ознаки професійного захворювання, а також про ситуацію, яка створює загрозу життю і здоров'ю людей;

знати терміни випробування захисних засобів і пристосувань, правила експлуатації, догляду та користування ними. Не дозволяється використовувати захисні засоби і пристосування з простроченим терміном перевірки;

виконувати тільки доручену роботу;

дотримуватися вимог інструкцій з експлуатації обладнання;

знати, де знаходяться засоби надання допомоги, первинні засоби пожежогасіння, головний і запасні виходи, шляхи евакуації в разі аварії або пожежі;

знати номери телефонів медичної установи (103) і пожежної охорони (101).

Електрик може відмовитися від виконання дорученої йому роботи, якщо виникла виробнича ситуація, яка становить загрозу для його життя і здоров'я оточуючих, або для навколишнього середовища, і доповісти про це своєму прямому керівнику.

На робочому місці заборонено курити, вживати алкогольні напої та інші речовини, які надають наркотичну дію на організм людини.

З метою запобігання отримання травм і виникнення травмонебезпечних ситуацій слід дотримуватися нижчеперелічених вимог:не можна залучати до роботи сторонніх осіб;

не починати роботу в разі відсутності умов для її безпечного виконання;

виконувати роботу тільки на справному обладнанні, зі справними пристроями та інструментом;

виявивши несправність терміново доповісти безпосередньому керівнику або усунути їх власними силами, якщо це відноситься до посадових обов'язків;

не торкатися неізольованих або пошкоджених проводів;

не виконувати роботу, яка не входить до професійних обов'язків.

Вміти надавати першу допомогу при кровотечах, переломах, опіках, ураженнях електричним струмом, раптовому захворюванні або отруєнні.

Дотримуватися правил особистої гігієни:

верхній одяг, головний убір і інші особисті речі слід залишати в гардеробі;

працювати в чистому спецодязі;

приймати їжу в призначеному для цього місці.

Вміти правильно користуватися ЗІЗ та засобами колективного захисту, первинними засобами пожежогасіння, протипожежним інвентарем, знати, де вони знаходяться.

Особи, які порушили цю інструкцію з охорони праці для електрика при виконанні робіт з ремонту та обслуговування електроустаткування, несуть дисциплінарну, адміністративну, матеріальну і кримінальну відповідальність відповідно до чинного законодавства України.

Вимоги безпеки перед початком роботи

Одягти спецодяг, провести огляд і підготовку робочого місця, прибрати зайві предмети.

Видалити із зони проведення робіт сторонніх осіб і звільнити робоче місце від сторонніх матеріалів та інших предметів, обгородити робочу зону і встановити знаки безпеки.

Переконатися в достатньому освітленні робочого місця, відсутність електричної напруги на відремонтованому обладнанні.

Оглянути на справність вимикачі, розетки електричної мережі, електровілок, електричних проводів, з'єднувальних кабелів, переконатися в наявності і справності ЗІЗ (засобів індивідуального захисту) і попереджувальних пристроїв (рукавичок діелектричних, окулярів захисних, калош, килимків і т. п.).

Виконуючи роботи з інструментом необхідно упевнитися в його справності, в відсутності механічних пошкоджень ізоляційного покриття і в своєчасності проходження випробувань інструменту.

Провести перевірку робочого місця на відповідність вимогам пожежної безпеки, на достатність освітлення робочого місця.

Виявивши недоліки і порушення з питань електричної і пожежної безпеки, негайно доповісти своєму безпосередньому керівнику.

4.2 Вимоги безпеки під час виконання робіт

Виконуючи посадові обов'язки, електрик зобов'язаний мати при собі посвідчення перевірки знань з питань охорони праці. За відсутності посвідчення або наявності посвідчення з терміном перевірки, працівник не отримує допуск до роботи.

Роботи в електричних установках щодо заходів безпеки поділяються на 3 категорії:

зі зняттям напруги;

без зняття напруги на струмопровідних частинах або біля них;

без зняття напруги віддалік від струмопровідних частин, що перебувають під напругою.

Працівники, які виконують спеціальні види робіт, до яких висуваються додаткові вимоги безпеки, повинні бути навчені безпечному проведенню таких робіт і мати про це відповідний запис в посвідченні про перевірку знань.

Працівникові, який обслуговує закріплені за ним електричні установки напругою до 1000 В одноосібно, необхідно мати III групу з електробезпеки.

Виконуючи роботи в електричних установках потрібно проводити організаційні заходи, що забезпечують безпеку робіт:

оформляти роботи нарядом-допуском, розпорядженням відповідно до переліку робіт, що виконуються в порядку поточної експлуатації;

проводити підготовку робочих місць;

допуск до роботи;

здійснювати контроль над виконанням робіт;

переводити на інше робоче місце;

установлювати перерви в роботі та її закінчення.

Для підготовки робочого місця до роботи, яка вимагає зняття напруги, необхідно застосувати, в певному порядку, наведені нижче технічні заходи:

виконати необхідні відключення і вжити всіх заходів, що виключають помилкове або самовільне включення комутаційної апаратури;

розвісити заборонні плакати на приводах ручного і на ключах дистанційного керування комутаційною апаратурою;

провести перевірку на відсутність напруги на струмопровідних частинах, які повинні бути заземлені для захисту людей від ураження електричним струмом;

встановити заземлення (включити заземлюючі ножі, застосувати переносні заземлення);

встановити огорожі, якщо необхідно, близько робочих місць або струмоведучих частин, що залишилися під напругою, а також вивісити на даних огорожах плакати безпеки.

в залежності від місцевих умов, струмовідні частини обгородити до чи після їх заземлення.

Працювати без зняття напруги на струмопровідних частинах або поблизу них слід як мінімум двом працівникам, одному з них, керівнику робіт, необхідно мати групу IV; іншим групу III з обов'язковим оформленням роботи нарядом-допуском або розпорядженням.

При знятті і встановлення запобіжників під напругою в електроустановках напругою до 1000 В слід заздалегідь відключити всі навантаження, які підключені до зазначених запобіжників; використовувати при цьому ізолюючі кліщі або діелектричні рукавички, а якщо є відкриті плавкі вставки, то і захисні окуляри.

Роботу з використанням драбин потрібно проводити вдвох, один з працівників повинен перебувати знизу. Стояти на ящиках або інших предметах забороняється. При установці приставних драбин на балках,

елементах металевих конструкцій і т. п. слід надійно закріпити верхню і нижню частину драбини на конструкціях.

Під час обслуговування та ремонту електричних установок користуватися металевими драбинами забороняється.

4.3 Вимоги безпеки після закінчення робіт з ремонту та обслуговування електроустаткування

Відключити (від'єднати) необхідне електрообладнання, електроінструмент від мережі.

Навести порядок на робочому місці, прибрати в спеціальні місця деталі, матеріал, сміття і відходи.

Прибрати у відведене місце весь інструмент і пристосування.

Зняти і прибрати спецодяг, ЗІЗ, ретельно помити руки.

Провести огляд робочого місця на відповідність його всім вимогам протипожежного захисту.

Повідомити своєму безпосередньому керівнику про недоліки і несправності, які були під час виконання роботи. Зафіксувати це в оперативному журналі.

Вимоги безпеки в аварійних ситуаціях

У випадку пожежі:

вимкнути електричне обладнання, припливно-витяжну вентиляцію, якщо вона є;

повідомити в пожежну частину за телефоном 101 і доповісти про це своєму керівнику, а при його відсутності іншій посадовій особі;

приступити до ліквідації осередка загоряння, застосовуючи передбачені для цього засоби пожежогасіння. Виконувати гасіння електричного обладнання, що знаходиться під напругою, можна тільки

вуглекислотними вогнегасниками типу ОУ або піском. Гасити їх водою або пінним вогнегасником забороняється.

Електрик повинен пам'ятати, що при раптовому відключенні напруги, вона може бути подана знову без попередження.

Слід швидко відключити механізми і пристрої:

в разі раптового відключення електроенергії,;

якщо подальша їх робота загрожує безпеці працівників;

в разі відчуття дії електричного струму при торканні металевих частин пускової апаратури;

в разі іскріння;

при найменших ознаках загоряння, появі диму, запаху гару;

якщо з'явився незнайомий шум.

У разі короткого замикання в мережі електроживлення необхідно знеструмити обладнання і повідомити своєму прямому керівнику.

Якщо сталося ураження електричним струмом, слід звільнити потерпілого від дії електричного струму, для чого відключити електричну мережу або від'єднати потерпілого від струмопровідних частин за допомогою діелектричних захисних засобів та інших ізолюючих речей і предметів (сухий одяг, суха жердина, прогумований матеріал і т. п.), або перерізати (перерубати) провід будь-яким інструментом з ізолюючою рукояткою, обережно, без додаткового нанесення травм потерпілому. До прибуття медпрацівника необхідно надати потерпілому першу допомогу.

При нещасних випадках (травмуванні людини) негайно повідомити про це безпосереднього керівника.

4.4 Розрахунок захисного заземлення

Захисне заземлення забезпечує зниження напруги дотику при замиканні на корпус до відносно безпечних значень шляхом зменшення потенціалу заземленого обладнання, вирівнювання потенціалів підвищенням потенціалів місця, на якому стоїть людина, до значень, що близькі до потенціалу заземлених конструктивних частин обладнання.

Розрахунок захисного заземлення має на меті визначення основних параметрів заземлення – кількість, розміри та порядок розміщення одиночних заземлювачів та заземлюючих провідників, при яких напруга дотику та кроку в період замикання фази на заземлений корпус не перевищує допустимих значень.

Розрахунок захисного заземлення здійснюється для випадку розташування заземлювача в однорідній землі. При цьому враховується опір верхнього шару землі (шар сезонних змін), який обумовленний замерзанням або засухою грунту. Розрахунок, який заснований на коефіцієнтах використання провідності заземлювача називається способом коефіцієнтів використання. Його виконують, як при простих, так і при складних конструкціях групових заземлювачів.

Загальні вимоги електробезпеки повинні відповідати ДСТУ 7237:2011. Для захисту від уражень електричним струмом використовують захисне заземлення. Воно повинно захищати людей від уражень електричним струмом у випадку дотику до металевих неструмопровідних частин, які можуть опинитись під напругою внаслідок пошкодження ізоляції, це досягається з’єднанням металевих частин електроустановок з землею, або її еквівалентом.

Згідно з класифікацією приміщень за ступенем небезпеки ураження електричним струмом (ПУЕ 1.1.6.), приміщення в якому проводяться всі роботи відноситься до першого класу (без підвищеної небезпеки). Під час

роботи використовуються електроустановки з напругою живлення 36 В, 220 В, та 360 В. Опір контура заземлення повинен мати не більше 4 Ом.

Розрахунок проводять за допомогою методу коефіцієнта використання (екранування) електродів. Коефіцієнт використання групового заземлювача η – це відношення діючої провідності цього заземлювача до найбільш можливої його провідності за нескінченно великих відстаней між його електродами.

При розрахунку заземлювачів в однорідній землі способом коефіцієнтів використання значення опору R захисного заземлення визначаємо в наступному порядку:

обчислюємо опір пристрою заземлення $R_з$. Згідно правил улаштування електроустановк (ПУЕ) найбільш припустимі значення $R_з$, складають для установок до 1000 В:

10 Ом при сумарній потужності генераторів або трансформаторів, що живлять дану мережу, не більше 100 кВА;

4 Ом у всіх інших випадках.

- визначаємо необхідний опір штучного заземлювача Rш:

$$R_{ш} = \frac{R_e \cdot R_з}{R_e - R_з}, \quad (4.1)$$

де R_e – опір розтікання природного заземлювача, Ом; $R_з$ – необхідний опір заземлюючого пристрою, Ом.

- обчислюємо кількість вертикальних і довжину горизонтальних електродів:

$$n = \frac{4 \cdot \sqrt{S}}{a'}, \quad (4.2)$$

де n – кількість вертикальних електродів, штук; S – площа цеху, м^2; a' – задана відстань між електродами, м.

$$l_г = 2a + 2b, \quad (4.3)$$

де $l_г$ – сумарна довжина горизонтальних електродів, м; а – ширина сторони цеху, м; b – довжина сторони цеху, м.

- розраховуємо опори розтікання вертикального Rв та горизонтального Rг електродів:

$$R_в = \frac{\rho_{роз.в}}{2 \cdot \pi \cdot l_в}\left(\ln\frac{2l_в}{d} + \frac{1}{2}\ln\frac{4t + l_в}{4t - l_в}\right), \quad (4.4)$$

де $\rho_{розв}$ – розрахунковий питомий опір землі для вертикального електрода, Ом·м;

$l_в$ – довжина вертикальних стрижневих електродів, м; d – діаметр електрода, мм;

t – глибина занурення в землю верхнього кінця електрода, м;

$$R_г = \frac{\rho_{роз.г}}{2 \cdot \pi \cdot l_г} \cdot \ln\frac{2l_г}{0{,}5 \cdot b' \cdot t}, \quad (4.5)$$

де $\rho_{роз.г}$ – розрахунковий питомий опір для горизонтального електрода, Ом·м;

$l_г$ – довжина горизонтальних електродів, м; b' – товщина горизонтального електрода, м.

- за даними таблиці 4.1 та таблиці 4.2 визначаємо коєфіцієнти використання для вертикальних та горизонтальних електродів $\eta_в$ та $\eta_г$ та з їх врахуванням обчислюємо розрахунковий опір заземлювача за виразом:

$$R_г = \frac{R_в \cdot R_г}{R_в \cdot \eta_г + R_г \cdot \eta_в \cdot n}, \quad (4.6)$$

Для розрахунку заземлювача задаємось такими вихідними даними: виробничий цех площею S=5000 м і з понижуючою підстанцією 10/0,4 кВ. Заземлювач передбачається виконати з вертикальних стрижневих електродів довжиною $l_в$=5 м, діаметром d=12 мм і відстанню між ними a'=5 м та горизонтальних електродів (сталева смуга перетином 440 мм) на глибині t=0,8 м. Розрахункова величина питомого опору ґрунту у місці спорудження захисного заземлення береться з таблиці 6.3 (для чорнозема с=20 Ом/м). Коєфіцієнти вертикальної прокладки $К_в$ і горизонтальної прокладки $К_г$ приймаються з таблиці 6.4 (для третього кліматичного району $К_в$=1,3, $К_г$=2,5).

Талиця 4.1 – Коефіцієнт використання горизонтального стрічкового електрода, що з’єднує вертикальні електроди (труби, кутики і ін.) групового заземлювача

Відношення відстані між вертикальним і електродами до їх довжин	Число вертикальних електродів							
	2	4	6	10	20	40	60	100
Вертикальні електроди розміщені в ряд								
1.	0,85	0,77	0,72	0,62	0,42	-	-	-
2.	0,94	0,80	0,84	0,75	0,5	-	-	-
3.	0,96	0,92	0,88	0,82	0,68	-	-	-
Вертикальні електроди розміщені по контуру								
1.	-	0,45	0,40	0,34	0,27	0,22	0,20	0,19
2.	-	0,55	0,48	0,40	0,32	0,29	0,27	0,23
3.	-	0,70	0,64	0,56	0,45	0,39	0,36	0,33

Талиця 4.2 – Коефіцієнт використання вертикальних електродів групового заземлювача (труб, кутиків, і т. ін.) без урахування впливу стрічки зв’язку

Число заземлювачів	Число вертикальних електродів					
	1.	2.	3.	1.	2.	3.
	Електроди, розміщені в ряд			Електроди, розміщені по контуру		
2	0,85	0,91	0,94	-	-	-
4	0,73	0,83	0,89	0,69	0,78	0,85
6	0,65	0,77	0,85	0,61	0,73	0,80
10	0,59	0,74	0,81	0,56	0,68	0,76
20	0,48	0,67	0,76	0,47	0,63	0,71
40	-	-	-	0,41	0,58	0,66
60	-	-	-	0,39	0,55	0,64
100	-	-	-	0,36	0,52	0,62

Таблиця 4.3 – Розрахункові значення питомих електричних опорів ґрунтів

Ґрунт	Значення, які рекомендуються для розрахунків, Ом/м
Пісок	700
Супісок	300
Суглинок	100
Глина	40
Чорнозем	20
Торф	20

Таблиця 4.4 – Значення підвищувальних коефіцієнтів $К_г$, $К_в$ за кліматичними зонами

Клімати чна зона	Тип заземлювачів	
	Горизонтально прокладені заземлювачі (смугові та ін.) при глибині від поверхні ґрунту t=0,8 м, $К_г$	Стрижневі вертикально встановлені заземлювачі при глибині від поверхні землі t=0,5–0,8 м, К
I	4,5–7	1,8–2
II	3,5–4,5	1,6–1,8
III	2,5–4	1,4–1,6
IV	1,5–2	1,2–1,4

Розрахункові питомі опори ґрунту для вертикальних і горизонтальних заземлювачів визначаються відповідно так:

$$\rho_{\text{роз.в}} = K_{\text{в}} \cdot \rho, \text{ Ом/м}, \quad (6.7)$$

$$\rho_{\text{роз.г}} = K_{\text{г}} \cdot \rho, \text{ Ом/м} \quad (6.8)$$

Таким чином за формулами (6.7), (6.8), для чорнозему:

$$\rho_{\text{роз.в}} = 1{,}3 \cdot 20 = 26, \text{ Ом/м};$$

$$\rho_{\text{роз.в}} = 2{,}5 \cdot 20 = 50, \text{ Ом/м}.$$

У якості природного заземлювача використовуємо металеву технологічну конструкцію з опором розтікання природного заземлювача

$R_е$=15 Ом.

Здійснюємо розрахунок у відповідності з зазначеною послідовністю:

- згідно ПУЕ необхідний опір заземлюючого пристрою складає:

$$R_З=4 \text{ Ом};$$

за формулою (6.1) визначеємо необхідний опір штучного заземлювача $R_ш$:

$$R_ш = \frac{15 \cdot 4}{15-4} = 5{,}5 \text{ Ом};$$

за формулами (4.2), (4.3) обчислюємо кількість вертикальних та довжин горизонтальних електродів:

$$n = \frac{4 \cdot \sqrt{5000}}{5} = 5 \text{ штук},$$

$$l_г = 2 \cdot 50 + 2 \cdot 100 = 300 \text{ м};$$

за формулами (4.4), (4.5) розраховуємо опори розтікання вертикального $R_в$ та горизонтального $R_г$ електродів:

$$R_в = \frac{26}{2 \cdot \pi \cdot 5}\left(\ln\frac{2 \cdot 5}{0{,}012} + \frac{1}{2}\ln\frac{4 \cdot 3{,}3 + 5}{4 \cdot 3{,}3 - 5}\right) = 5{,}7 \text{ Ом},$$

$$R_г = \frac{50}{2 \cdot \pi \cdot 300} \cdot \ln\frac{2 \cdot 300}{0{,}5 \cdot 0{,}04 \cdot 0{,}8} = 0{,}3 \text{ Ом};$$

за даними таблиць (4.10, (4.11) обираємо коєфіцієнти використання для вертикальних та горизонтальних електродів $К_в$=0,4 та $К_г$=0,21;

обчислюємо розрахунковий опір заземлювача R за формулою (4.6):

$$R_г = \frac{5{,}7 \cdot 0{,}3}{5{,}7 \cdot 0{,}21 + 0{,}3 \cdot 0{,}4 \cdot 56} = 0{,}22 \text{ Ом}.$$

Таким чином, проєктований заземлювач є контурним, складається з 56 вертикальних стрижневих електродів довжиною 5 м і діаметром 12 мм та горизонтального електрода у вигляді сталевої смуги довжиною 300 м, перетином 440 мм^2, занурених у землю на 0,8.

ОСНОВНІ ВИСНОВКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

У роботі було проведено розробку автоматизованої системи керування транспортним засобом з блютуз керування.

Підсумовуючи, створено основу роботизованої платформи для підключених і автоматизованих транспортних засобів. Платформа наразі може використовувати 14 функціональних автомобілів-роботів, здатних переміщатися по ланцюгах, дотримуючись чорної лінії на білій землі, вимірювати відстань до автомобіля попереду та контролювати їх швидкість до 50 см/с. Ними також можна дистанційно керувати з комп’ютера, реалізуючи зв’язок по блютузу.

Для створення різних схем на платформі є велика біла складна вінілова стрічка та чорна вінілова стрічка, тому її можна згортати та розгортати в різних місцях.

ПЕРЕЛІК ПОСИЛАНЬ

1. Arduino Robot. URL: http://www.arduino.cc/en/Main/Robot.
2. Robot Shield with Arduino. URL: https://www.parallax.com/product/32335.
3. Zumo Robot for Arduino. URL: https://www.pololu.com/product/2510.
4. Robot Kits and More. URL: http://www.makershed.com/products/make-rovera-2wd-arduino-robot-kit.
5. Sparki Robot. URL: http://www.dfrobot.com/index.php?route=product/product&path=37&product_id=1092.
6. SparkFun XBee Shield. URL: https://www.sparkfun.com/products/12847.
7. Arduino Stackable Header Kit. URL: https://www.sparkfun.com/products/11417.
8. Pixycam. URL: https://learn.adafruit.com/pixy-pet-robot-color-vision-follower-using pixycam/overview.
9. https://learn.sparkfun.com/tutorials/exploring-xbees-and-xctu.
10. http://cmucam.org/projects/cmucam5/wiki/Teach_Pixy_an_object.
11. https://learn.adafruit.com/pixy-pet-robot-color-vision-follower-using-pixycam/pixypet-code.
12. http://www.amazon.com/AmazonBasics-AA-Rechargeable-Batteries-Pack/dp/B007B9NV8Q/ref=sr_1_1?s=hpc&ie=UTF8&qid=1439411027&sr=1-1&keywords=rechargeable+batteries.
13. http://www.amazon.com/AmazonBasics-High-Capacity-Rechargeable-BatteriesPre-charged/dp/B00HZV9WTM/ref=sr_1_3?s=hpc&ie=UTF8&qid =1439411027&sr=1- 3&keywords=rechargeable+batteries.
14. Y. Sugiyama, M. Fukui, M. Kikuchi, K. Hasebe, A. Nakayama, K. Nishinari, S.-i. Tadaki and S. Yukawa. Traffic jams without bottlenecks - experimental evidence for the physical mechanism of the formation of a jam. New Journal of Physics. vol. 10, 2008.

15. Rechargeable batteries. URL: http://www.amazon.com/EBL%C2%AE-Version-Charger-RechargeableBatteries/dp/B00EB7812C/ref=sr_1_2?s=hpc&ie=UTF8&qid=1439411027&sr=1- 2&keywords=rechargeable+batteries.

16. А.Г. Микитишин, М.М. Митник, П.Д. Стухляк, В.В. Пасічник Комп'ютерні мережі. Книга 1. [навчальний посібник] (Лист МОНУ №1/11-8052 від 28.05.12р.) - Львів, "Магнолія 2006", 2013. – 256 с.

17. А.Г. Микитишин, М.М. Митник, П.Д. Стухляк, В.В. Пасічник Комп'ютерні мережі. Книга 2. [навчальний посібник] (Лист МОНУ №1/11-11650 від 16.07.12р.) - Львів, "Магнолія 2006", 2014. – 312 с.

18. Микитишин А.Г., Митник, П.Д. Стухляк. Комплексна безпека інформаційних мережевих систем: навчальний посібник – Тернопіль: Вид-во ТНТУ імені Івана Пулюя, 2016. – 256 с.

19. Микитишин А.Г., Митник М.М., Стухляк П.Д. Телекомунікаційні системи та мережі : навчальний посібник для студентів спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» – Тернопіль: Тернопільський національний технічний університет імені Івана Пулюя, 2017 – 384 с.

20. Введення в компютерну графіку та дизайн: Навчальний посібник для студентів спеціальності 174 "Автоматизація, компютерно-інтегровані технології та робототехніка"/Укладачі: О.В. Тотосько, П.Д. Стухляк, А.Г. Микитишин, В.В. Левицький, Р.З. Золотий - Тернопіль: ФОП Паляниця В.А., 2023 - 304с. http://elartu.tntu.edu.ua/handle/lib/41166.

21. Пилипець М. І. Правила заповнення основних форм технологічних документів : навч.-метод. посіб. / Уклад. Пилипець М. І., Ткаченко І. Г., Левкович М. Г., Васильків В. В., Радик Д. Л. Тернопіль : ТДТУ, 2009. 108 с. https://elartu.tntu.edu.ua/handle/lib/42995.