МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

ФАКУЛЬТЕТ МЕХАТРОНІКИ ТА КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Алгоритмічне та програмне забезпечення
для тестування сенсорних пристроїв, напрями удосконалення»

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки

Освітня програма Комп'ютерні науки

Виконав: студент групи МгІТ-2-22

Сурженко Ростислав Русланович

Науковий керівник к.т.н., доц. Колиско О.З.

Рецензент

Київ 2023

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Факультет мехатроніки та комп'ютерних технологій
Кафедра комп'ютерних наук
Рівень вищої освіти другий (магістерський)
Спеціальність 122 Комп'ютерні науки
Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
_____Володимир ЩЕРБАНЬ.
"____" _____2023 року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Сурженку Ростиславу Руслановичу

1. Тема кваліфікаційної роботи Алгоритмічне та програмне забезпечення для тестування сенсорних пристроїв, напрями удосконалення, науковий керівник роботи Колиско Оксана Зенонівна, доц.,к.т.н. затверджені наказом КНУТД від "_12_"вересня_2023 року №_210-уч_

2. Вихідні дані до роботи: Розробки кафедри комп'ютерних наук; рекомендована література, додатки.

3. Зміст дипломної роботи: Вступ; РОЗДІЛ 1 Постановка задачі; РОЗДІЛ 2 Проектування; РОЗДІЛ 3 Програмна реалізація; Висновки; Список літератури; ДОДАТОК А Окремі фрагменти програмного коду; ДОДАТОК Б Презентація.

4. Дата видачі завдання_1 вересня 2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапу кваліфікаційної роботи	Орієнтовний термін виконання	Примітка про виконання
1	Вступ	*15.09.2023*	
2	Розділ 1. Постановка задачі	*20.09.2023*	
3	Розділ 2 Проектування	*30.09.2023*	
4	Розділ 3. Програмна реалізація	*10.10.2023*	
5	Висновки	*25.10.2023*	
6	Оформлення (чистовий варіант)	*1.11.2023*	
7	Подача кваліфікаційної роботи науковому керівнику для відгуку (за 14 днів до захисту)	*4.11.2023*	
8	Подача кваліфікаційної роботи для рецензування (за 12 днів до захисту)	*6.11.2023*	
9	Перевірка кваліфікаційної роботи на наявність ознак плагіату (за 10 днів до захисту)	*8.11.2023*	
10	Подання кваліфікаційної роботи на затвердження завідувачу кафедри (з 7 днів до захисту)	*10.11.2023*	

З завданням ознайомлений:

Студент _____ Ростислав СУРЖЕНКО

Науковий керівник _____ Оксана КОЛИСКО

АНОТАЦІЯ

Сурженко Р.Р. Алгоритмічне та програмне забезпечення для тестування сенсорних пристроїв. напрями удосконалення. – Рукопис.

Дипломна магістерська робота за спеціальністю 122 – Комп'ютерні науки. – Київський національний університет технологій та дизайну, Київ, 2023 рік.

Дана магістерська робота присвячена дослідженню та розробці комплексних алгоритмів та програмного забезпечення для ефективного та точного тестування двох важливих категорій сенсорних пристроїв: інфрачервоних сенсорів та мікрохвильових сенсорів. Ці два типи сенсорних пристроїв відіграють важливу роль у сучасних.

У цій статті спочатку надається огляд сучасних розробок і характеристик інфрачервоних і мікрохвильових сенсорів, а також описується їхня базова структура і функції. Потім пропонуються методи та програмне забезпечення для створення та проведення різноманітних тестів для визначення точності, надійності та інших характеристик обох сенсорів.

Ключові слова Алгоритми, програмне забезпечення, тестування, сенсорні пристрої, інфрачервоні датчики, мікрохвильові датчики, розробка, огляд, точність, сучасність, структура, функціональність, стабільність.

ANNOTATION

Surzhenko R.R. Algorithmic and software for testing sensor devices. areas of improvement. - Manuscript.

Master's degree work on the specialty 122 Computer science. - Kyiv National University of Technology and Design, Kyiv, 2023.

This master's thesis is dedicated to the development and development of complex algorithms and software for the effective and accurate testing of two important categories of sensory devices: infrared sensors and microhelical sensors. These two types of sensory devices play an important role in everyday life.

This article first provides an overview of the current developments and characteristics of infrared and microvector sensors, and also describes their basic structure and function. Then, methods and software will be introduced to create and conduct various tests to determine the accuracy, reliability and other characteristics of both sensors.

Keywords: Algorithms, software, testing, sensory devices, infrared sensors, micro-hair sensors, analysis, inspection, accuracy, consistency, structure, functionality, stability.

ЗМІСТ

ВСТУП

Нині, досягнення сучасних технологій мають значний вплив на багато аспектів людської життя. Розробка нових систем та технологій потребує відповідного програмного забезпечення, призначеного для тестування та перевірки надійності, точності та коректності роботи різних сенсорних пристроїв.

Однією з ключових проблем є удосконалення процесу тестування та розробка універсального програмного забезпечення для тестування сенсорних пристроїв. Важливість цього питання особливо очевидна при тестуванні датчиків руху, таких як інфрачервоні та мікрохвильові сенсори, які мають високу чутливість і тому вимагають високої точності.

На сьогоднішній день створення універсальних програм, які можуть візуально контролювати реакції та спрацьовування сенсорів, зберігати та аналізувати дані є актуальним завданням у розвитку сучасних технологій.

Метою даної магістерської роботи є створення універсального програмного забезпечення для тестування сенсорних пристроїв, зокрема інфрачервоних та мікрохвильових сенсорів, та визначення їх точності, стійкості до зовнішніх впливів та інших характеристик.

Цілі даної роботи полягають у наступному:

1. Розробити алгоритми та програмне забезпечення для тестування та аналізу різних сенсорних пристроїв, у тому числі інфрачервоних та мікрохвильових сенсорів
2. Проведення експериментальних досліджень для визначення точності, стабільності та інших характеристик інфрачервоних та мікрохвильових сенсорів
3. Виявлення несправностей та формулювання рекомендацій щодо покращення характеристик сенсорних пристроїв на основі отриманих результатів.

Об'єктами дослідження є інфрачервоні та мікрохвильові сенсори. Інфрачервоні та мікрохвильові сенсори призначені для виявлення руху об'єктів

у межах свого робочого діапазону та відіграють важливу роль у різних додатках завдяки своїй здатності виявляти рух.

Наукові інтереси включають розробку програмного забезпечення та алгоритмів для підвищення функціональності та надійності інфрачервоних та мікрохвильових сенсорів у різних сферах систем безпеки.

Науковою новизною є розробка комплексного програмного забезпечення, що включає алгоритми для тестування різних типів датчиків, у тому числі ІЧ та СВЧ сенсорів. Іншою інновацією є використання алгоритмів тестування та аналізу результатів, що дозволяє виявляти помилки в роботі сенсорних пристроїв.

Практичне значення полягає в тому, що розроблене програмне забезпечення та алгоритми підвищують якість та надійність сенсорних пристроїв. Це дозволяє економити час та ресурси під час тестування, що є особливо важливим у відповідальних галузях промисловості.

Для досягнення поставлених цілей були використані наступні методи дослідження

- Аналіз наукової літератури та патентів для вивчення існуючих методів тестування сенсорних пристроїв.
- Моделювання та емуляція для створення тестових сценаріїв та експериментальних умов.
- Експериментальні дослідження з використанням реальних інфрачервоних та мікрохвильових сенсорів для перевірки розробленого програмного забезпечення та алгоритмів.

Результати цього дослідження були успішно апробовані в компанії, що спеціалізується на виготовленні та вдосконаленні сенсорних пристроїв. Позитивні відгуки та рекомендації, отримані від фахівців, підтвердили, що розроблене програмне забезпечення та алгоритми є придатними та ефективними для тестування сенсорних пристроїв.

РОЗДІЛ 1. ОГЛЯД ТЕХНІК И МЕТОДОЛОГІЙ ТЕСТУВАННЯ

1.1 Поняття тестування

Тестування програмного забезпечення передбачає насамперед, порівняння фактичної та очікуваної функціональності програми. Ця процедура розглядає наскільки відповідає програмний продукт визначеним характеристикам та вимогам. Вона також включає оцінку безпеки, зручності використання та ергономіки.

В залежності від мети тестування аналізуються конкретні характеристики програми або веб-сайту. Процес тестування зазвичай документується шляхом створення плану тестування та тестових кейсів, а також іншої інформації. План тестування описує стратегії, методи, інструменти, процедури та інші аспекти тестування.

У процесі тестування можна оцінити наступні аспекти

- Функціональність програми
- Швидкість роботи
- Стабільність під великим навантаженням
- Споживання ресурсів, наприклад, пам'яті та диска
- Стабільність роботи
- Сумісність з різними платформами
- Зручність користувацького інтерфейсу
- Робота в мережі
- Взаємодія з апаратним забезпеченням тощо.

Тест-кейси детально описують ряд етапів, на яких перевіряється функціональність додатку або веб-сайту.

1.2 Класифікація тестування

Спочатку детальніше розглянемо класифікацію тестів.

Таблиця 1 ілюструє всі методи класифікації одночасно. Тут найбільш характерні випадки позначені межами блоків у вигляді набору кольорів та крапок.

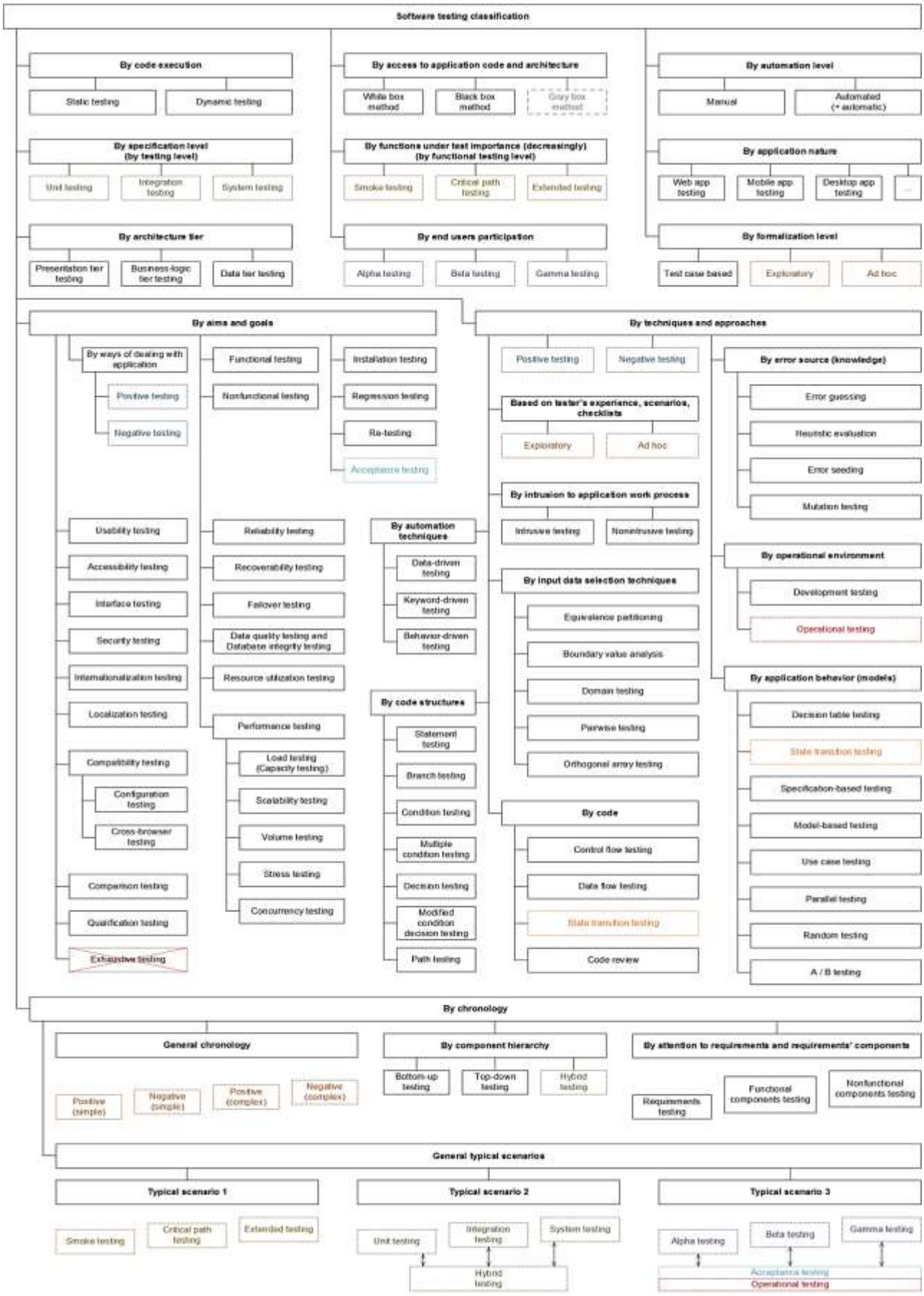
Наведена класифікація допоможе впорядкувати знання, значно прискорити процес розробки та планування тестів та оптимізувати трудовитрати.

У той же час, ніщо не заважає створювати свої власні класифікації, як розроблені з нуля для конкретного завдання або компанії, так і комбінації чи модифікації з класифікацій, представлених нижче.

Доцільно зупинитися на основних модулях, які часто використовуються і тісно пов'язані між собою:

1. Модульне тестування;
2. Інтеграційне тестування;
3. Системне тестування;
4. Приймальне тестування.

Таблиця 1.Класифікація тестування



1.2.1 Модульне тестування для підприємств

По суті, модульне тестування – це перевірка програми на рівні окремих модулів, функцій чи класів. Мета модульного тестування полягає у виявленні помилок, які можуть бути локалізовані в конкретному модулі, що стосуються реалізації алгоритмів, а також оцінці готовності системи для переходу на наступний етап розробки та тестування. Модульне тестування є необхідною та актуальною складовою для підприємств, що працюють у галузі розробки датчиків руху з наступних причин:

1. Виявлення несправностей на ранній стадії: модульне тестування дозволяє виявити та усунути помилки у реалізації окремих модулів та програмних компонентів на ранній стадії розробки, тим самим підвищуючи загальну якість продукту та знижуючи витрати на подальші роботи.

2. Підвищити надійність системи: модульне тестування дозволяє визначити надійність окремих модулів та компонентів. Це важливо для забезпечення безперебійної роботи пристрою та запобігання збоям.

3. Ефективність розробки: дозволяє розробникам сконцентруватися на окремих частинах програмного коду, спрощуючи процес налагодження та покращуючи функціональність.

4. Оптимізація ресурсів: виявлення проблем на ранній стадії знижує вартість подальших виправлень та економить ресурси компанії.

5. Забезпечення сумісності: модульне тестування дозволяє перевірити, як окремі модулі взаємодіють один з одним і чи сумісні вони з іншим обладнанням та програмним забезпеченням.

Усі ці аспекти роблять модульне тестування необхідним етапом у розробці датчиків руху, забезпечуючи покращення якості продукту та зменшення ризиків.

1.2.2 Інтеграційне тестування

Інтеграційне тестування - це тестування частини системи, яка складається з двох чи більше модулів. Основною метою інтеграційного тестування є пошук дефектів, пов'язаних з помилками в реалізації та

інтерпретації взаємодії між модулями. Основна відмінність між модульним і інтеграційним тестуванням полягає в типах виявлених дефектів.

Воно дає можливість перевірити взаємодію окремих компонентів, з яких складається датчик руху. Це включає сенсори, мікроконтролери, інтерфейси та інше.

Інтеграційне тестування дозволяє вчасно виявити помилки, що можуть виникнути під час об'єднання окремих компонентів. Деякі проблеми можуть бути складніше виявити на ранніх етапах розробки.

Важливо переконатися, що датчик руху може працювати спільно з іншими системами та пристроями, такими як смартфони, комп'ютери, системи безпеки тощо. Інтеграційне тестування гарантує сумісність з іншим обладнанням.

Під час інтеграційного тестування можна оцінити стабільність та продуктивність пристрою у реальних умовах. Це важливо для забезпечення комфортного користування.

Якщо датчик руху планується для продажу, то важливо впевнитися, що він відповідає стандартам та вимогам. Інтеграційне тестування допомагає готувати пристрій до сертифікації.

1.2.3 Тестування системи.

Головною метою проведення системного тестування є виявлення широкого спектра дефектів, які впливають на функціонування системи в її загальному вигляді. Серед цих дефектів можуть бути такі аспекти:

1. Некоректне використання ресурсів, яке може призвести до неефективного або навіть вичерпного використання ресурсів системи.

2. Непередбачені та недопущені комбінації даних, що виникають на рівні користувача та можуть призвести до виникнення помилок чи некоректних результатів.

3. Несумісність з програмним оточенням, що може перешкоджати взаємодії системи з іншими компонентами або програмами.

4. Непередбачені сценарії використання, які можуть виникнути в

реальних умовах експлуатації та вимагати виправлень чи додаткового функціоналу.

5. Відсутність або неправильна функціональність, що включає в себе недоліки в реалізації функцій, що вимагають вдосконалення або виправлень.

6. Незручність в застосуванні, яка може вплинути на зручність користування системою і вимагати розробки інтерфейсу або внесення змін у взаємодію з користувачем та інші аспекти.

Виходячи з міркувань, такий ретельний аналіз та виявлення дефектів на рівні системи в цілому сприяють покращенню її продуктивності, надійності та забезпечують більш зручне користування, сприяючи високій якості та ефективності системи.

1.2.4 Приймальне тестування.

Приймальне тестування - це формальний процес тестування, який перевіряє відповідність системи вимогам і проводиться з метою:

Визначення, чи відповідає система критеріям прийому;

•Прийняття рішення замовником або іншою уповноваженою особою про те, чи приймати додаток чи ні.

Приймальне тестування проводиться на основі набору типових тестових випадків і сценаріїв, які розробляються на основі вимог до даного додатка. Рішення про проведення приймального тестування приймається тоді, коли:

•Продукт досяг необхідного рівня якості;

•Замовник ознайомлений з планом робіт з прийому або іншим документом, в якому описаний набір дій, пов'язаних з проведенням приймального тестування, дата проведення, відповідальні особи тощо.

Фаза приймального тестування триває до того часу, поки замовник не приймає рішення про подальші кроки, такі як виправлення додатка чи його прийом. На прикладі підприємства з розробки та тестування датчиків руху, проведення приймального тестування виграє особливу актуальність і значення за декількома фундаментальними причинами:

1. Забезпечення якості: Приймальне тестування є необхідним кроком для переконання, що виготовлені датчики руху відповідають встановленим стандартам та вимогам якості. Це допомагає забезпечити високу якість та надійність продукції, що є критично важливим для задоволення потреб споживачів та уникнення можливих проблем і ризиків.

2. Аспекти безпеки: У багатьох випадках датчики руху використовуються як складові частини систем безпеки та систем контролю доступу. Несправні датчики можуть призвести до небажаних ситуацій і порушити безпеку. Проведення приймального тестування гарантує відповідність продукції стандартам безпеки.

3. Відповідність вимогам: Замовники зазвичай мають конкретні вимоги до датчиків руху, які повинні бути точно виконані. Приймальне тестування дозволяє переконатися, що вироблені датчики відповідають цим вимогам.

4. Надійність і тривалість служби: Датчики руху використовуються в різних галузях, включаючи промисловість, домашнє використання та багато інших сфер. Вони мають бути надійними та мають тривати довгий час. Проведення приймального тестування допомагає переконатися, що вони можуть працювати без збоїв на протязі тривалого періоду.

5. Задоволення клієнтів: Висока якість продукції є ключовою для задоволення потреб і очікувань клієнтів. Проведення приймального тестування гарантує, що клієнти отримують надійні та високоякісні датчики руху, що може сприяти покращенню репутації компанії.

Це дозволяє зробити висновок, що приймальне тестування є важливим кроком у виробництві датчиків руху, оскільки воно допомагає забезпечити відповідність продукції вимогам якості, безпеки та функціональності, а також сприяє задоволенню клієнтів та підвищує надійність та тривалість служби виробленої продукції.

1.3 Види тестування

В першу чергу, слід приділити особливу увагу вибору виду тестування.

Правильний вибір виду та підходу у тестуванні є фундаментом та важливим кроком у розробці датчиків руху, і він може суттєво вплинути на якість та надійність кінцевого продукту.

1.3.1 Функціональне тестування.

Функціональне тестування розглядає заздалегідь вказану поведінку і ґрунтується на аналізі специфікацій функціональності компоненту або системи в цілому.

Функціональні тести базуються на функціях, які виконує система, і можуть проводитися на всіх рівнях тестування (компонентному, інтеграційному, системному, прийомному). Зазвичай ці функції описуються в вимогах, функціональних специфікаціях або у вигляді сценаріїв використання системи.

Звідси виходить, що тестування функціональності може проводитися з двох поглядів:

• Вимоги;

• Бізнес-процеси.

Тестування з погляду "вимог" використовує специфікацію функціональних вимог до системи як основу для проектування тестових кейсів. У цьому випадку потрібно створити список того, що буде тестуватися, і що не буде, встановити пріоритет вимог на основі ризиків (якщо цього не зроблено в документі з вимогами), і на цій основі розробити тестові сценарії. Це дозволить зосередитися на найважливішій функціональності під час тестування.

Тестування з погляду "бізнес-процесів" використовує знання саме цих бізнес-процесів, які описують сценарії щоденного використання системи. У цій перспективі тестові сценарії, як правило, базуються на випадках використання системи.

Переваги функціонального тестування:

• Імітує фактичне використання системи; Недоліки функціонального тестування:

• Ємність пропускання логічних помилок у програмному забезпеченні;
• Ймовірність надмірного тестування.

1.3.2 Конфігураційне тестування.

Конфігураційне тестування дозволяє перевірити, як програма себе веде при різних роздільних здатностях екрану, в різних браузерах, на різних операційних системах, з різним програмним та апаратним забезпеченням.

1.3.3 Тестування на навантаженість.

Цей тип тестування дозволяє визначити рівень критичних навантажень під час роботи з БД, інтернет-серверами, мережами та іншими ресурсами. За допомогою автоматизованих тестів можна відтворити типові сценарії дій користувача та багаторазово збільшити їх кількість, таким чином, симулюючи, як система поводить себе при 100 або 10 000 активних користувачах.

1.3.4 Тестування usability.

Оцінюється зручність користування продуктом. Досліджується на прикладі групи випробуваних того, як користувач сприймає продукт, як він уявляє шляхи його використання, скільки часу витрачає на певну дію, які проблеми виникають і чи може він їх вирішити.

1.4 Автоматизоване тестування

По перше, автоматизоване тестування програмного забезпечення - це частина процесу контролю якості на етапі розробки програмного забезпечення. Воно використовує програмні засоби для виконання тестів та перевірки результатів, що допомагає скоротити час тестування і спростити процес.

Тестування в автоматичному режимі за допомогою програмних сценаріїв дозволяє автоматизувати процес. Це зручно, коли продукт часто змінюється, але при цьому має велику функціональність. Автоматизоване тестування дозволяє переконатися, що всі функції продукту залишаються стабільними після внесення змін.

Автоматизоване тестування можна включити в цикл розробки програм,

виконуючи його після кожної нової збірки або створення нової версії. При цьому не потрібно присутність людини.

Поруч з цим, автоматизоване тестування - це не лише виконання тестів. Автоматизація може бути представлена в більшості процесів тестування:

1) **Планування і контроль.** Тут обирається необхідний обсяг тестів на основі ризиків або іншої методики; розподіл тестів між співробітниками; контроль за їх виконанням; визначення кількості тестів, їх стану та властивостей.

2) **Аналіз і звітність.** Передбачає побудову звітів різних форм та напрямків на основі даних про проведену роботу, як для потреб робочого колективу, так і для вищого керівництва.

3) **Контроль помилок**. Сюди входять багтрекінгові системи різних варіацій.

Створення тестів на основі отриманих вимог. Можлива як генерація методом запису і відтворення, так і ручне написання тестів на одній із багатьох підтримуваних обраною системою тестування мов програмування.

4) **Аналіз покриття/трасування.** Виконується для оцінки покриття коду, вимог або певного обсягу функціональності різними типами тестів, створеними на попередньому етапі.

5) **Виконання тестів.** Запуск тестових сценаріїв та аналіз отриманих результатів.

Відповідним чином, автоматизоване тестування має свої переваги:

• Розширене охоплення коду;

• Мінімізація впливу людського фактору при повторному перевірянні тест-кейсів;

• Записані кроки можна використовувати безліч разів;

• Забезпечення, що жоден тест не буде пропущений;

• Перевірка однакових кроків в тестах. Але є й недоліки:

• Помилки переважно виявляються при створенні скрипта;

• У скриптах також можуть бути помилки.

Іноді зусилля, витрачені на розробку автоматизованих тестів, можуть значно перевищити ручний варіант виконання, навіть якщо тести доводиться виконувати щодня. Деякі випадки тестування просто неможливо автоматизувати з різних причин:

• Складність алгоритмізації перевірки;

• Нестандартна конфігурація стенду, яку потрібно змінювати для різних тестів;

• Стороннє ПЗ, яке створює додаткові проблеми;

• Та інші причини.

Виходить, автоматизація стає корисною лише для продуктів, розробка та підтримка яких тривають достатньо довго, або для невеликих продуктів з максимально схожими інтерфейсами та API.

1.5 Конклюзія до автоматизованого тестування

Спершу слід відзначити, що ручне і автоматизоване тестування - це технології, які доповнюють одна одну. Тому в сучасних умовах важко створити якісний продукт без автоматизації, але також неможливо обійтися без ручного тестування.

По-друге, слід відзначити, що існують різні види автоматизованих тестів, і слід вибирати ті, які найбільше відповідають конкретному проекту.

Отже:

- Автоматизація може застосовуватися в більшості процесів та на всіх рівнях тестування.
- Ручне та автоматизоване тестування - це взаємодоповнюючі технології.
- Автоматизоване тестування передбачає участь людини.
- Автоматизоване тестування потребує додаткових інвестицій, але дозволяє підвищити якість продукту.
- Автоматизоване тестування - це розробка (програмування).
- Автоматизоване тестування гарантує детерміновану перевірку функціональності.

- Супровід автоматизованих тестів вимагає додаткових витрат при активних змінах системи, яку тестують.

1.6 Моделі розробки програмного забезпечення

Для кращого розуміння взаємозв'язку між тестуванням, програмуванням та іншими аспектами проектної роботи, давайте спочатку розглянемо основні поняття, такі як моделі розробки програмного забезпечення.

Модель розробки програмного забезпечення (Software Development Model, SDM) - це систематична структура, яка організовує різні типи проектної діяльності, їх взаємодію та послідовність в процесі розробки програмного забезпечення.

Вибір конкретної моделі залежить від різних чинників, таких як розмір і складність проекту, галузь застосування, наявні ресурси та багато інших.

1.6.1 Водоспадна модель

Водоспадна (каскадна) модель життєвого циклу ПЗ, також відома як модель водоспаду, представляє собою послідовний підхід до розробки програмного забезпечення. Ім'я "водоспадна" походить від структури цієї моделі, яка нагадує водоспад, де один етап слідує іншому

Цей метод розробки взятий з інших галузей, таких як системна інженерія в виробництві і будівництві, де зміни на пізніших етапах процесу дуже дорогі або навіть неможливі. Наприклад, у будівництві зміни фундаменту після закінчення будівництва даху вимагають великих витрат. Саме тому на ранніх етапах проектування розробка повинна бути максимально докладною і точною.

Перші розробники програмного забезпечення, які прийшли з інших галузей, просто адаптували цей метод, оскільки на той час не було спеціалізованих методологій для розробки ПЗ.

Ця модель була формалізована В. В. Ройсом у 1970 році і, незважаючи на те, що його стаття містила критику цього методу, вона стала дуже популярною і досі використовується в галузях, де вимоги є фіксованими і

потребують високої надійності, наприклад у військовій або медичній сферах.

Основною особливістю водоспадної (каскадної) моделі є те, що перехід на наступний етап відбувається лише після повного завершення роботи на поточному етапі. Назад переходити на попередні стадії не передбачено.

Кожен етап завершується створенням результатів, які стають вхідними даними для наступного етапу, і включає розробку повного набору документації. Вимоги до програмного забезпечення, сформульовані на стадії формування вимог, документуються у вигляді технічного завдання і залишаються незмінними на протязі всього процесу розробки.

Ключовим показником якості розробки за такої моделі є точність виконання специфікацій технічного завдання.

1.6.2 V-подібна модель

V-подібна модель представляє собою логічний розвиток водоспадної моделі. Слід відзначити, що в загальному випадку як водоспадна, так і V-образна моделі життєвого циклу програмного забезпечення можуть містити однаковий набір етапів, але вирішальна відмінність полягає в способі використання цієї інформації під час реалізації проекту.

У зв’язку з цим, при використанні V-образної моделі на кожному етапі "спуску" необхідно розмірковувати про те, як саме відбудеться наступний етап "підйому". Тестування вже починається на дуже ранніх стадіях розробки проекту, що дозволяє мінімізувати ризики і виявити та виправити безліч потенційних проблем до того, як вони стануть реальними проблемами.

1.6.3 Ітеративно-інкрементна модель

Модель ітеративно-інкрементної розробки є основою сучасного підходу до створення програмного забезпечення. Звідси випливає подвійність:

• З точки зору життєвого циклу, ця модель є ітеративною, оскільки передбачає багатократне повторення одних і тих самих етапів;

• З точки зору розвитку продукту (додавання його корисних функцій), модель є інкрементною.

Основною особливістю цієї моделі є розбиття проекту на відносно невеликі періоди (ітерації), кожна з яких загалом може включати всі класичні стадії, притаманні водоспадним і V-подібним моделям. Результатом ітерації є приріст (інкремент) функціональності продукту, виражений у проміжному збірці.

Довжина ітерацій може змінюватися в залежності від числа факторів, але сам принцип багатократного повторення гарантує активне застосування тестування і демонстрування продукту кінцевому замовнику (з отриманням зворотного зв'язку) з самого початку і протягом усього періоду розробки проекту.

У багатьох випадках допускається паралельне проведення окремих стадій в межах ітерації та активне доопрацювання для виправлення недоліків, виявлених на будь-якій з (попередніх) стадій.

Модель ітеративно-інкрементної розробки добре себе зарекомендувала на великих і складних проектах, що виконуються великими командами протягом тривалих періодів. Однак до основних недоліків цієї моделі часто відносять високі накладні витрати, зумовлені високою "бюрократизованістю" і загальною важкістю моделі.

1.6.4 Спіральна модель

Спіральна модель є варіацією ітеративно-інкрементної моделі та приділяє особливу увагу управлінню ризиками, зокрема тими, що впливають на організацію процесу розробки проекту та контрольні точки.

В спіральній моделі чітко можна виділити чотири ключові фази:

• Розробка цілей, альтернатив і обмежень;

• Аналіз ризиків і створення прототипів;

• Розробка (проміжної версії) продукту;

• Планування наступного циклу.

С точки зору тестування та управління якістю велика увага до ризиків є значущою перевагою при використанні спіральної моделі для розробки концептуальних проектів, де вимоги натуральним чином складні та

нестабільні і можуть багаторазово змінюватися протягом розробки проекту.

1.6.5 Гнучка модель

Гнучка модель є сукупністю різних підходів до розробки програмного забезпечення та базується на так званому "гнучкому маніфесті", що включає в себе наступні принципи:

• Люди і їх взаємодія більш значуще, ніж процеси і інструменти.

• Робочий продукт більш вагоміший, ніж вичерпна документація.

• Співпраця зі замовником важливіше, ніж узгодження умов контракту.

• Готовність до змін у пріоритеті, ніж дотримання початкового плану.

Легко помітити що, ці підходи закладені в основу гнучкої моделі, є логічним розвитком і продовженням всього, що було створено та випробувано протягом десятиліть в моделях, таких як водоспадна, V-подібна, ітеративно-інкрементальна, спіральна та інші. Важливою рисою є значне зменшення бюрократичних процедур і максимальна адаптація процесу розробки програмного забезпечення до миттєвих змін на ринку та вимог замовника.

Зауважимо, що в гнучкій моделі значно менше документації в порівнянні з іншими моделями. Важливо пам'ятати про переваги та недоліки "гнучкого маніфесту".

Виходячи зі сказаного вище, можна зробити рішення, та виділити основні переваги та недоліки при виборі моделі розробки ПЗ.

Таблиця 2. Порівняльна таблиця моделей розробки ПЗ

Модель	Переваги	Недоліки	Тестування
Водоспадна	• Кожна стадія має точний результат. • Кожного часу команда виконує один вид роботи. • Добре працює для невеликих задач.	• Повна нездатність адаптувати проект до змін у вимогах. • Надзвичайно пізнє створення працюючого продукту.	• З середини проекту.
V-подібна	• Кожна стадія має точний результат.	• Недостатня гнучкість та	• На переходах між стадіями.

	• Увага тестування приділяється з першої стадії. • Добре працює для проектів зі стабільними вимогами.	адаптованість. • Відсутнє раннє прототипування. • Складність усунення проблем, пропущених на ранніх стадіях розвитку проекту.	
Ітеративно-інкрементальна	• Досить раннє прототипування. • Простота керування ітераціями. • Декомпозиція проекту на керовані ітерації.	• Недостатня гнучкість усередині ітерацій. • Складність усунення проблем, пропущених на ранніх стадіях розвитку проекту.	• У певні моменти ітерації. • Повторне тестування (після доопрацювання) вже перевіреного раніше.
Спіральна	• Глибокий аналіз ризиків. • Підходить для великих проектів. • Досить раннє прототипування.	• Високі витрати. • Складність застосування для невеликих проектів. • Висока залежність успіху від якості аналізу ризиків.	
Гнучка	• Максимальне залучення замовника. • Багато роботи із вимогами. • Тісна інтеграція тестування та розробки. • Мінімізація документації.	• Складність реалізації великих проектів. • Складність побудови стабільних процесів.	• У певні моменти ітерацій та у будь-який необхідний момент.

Вибір моделі повинен базуватися на конкретних потребах та характеристиках проекту. Важливо враховувати стабільність вимог, рівень ризику, доступні ресурси та участь замовника. Зазвичай комбінація різних методів розробки може бути найкращим підходом для досягнення успіху у

розробці програмного забезпечення.

1.7. ІЧ-сенсор та принцип його роботи

ІЧ-сенсор (Рисунок 1.1), що також називається інфрачервоним сенсором, представляє собою пристрій, призначений для реєстрації інфрачервоного випромінювання (теплового випромінювання) і його подальшого використання для різних завдань. Ці сенсори можуть функціонувати в різних спектральних діапазонах інфрачервоного випромінювання і, зазвичай, використовуються для виявлення руху, вимірювання температури та інших характеристик оточуючого середовища.

Рисунок 1.1. ІЧ-сенсор

Важливо зауважити, що ІЧ-сенсори мають широкий спектр застосування в різних галузях, таких як безпека, автоматизація, медицина, а також в побутових пристроях та інших сферах. Вони можуть бути використані для створення систем автоматичного керування, контролю мікроклімату, виявлення руху в системах безпеки, а також для віддалених вимірювань температури та інших параметрів.

Залежно від конкретної мети використання, вони можуть бути обладнані різними видами датчиків та оптичних компонентів для отримання необхідних параметрів та характеристик.

Датчик ґрунтується на двох ключових компонентах: пасивних інфрачервоних приймачах та лінзі Френеля, і використовується для виявлення теплового випромінювання. Зазвичай ці датчики використовуються в системах безпеки або автоматичних вимикачах світла. Однак, важливо зауважити, що предмети у будь-якому приміщенні є джерелами інфрачервоного випромінювання (ІЧ-випромінювання) в різних спектральних діапазонах.

Загалом спектр інфрачервоного випромінювання може бути поділений на три основні групи:

Область коротких хвиль: λ = 0,74 ... 2,5 мкм.

Область середніх хвиль: λ = 2,5 ... 50 мкм.

Область довгих хвиль: λ = 50 ... 2000 мкм.

Спектр ультрафіолетового, видимого людським оком та інфрачервоного випромінювання

Інфрачервоне випромінювання, яке називається "тепловим" в тих випадках, коли енергія випромінюється нагріванням, може змінювати свою довжину хвилі в залежності від температури тіла. При цьому людське тіло також випромінює тепло, і його максимальна інтенсивність відзначається на довжині хвилі близько 9,36 мкм. Однак при входженні людини до приміщення, загальна температура в приміщенні майже не змінюється, що ускладнює виявлення її присутності. Для цього використовується явище переміщення теплової плями.

Пасивний інфрачервоний приймач співпрацює з зовнішньою оптичною системою, особливо з лінзою Френеля.

Ця лінза розділяє простір на прозорі та непрозорі сектори та фокусує інфрачервоне випромінювання контрольованого обсягу на чутливий елемент. Коли людина перетинає ці сектори, генерується змінний тепловий сигнал від її руху. PIR-елемент датчика реагує на різке зміни інтенсивності інфрачервоного випромінювання, що до нього потрапляє.

Принцип дії піроелектричних приймачів полягає в тому, що вони генерують електричні заряди під впливом інфрачервоного випромінювання. Різниця потенціалів у чутливому елементі при опромінюванні становить 1 мВ. Коли людина знаходиться недалеко від детектора, перетинаючи кілька променів, створюється сигнал на піроприймачі (Рисунок 1.2.). Сила сигналу залежить від ступеня перекриття променів та відстані до детектора.

При проектуванні оптичних систем PIR-датчиків часто використовується лінза Френеля, яка проста у виготовленні та має додаткову

перевагу - одну лінзу можна використовувати в різних датчиках руху.

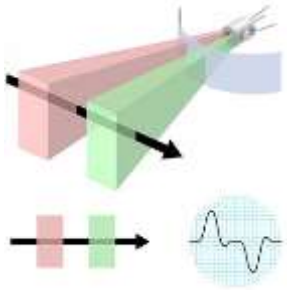


Рисунок 1.2. Перетин променів температурних секторів

Зазвичай кожен сегмент лінзи Френеля формує свої власні промені відображення (див. Рис. 1.3.). Сучасні лінзи Френеля надають змогу прогнозувати чутливість PIR-елемента до всіх цих променів. При створенні лінз Френеля враховується низка параметрів, які безпосередньо впливають на чутливість PIR-елемента, такі як площа сегмента лінзи, кількість активних променів і їх кут нахилу, а також матеріал, з якого виготовлена лінза.

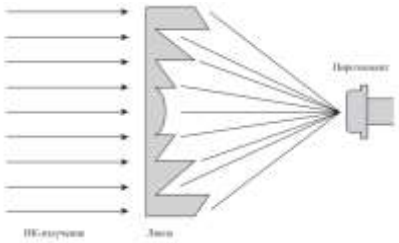


Рисунок 1.3. Формування променю через лінзу Френеля

Матеріал, використовуваний для лінз сучасних датчиків присутності, дозволяє створити лінзу Френеля, яка точніше фокусує інфрачервоне випромінювання і зменшує потрапляння інших частин спектра випромінювання, наприклад, видимого світла від штучних джерел, поза інфрачервоним діапазоном хвиль випромінювання людського тіла.

Для ІЧ-датчиків основними джерелами помилкових спрацьовувань можуть бути:

- Різні пристрої для регулювання клімату, які створюють теплові потоки.
- Випромінювання сонця та світло від штучних джерел.
- Різноманітні радіо- та електромагнітні перешкоди.
    - Термічні напруги в лінзі.
    - Домашні тварини.

1.8. МХ-сенсор та принцип його роботи

МХ-сенсор, або мікрохвильовий сенсор, це пристрій, який використовує мікрохвильове випромінювання для виявлення руху чи наявності об'єктів у навколишньому середовищі. Він працює на основі принципу вимірювання змін у відбитих мікрохвильових сигналах.

Мікрохвильовий датчик руху може виявляти рух об'єктів (наприклад, людей, тварин, металевих предметів тощо), які повністю або частково відбивають мікрохвилі, навіть коли вони перебувають за перешкодами, такими як дерева, двері, стіни, виготовлені з гіпсу, бетону, пластику, скла тощо.

Основним принципом роботи цього датчика є використання ефекту Доплера (Рисунок 1.4), який полягає в зміні частоти відбитих мікрохвиль, що виникає внаслідок руху самого датчика, об'єкта або перешкоди.

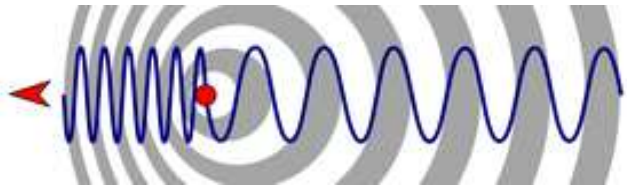


Рисунок 1.4. Ефект Доплера

Ефект Доплера - це помітна зміна частоти або висоти звуку/хвилі, що виникає, коли джерело звуку рухається до або від слухача, або коли слухач рухається до або від джерела звуку. Цей принцип, відкритий австрійським фізиком Крістіаном Допплером, є застосовним до всіх видів хвильового руху.

Декілька слів щодо ефекту Доплера, спочатку припустимо, що частота звуку, що виходить з джерела, залишається постійною. Довжина хвилі звуку також залишається постійною. Якщо і джерело, і приймач звуку залишаються нерухомими, приймач буде чути звук тієї ж самої частоти, що і виробляється джерелом. Це відбувається через те, що приймач отримує ту саму кількість хвиль за секунду, що і джерело.

Тепер, якщо джерело, приймач, або обидва рухаються назустріч один одному, приймач буде сприймати звук вищої частоти. Це відбувається тому, що приймач отримує більше хвиль звуку за секунду і інтерпретує їх як звук

вищої частоти. З іншого боку, якщо джерело та приймач віддаляються один від одного, приймач буде отримувати менше хвиль звуку за секунду і сприймати звук нижчої частоти. У обох випадках частота звуку, що видається джерелом, залишається постійною.

Слід навести приклад (Рисунок 1.5), сирена на швидкісній машині коли вона наближається здається дзвінкішим, ніж коли віддаляється.

Незважаючи на те, що сирена генерує звук із постійною частотою, і ці звукові хвилі рухаються повітрям із однаковою швидкістю у всі напрямки, проте відстань між машини, яка наближається, та людиною, яка слухає, скорочується і завдяки цьому кожна хвиля повинна подолати менший шлях, щоб дістатися до слухача порівняно із ситуацією описаною вище. Із цього випливає, що хвилі приходять до слухача з меншими інтервалами часу між ними.

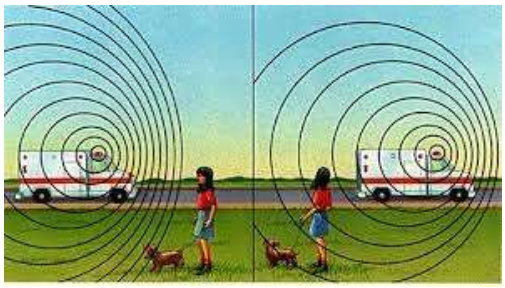


Рисунок 1.5. Розповсюдження хвиль з різною частотою при русі джерела звуку

Ефект Доплера - це явище, яке використовується у багатьох галузях для вимірювання та аналізу руху об'єктів. Наприклад:

1. У астрономії він допомагає визначити рух світлових джерел у космосі, таких як зірки та галактики.
2. В медицині використовується для вимірювання швидкості кровотоку та інших параметрів у пацієнтів за допомогою ультразвукових приладів.
3. У транспорті - для визначення швидкості руху автомобілів та локації об'єктів за допомогою радарів і сонарів.
4. У метеорології - для вимірювання швидкості вітру та виявлення опадів за допомогою Доплерівських радарів.
5. У телекомунікаціях - для стабілізації сигналів у супутниковому зв'язку та

рухливих комунікаційних системах.

6. В авіації і аерокосмічній промисловості - для вимірювання швидкості та руху літаків і космічних апаратів.

7. У акустичній науці - для дослідження руху звукових хвиль в різних середовищах.

8. У охоронних системах – для створення датчиків руку з інтегрованими у них мікрохвильовими сенсорами.

Висновок до Розділу 1

У зв'язку з вищесказаним, ми розуміємо, що глибоке вивчення і осмислення різних технік та методологій тестування має величезне значення для успішної розробки програмного забезпечення. Обираючи оптимальну стратегію, модель розробки ПЗ, конкретні види та техніки тестування, ми будуємо стійку основу для якісних досліджень та аналізу у галузі розробки та тестування програмного забезпечення.

Цей підхід відіграє критичну роль у досягненні оптимальних результатів та раціонального використання ресурсів та часу. Важливо зрозуміти, що успішна розробка програмного забезпечення потребує від нас глибокого розуміння всіх аспектів тестування та вибір правильних підходів.

Після ретельного вивчення та аналізу понять та методів, пов'язаних з інфрачервоними та мікрохвильовими сенсорами, ми можемо надати вагому підставу для подальших досліджень та розробки програмного забезпечення. Це також сприяє раціональному використанню ресурсів та оптимізації процесу розробки.

Такий підхід особливо корисний в контексті галузей, де точність та ефективність мають першорядне значення, таких як астрономія, медицина, транспорт, метеорологія, телекомунікації, авіація та інші. Отже, глибоке розуміння тестування та принципів роботи сенсорів є важливим інструментом для досягнення високих стандартів у цих галузях.

РОЗДІЛ 2. СХЕМАТИЧНЕ ТА АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ТЕСТУВАННЯ СЕНСОРНИХ ПРИСТРОЇВ

2.1. Схематика ІЧ-сенсора

Дослідженням встановлено, що деякі кристалічні речовини здатні набувати поляризації під впливом падаючого на них випромінювання. Зі зміною інтенсивності цього випромінювання змінюється ступінь поляризації і, отже, електрична напруга всередині кристала. Ці матеріали, що мають описану властивість, називаються піроелектриками.

За допомогою вимірювання різниці електричного потенціалу між різними точками всередині кристала можна оцінити інтенсивність падаючого випромінювання. Важливо, що різниця електричного потенціалу знижується внаслідок наявності заряджених частинок у навколишньому середовищі, які потрапляють на кристал. Ця обставина робить піроелектричні датчики неоптимальним вибором для точного вимірювання постійної інтенсивності випромінювання.

Однак саме зміни інтенсивності випромінювання можуть бути надійно ідентифіковані, що надає їм високу цінність у розробці пристроїв, пов'язаних з виявленням руху. Очевидно, процес простий, але тут виникає проблема. Нас не цікавить абсолютна зміна випромінювання, а скоріше зміна, спричинена впливом порушника. Однак зовнішні чинники, такі як добові зміни дня та ночі, сезонні зміни, а також активація та вимкнення опалення призводять до значних коливань інфрачервоного випромінювання, хоча подібна зміна не пов'язана з реальними загрозами, наприклад проникненню грабіжника у будинок. Очевидно, що пристрій, що реагує на подібні флуктуації, має обмежену практичну цінність. Цю складність можна уникнути досить простим чином: замість одного чутливого елемента слід застосувати два. Їхнє підключення здійснюється послідовно так, щоб зміни напруги на них відбувалися в протилежних напрямках.

Крім того, при конструюванні слід врахувати, що глобальні зміни рівня інфрачервоного випромінювання, такі як зміна температури повітря, мають

впливати на них однаковою мірою, тоді як локальні зміни (наприклад, пересування людини) впливатимуть на них нерівномірно .

Для наочності схема працює наступним чином (Рисунок 2.1).

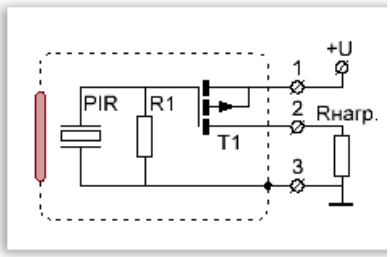


Рисунок 2.1.. Спрощена схема PIR-сенсора

Усередині металевого корпусу розміщено два кристали піроелектрика. Корпус датчика оснащений віконцем, закритим спеціальним фільтром, який пропускає лише певний діапазон довжин хвиль, необхідний вимірюваного випромінювання. Перед вікном знаходиться оптична система, яка формує бажану діаграму спрямованості датчика. Двоопукла лінза зображена схематично; у реальних датчиках застосовуються лінзи Френеля, виготовлені методом штампування на пластиці. Хоча, існують багато інших варіацій конструкційних рішень, яким чином можна збирати та фокусувати випромінювання, такі як дзеркала.

Біля кристалів всередині корпусу розміщується полевий транзистор, зі стоком та джерелом, що виведені назовні, для того щоб зберегти вимірювану величину без втрат на далекій відстані. Як відомо, цей пристрій керується електричним зарядом, зміна якого на кристалі дійсно і вимірюється. Біполярний транзистор не підійшов би, оскільки це пристрій, який керується струмом, а піроелектрика не може видавати струм. Резистор необхідний для розрідження паразитних статичних зарядів, хоча й при цьому трохи погіршує чутливість приладу.

2.2. Обробка сигналу ІЧ-сенсора

Для початку, детальніше розглянемо оптичні лінзи. Без лінз датчик має дуже широку діаграму напрямку - приблизно 100-120° по вертикалі та горизонталі.

У цьому разі ,загальновизнано, один кристал бачить одну половину простору, а інший - іншу. Тобто ми маємо конус, який розрізаний площиною, орієнтація якої залежить від взаємного розташування кристалів. Зазвичай ця площина вертикальна. За допомогою лінзи з двох отриманих пів-конусів формуються два відносно вузькі промені діаграми напрямку. Це підвищує чутливість датчика на відстані та знижує паразитні шуми. Проте вузька діаграма напрямку дає можливість зламувати зони чутливості зловмиснику. Щоб цього уникнути, лінз роблять декілька і відповідно формують кілька пар променів. Таким чином, лінза Френеля складається з множини маленьких лінз, кожна з яких фокусує ІЧ-світло на площину фотоелемента, а одна з них безпосередньо на сам фотоелемент і відбувається реєстрація сигналів (Рисунок 2.2).

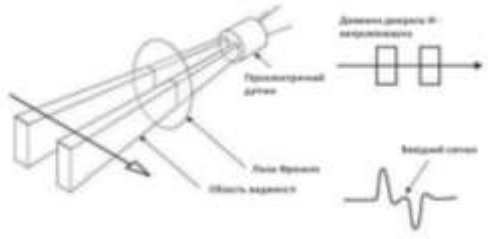


Рисунок 2.2. Схема обробки сигналу

Відповідним чином, під час переміщення об'єкт потрапляє в зону видимості одного кристалу, і останній генерує напруговий імпульс, відповідно до зміни рівня інфрачервоного випромінювання. Коли об'єкт опиняється в зоні видимості іншого кристалу, той також генерує імпульс, але іншого напрямку (при чому кристали включені в протилежні полярності). Якщо об'єкт перетне кілька пар променів, то весь цей процес повториться.

Від цього моменту стає зрозумілим, яким чином можна визначити напрям руху. Якщо спочатку фіксується позитивний імпульс, а потім від'ємний, то напрям один, якщо навпаки - інший. Проте в бітових датчиках, про які мова, ця властивість не використовується.

Використання сигналу безпосередньо з сенсора неможливе, оскільки він занадто слабкий, а його середнє значення коливається в широких межах.

Сигнал потрібно посилення, причому в сотні тисяч разів, позбутися середнього значення і перетворити його на дискретну форму - є рух/немає руху.

Звичайна структурна схема датчика руху (Рисунок 2.3.), яка вирішує ці завдання, має такий вигляд:

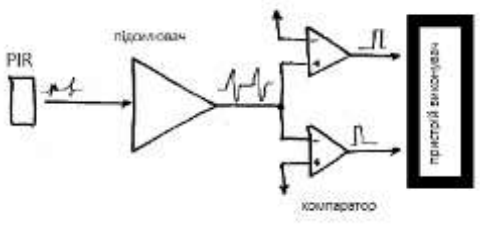


Рисунок 2.3.. Структурна схема датчика руху

Слабкий сигнал сенсора посилюється і подається на пару компараторів, один із яких реєструє перевищення встановленої амплітуди позитивним імпульсом, а інший - від'ємним. Вже дискретний сигнал з компараторів подається на виконавчий пристрій. Останнім часто є чекаючий пристрій, який протягом певного фіксованого часу увімкнутий, а потім вимикає контакти охоронної петлі.

В більш вдосконалених датчиках можуть використовуватися складніші схеми забезпечені, щоб запобігти помилковим спрацюванням, різкими змінами параметрів навколишнього середовища, реакціями на рух дрібних тварин, додатковими реакціями на пожежу і так далі.

2.3. Алгоритмізація роботи ІЧ-сенсору

У першу чергу, блок обробки визначає корисний сигнал, відокремлюючи його від перешкод. Це досягається завдяки спеціальній обробці параметрів сигналів, які надходять з піродатчика (сила сигналу, тривалість та форма). Коли людина перетинає зони чутливості датчика, створюється двополярний симетричний сигнал. Тривалість цього сигналу залежить від відстані між людиною та PIR-сенсором та її швидкості руху перед датчиком. При швидкості руху людини від 0,2 до 5 м/с тривалість сигналу на піродатчику може коливатися в діапазоні від 0,02 до 10 секунд.

Перший алгоритм спрацювань ІЧ-сенсору полягає у наступному ключі. Так як, тривалість сигналів при виникненні перешкод зазвичай відрізняється від аналогічного параметра при русі людини, також сигнали можуть мати асиметричну форму. Проте основним параметром для датчика є величина сигналу (Рисунок 2.4.). Цей параметр може бути єдиним для визначення спрацювання детектора. Його значення порівнюється з певним пороговим значенням. Це визначає чутливість детектора.

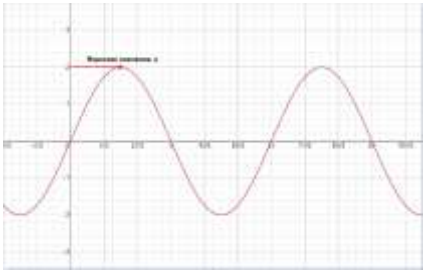


Рисунок 2.4.. Основний параметр сигналу ІЧ-сенсору, величина.

Проте, за такого підходу кількість помилкових спрацювань може бути набагато більше, тому треба шукати альтернативу цьому методу.

З метою зменшення кількості помилкових спрацювань у подібних конструкціях датчиків застосовується облік кількості імпульсів, які перевищують порогове значення. При перетині людиною активного зони неодноразово або декількома променями одночасно, детектор реєструє імпульси.

Наступний алгоритм працює так, спрацювання не відбудеться при першому перевищенні порогового значення, а лише тоді, коли протягом встановленого інтервалу часу, де червоні точки - кількість перевищень перетне встановлений параметр, (Рисунок 2.5).

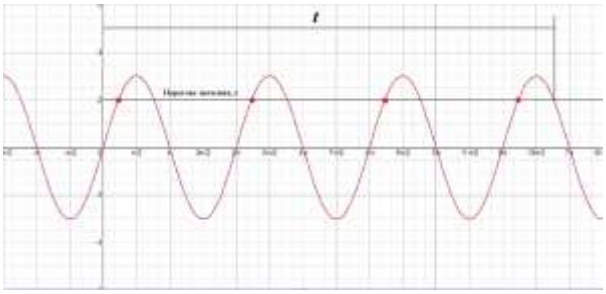


Рисунок 2.5. Алгоритм спрацювання ІЧ-сенсору підраховуючи кількість перевищених порогових значень за певний проміжок часу.

До недоліків такого підходу можна віднести затримки в спрацюванні при реальному появі людей в зоні дії датчика. Крім того, при обліку імпульсів можливі помилкові спрацювання, пов'язані з різними перешкодами, такими як електромагнітне випромінювання або теплові перешкоди.

Тому я пропоную новий алгоритм детектування руху. В основі алгоритму лежить фіксація прискореного росту величини, такий метод дозволяє максимально швидко спрацювати датчику (Рисунок 2.6).

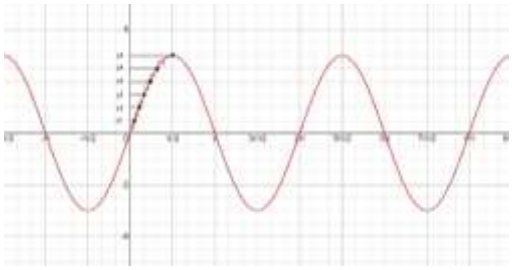


Рисунок 2.6. Алгоритм спрацювання ІЧ-сенсору за зафіксованим прискоренням росту величини сигналу

Суть алгоритму у тому, щоб зафіксувати наростання величини імпульсу (залежність величин між контрольними точками за певний проміжок часу), що і буде тригером спрацювання. Залежність між точками можна конфігурувати, задіяти від 2-ох точок для спрацювання. Такий алгоритм дозволяє відреагувати якомога швидше на рух об’єктів. Проте він також певно має недоліки які треба тестувати на практиці.

Так як чутливість висока, нарощування сигналу високе, сенсор буде реагувати та спрацьовувати на незначний рух гілок, листя, колихання штор, руху теплих потоків повітря від кондиціонеру, обігрівачів і тд. Тому таким чином, сенсор буде давати «хибні» спрацювання, які нам не потрібні.

Проаналізувавши вище описані алгоритми, мною був спроектований ще один альтернативний варіант алгоритму спрацювань ІЧ-сенсору. Для того щоб усунути проблему хибних спрацювань на рухомі предмети від вітру і т.п., сенсор певним чином не має сприймати їх за рухи наступним чином.

Так як, рух таких об’єктів, як наприклад штори, відбувається з певною амплітудою, такі рухи відбуваються лише в одній умовній області, десь в

одному захисному промені одного фоточутливого елемента, тобто кристала піроелемента, і не переходять на другий захисний промінь. Як наслідок, такі мілкі рухи будуть зафіксовані лице одним фоточутливим елементом ІЧ-сенсору. Сигнал сенсору буде коливатись лише в одній із полярностей, або у додатній, або у від'ємній, в залежності від того у якій захисній області припаде рух відносно сенсору (Рисунок2.7).

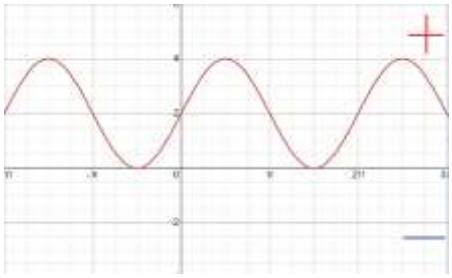


Рисунок 2.7. Сигнал коливальних рухів об'єктів у зоні охоплення одного фоточутливого елемента ІЧ-сенсору

Відповідно, це і є різниця «хибних» рухів об'єктів від руху людини у просторі. Рух людини, наприклад звичайне пересування перед датчиком, тобто переміщення звичайним кроком перпендикулярно датчику, означає що людина перейде декілька захисних зон. Отже буде задіяно два чи більше фоточутливих елементів. Візуально сигнал буде переходити постійно то з додатної у від'ємну полярність, або навпаки (Рисунок 2.8).

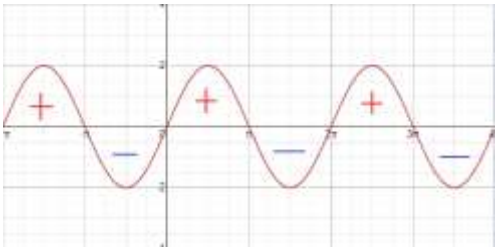


Рисунок 2.8. Сигнал від проходження людиною перпендикулярно датчику, де задіяні 2 фоточутливих елементи ІЧ-сенсору

У зв'язку з вище викладеним, за валідний рух ми тепер можемо сприймати рух через декілька полярностей. Тобто тепер нам достатньо лише зафіксувати один період синусоїди, точніше кажучи перехід сигналу із додатної у від'ємну область, або навпаки. Це буде означати що об'єкт здійснив рух через два захисних променя фоточутливого елементу (Рисунок 2.9). Таким чином, спрацювання не буде генеруватись при русі на одному

місці, тобто в одній зоні покриття одним з фоточутливого елемента.

За валідний рух вважаємо перехід з додатної зони покриття у від'ємну. Та відповідне спрацювання, і перехід до виконуючого пристрою далі (виконання наступної функції).

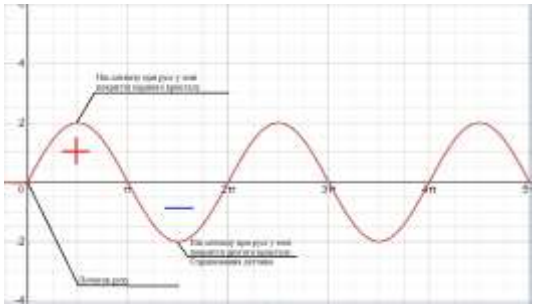


Рисунок 2.9. Сигнал від проходження людиною перпендикулярно датчику, де задіяні 2 фоточутливих елементи ІЧ-сенсору з відповідним спрацювання датчиком

Крім того, однією з додаткових методик обробки сигналів піроприймачів є автоматична термокомпенсація. Її суть полягає в тому, що при температурі повітря, яка зазвичай коливається в межах від +25 до +37 °С, чутливість детектора спадає, оскільки різниця між температурою людини та тепловими полями навколишнього середовища стає меншою. Простим рішенням цієї проблеми є автоматичне підсилення сигналу під час збільшення температури навколишнього середовища. Проте в цьому випадку потрібно вимірювати температуру. Автоматична термокомпенсація дозволяє підтримувати сталий рівень чутливості детекторів незалежно від змін температури навколишнього середовища, які можуть виникати протягом року.

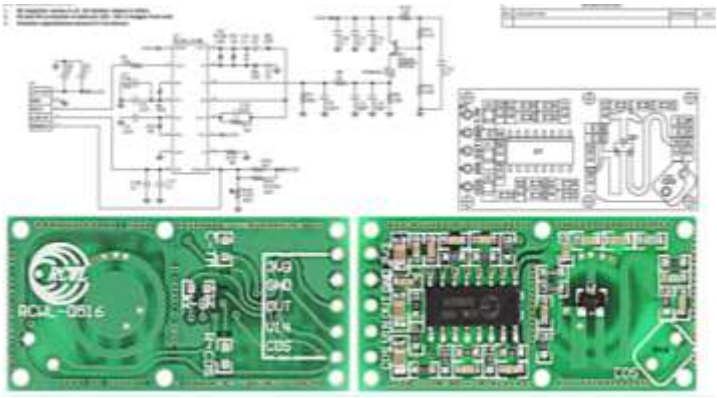
Звідси, у найсучасніших інфрачервоних датчиках застосовуються цифрові методи обробки, які базуються на мікроконтролерах та спеціалізованому програмному забезпеченні. Це рішення гарантує найвищу надійність коректної роботи PIR-детектора та мінімізацію кількості помилкових спрацювань. Крім того, використання схем на сучасних мікроконтролерах дозволяє знизити споживання енергії та виділення тепла датчика, що безпосередньо впливає на строк служби пристрою.

2.4. Схема MW-сенсора

Як було сказано, інфрачервоний сенсор може давати неправдиві спрацювання через вплив теплового потоку систем опалення та кондиціювання, з цієї ж причини нестійко працює на вулиці.

Проте, мікрохвильовий датчик має велику зону виявлення та може реагувати на рух за легкими стінами, дверима тощо. Зміна температури приміщення не впливає на роботу пристрою. Цей датчик може спрацьовувати на малий рух людини.

Цей датчик руху генерує радіохвилі високої частоти. В основу роботи датчика закладено ефект Доплера - зміна частоти відбитої хвилі внаслідок руху випромінювача, приймача або відбивача. У сенсорі частота радіохвилі, що випромінюється ним, змінюється внаслідок руху відбивача (перешкоди). Датчик спрацює, якщо приймач прийме сигнал, частота якого трохи відрізняється від частоти сигналу передавача.



Розглянемо на прикладі сенсора RCWL-0516 (Рисунок 2.10).

Рисунок 2.10. Схема сенсора RCWL-0516.

В основному, більшість MX-сенсорів мають схожі призначення контактів:

- VIN, GND — живлення модуля від 4 до 28 В постійної напруги;
- OUT – цифровий вихід 3.3 В;
- 3V3 – вихід постійного струму (максимум 100 мА);
- CDS — Вхід вимикання датчика (LOW — Вимкнено)

Отже, ще раз, якщо в зоні дії датчика немає об'єктів здатних відбивати радіохвилі, приймач нічого не отримає і датчик не спрацює. Якщо в зоні дії датчика є нерухомі об'єкти здатні відбивати радіохвилі, то приймач прийме

радіохвилю передавача, відбиту від цих об'єктів, але частота прийнятої радіохвилі дорівнюватиме частоті сигналу передавача і датчик не спрацює.

Якщо в зоні дії датчика є об'єкт здатний відбивати радіохвилі, який наближається до датчика (рухається), то приймач прийме відбиту від об'єкта радіохвилю, частота якої буде вищою ніж сигнал передавача і датчик спрацює. Якщо в зоні дії датчика є об'єкт здатний відбивати радіохвилі, який віддаляється від датчика (рухається), то приймач прийме відбиту від об'єкта радіохвилю, частота якої буде нижчою ніж у сигналу передавача і датчик спрацює. Звідси вже зрозуміло як реалізовувати алгоритміку спрацювань.

2.5. Алгоритмізація роботи МХ-сенсора

Виходячи з принципу роботи МХ-сенсора, можна використати наступний алгоритм, проаналізувавши та вимірявши сигнал з сенсора.

Як було раніше досліджено, на виході МХ-сенсора ми отримуємо сигнал певної частоти, в залежності від типу руху об’єкта від якого хвилі відбиваються.

Цей алгоритм лежить на основі ефекту Допплера. Випромінювані хвилі однієї частоти, відбиваючись від рухомого об’єкта, будуть попадати на приймач вже з іншою частотою. Чим більша інтенсивність відбитих хвиль, тим рівень сигналу вищий. Розглянемо на прикладі з радаром та рухомим авто (Рисунок 2.11).

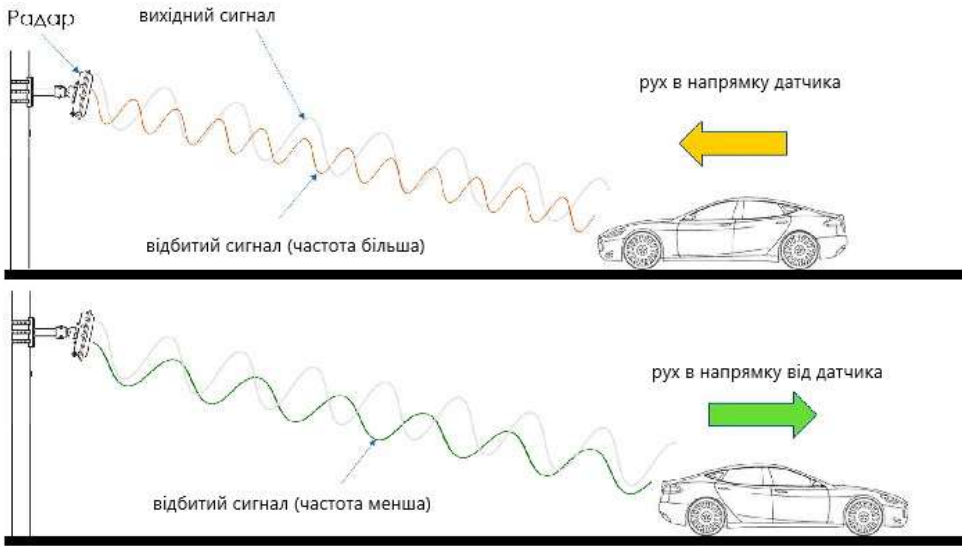


Рисунок 2.11. Зміна частоті сигналу з різни направленням руху авто

Якщо джерело хвиль рухається щодо середовища, то довжина хвилі (λ), відстань між її хребтами, залежить від швидкості та напрямку руху джерела. Коли джерело рухається у напрямку приймача, наближаючись до нього, довжина хвилі зменшується. Якщо джерело віддаляється від приймача, довжина хвилі збільшується. Це можна виразити наступною формулою:

$\lambda = 2\pi(c - v)/\omega_0$,

де ω_0 - кутова частота, з якою джерело випромінює хвилі,

c - швидкість поширення хвиль у середовищі,

v - швидкість джерела щодо середовища (позитивна, якщо джерело наближається до приймача, і від'ємна, якщо віддаляється).

Частота, яку реєструє нерухомий приймач, можна виразити так:

$\omega = 2\pi c/\lambda = \omega_0/(1 - v/c)$.

Аналогічно, якщо приймач рухається назустріч хвилям, він реєструє їх хребти з найвищою частотою, і навпаки. Для нерухливого джерела та рухомого приймача це можна описати такою формулою:

$\omega = \omega_0(1 + u/c)$,

де u - швидкість приймача щодо середовища (додаткова, якщо він рухається у напрямку джерела).

Якщо встановити значення частоти ω_0 з першої формули в другу, отримаємо загальний вираз:

$\omega = \omega_0(1 + u/c)/(1 - v/c)$.

Отже, тепер ми можемо працювати з вихідним сигналом з МХ-сенсору.

Нас цікавить дана частота Доплера f_D (Рисунок 2.12).

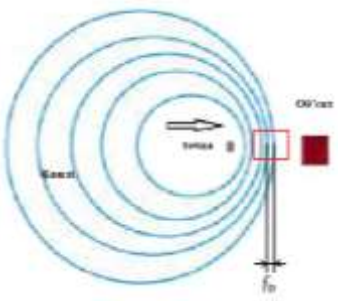


Рисунок 2.12. Частота Доплера при зближенні

У результаті, рівень сигналу зростає із збільшенням інтенсивності відбитих хвиль.

Наразі, вже можемо відрізняти тип руху об’єктів відносно приймача.

Розглянемо на прикладі, де порівняємо вихідну частоту хвиль від передавача, яка відмічена червоним кольором (Рисунок 2.13), з частотами хвиль відбитими від об’єктів (зображені синім кольором), що рухаються на сенсор (Рисунок 2.14), і від нього (Рисунок 2.15).

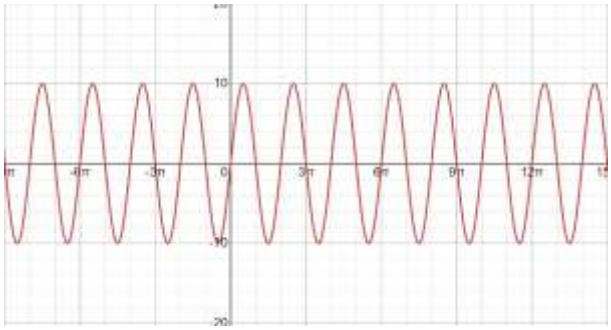


Рисунок 2.13. Вихідна (первісна) частота випромінювача МХ-сенсору.

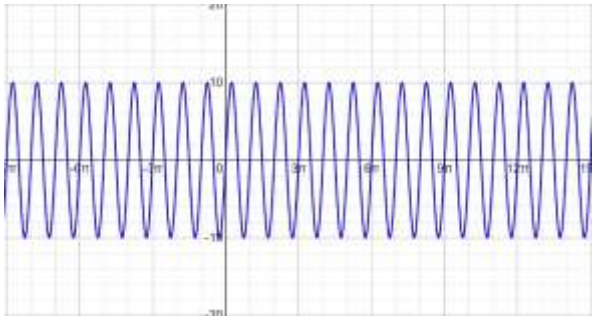


Рисунок 2.14. Частота відбитих хвиль від об’єкту, що рухається на сенсор.

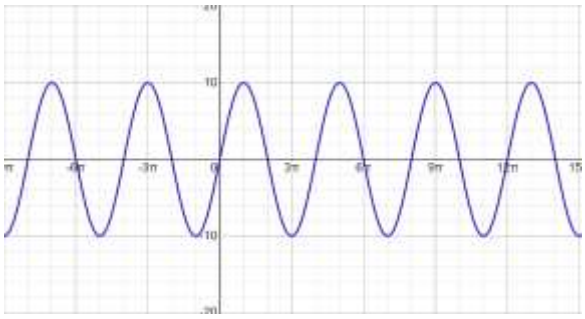


Рисунок 2.15. Частота відбитих хвиль від об’єкту, що рухається від сенсору.

2.6. Формування вимог до програми

Далі, вимоги до програми для логування сигналів з датчиків руху з інтегрованими інфрачервоними та мікрохвильовими сенсорами повинні включати наступні пункти:

1. Збір і аналіз сигналів: Програма повинна бути здатною збирати дані від

інфрачервоних і мікрохвильових сенсорі.

2. Інтерфейс користувача: Програма повинна мати інтуїтивний інтерфейс користувача для налаштування параметрів, відображення результатів та збереження логів.

3. Зберігання даних: Програма повинна забезпечувати можливість зберігання логів датчиків у вигляді файлів, баз даних або іншими методами зручними для подальшого аналізу та звітності.

4. Аналітика: Програма може надавати аналітичні звіти та графіки для моніторингу та аналізу активності.

5. Сумісність: Програма повинна бути сумісною з різними типами датчиків руху, інфрачервоними і мікрохвильовими сенсорами.

2.7.Вибір мови програмування

Для написання програми для збору сирого сигналу з інфрачервоного датчика можна використовувати безліч мов програмування, і вибір залежить від конкретних потреб, переваг та цілей проекту. Однак деякі мови програмування можуть бути більш зручними та поширеними для таких завдань.

Python - високорівнева мова, яка надає безліч бібліотек та фреймворків для обробки даних та сигналів. Він також добре підходить для прототипування та швидкого розробки. Він має безліч особливостей та можливостей:

Простий синтаксис: Python славиться своєю зрозумілою та читабельною мовою, що робить його ідеальним вибором для новачків у програмуванні.

Велика кількість бібліотек: Python має широкий спектр бібліотек і фреймворків для різних завдань, таких як наукові обчислення (NumPy, SciPy), машинне навчання (TensorFlow, PyTorch), веб-розробка (Django, Flask), обробка даних (Pandas) та багато інших.

Кросплатформенність: Python підтримує багато операційних систем і архітектур, що дозволяє писати код один раз і використовувати його на

різних платформах.

Динамічна типізація: Python автоматично визначає типи змінних, що робить код більш гнучким, але вимагає уваги до типів даних.

Велике спільнота та ресурси: Python має велику громаду розробників, що означає, що завжди можна знайти допомогу, документацію та рішення проблем.

Велика кількість галузей застосування: Python використовується в багатьох галузях, включаючи веб-розробку, наукові дослідження, розробку ігор, автоматизацію, машинне навчання та багато іншого.

Розширюваність: Python можна інтегрувати з кодом на C/C++, що дозволяє оптимізувати продуктивність важливих частин додатка.

Відкритий вихідний код: Python поширюється під відкритою ліцензією, що означає, що ви можете використовувати і змінювати його вільно.

Підтримка багатозадачності і асинхронності: Python підтримує багатозадачне і асинхронне програмування, що робить його ідеальним для створення високопродуктивних додатків.

Це лише деякі з основних можливостей і особливостей Python. Мова програмування Python широко використовується в різних сферах, і його простота, гнучкість та багата екосистема бібліотек роблять його популярним інструментом для розробки.

Крім того, для збору сирих сигналів з інфрачервоного датчика на мові програмування Python існують декілька корисних бібліотек. Ось декілька з них:

Adafruit CircuitPython: Ця бібліотека надає інструменти для роботи з різними датчиками та модулями, включаючи інфрачервоні датчики. Вона дозволяє зчитувати дані з датчика та обробляти їх у середовищі Python.

PySerial: Якщо інфрачервоний датчик використовує послідовне з'єднання для передачі даних, PySerial допоможе встановлювати зв'язок та зчитувати дані.

RPi.GPIO: Якщо працюємо з Raspberry Pi та інфрачервоними

датчиками, RPi.GPIO надає інструменти для роботи з GPIO (входи/виходи загального призначення) роз'ємами Raspberry Pi та підключення до датчиків.

MicroPython: Якщо використовуємо мікроконтролери, такі як ESP8266 або ESP32, то MicroPython надає підтримку для роботи з різними датчиками, включаючи інфрачервоні.

Python-miio: Ця бібліотека допоможе працювати з інфрачервоними пристроями та системами управління. Вона дозволяє взаємодіяти з різними пристроями, включаючи пульт дистанційного керування та інфрачервоні сенсори.

LIRC (Linux Infrared Remote Control): Якщо працюємо з Linux-системами і інфрачервоними пристроями, LIRC - це популярна бібліотека для прийому та відправки інфрачервоних сигналів. Вона дозволяє зчитувати та керувати пристроями через інфрачервоний зв'язок.

Інші бібліотеки: Для конкретних датчиків та пристроїв існують також спеціалізовані бібліотеки, які можуть надавати підтримку для їхньої роботи з Python. Якщо у вас є конкретний датчик чи пристрій, рекомендується знайти бібліотеку, спеціалізовану для цього пристрою.

Висновок до Розділу 2

У кінцевому підсумку, у цьому розділі було представлено схематичне та алгоритмічне забезпечення для тестування сенсорних пристроїв.

Була розглянута схематика ІЧ-сенсора, процес обробки сигналу ІЧ-сенсора та алгоритмізація його роботи, декілька видів алгоритмів по спрацюванню датчика, засновані на своїх унікальних принципах та аналізуючи закономірність явищ.

Також була представлена схематика МХ-сенсора та алгоритмізація його роботи. Була проаналізована схема МХ-сенсора, процес обробки МХ-сигналу та алгоритми його функціонування, розглянуто алгоритм і принцип, які використовуються для активації датчика, кожен з яких має свої унікальні принципи та враховує закономірності відповідних явищ.

Окрему увагу було приділено формуванню вимог до програми, включаючи вибір мови програмування. Важливим кроком було визначення вимог щодо реалізації програмного забезпечення для тестування сенсорних пристроїв.

І нарешті, висновок до розділу підсумовує ключові пункти, представлені в даному розділі, і вказує на їхню важливість для подальшого розроблення програмного продукту. Він надає основоположну інформацію, необхідну для розуміння принципів роботи сенсорних пристроїв та їхнього тестування, і визначає основу для подальших досліджень у цій області.

РОЗДІЛ 3. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ТЕСТУВАННЯ СЕНСОРНИХ ПРИСТРОЇВ

Насамперед, розділ цього дослідження приділяє увагу програмному забезпеченню, яке розроблено спеціально для тестування сенсорних пристроїв.

У цьому розділі ми розглянемо ключові аспекти розробки та застосування програмного забезпечення, яке допомагає в оцінці функціональності цих пристроїв. Докладно розглянемо основні класи програми та їх реалізацію, описуючи важливі кроки у розробці. Далі проаналізуємо алгоритми, які використовуються в програмному забезпеченні для вимірювання та оцінки працездатності сенсорних пристроїв.

Показавши графічні ілюстрації результатів, ми надамо чітке уявлення про те, як програма працює та які результати можна отримати.

Завершуючи цей розділ, ми висвітлимо можливості модернізації та майбутні проекти у цій сфері, роблячи акцент на тих аспектах, які можуть сприяти подальшому розвитку та вдосконаленню сенсорних пристроїв та їх програмного забезпечення.

3.1 Реалізація програмного коду, класів, функцій та алгоритмів.

Спочатку створимо візуальну форму програми.

Крім того, основне що треба реалізувати це місце для графіка, де буде виводитись інтенсивність сигналу з наших сенсорів у реальному часі.

Також, вивід даних у текстовому форматі у програмі, для слідкування, за якими точками будується графік. Далі, місце де ми зможемо вибирати порт підключення, через який мі будемо отримувати значення сигналів.

Наступне, вибір швидкості прийняття бітів даних через порт. І кнопка підключення до порту.

Треба реалізувати можливість запису файлів з даними сигналу.

Також, опції конфігурації масштабу графіку, тобто його максимальне значення по осі абсцис та ординат.

Між іншим, буде непоганим реалізувати можливість очищення графіку,

та можливості зробити скріншот графіку у будь-який момент часу.

Програма використовує бібліотеку Dear PyGui для створення графічного інтерфейсу користувача і взаємодії з сенсорними пристроями.

Dear PyGui складається з вікна програми (Рисунок 3.1), вікон та віджетів. Вікно програми є головним вікном нашої створюваної програми й створюється в кінці основного скрипта викликом start_dearpygui(). Програмний код для візуальної реалізації форми програми з графіком наведено в Додатку А.

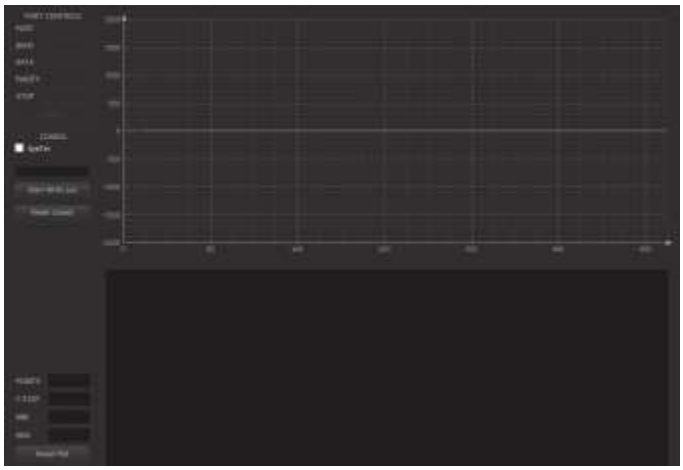


Рисунок 3.1. Вікно форми програми.

Далі нам потрібна функція оновлення даних update_data() що запускає потік, який безперервно оновлює дані із послідовного порту та оновлює графіки та інтерфейс у реальному часі. Програмний код див. Додаток А.:

Потім спрацьовує Thread: Це об'єкт потоку, який запускається під час виконання функції update_data.

Також потрібна функція open_port() (Рисунок 3.2.): Ця функція відповідає за відкриття та закриття послідовного порту для обміну даними з пристроєм. Код див. Додаток А

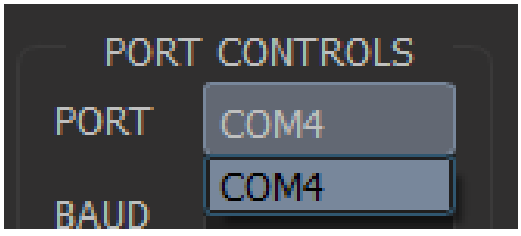


Рисунок 3.2.Функція відкриття портів

Коли підключити пристрої та подивитись як працює програма та відображає графік побачимо наступне (Рисунок 3.3.).

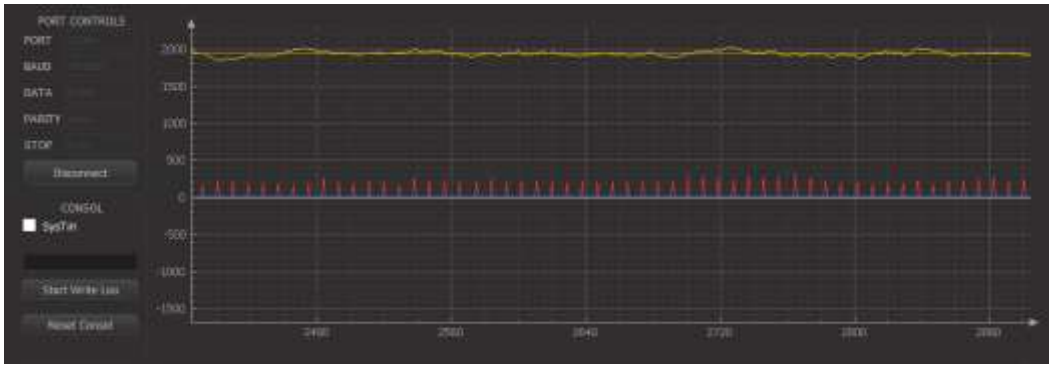
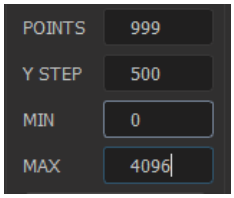


Рисунок 3.3. Візуалізація сигналів

Бачимо, що сигнал з ІЧ- та МХ-сенсору приймаються та відображається з потрібною нам швидкістю, де жовтий графік – сигнал з інфрачервоного сенсору, та червоний – з мікрохвильового сенсору. Тут можемо відредагувати масштабність вручну(Рисунок 3.4.).



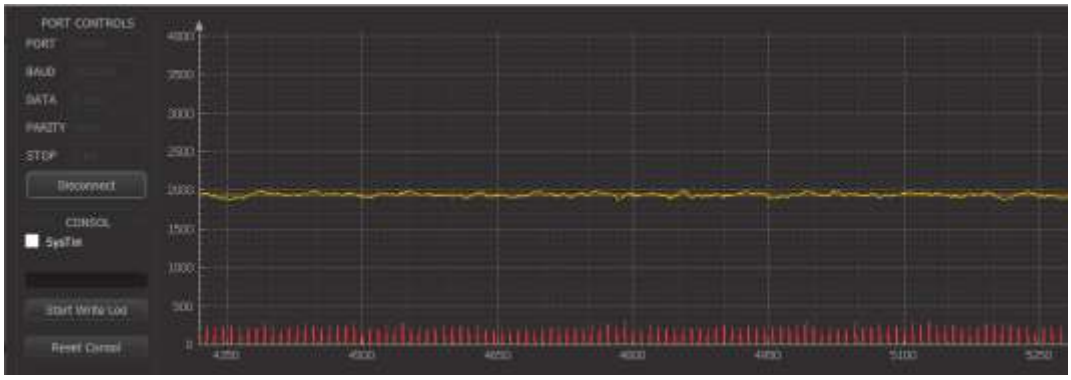
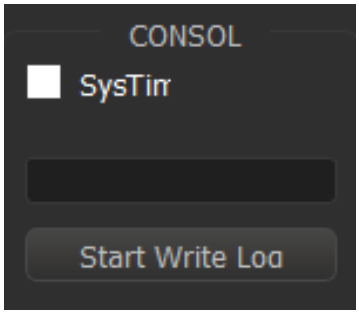


Рисунок 3.4. Регулювання масштабності

Тепер треба реалізувати зберігання запису в файлі з даними сигналами графіка. Треба організувати діалог(Рисунок 3.5)..



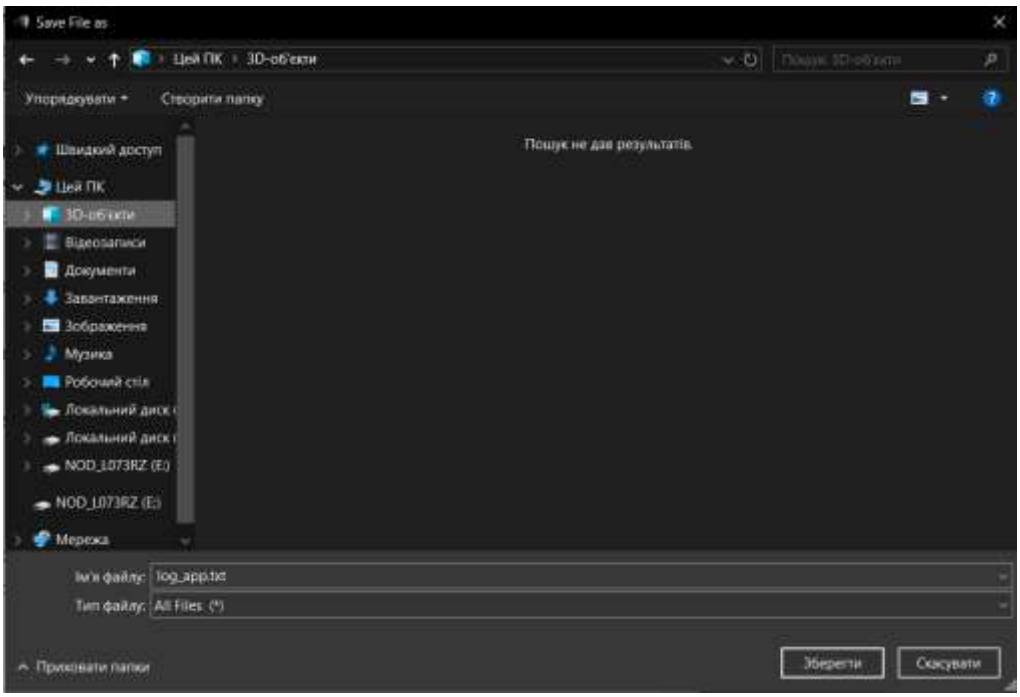


Рисунок 3.5. Діалог зберігання та зчитування файлів

Збережений файл з логом виглядає так:

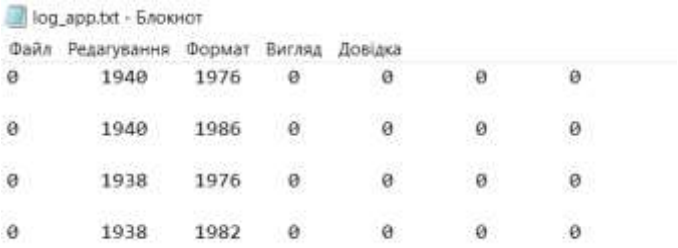


Рисунок 3.6. Файл з інформацією

Також потрібні наступні методи:

- dpg.setup_dearpygui(): Цей метод запускає Dear PyGui та готує його до виконання `dpg.show_viewport()`

- dpg.start_dearpygui(): Цей метод починає виконання Dear PyGui і очікує подій взаємодії користувача.

- dpg.destroy_context(): Цей метод знищує контекст Dear PyGui та завершує роботу програми.

- resource_path(relative_path): Ця функція повертає абсолютний шлях до ресурсів. Вона використовується для визначення шляху до файлів незалежно від того, чи працює програма в режимі розробки, чи була упакована за допомогою PyInstaller.

Тепер треба додати консоль куди будуть виводитись значення даних за якими будується графік сили сигналу з певним інтервалом часу.(Рисунок 3.7.) Див. Додаток А..

Рисунок 3.7. Вікно з виводом значень

Щоб побудувати графік з записаного раніше лог-файлу треба додати функцію побудови графіка.

Відкриємо записаний нами раніше лог-файл (Рисунок 3.8.)

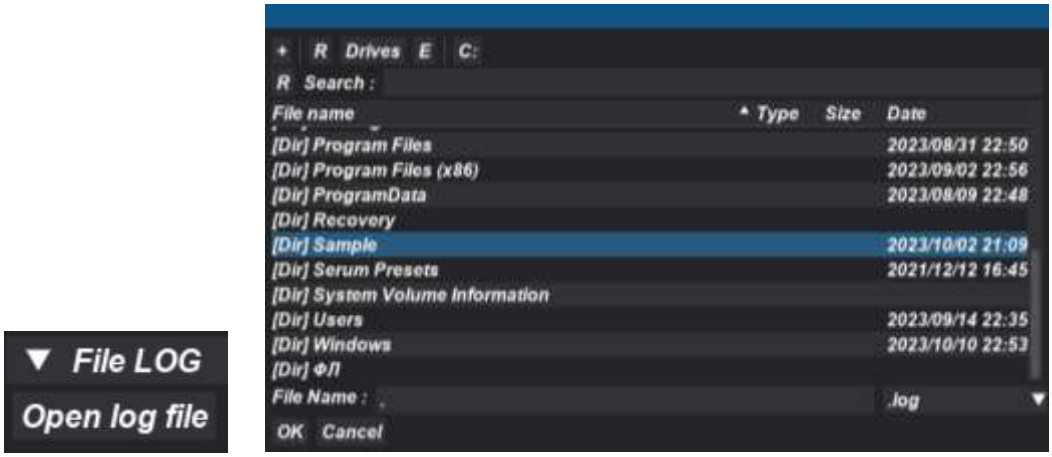


Рисунок 3.8. Діалог відкриття файлу

та подивимось на побудований графік (Рисунок 3.9).

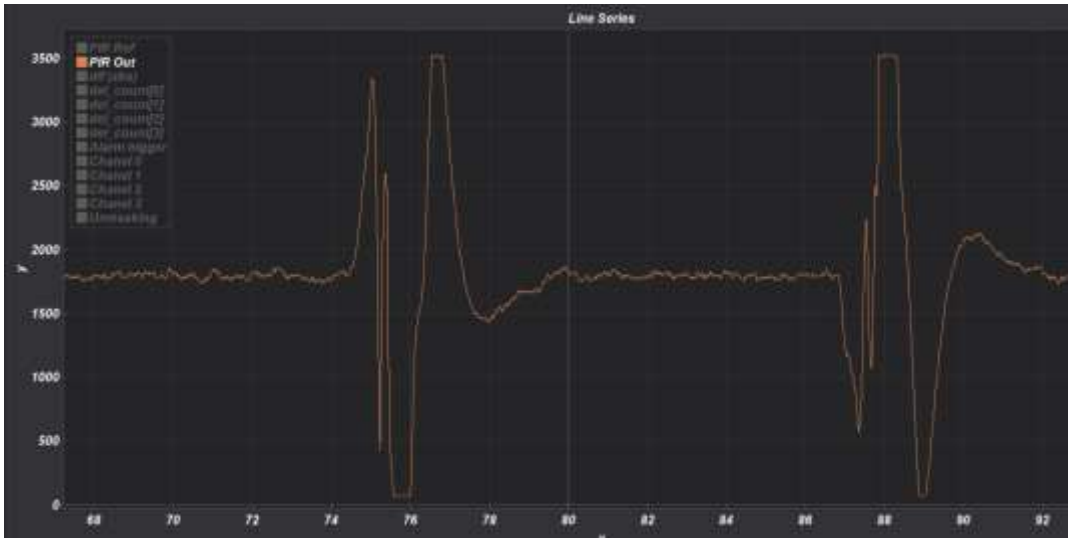


Рисунок 3.9. Графік спрацювання

Додамо фунцію скидання даних з буфера, після натискання кнопки Reset Plot, побудований раніше графік буде очищений і залишиться тільки порожня система координат (Рисунок 3.10).

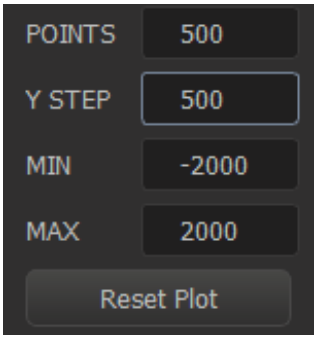


Рисунок 3.10. Скидання графіка

Наступним кроком, для зручності складання звітів, реалізуємо функцію

скріншоту графіку сигналу, та зберігання його у форматі .JPG .

3.2. Застосування алгоритмів спрацювання.

Для початку, протестуємо ІЧ-сенсор простими проходами кроком перпендикулярно датчику на відстанях 6м, 9м, 12м (Рисунок 3.11-3.13), використовуючи лінзу Френеля (яка ділить захисне покриття на сектори) та проаналізуємо результати.

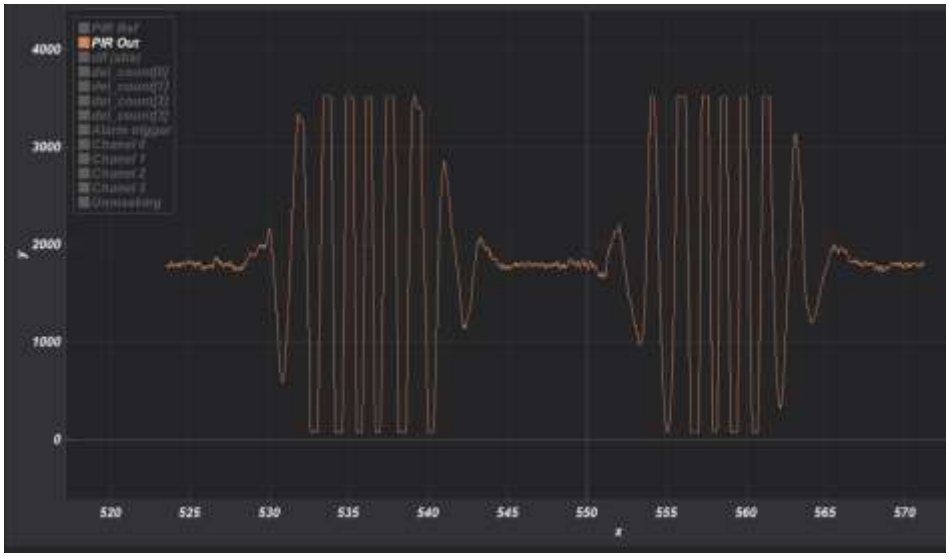


Рисунок 3.11. Тестування ІЧ-сенсору проходками кроком перед датчиком на відстані 6м перпендикулярно.

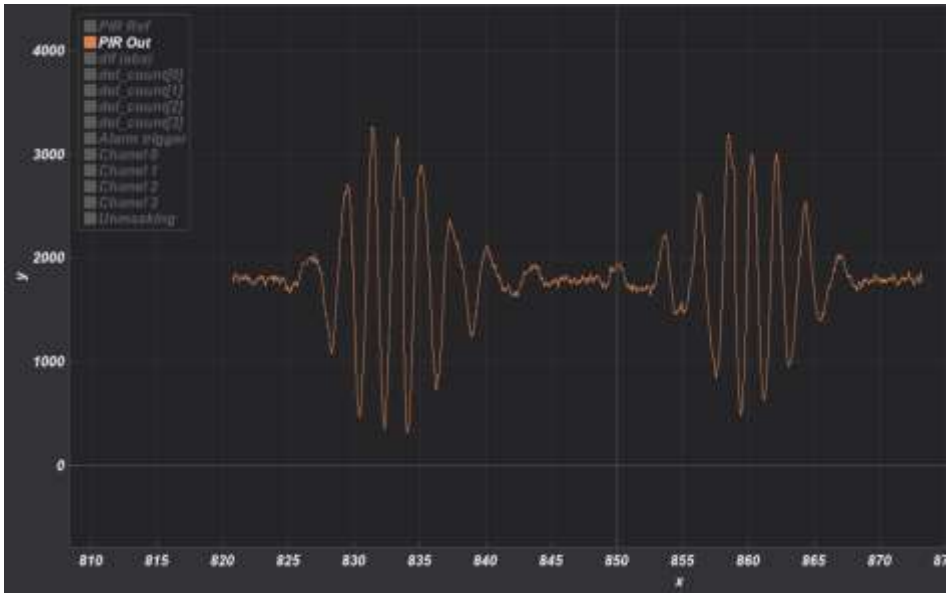


Рисунок 3.12. Тестування ІЧ-сенсору проходками кроком перед датчиком на відстані 9м перпендикулярно.

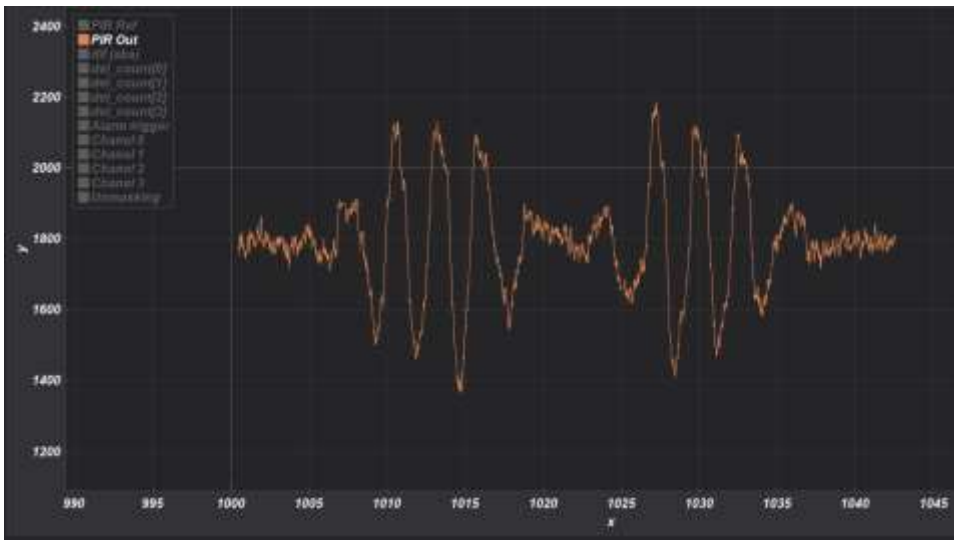


Рисунок 3.13. Тестування ІЧ-сенсору проходками кроком перед датчиком на відстані 12м перпендикулярно.

Отже, виходячи з результатів ми можемо сказати, що програма працює і користувач може тестувати сенсорні пристрої та записувати сигнал з них.

Дивлячись на лог-файл, можна зробити деякі висновки.

На графіку видно як змінювався сигнал з ІЧ-сенсору під час руху людини перпендикулярно датчику.

Можна побачити де саме були переходи людини із поля зору кожної з зон покриття фоточутливих елементів, від додатного до від’ємного (Рисунок 3.14).

Наочно видно декілька зон променів покриття (8 променів), завдяки конструкції лінзи Френеля.

Сигнал по мірі віддалення людини зменшується.

Перед початком руху бачимо певний рівень сигналу, він буде залежати від температури навколишнього середовища.

За графіком також можна визначити і швидкість руху об’єкту.

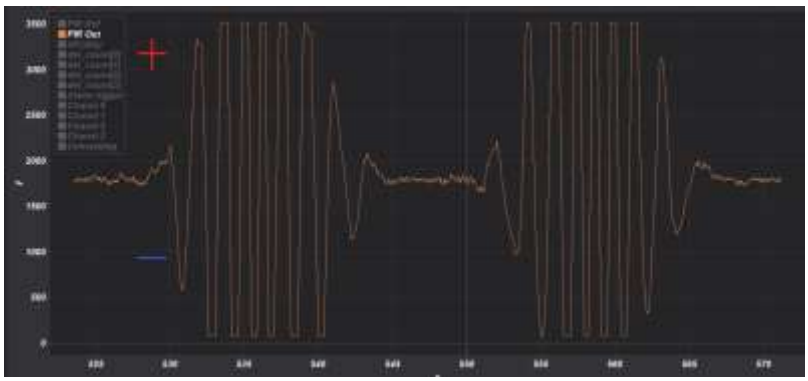


Рисунок 3.14. Сигнал при перпендикулярному русі об’єкту у декількох зонах охоплення, поділеними лінзою Френеля, двох фоточутливих елемента ІЧ-сенсору.

Надалі реалізуємо перший алгоритм по підрахунку перевищених порогових значень. Відмічається спрацювання пристрою на графіку спеціальним тригером Alarm trigger, тож промоделюємо ситуацію.

Граничним значенням встановлено сигнал до 2500 од, для спрацювання – 2 перевищення. Код див. Додаток А.

Спрацювання сенсору на прохід кроком людиною в 3,6,12 метрах від сенсору виглядатиме наступним чином (Рисунок 3.15-3.17).

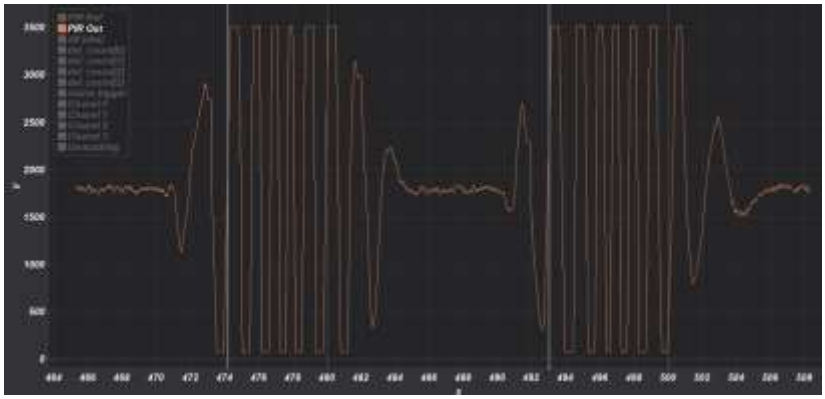


Рисунок 3.15.Спрацювання сенсору на прохід кроком людиною в 3 метрах від сенсору.

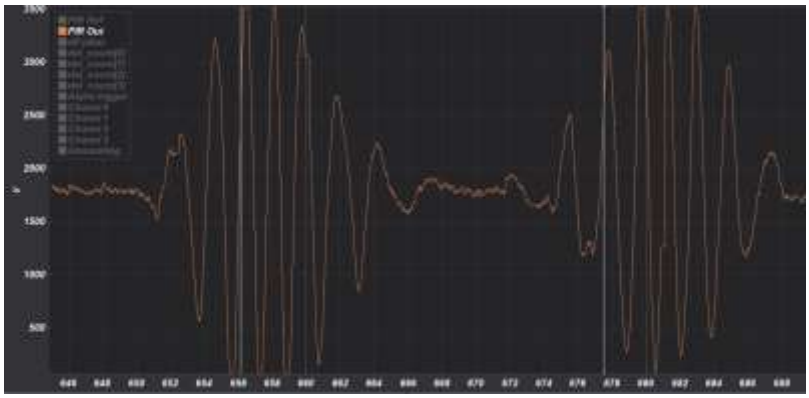


Рисунок 3.16.Спрацювання сенсору на прохід кроком людиною в 6 метрах від сенсору.

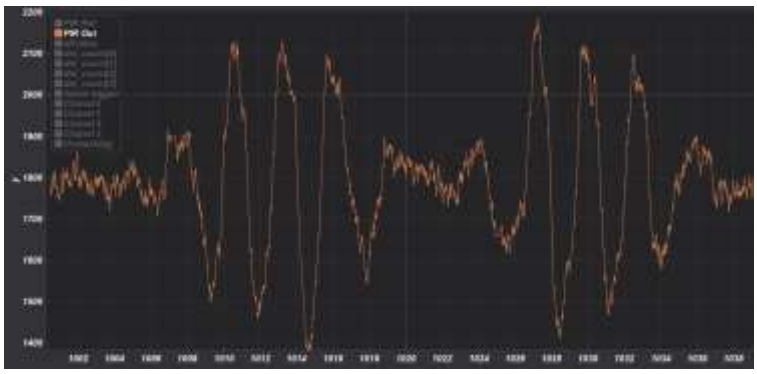


Рисунок 3.17.Спрацювання сенсору на прохід кроком людиною в 12 метрах від сенсору.

На 12 метрів немає спрацювань, адже сигнал навіть не досягнув граничного значення у 3000 од., а лише до 2189 од. Програмно можна знизити порогове значення сигналу, щоб досягти спрацювання, проте маємо великий ризик хибних спрацювань, від фонової температури середовища. Такий підхід не дає високих результатів та надійність його низька.

Тепер розглянемо другий алгоритм спрацювання який враховує прискорення об’єкта спостереження. Відмічати спрацювання пристрою будемо на графіку спеціальним тригером Alarm trigger та промоделюємо ситуацію.

Встановимо діапазон спрацювань для того щоб уникнути хибних спрацювань від мінімальної зміни сигналу. Алгоритм буде рахувати максимальні точки за короткі проміжки часу, так рахуватиметься сковзне середнє, і різниця кожного із контрольних значень за інтервал часу (Рисунок 3.18). Для підвищення точності, інтервал часу можна зменшувати софтово.

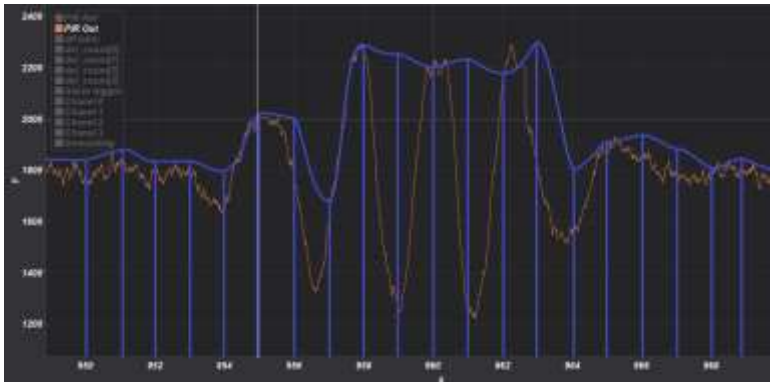


Рисунок 3.18. Графічне представлення алгоритму по підрахунку ковзного середнього (синій колір) за короткі проміжки часу з сигналу ІЧ-сенсору (помаранчевий колір).

Тепер спробуємо промоделювати ту саму ситуацію з проходженням людини перед сенсором. Як спрацює сенсор на прохід кроком людиною в 6 метрах від сенсору.

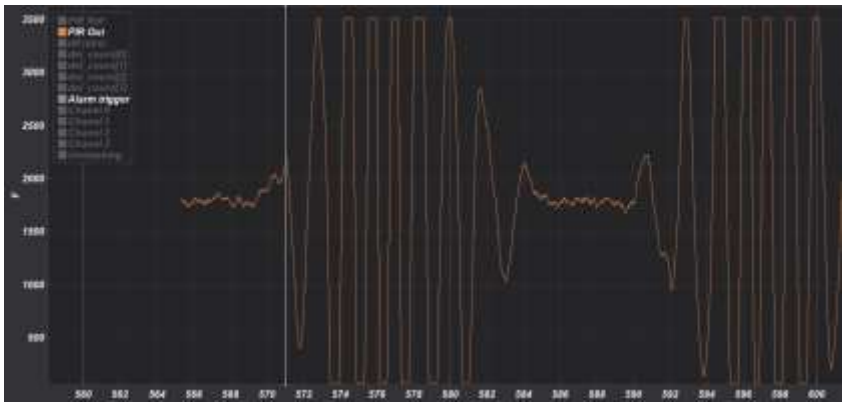


Рисунок 3.19.Спрацювання сенсору на прохід кроком людиною в 6 метрах від сенсору.

Далі спрацювання сенсору на прохід кроком людиною в 9 метрах від сенсору.

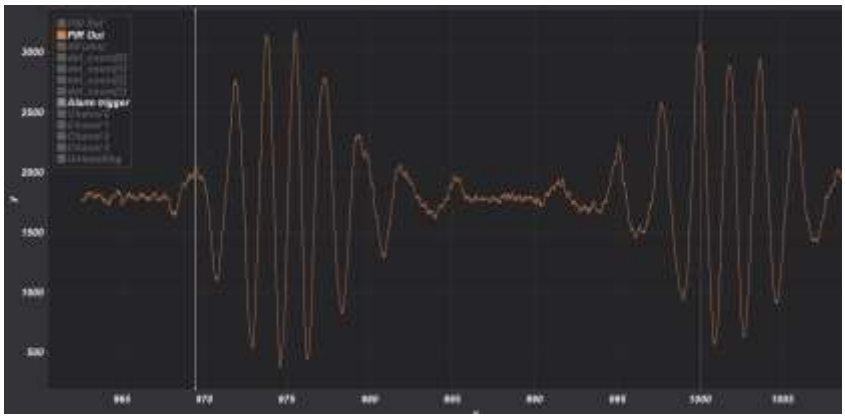


Рисунок 3.20.Спрацювання сенсору на прохід кроком людиною в 9 метрах від сенсору.

Спрацювання сенсору на прохід кроком людиною в 12 метрах від сенсору.

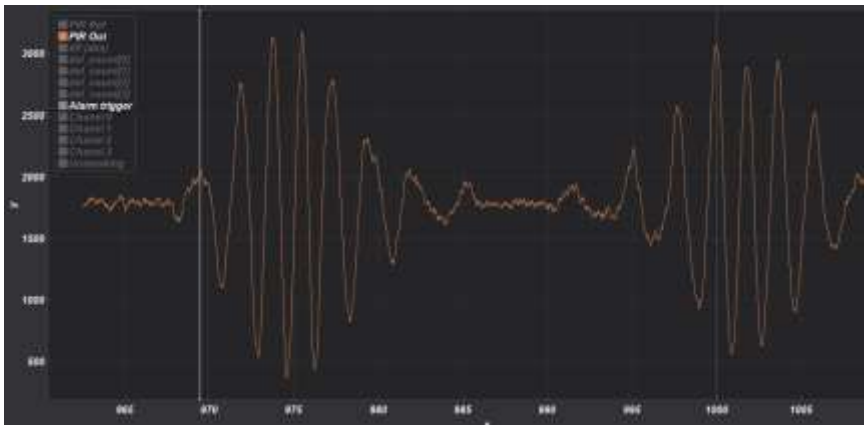


Рисунок 3.21.Спрацювання сенсору на прохід кроком людиною в 12 метрах від сенсору.

Виходячи з результатів, можна сказати, що даний алгоритм працює краще, швидше, і детектує рух раніше. Проте загроза хибних спрацювань на рух непередбачених предметів, або гілок, штор дуже велика.

Наразі, реалізуємо третій алгоритм спрацювання датчика. Його суть полягає у наступному, спрацювання відбувається після виходу із певного діапазону шуму сигналу (виділеним зеленим кольором), діапазон так званого спокою, де не відбувається рухів. Після виходу за межі діапазона, починає працювати алгоритм, який фіксує перехід з додатної зони у від'ємну, простими словами нас цікавить один період синусоїди, і не важливо як відбувається перехід, або від негативної у додатну, або навпаки.

Логіка алгоритму така, що спрацювання відбуватиметься після фіксування одного періоду синусоїди (Рисунок 3.22). На практиці такий сигнал можна отримати, пройшовши через 2 фоточутливих елемента почергово, і цього буде достатньо для спрацювання, при умові що сигнал досить сильний.

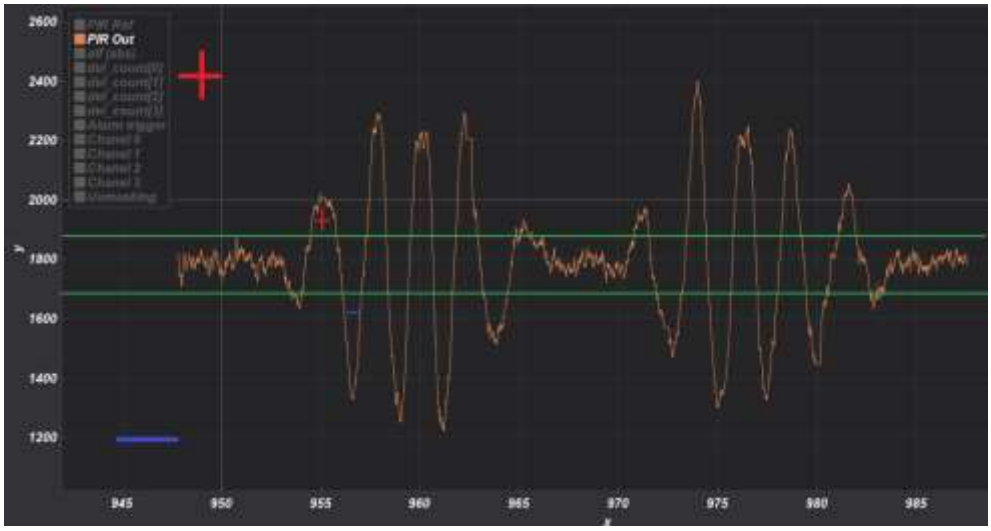
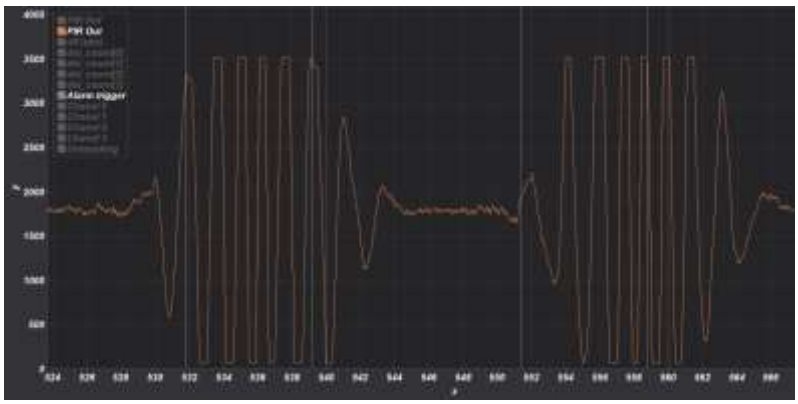
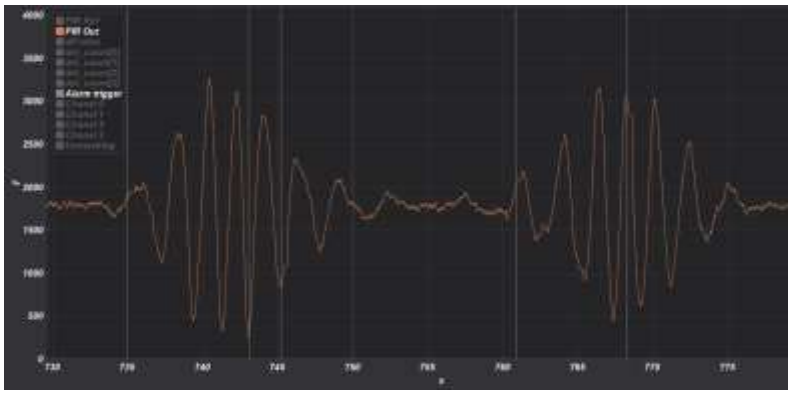


Рисунок 3.22. Графічне представлення алгоритму по підрахунку одного періоду синусоїди сигналу ІЧ-сенсору.

Відтворимо модуляцію ситуації та перевіримо спрацювання алгоритму. (Рисунок 3.23-3.25).

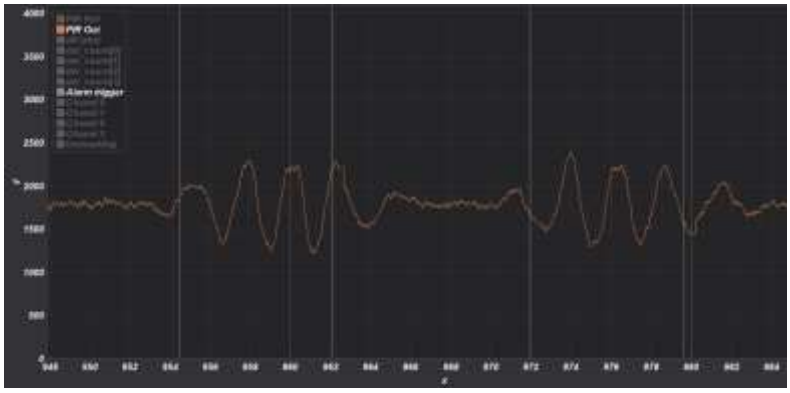


3.23. Спрацювання сенсору на прохід кроком людиною в 6 метрах від сенсору.



3.24. Спрацювання сенсору на прохід кроком людиною в 6 метрах від сенсору.

Спрацювання сенсору на прохід кроком людиною в 12 метрах від сенсору.



3.25. Спрацювання сенсору на прохід кроком людиною в 12 метрах від сенсору.

Як бачимо, алгоритм працює стабільніше, детектування руху відбуваються, як і очікувалось, за один синусоїдний період сигналу. Такий алгоритм дозволяє одночасно зберігати стійкість до хибних спрацювань, невеликих коливальних рухів у зоні покриття одного фоточутливого елементу ІЧ-сенсору, а також має доволі стабільну швидкість детектування руху у полі зору датчика.

Висновок до Розділу 3

У цьому розділі було досліджено та розроблено програмне забезпечення, призначене для тестування сенсорних пристроїв. Робота включала в себе реалізацію програмного коду, створення класів, функцій і алгоритмів, необхідних для забезпечення коректної роботи пристрою і збору з нього даних. Важливою частиною цього розділу було також застосування розробленого алгоритму тригера, що гарантує коректне опрацювання отриманих даних та їх інтерпретацію.

Дослідження показали, що розроблене програмне забезпечення може бути використане для ефективного тестування сенсорних пристроїв та збору даних з високою точністю. Крім того, воно виявилося досить гнучким для адаптації до різних типів сенсорних пристроїв, що робить його корисним інструментом, який може бути використаний в різних додатках та галузях.

Загальний висновок полягає в тому, що розробка програмного забезпечення для тестування сенсорних пристроїв є важливою та актуальною проблемою у сучасній інформаційній середовищі. Результати досліджень та розробок, наведених у цьому розділі, будуть корисними для подальших досліджень та розробок цієї галузі, а також для практичного застосування в промисловості та наукових дослідженнях.

ВИСНОВОК

В результаті досліджень, представлених у цій роботі, були детально розглянуті методики та технології тестування сенсорних пристроїв, а також розроблено програмне забезпечення для їх тестування. Робота складається з трьох основних частин, що відображають важливі аспекти тестування та розробки програмного забезпечення.

У першому розділі детально розглянуто поняття тестування, класифікації, види тестування та моделі розробки програмного забезпечення. Цей розділ закладає базову основу для розуміння процесу тестування та вибору відповідних методів та підходів при розробці програмного забезпечення.

Таким чином, складається повне представлення проблеми та спосіб її рішення. Визначивши основні ризики та специфіку тестування, формуємо фундаментальні заключення та факти, якими оперуватимемося впродовж алгоритмічного, а потім і програмного забезпечення.

В другому розділі було проведено аналіз схем та алгоритмічного забезпечення для тестування інфрачервоних та мікрохвильових сенсорів, визначено основні засади їхньої роботи, сформовано вимоги до програми та обрано мову програмування. Цей розділ був необхідний для визначення окремих деталей програмного забезпечення, для повного розуміння структури програми, а також для планування реалізації основних функцій та можливостей и способами взаємодії з пристроями, прийняття та обробка вихідних сигналів.

У третьому розділі було розроблено програмне забезпечення для тестування сенсорних пристроїв. Було реалізовано програмний код, створено класи, функції та алгоритми, які забезпечують ефективне тестування та збір даних із пристрою.

Висновки, зроблені в цьому розділі, підкреслюють корисність розробленого програмного забезпечення для практичного застосування в промисловості та наукових дослідженнях.

Створена програма є доволі універсальною для тестування сенсорних пристроїв. Вона підходить для тестування сенсорних пристроїв з їх різними модифікаціями, наприклад, при зміні параметрів лінз Френеля, їх форми, матеріалу, та візерунків фокусуючих частин лінз, або дзеркал, які також відбивають та фокусують інфрачервоне випромінювання на сенсорний пристрій.

Крім того, протестувати можна будь-який сенсор з аналоговим сигналом на виході. А також, закладені алгоритми дозволяють промоделювати спрацювання датчика на зміну сигналів, або модернізувати застарілий алгоритм чи створити новий. Загальні висновки відображають результати проведених досліджень та розробок.

Метою даної роботи є розширення знань у галузі тестування сенсорних пристроїв та надання практичних рекомендацій щодо використання розробленого програмного забезпечення.

Таким чином, у роботі закладено підґрунтя для подальших досліджень з удосконалення сенсорної техніки та відповідного програмного забезпечення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Люта М. В., Розломій І. О., Новикова К. В. Аналіз методів тестування програмного забезпечення./ Сучасні перспективи розвитку науки: матеріали Міжнародної науково-практичної конференції м. Київ, 15-16 вересня 2017 року.–Київ.: МЦНД, 2017.–86 с.
2. Кожем’яко В. П. Оптоелектронні системи та пристрої : навчальний посібник / Кожем’яко В. П., Гаркушевський В. С., Петрук В. Г.– Вінниця ВНТУ, 2005.
3. Дідковська М.В., Тимошенко Ю.О. «Тестування: Основні визначення, аксіоми та принципи. Текст лекцій» (http://mmsa.kpi.ua /sites/default/files/disciplines/didkovska_m_v_testing_definition_part1.pdf)
4. Молодцова О.П. Управління якістю програмної продукції [Текст]: навчальний посібник /О.П. Молодцова. —К. КНЕУ, 2021. —248с.— ISBN 966-574-230-2
5. Популярні життєві цикли розробки ПЗ. Електронний ресурс. https://training.qatestlab.com/blog/technical-articles/popular-software-development-life-cycles/
6. Іванов А. О. Теорія автоматичного керування: Підручник. — Дніпропетровськ: Національний гірничий університет. — 2014. — 250 с.
7. Енциклопедія кібернетики. тт. 1, 2. — К.: Головна редакція УРЕ, 2016.
8. Офіційний сайт Arduino. – Режим доступа: http://www.arduino.cc/. Дата доступа: 23.09.2023.
9. Офіційний сайт 8Devices. – Режим доступа: http://www.8devices.com. - Дата доступа: 29.09.2023.
10. Форум іноваційних технологій – Режим доступу: http://innotech.kiev.ua/. - Дата доступа: 01.10.2023.
11. Охрімчук О.Б. Оптоелектронні сенсори. Перспективні технології та прилади. 2023. № 22.
12. J. Humble and D. Farley, Continuous delivery: reliable software releases through build, test, and deployment automation. Pearson Education, 2010.
13. 19. E. Dustin, T. Garrett, and B. Gauf, Implementing Automated Software Testing. Pearson Education, 2019.
14. A A. Y. Head first python. International journal of engineering in computer science. 2021. Vol. 3, no. 1.
15. Automate the boring stuff with python–an innovative solution to borehole salinity estimation / M. Dutta et al. SPE/IADC middle east drilling technology conference and exhibition, Abu Dhabi, UAE, 23–25 May 2023.

16. Charles T. F. Python programming crash course. Charlie Creative Lab, 2020.
17. Cicolani J. A crash course in python. Beginning robotics with raspberry pi and arduino. Berkeley, CA, 2018.
18. Computer Science Computer Science Academy. Python for data analysis: a complete guide for beginners, including python statistics and big data analysis. Independently Published, 2020.
19. Deep L. P. I. Python for beginners: ride the wave of artificial intelligence and machine learning with this crash course on python programming, deepening data analysis and science with real computer programming codes. Independently Published, 2019.
20. Pilgrim M. Your first python program. Dive into python 3. Berkeley, CA, 2019.
21. Programming P. C. Python data science: the utimate crash course for beginners. learn python in a week and master it. Independently Published, 2020.
22. Python cookbook / ed. by M. Alex, A. David. Sebastopol, CA: O'Reilly, 2020.
23. Ramalho L. Fluent python. O'Reilly Media, Incorporated, 2015.
24. Slatkin B. Effective python: 90 specific ways to write better python. Pearson Education, Limited, 2019.
25. ДСТУ ГОСТ 3008-95 «Документація. Звіти у сфері науки і техніки. Структура і правила оформлення»