НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих

комп'ютерних систем

До захисту допущено:

Завідувач кафедри

_____ Віталій РОМАНКЕВИЧ

«___» червня 2025 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою

«Системне програмування та спеціалізовані комп'ютерні системи»

спеціальності 123 «Комп'ютерна інженерія»

на тему: «Програмно-апаратний комплекс детекції руху на базі мікроконтролера STM32»

Виконав (-ла):
студент (-ка) IV курсу, групи КВ-12
Ступницька Софія Миколаївна _____

Керівник:
ст. викладач,
Радченко Костянтин Олександрович _____

Консультант з нормконтролю:
доц. кафедри СПСКС, к.т.н.
Клятченко Ярослав Михайлович _____

Рецензент:
ст. викладач каф. обчислювальної техніки ФІОТ,
Алещенко Олексій Вадимович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики
Кафедра системного програмування і
спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Системне програмування та спеціалізовані комп'ютерні системи»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
«___» _____ 2024 р.

ЗАВДАННЯ
на дипломний проєкт студенту
Ступницької Софії Миколаївни

1. Тема проєкту «Програмно-апаратний комплекс детекції руху на базі мікроконтролера STM32», керівник проєкту Радченко Костянтин Олександрович, ст. викладач, затверджені наказом по університету від «29» травня 2025 р. № 1808

2. Термін подання студентом проєкту _____

3. Вихідні дані до проєкту: див. Технічне завдання

4. Зміст пояснювальної записки:

- аналіз існуючих рішень та обґрунтування теми дипломного проєкту;
- аналіз обраних технологій та апаратних засобів;
- програмно-апаратний комплекс моніторингу стану повітря.

5. Перелік графічного матеріалу:

- концептуальна схема архітектури системи;

- схема електрична функціональна;
- схема алгоритму циклу вимірювань;
- структурна схема програмного забезпечення;

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормконтроль	Клятченко Я.М., доц. кафедри СПіСКС, к.т.н		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Видача завдання на дипломне проєктування	30.10.2024	
2	Вивчення літератури за тематикою проєкту	09.11.2024	
3	Розроблення та узгодження технічного завдання	18.11.2024	
4	Аналіз існуючих рішень	12.12.2024	
5	Підготовка матеріалів першого розділу дипломного проєкту	28.02.2025	
6	Підготовка матеріалів другого та третього розділів дипломного проєкту	15.04.2025	
7	Підготовка графічної частини дипломного проєкту	19.05.2025	
8	Оформлення документації дипломного проєкту	20.05.2025	
9	Попередній огляд матеріалів проєкту на кафедрі	03.06.2025	

Студент Софія СТУПНИЦЬКА

Керівник Костянтин РАДЧЕНКО

АНОТАЦІЯ

Кваліфікаційна робота включає пояснювальну записку (57 с., 21 рис.).

Об’єктом розробки є програмно-апаратний комплекс охоронного призначення, що реалізований на базі мікроконтролера STM32. Система розроблена для виявлення несанкціонованих дій у контрольованій ділянці та оперативного інформування користувача про потенційні загрози. Комплекс поєднує в собі функції детекції руху, інформування про небезпечні зміни параметрів навколишнього середовища. Крім того, реалізовано функціонал розумного спостереження з гнучким налаштуванням режимів охорони.

Сенсорний дисплей забезпечує користувачу доступ до налаштування параметрів охорони та режимів роботи установки. Графічний інтерфейс реалізовано з інтуїтивно зрозумілою структурою, що гарантує зручну навігацію між функціональними елементами. Програмна реалізація здійснюється за допомогою мови програмування С, з використанням бібліотеки HAL, що забезпечує абстракцію від апаратної частини. Для обробки подій у реальному часі, використано операційну систему FreeRTOS.

У ході розробки було здійснено аналіз сучасних рішень у галузі систем охорони та попередження загроз; описано чіткі вимоги до програмно-апаратного комплексу; розроблена структура взаємодії між компонентами системи; розроблено програмне забезпечення; реалізовано логіку обробки подій тривоги та попередження; виконано тестування працездатності комплексу в умовах, наближених до реальних.

Реалізована система може бути використана у приміщеннях різного призначення – від побутових до комерційних. Установка вирізняється енергоефективністю, портативністю та масштабованістю, що дозволить адаптувати її під вимоги замовника.

Ключові слова: ПРОГРАМНО-АПАРАТНИЙ КОМПЛЕКС, СИСТЕМА ОХОРОНИ, МІКРОКОНТРОЛЕР STM32, РОЗУМНИЙ ДІМ, СЕНСОРНИЙ ІНТЕРФЕЙС, ДЕТЕКЦІЯ РУХУ, СИГНАЛІЗАЦІЯ, FREERTOS, HAL.

ABSTRACT

Qualification work includes an explanatory note (57 p., 21 img.).

The object of development is a hardware–software complex for security purposes, implemented on the basis of the STM32 microcontroller. The system is designed to detect unauthorized actions within a controlled area and to promptly inform the user of potential threats. The complex combines functions of motion detection, indication of various types of alarms, and notification of dangerous changes in environmental parameters. In addition, a smart monitoring functionality has been implemented, enabling the user to configure different security modes.

The touch display provides the user with access to configuring security parameters and operating modes of the installation. The graphical interface is implemented with an intuitively clear structure, which guarantees simplicity of interaction and convenient navigation among functional elements. The software implementation is carried out using the C programming language, with the HAL library providing hardware abstraction. For efficient task management and real-time event handling, the FreeRTOS operating system is used.

During the development process, a comprehensive analysis of current security and threat-prevention solutions was carried out, followed by the formulation of specific requirements for the hardware–software system. The architecture for component interaction was designed, and the software was subsequently implemented. Functionality for handling alarm and warning events was developed, and the entire system was tested under conditions closely simulating real-world usage to ensure its reliability and effectiveness.

The implemented system can be used in premises of various purposes – from residential to commercial. The installation is distinguished by energy efficiency, portability, and scalability, allowing adaptation to the customer’s requirements.

Keywords: HARDWARE–SOFTWARE COMPLEX, SECURITY SYSTEM, STM32 MICROCONTROLLER, SMART HOME, TOUCH INTERFACE, MOTION DETECTION, ALARM SIGNALING, FreeRTOS, HAL.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.467200.002 ТЗ	Програмно-апаратний комплекс детекції руху на базі мікроконтролера STM32. Технічне завдання	4		
	А4	ІАЛЦ.467200.003 ТП	Програмно-апаратний комплекс детекції руху на базі мікроконтролера STM32. Відомість технічного проекту	3		
	А4	ІАЛЦ.467200.004 ПЗ	Програмно-апаратний комплекс детекції руху на базі мікроконтролера STM32. Пояснювальна записка	57		

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.46720.001 ОА		
Розробив		Ступницька С. М.			Програмно-апаратний комплекс детекції руху на базі мікроконтролера STM32 **Опис альбому**	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Радченко К. О.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ КВ-12	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В. О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.467200.005 Д1	Блок-схема алгоритму	1		
			детекції небезпеки.			
			Схема алгоритму			
	А4	ІАЛЦ.467200.006 Д2	Блок-схема алгоритму	1		
			вибірки даних.			
			Схема алгоритму			
	А4	ІАЛЦ.467200.007 Д3	Схема взаємодії	1		
			програмного			
			забезпечення.			
			Схема структурна			
	А4	ІАЛЦ.467200.008 Д4	Схема станів охорони.	1		
			Схема структурна			
		Диск CD-ROM	Текст пояснювальної	1		
			записки.			
			Графічний матеріал			

ЗМІСТ

					ІАЛЦ.467200.002 ТЗ		
	Арк.	№ докум.	Підпис	Дата			
Розроби		Ступницька С.М.			*Програмно-апаратний комплекс детекції руху на базі мікроконтролера STM32* ***Технічне завдання***	Літ.	Арку 1 / Аркушів 4
Перевір		Радченко К. О.					
Н. контроль		Клятченко Я.М.				КПІ ім. Ігоря Сікорського, ФПМ КВ-12	
Затвердив		Романкевич В.О.					

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Програмно-апаратний комплекс детекції руху на базі мікроконтролера STM32».

Галузь застосування: детекція руху та сигналізація порушень для підвищення рівня безпеки у приміщеннях та на об’єктах різного призначення.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на дипломне проектування на здобуття першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп’ютерних систем Національного технічного університету України «Київський Політехнічний Інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проєкту є створення програмно-апаратного комплексу для виявлення руху та оповіщення про тривожні події з можливістю налаштування режимів роботи через сенсорний інтерфейс. Призначенням системи є підвищення рівня безпеки приміщень шляхом автоматичної фіксації загроз та інформування користувача про їх виникнення у реальному часі.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації у періодичних виданнях та електронні статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється

- можливість виявлення руху за допомогою сенсора присутності (PIR);
- можливість визначення стану попередження або тривоги на основі показників температури та вологості;
- реалізація звукової (бузер) та візуальної (LED) сигналізації у разі загрози;
- наявність графічного інтерфейсу користувача (GUI) на сенсорному екрані для налаштування режимів охорони;
- можливість перегляду поточного стану системи через екран;
- підтримка взаємодії модулів у режимі реального часу з використанням FreeRTOS;
- гнучкість та масштабованість програмної структури для подальшого розширення функцій.

5.2. Вимоги до апаратного забезпечення

- наявність PIR-сенсора для виявлення руху;
- вбудований сенсор температури та вологості;
- сенсорний дисплей для взаємодії з користувачем;
- наявність звукового сигналізатора (пасивний бузер);
- наявність світлодіодів для візуального інформування.

5.3. Вимоги до програмного та апаратного забезпечення користувача

- наявність фізичного доступу до плати управління з сенсорним екраном.

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури за тематикою проекту	19.11.2024
2.	Розроблення та узгодження технічного завдання	30.12.2024
3.	Аналіз існуючих рішень	05.03.2025
4.	Підготовка матеріалів першого розділу дипломного проекту	10.04.2025
5.	Підготовка матеріалів другого розділу дипломного проекту	18.04.2025
6.	Підготовка графічної частини дипломного проекту	20.05.2025
7.	Оформлення документації дипломного проекту	25.05.2025
8.	Попередній огляд матеріалів диплому на кафедрі	03.06.2025

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.467200.002 ТЗ	Програмно-апаратний комплекс детекції руху на базі мікроконтролера STM32. Технічне завдання	4		
	А4	ІАЛЦ.467200.003 ТП	Програмно-апаратний комплекс детекції руху на базі мікроконтролера STM32. Відомість технічного проекту	3		
	А4	ІАЛЦ.467200.004 ПЗ	Програмно-апаратний комплекс детекції руху на базі мікроконтролера STM32. Пояснювальна записка	57		

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.467200.003 ТП		
Розробив		Ступницька С. М.			Програмно-апаратний комплекс детекції руху на базі мікроконтролера STM32 **Відомість технічного проєкту**	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Радченко К. О.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ КВ-12	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В. О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.467200.005 Д1	Блок-схема алгоритму	1		
			детекції небезпеки.			
			Схема алгоритму			
	А4	ІАЛЦ.467200.006 Д2	Блок-схема алгоритму	1		
			вибірки даних.			
			Схема алгоритму			
	А4	ІАЛЦ.467200.007 Д3	Схема взаємодії	1		
			програмного			
			забезпечення.			
			Схема структурна			
	А4	ІАЛЦ.467200.008 Д4	Схема станів охорони.	1		
			Схема структурна			
		Диск CD-ROM	Текст пояснювальної	1		
			записки.			
			Графічний матеріал			

					ІАЛЦ.467200.003 ТП	Арк.
						2
	Арк.	№ докум.	Підпис	Дата		

Пояснювальна записка
до дипломного проєкту

на тему: Програмно-апаратний комплекс детекції руху на базі мікроконтролера STM32

Київ – 2025

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

STM32 – (STMicroelectronics 32-bit) серія 32-бітних мікроконтролерів компанії STMicroelectronics на базі архітектури ARM Cortex-M.

MCU – (Microcontroller Unit) вбудований модуль, який об’єднує процесор, пам’ять і периферійні інтерфейси.

HAL – (Hardware Abstraction Layer) програмний рівень, що спрощує доступ до апаратних ресурсів мікроконтролера.

BSP – (Board Support Package) пакет підтримки апаратної плати, який містить драйвери і засоби ініціалізації.

RTOS – (Real-Time Operating System) операційна система, що забезпечує виконання задач у реальному часі з гарантованими часовими обмеженнями.

I2C – (Inter-Integrated Circuit) синхронний двопровідний інтерфейс для передачі даних між мікросхемами.

SPI – (Serial Peripheral Interface) синхронний послідовний інтерфейс з повнодуплексною передачею даних.

UART – (Universal Asynchronous Receiver/Transmitter) асинхронний інтерфейс для серійного обміну даними.

GPIO – (General-Purpose Input/Output) універсальні цифрові порти для введення або виведення сигналів.

DMA – (Direct Memory Access) механізм прямого обміну даними між периферією та пам’яттю без участі процесора.

ILI9341 – контролер керування кольоровим TFT-дисплеєм.

FT6206 – контролер ємнісного сенсорного екрана з підтримкою мультидотику.

PIR – (Passive Infrared Sensor) сенсор, що реагує на інфрачервоне випромінювання рухомих об’єктів.

HTS221 – цифровий сенсор для вимірювання температури та вологості.

ВСТУП

У сучасних умовах зростаючих загроз безпеці об’єктів різного призначення (від житлових будинків до комерційних приміщень) дедалі важливішим стає своєчасне виявлення несанкціонованого руху та швидке реагування на потенційні ризики. Інтеграція «розумних» систем охорони в інфраструктуру «розумного дому» й промислових об’єктів дозволяє забезпечити безперервний моніторинг, автоматизувати процес оповіщення та знизити ймовірність людської помилки. Саме це обумовлює високу актуальність розробки компактних, енергоефективних і масштабованих апаратно-програмних рішень для детекції руху.

Програмно-апаратний комплекс, реалізований на базі мікроконтролера STM32, поєднує в собі переваги високопродуктивного апаратного ядра, сучасних периферійних інтерфейсів та гнучкого програмного забезпечення.

Метою даної роботи є створення інтуїтивно зрозумілого та адаптивного комплексу детекції руху з можливістю вибору режимів охорони, індикації тривожних подій та гнучкого налаштування користувачем. Досягнення поставленої мети передбачає: аналіз існуючих рішень, формалізацію вимог, розробку апаратної архітектури, реалізацію програмних модулів, створення графічного інтерфейсу й проведення тестування в умовах, наближених до експлуатаційних. Запропонований комплекс може бути застосований у житлових, комерційних і промислових приміщеннях, гарантуючи надійність, енергоефективність та масштабованість відповідно до потреб користувача.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

1.1 Вступ

Розробка програмно-апаратного комплексу для виявлення руху на основі мікроконтролера STM32 потребує попереднього аналізу сучасних апаратних і програмних рішень у сфері охоронних технологій [1]. На цьому етапі важливо дослідити як готові комерційні продукти, так і відкриті технічні розробки, що дає змогу оцінити їхні функціональні можливості, архітектуру та рівень ефективності. Особливу увагу слід приділити системам, які поєднують сенсори руху, засоби керування, інтерфейси відображення інформації та механізми сповіщення.

1.2 Аналіз комерційних охоронних систем

1.2.1 Загальна характеристика

Комерційні охоронні системи зазвичай представляють собою комплексне рішення, що охоплює весь процес забезпечення безпеки — від виявлення руху та передавання сповіщень до централізованого моніторингу й організації реагування. Основною метою таких систем є своєчасне виявлення потенційної загрози, передача сповіщення користувачеві або службі безпеки, а також збереження записів подій для подальшого аналізу [2].

Базова архітектура комерційної системи зазвичай включає:

- набір детекторів руху (PIR, мікрохвильові, ультразвукові або комбіновані),
- центральний контролер (хаб),
- модулі зв’язку (Wi-Fi, GSM, Ethernet),
- засоби сповіщення (сирени, push-нотифікації, SMS),
- а також, інтегроване програмне забезпечення з інтерфейсом керування.

Найпоширенішим прикладом датчиків є пасивні інфрачервоні сенсори (PIR), які реагують на зміну температурного фону — зокрема, на теплове випромінювання людського тіла. Для підвищення надійності багато сучасних систем застосовують комбіновані детектори, які поєднують PIR із мікрохвильовим радаром. Це дозволяє значно зменшити кількість хибних спрацювань, фіксуючи рух лише тоді, коли обидва сенсори виявляють зміну навколишнього середовища [2].

Центральний контролер виконує функції обробки даних, координації роботи сенсорів, шифрування трафіку, а також управління логікою сценаріїв охорони. У сучасних рішеннях часто застосовується апаратне шифрування AES, що забезпечує захист даних у бездротових мережах. Зв’язок із зовнішнім світом може відбуватися через GSM/3G/4G, Ethernet або Wi-Fi. Окремі моделі передбачають резервні канали передачі — наприклад, при втраті інтернету активується SMS-повідомлення або дзвінок на вказаний номер [3].

Інсталяція таких систем здійснюється професіоналами. За допомогою спеціалізованого ПЗ інженер налаштовує кожен сенсор, вказуючи чутливість, тип тригера, затримку на активацію, а також об'єднуючи пристрої в логічні групи — наприклад, "периметр", "вхідна група", "внутрішнє приміщення". Це дозволяє реалізувати складні сценарії: часткова охорона, нічний режим, автоматичне перемикання зон залежно від часу тощо.

1.2.2 Приклади сучасних комерційних рішень

Ajax Systems — українська компанія, яка виробляє бездротові охоронні системи з високим рівнем захисту. Їхні рішення відзначаються продуманим дизайном, розширеним набором датчиків (у тому числі з детекцією розбиття скла, витоку води, задимлення) та фірмовим протоколом Jeweller, що забезпечує швидку та стабільну бездротову передачу даних (рис. 1).

Hikvision — китайський виробник, відомий переважно системами відеонагляду, проте також пропонує гібридні охоронні рішення з інтеграцією відеоаналітики, детекції руху, керування доступом і тривожних сповіщень [4].

Dahua Technology — ще один великий гравець у галузі безпеки, що спеціалізується на розробці мережевих камер і комплексних систем, які поєднують охоронну сигналізацію з аналітичними можливостями на базі ШІ.

Рис. 1 – Приклад комерційного датчика руху від Ajax Systems

1.2.3 Переваги комерційних систем

- Комерційні системи проходять сертифікацію за європейськими стандартами безпеки (наприклад, EN 50131), що гарантує їхню працездатність за будь-яких умов, включаючи температурні перепади, електромагнітні перешкоди чи відключення живлення.
- Апаратне та програмне шифрування, ізоляція каналів зв’язку, захист від перехоплення сигналів — усе це забезпечує високий рівень конфіденційності інформації.
- Фірмові додатки дозволяють керувати системою з будь-якої точки світу, переглядати сповіщення, налаштовувати сценарії охорони та віддалено змінювати конфігурацію пристроїв.
- Регулярна перевірка рівня заряду батарей, сили сигналу, стану зв’язку з хабом. У разі проблем користувачі можуть розраховувати на професійну підтримку виробника.

- Можливість побудови гнучких конфігурацій системи з урахуванням типу об’єкта, його планування та специфіки загроз.

1.2.4 Недоліки комерційних систем

- Повноцінна система з декількома датчиками, центральним хабом, сиреною та підтримкою GSM може коштувати кілька сотень євро,
- не враховуючи витрат на монтаж і обслуговування. Додатково щомісячна оплата за доступ до хмарних функцій може становити 30–50 євро.
- Більшість рішень не допускають кастомізації. Неможливо змінити алгоритми виявлення, додати нові типи сенсорів або інтегрувати нестандартні модулі без дозволу або підтримки виробника.
- У разі збоїв у мережі мобільного оператора або провайдера порушується зв’язок із хмарним сервером, а отже — недоступні повідомлення, аналітика та віддалене керування.
- Якщо користувач хоче розгорнути систему в умовах повної відсутності інтернету або GSM-зв’язку, значна частина функціоналу може бути недоступною або не працювати коректно.

1.3 Аналіз відкритих програмних платформ та DIY-підходи

1.3.1 Загальна характеристика

На противагу закритим комерційним системам, які передбачають мінімальний вплив користувача на внутрішні процеси, світ відкритих платформ дає змогу реалізувати повністю кастомізовану систему безпеки — із нуля або на основі вже наявних модулів. Цей підхід отримав назву DIY (Do It Yourself), і він особливо популярний серед інженерів-ентузіастів, студентів та розробників систем розумного дому (рис. 2) [5].

Ключовими компонентами DIY-систем є відкриті апаратні платформи — зокрема, ESP32, STM32 Nucleo, Raspberry Pi Pico, Teensy та інші мікроконтролери з відкритим SDK. У парі з ними працюють прості сенсори: HC-SR501 (інфрачервоний детектор руху), RCWL-0516 (мікрохвильовий радар), DHT11/22 (температура і вологість) або VL53L0X (лазерна дальність).

Програмна частина базується на таких відкритих середовищах:

- ESPHome — фреймворк для налаштування прошивок пристроїв на ESP;
- Tasmota — легка ОС для ESP-чіпів з веб-інтерфейсом керування;
- OpenHAB, Node-RED, Home Assistant — платформи для побудови локального або гібридного (локально-хмарного) контролю «розумного дому».

Для обміну подіями широко використовуються протоколи MQTT (легкий брокер повідомлень), HTTP/HTTPS API, а також WebSocket для синхронної взаємодії між мікроконтролером і сервером. У результаті DIY-система може реагувати на спрацювання сенсора, автоматично запускати камеру, надсилати push-повідомлення на смартфон або активувати сирену — і все це без централізованого хабу, виключно через сценарії в Node-RED чи YAML-опис у Home Assistant [6].

Рис. 2 – Приклад DIY рішення

1.3.2 Переваги DIY-рішень

- Вартість базової DIY-конфігурації може бути в 5–10 разів меншою, ніж аналогічної комерційної. Наприклад, ESP32 з датчиком руху можна зібрати менш ніж за 10 євро. Жодних ліцензій чи платних оновлень — усе програмне забезпечення безкоштовне.
- Користувач сам визначає сценарії охорони, алгоритми фільтрації, умови затримки тощо. Можна реалізувати часткову охорону окремих зон, часові профілі, адаптивне керування освітленням чи камерами залежно від типу тривоги.
- Відкриті протоколи (MQTT, REST API, Webhook) дозволяють легко під'єднувати DIY-систему до Alexa, Google Assistant, Telegram-бота або власного веб-інтерфейсу без складного шлюзу.
- До однієї плати можна під'єднати кілька різних сенсорів, дисплеїв, модулів GSM або LoRa, створюючи систему, адаптовану до конкретного об'єкта — будинку, гаража, складу тощо.

1.3.3 Обмеження та виклики

- DIY-плати не проходять сертифікаційного тестування. Їхня робота в умовах перепадів напруги, екстремальних температур або високої вологості не гарантується. Система може давати збої, що критично для безпеки.
- Arduino або Raspberry Pi мають значне базове споживання енергії (до кількох сотень мА), що робить їх менш придатними для автономної роботи. Для живлення від батарей потрібна оптимізація режимів сну або застосування спеціалізованих плат із low-power-режимами.
- Різні пристрої можуть працювати на різних прошивках, з різними версіями бібліотек, конфігураціями й навіть способами

комунікації. Це ускладнює централізоване оновлення, синхронізацію або відновлення після збою.

- Без належного шифрування, авторизації та сегментації мережі DIY-системи стають легкою мішенню для зловмисників. Особливо це стосується випадків відкритого доступу до MQTT-брокера або веб-інтерфейсу без паролю.
- Хоча багато рішень мають графічні інтерфейси, справжнє налаштування вимагає знань у C/C++, Python або JavaScript, розуміння логіки GPIO, UART/I2C/SPI.

1.4 Переваги власного рішення на базі STM32

1.4.1 Інтеграція гнучкості та надійності

Власний апаратно-програмний комплекс, побудований на мікроконтролерах STM32, поєднує в собі дві найважливіші якості — відкритість і прогнозованість. По-перше, архітектура STM32 дозволяє заглибитися в будь-які внутрішні механізми: від налаштування параметрів тактування до оптимізації режимів енергоспоживання. Це означає, що, з одного боку, можна вільно модифікувати драйвери й алгоритми обробки, а з іншого — впевнено покладатися на перевірені на виробництві й багатократно протестовані апаратні блоки, серед яких ядро Cortex-M4 із вбудованим блоком плаваючої точки. Саме завдяки такому поєднанню складні обчислювальні процедури, наприклад адаптивне фільтрування шуму або аналіз часових закономірностей у відгуку сенсорів, виконуються миттєво, без затримок, що вкрай важливо для систем безпеки [7].

У режимі очікування мікроконтролер може споживати лічені мікроампери струму, що робить можливим довготривалу автономну роботу від батарей чи сонячних панелей. Такий рівень енергоефективності неможливо досягти з використанням універсальних одноплатних комп’ютерів чи

Arduino-рішень стандартного класу. При цьому у звичайному активному режимі STM32 здатен обробляти дані від десятків сенсорів, відправляти їх через кілька інтерфейсів та оновлювати графічний інтерфейс без жодних пропусків або збоїв, що забезпечує безперервний захист в умовах постійної активності.

1.4.2 Модульність та масштабованість

Програмно-апаратний комплекс на базі STM32L475 розроблений за принципами модульності, що дозволяє з легкістю адаптувати й розширювати систему відповідно до змін вимог чи нових сценаріїв використання. Архітектура побудована на середовищі FreeRTOS, де кожна функціональна частина — окрема задача з власними пріоритетами та контекстом виконання. Завдяки цьому підхід до інтеграції нових компонентів зводиться до додавання окремого модуля без необхідності змін у вже працюючій частині системи [8].

У поточному рішенні використовуються лише обрані можливості плати B-L475E-IOT01A: зокрема, модулі детекції руху через PIR-сенсор, сенсор температури та вологості. Проте ця плата оснащена значно ширшим спектром вбудованих периферійних модулів, які залишаються незадіяними в межах даного проєкту, але можуть бути з легкістю інтегровані в подальшому. Серед них [9]:

- акселерометр/гіроскоп LSM6DSL — дозволяє виявляти вібрації, падіння або нахили, що може бути корисно для розширення охоронних сценаріїв;
- магнітометр LIS3MDL — здатен визначати орієнтацію пристрою в просторі, відкриваючи можливості для навігаційних або позиційних сценаріїв;
- барометр LPS22HB — забезпечує дані про атмосферний тиск, що дозволяє виявляти відкриття дверей або вікон за різкими змінами тиску;
- мікрофон MP34DT01 — може використовуватись для аналізу звуків;

- модуль Bluetooth (BlueNRG-MS) — забезпечує локальний бездротовий зв’язок із мобільними пристроями або іншими вузлами системи;
- Wi-Fi-модуль ISM43362-M3G-L44 — дозволяє підключити систему до мережі Інтернет без додаткового зовнішнього обладнання.

Кожен із цих модулів уже має підтримку в бібліотеках STM32Cube BSP і може бути підключений до системи. Завдяки абстрагованій структурі коду, нові компоненти легко вбудовуються в існуючу екосистему: розширюється список задач, додаються обробники подій, але ядро системи залишається незмінним.

Більше того, FreeRTOS дозволяє чітко визначити пріоритети виконання задач, що забезпечує передбачувану поведінку навіть у складних конфігураціях. Наприклад, задачі збору кліматичних даних можуть виконуватись із низьким пріоритетом, а задачі, пов’язані з тривожним реагуванням (детекція руху, порушення меж), отримують максимальний пріоритет і час виконання [9].

Таким чином, хоча в межах поточного проєкту використовуються лише базові компоненти, закладена структура дозволяє у будь-який момент додати нові функції — від акустичної тривоги до аналізу просторового положення — без необхідності редизайну всього комплексу. Це робить рішення не лише адаптивним, а й перспективним для довготривалого розвитку.

1.4.3 Зручність для користувача та економічна доцільність

Інтерфейс користувача — не менш важливий елемент будь-якого «смарт» продукту. Рішення передбачає сенсорний TFT-дисплей із власною GUI-бібліотекою, що забезпечує візуально приємний та інтуїтивно зрозумілий досвід налаштування. Користувач не мусить вивчати командний рядок або звертатися до інженера щоразу, коли треба змінити чутливість датчика чи додати нову зону охорони.

Ще однією важливою перевагою є можливість «повітряного» оновлення прошивки через USB-порт або бездротовий модуль. Це означає, що вдосконалення алгоритмів, виправлення помилок чи додавання нових функцій відбуваються без фізичного доступу до пристрою. Поєднання простоти використання, можливості розширення та економічної доцільності робить систему на базі STM32 не просто альтернативою комерційним або DIY-рішенням, а самостійним і перспективним продуктом на ринку охоронних технологій [10].

1.5. Висновки до розділу

Проведений аналіз існуючих підходів до створення охоронних систем дозволив виявити основні напрями розвитку цієї галузі та сформувати уявлення про ключові переваги і недоліки кожного з них. Сучасний ринок пропонує як готові комерційні рішення, так і відкриті інструменти для самостійного проєктування, що охоплюють різні цільові аудиторії — від бізнесу до індивідуальних користувачів та дослідників.

Комерційні охоронні комплекси характеризуються високим рівнем інтеграції, сертифікованою надійністю та зручністю користування, однак їхня закритість, обмеженість у модифікаціях та висока вартість створюють бар’єри для гнучкого впровадження в нестандартні середовища чи обмежені бюджети.

У противагу їм відкриті програмно-апаратні платформи забезпечують максимальну свободу конфігурації, швидке прототипування та низький поріг входу з точки зору вартості. Проте вони вимагають більш глибоких технічних знань, не гарантують стабільності в складних умовах та можуть бути менш захищеними від зовнішніх загроз.

У цьому контексті найбільш збалансованим підходом є створення власного програмно-апаратного рішення на базі мікроконтролера, що дозволяє зберегти контроль над архітектурою, реалізувати критично важливі функції з урахуванням специфіки об’єкта, а також забезпечити необхідну

масштабованість, енергоефективність і надійність. Використання реального часу, чіткої модульної структури та можливості інтеграції широкого спектру сенсорів робить такий підхід особливо перспективним для побудови адаптивних систем безпеки нового покоління.

2. АНАЛІЗ ОБРАНИХ ТЕХНОЛОГІЙ

2.1 Комплексне середовище розробки STM32CubeIDE

STM32CubeIDE є офіційним інструментом від STMicroelectronics, який об’єднує всі необхідні компоненти для створення, відлагодження та програмування мікроконтролерів STM32. Перш за все, це зручний редактор коду з підтримкою мов C і C++, а також вбудований компілятор і відладчик, що дають змогу здійснювати повний цикл розробки в одному середовищі без залучення додаткових зовнішніх утиліт. Завдяки цьому можна зосередитися на логіці застосунку, не витрачаючи час на налаштування сторонніх інструментів [11].

Другим ключовим елементом є інтегрований графічний конфігуратор CubeMX: через інтуїтивно зрозумілий інтерфейс можна вибрати конкретну серію STM32, налаштувати тактування, GPIO та інші периферійні модулі, після чого середовище автоматично генерує початковий (boilerplate) код. Завдяки такій інтеграції мінімізується ризик помилок у конфігурації апаратури, оскільки IDE попереджає про можливі конфлікти ресурсів і самостійно формує функції ініціалізації.

Важливою перевагою є потужні відлагоджувальні інструменти: STM32CubeIDE підтримує роботу зі SWD (ST-Link) та надає можливості для встановлення брейкпоінтів, покрокового виконання коду та перегляду регістрів периферії в реальному часі. Окрім цього, середовище дозволяє використовувати трасування для аналізу виконання програми, що спрощує виявлення та виправлення помилок у роботі з різноманітними блоками мікроконтролера [11].

Ще однією сильною стороною STM32CubeIDE є вбудована підтримка FreeRTOS: достатньо активувати відповідний middleware у проекті, і IDE автоматично згенерує файли конфігурації, створить каркас для завдань та

налаштує планувальник. Таким чином, робота з операційною системою реального часу стає максимально простою та прозорою.

Нарешті, вбудована інтеграція зі сторонніми сховищами коду (наприклад, Git) дозволяє здійснювати коміти, працювати з гілками й виконувати мерджі, не виходячи з середовища. Це спрощує керування версіями і дозволяє відслідковувати історію змін без додаткових зовнішніх інструментів [12].

У підсумку, STM32CubeIDE забезпечує:

- єдину інтегровану платформу для написання, налагодження й програмування STM32-проєктів;
- швидке налаштування й автоматичну генерацію початкового коду через графічний конфігуратор;
- потужні засоби відлагодження з підтримкою SWD і трасування;
- вбудовану інтеграцію FreeRTOS для швидкого старту з ОС реального часу;
- можливість безпосередньої роботи з Git, що спрощує керування версіями.

Ці можливості роблять STM32CubeIDE зручним, ефективним і надійним рішенням для розробки проєктів будь-якого рівня складності, від прототипу до готового комерційного продукту.

2.2 Мова С для мікроконтролерного програмування

Мова програмування С є класичним вибором для вбудованих систем завдяки збалансованому поєднанню низькорівневого контролю апаратури та достатньо високого рівня абстракції для зручності розробки. У проєкті програмно-апаратного комплексу детекції руху на базі STM32 використання С дозволяє максимально ефективно взаємодіяти з регістрами мікроконтролера, оптимізувати споживання пам’яті та забезпечити достатню швидкість виконання критичних ділянок коду [13].

По-перше, мова С надає прямий доступ до пам’яті через вказівники та можливість оперувати окремими бітами в регістрах. Завдяки цьому можна встановлювати й зчитувати стани виходів, конфігурувати периферійні модулі (таймери, UART, SPI, I²C тощо) і регулювати режими їх роботи з мінімальними накладними витратами. Вбудовані побітові операції (наприклад, &, |, <<, >>) спрощують маніпуляції з апаратними флагами, дозволяють швидко змінювати окремі розряди в регістрі та реалізовувати ефективні алгоритми обробки сигналів.

По-друге, компактність і передбачуваність коду С роблять його ідеальним для виконання на мікроконтролерах із обмеженими ресурсами. Оскільки вбудовані застосунки часто працюють із суворо фіксованим обсягом Flash- і RAM-пам’яті, важливо, щоб згенерована двійкова прошивка займала якомога менше місця. При правильній побудові модулів та використанні оптимізації компілятора (-O2, -Os) можна досягти мінімального розміру і водночас задовольнити вимоги до продуктивності.

По-третє, структура С-коду сприяє модульності та простоті підтримки. Розбиття логіки на окремі .c і .h файли дає змогу виділяти блоки для роботи з апаратурою, обробки даних, керування системними таймерами чи інтерфейсами зв’язку. Це полегшує читання коду, пошук помилок і подальшу розширюваність. Наприклад, функції, що керують шиною обміну чи ініціалізують тактування, розміщуються в окремому драйвері, тому при заміні мікроконтролера чи апаратної платформи достатньо змінити лише цей модуль, залишаючи незайманими алгоритми вищого рівня [13].

Нарешті, поширеність С серед інженерів-розробників і багата екосистема бібліотек полегшують інтеграцію сторонніх модулів та прикладних рішень. У випадку STM32 основним інструментарієм є HAL (Hardware Abstraction Layer) та LL (Low-Layer), що реалізовані саме на С. Ці бібліотеки забезпечують уніфіковані API для різних серій контролерів,

полегшують переїзд між лінійками STM32 і скорочують час розробки низькорівневих драйверів.

Незважаючи на численні переваги, мова С має також певні обмеження. Відсутність автоматичного управління пам’яттю змушує розробника самостійно стежити за виділенням і звільненням динамічних буферів, що може призводити до помилок (витоків пам’яті або «висячих» вказівників). Крім того, у стандартній бібліотеці С немає вбудованих засобів для реалізації багатозадачності чи синхронізації — для цього зазвичай залучають сторонні рішення (наприклад, FreeRTOS).

2.3 STM32 та стандартна бібліотека HAL

STM32—це сімейство 32-розрядних мікроконтролерів на базі ARM Cortex-M, яке відоме своєю продуктивністю та енергоефективністю. Завдяки вбудованим таймерам, АЦП, комунікаційним інтерфейсам (I^2C, SPI, UART) і низці інших периферійних блоків, STM32 підходить як для простих задач керування, так і для складніших вбудованих систем.

Для спрощення роботи з усіма цими модулями STMicroelectronics надає стандартну бібліотеку HAL (Hardware Abstraction Layer). Головна ідея HAL—забрати необхідність писати власний код безпосередньо в регістри контролера.

Передача даних: реалізовані функції забезпечують зручний обмін інформацією без необхідності прямого втручання в низькорівневі регістри. Завдяки підтримці DMA стає можливим передавати великі обсяги даних у фоновому режимі, не блокуючи основне ядро та дозволяючи паралельне виконання інших задач [14].

Обробка подій: після завершення передавання або приймання даних автоматично викликаються callback-функції, що дозволяє організувати асинхронну логіку без постійного опитування периферійних модулів. Це значно спрощує структуру коду та підвищує його ефективність.

Переваги HAL:

- Уніфікація: однакові назви функцій і структур для різних серій STM32.
- Швидка розробка: не потрібно вручну налаштовувати регістри, можна швидко отримати працюючу конфігурацію.
- Асинхронність: підтримка переривань і DMA без додаткових зусиль.
- Інтеграція з FreeRTOS: просте поєднання з ОС реального часу, згенеровані налаштування для NVIC і пріоритетів переривань.

Обмеження HAL:

- Більше накладні витрати: через абстракцію код займає трохи більше місця у Flash і може працювати трохи повільніше, ніж впритул оптимізовані регістрові звернення або LL-драйвери.
- Менш гнучкий: в разі потреби максимальної оптимізації чи нестандартних конфігурацій доводиться відходити від HAL на рівень LL або безпосередніх регістрових записів.

Загалом, HAL поєднує простоту використання з достатньою продуктивністю. Для дипломного проєкту вибір HAL дозволяє швидко реалізувати всі необхідні модулі—від роботи з таймерами до обміну даними—без глибокого занурення в бітові маски регістрів. Це зменшує час розробки й робить код зрозумілішим для подальшого супроводу [15].

2.4 Основні апаратні інтерфейси для передачі даних (I²C, SPI, UART)

У кожному мікроконтролері є кілька спеціальних “шляхів” для обміну даними між самим контролером і зовнішніми пристроями. У випадку STM32 найчастіше використовують три основні інтерфейси — I²C, SPI та UART — які відрізняються за кількістю ліній, швидкістю передачі та складністю протоколу.

I²C працює на двох лініях: одна для даних (SDA), інша для такту (SCL). Контролер, який ініціює обмін, називається майстром, а всі інші —

підлеглими. Майстер посилає адресу потрібного пристрою, і, якщо підлеглий відповідає “так” (через ACK), починається обмін байтами. Цей інтерфейс добре підходить, коли потрібно під’єднати багато невимогливих до швидкості датчиків, оскільки для них достатньо помірної швидкості до приблизно одного мегагерца. У HAL все налаштовується через одну структуру, і розробнику залишається лише викликати готові функції, не заглиблюючись у тонкощі регістрів.

SPI зазвичай використовує чотири лінії: MOSI (від майстра до підлеглого), MISO (від підлеглого до майстра), SCK (тактування) і CS (вибір, до якого підлеглого звертатися). Майстер піднімає лінію CS у стан низького сигналу для обраного пристрою, формує тактові імпульси і одночасно передає й отримує дані. Через повнодуплексну передачу SPI може досягати значно вищих швидкостей, ніж I²C, тому його часто використовують для дисплеїв чи швидкої флеш-пам’яті. У HAL процес налаштування полягає в тому, щоб заповнити структуру параметрів, після чого можна викликати готові функції для відправлення та отримання даних.

UART (або USART у асинхронному режимі) працює простіше: потрібні лише дві лінії — TX для передачі та RX для прийому. Досить домовитися про швидкість (наприклад, 115200 біт/с), і дані передаються послідовно одним байтом за раз, з визначеним форматом кадра (стартовий, біти даних, стопові). UART не потребує спільного тактування, кожен біт синхронізується всередині приймача. Цей інтерфейс зручно використовувати для надсилання текстових повідомлень або команд між контролером і, наприклад, модулем Bluetooth чи зовнішнім комп’ютером. Через HAL налаштування зводиться до визначення швидкості, формату кадра та вибору режиму роботи (блокуючий чи з перериваннями).

2.5 Реалізація багатозадачного керування на базі FreeRTOS

FreeRTOS є однією з найпопулярніших операційних систем реального часу, яка активно використовується у вбудованих системах. Вона дозволяє виконувати кілька незалежних задач одночасно, що особливо важливо для складних програмно-апаратних рішень. Завдяки FreeRTOS можна розділити функціональність системи на окремі частини, кожна з яких виконується паралельно, відповідно до заданого пріоритету. Це дозволяє ефективно організувати роботу пристрою, уникнути затримок у виконанні важливих дій і забезпечити стабільність у режимі реального часу [8].

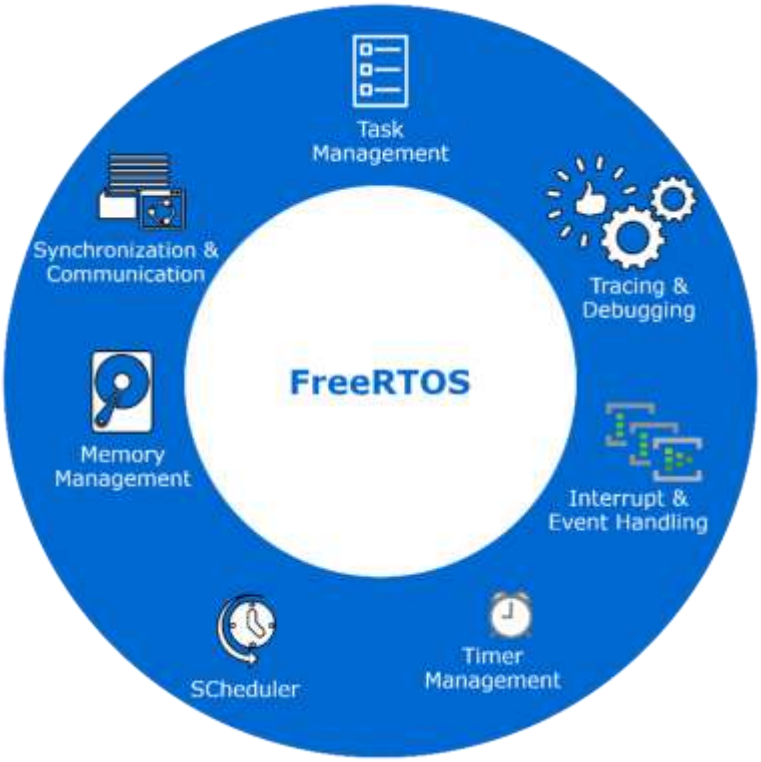


Рис. 3 – Можливості FreeRTOS

Основними складовими FreeRTOS є (рис. 3):

– планувальник задач (Scheduler), який визначає порядок виконання задач і перемикає їх відповідно до заданих пріоритетів;

– задачі (Tasks), кожна з яких є окремою функцією, що виконується незалежно від інших;

– черги (Queues), які дозволяють передавати дані між задачами без конфліктів;

– семафори та м’ютекси (Semaphores and Mutexes), що забезпечують синхронізацію між задачами та контроль доступу до спільних ресурсів;

– таймери (Timers), які використовуються для періодичного виконання певних дій.

FreeRTOS було обрано для цього проєкту завдяки її простоті у використанні, широкій підтримці мікроконтролерів STM32, мінімальним вимогам до ресурсів і великій спільноті користувачів. Її API є доступним і зрозумілим, що дозволяє швидко налагодити багатозадачне середовище навіть у невеликих проєктах. FreeRTOS також легко масштабується та підтримує розширення, наприклад, стек TCP/IP або модулі для роботи з файловими системами [11].

У межах розробленого програмно-апаратного комплексу FreeRTOS використовується для організації чіткої, структурованої роботи окремих функціональних блоків. Основними задачами є:

- опитування сенсорів і збір даних про рух;
- обробка отриманої інформації та прийняття рішення щодо подальших дій;
- оновлення даних на дисплеї, щоб користувач міг бачити поточний стан системи;
- передача обробленої інформації по бездротовому каналу до зовнішнього пристрою.

Такий підхід дозволяє ефективно розподілити ресурси мікроконтролера та забезпечити стабільну, безперебійну роботу всієї системи в режимі реального часу.

2.6 Теоретичні основи детекції руху

Детекція руху — це процес виявлення зміни положення об'єктів у просторі за певний інтервал часу. У контексті вбудованих систем, які працюють в реальному часі, завдання виявлення руху полягає в отриманні достовірного сигналу про наявність або відсутність руху в зоні спостереження. Цей сигнал може використовуватись для активації подальших дій — наприклад, ввімкнення сигналізації, запису відео, надсилання повідомлення або оновлення інформації на інтерфейсі користувача.

У практиці проектування подібних систем розрізняють кілька основних підходів до реалізації детекції руху, які поділяються на апаратні та програмні. Апаратна детекція базується на спеціалізованих сенсорах (наприклад, PIR або мікрохвильових), тоді як програмна зазвичай застосовується у відеоаналітиці або робототехніці й ґрунтується на аналізі змін у послідовності зображень [8].

На апаратному рівні найпоширенішими засобами виявлення руху є:

- Пасивні інфрачервоні сенсори (PIR) — виявляють зміну інфрачервоного випромінювання в полі зору.
- Мікрохвильові сенсори — працюють за принципом ефекту Доплера, генеруючи радіохвилі і фіксуючи їхнє відбиття.
- Ультразвукові сенсори — подібно до мікрохвильових, але використовують звукові хвилі.
- Комбіновані сенсори — поєднують декілька фізичних методів для підвищення точності.

З точки зору електронної реалізації, PIR-сенсор є найпростішим у використанні: він видає логічну одиницю при фіксації теплового руху, що дозволяє напряму підключити його до мікроконтролера.

Мікрохвильові сенсори, хоча і більш чутливі, часто створюють хибні спрацювання внаслідок високої проникності сигналу через стіни, вікна чи інші перешкоди. Тому в багатьох промислових системах використовується

гібридний підхід — коли PIR працює у зв’язці з мікрохвильовим сенсором для верифікації сигналу.

Після отримання сигналу про рух постає завдання його правильної інтерпретації. Примітивні системи реагують одразу на одиничний імпульс з сенсора, однак такий підхід створює ризик хибних спрацювань — через пил, протяги, комах або теплові хвилі.

Більш складні системи, особливо у відеоаналітиці або робототехніці, застосовують цифрову обробку зображень — порівняння кадрів, обчислення різниці пікселів, побудову карти руху. Такі алгоритми потребують високої обчислювальної потужності та значного обсягу оперативної пам’яті, що виходить за межі типових можливостей мікроконтролерів STM32 базового рівня.

У сучасних системах також актуальними є підходи з елементами машинного навчання — аналіз патернів руху, розпізнавання аномальних поведінкових сценаріїв, автоматичне налаштування чутливості тощо. Проте реалізація таких функцій потребує використання зовнішніх процесорів, FPGA або Edge AI-модулів, що ускладнює архітектуру.

2.7 Основи виявлення руху за допомогою інфрачервоних сенсорів

У мікроконтролерних системах на STM32 зазвичай застосовуються спрощені евристичні моделі: рух визначається як перехід сигналу з "0" у "1" від PIR-сенсора, після чого система активує відповідну функцію. Така схема дозволяє досягти балансу між простотою реалізації, точністю реагування та ефективністю витрат енергоресурсів, що є критичним у вбудованих системах, які працюють автономно та мають обмежені ресурси.

PIR-сенсори не випромінюють сигнал самостійно, а працюють за принципом фіксації змін інфрачервоного випромінювання у полі зору. Цей підхід дозволяє зменшити енергоспоживання та уникнути впливу оптичних або звукових перешкод.

За відсутності руху детектор отримує певну кількість інфрачервоного випромінювання, і вихідний сигнал сенсора залишається нульовим. Коли ж у полі зору з'являється об'єкт, наприклад, людина, яка рухається, баланс порушується, і сенсор формує цифровий сигнал, що подається на контролер. У системі на основі STM32 це дозволяє швидко і без зайвих обчислень ініціювати відповідну подію — увімкнення сигналізації, оновлення інтерфейсу або інша реакція [2].

До переваг PIR-сенсорів можна віднести низьку вартість, компактні розміри, високу чутливість у межах від кількох до десятків метрів та простоту інтеграції в мікроконтролерні системи. Крім того, більшість модулів мають налаштування чутливості та часу затримки після спрацювання, що дозволяє гнучко адаптувати їх до конкретного середовища. Проте існують і обмеження: PIR-сенсори чутливі лише до теплового руху, не працюють через скляні або непрозорі перешкоди, а також можуть давати хибні спрацювання при наявності джерел тепла або потоків повітря.

У межах розробленого проєкту PIR-сенсор виконує роль основного датчика виявлення руху. Його підключення до мікроконтролера через цифровий вхід з налаштуванням на переривання дозволяє досягти ефективного реагування у реальному часі без постійного опитування, що економить ресурси процесора. У поєднанні з іншими засобами відображення інформації та взаємодії з користувачем PIR-сенсор формує базу для функціонального охоронного сценарію.

2.8 Технологія збору даних про температуру та вологість

Контроль мікроклімату є важливою частиною багатьох систем моніторингу та автоматизації. У проєкті використано цифровий сенсор HTS221, інтегрований на відладній платі STM32. Цей сенсор є компактним і високоточною мікросхемою для одночасного вимірювання температури та відносної вологості повітря. Він забезпечує стабільну роботу в широкому

діапазоні температур та рівнів вологості, а також має цифровий інтерфейс зв’язку, що значно спрощує його інтеграцію.

HTS221 використовує ємнісний метод вимірювання вологості та терморезистивний для температури. Його перевагою є наявність вбудованих калібрувальних коефіцієнтів, що дозволяють мікроконтролеру точно обробляти "сирі" показники та одразу отримувати фізично інтерпретовані значення. Передача даних здійснюється через інтерфейс I2C, що дозволяє заощадити виводи мікроконтролера та організувати стабільний зв'язок навіть на великій відстані всередині плати [3].

У рамках даного проєкту дані із сенсора HTS221 зчитуються з певною періодичністю за допомогою окремої задачі FreeRTOS. Отримані значення використовуються не лише для виводу на екран, але й можуть бути основою для майбутньої інтеграції з адаптивними сценаріями охорони або вентиляції. Така реалізація дозволяє забезпечити гнучкість, модульність та масштабованість системи.

2.9 Протоколи взаємодії з сенсорними екранами: I2C і координатна система

Інтеграція сенсорного інтерфейсу у склад програмно-апаратного комплексу істотно підвищує ергономіку та зручність взаємодії з користувачем, особливо у випадках, коли традиційні кнопки або перемикачі є недоцільними через обмежений простір чи потребу в гнучкості інтерфейсу. У розробленій системі використано ємнісний сенсорний контролер FT6206, інтегрований у модуль графічного дисплея Adafruit 1947. Даний чип широко застосовується у вбудованих системах завдяки своїй простоті у використанні, надійності та підтримці багаторазових торкань.

FT6206 є автономним сенсорним контролером, який виконує попередню обробку сигналів з сенсорного шару (скляної панелі з провідною сіткою) та

видає координати точок дотику у вигляді цифрових значень. Він підтримує до двох одночасних дотиків та надає інформацію про [5]:

- кількість активних торкань;
- координати X та Y кожного дотику;
- ідентифікатор точки (ID), що дозволяє відстежувати переміщення.

Передача даних до мікроконтролера здійснюється через I2C-протокол, що робить підключення до мікроконтролера STM32 надзвичайно простим — достатньо лише двох ліній: SCL (Serial Clock Line) та SDA (Serial Data Line).

У цій системі STM32 виконує роль ведучого пристрою (master), ініціюючи комунікацію, а FT6206 — роль підлеглого (slave), з типовою адресою 0x38. Передача даних відбувається за запитом — мікроконтролер читає відповідні регістри FT6206, в яких зберігаються координати дотиків.

Контролер FT6206 передає координати дотиків у вигляді послідовності байтів, де кожен дотик описується двома координатами (X і Y), що мають 12-бітну роздільну здатність і об'єднуються з двох байтів.

Система координат має початок у верхньому лівому куті дисплея (точка з координатами 0,0), що відповідає загальноприйнятому підходу у графічних інтерфейсах. Це дозволяє точно співставляти положення дотику з візуальними елементами інтерфейсу.

Для взаємодії з сенсором використовується програмна обгортка — драйвер FT6X06, який реалізує абстрактний доступ до функцій сенсора.

Основні функції драйвера включають:

- ft6x06_TS_Init() — ініціалізація сенсора;
- ft6x06_TS_DetectTouch() — перевірка наявності дотику та зчитування кількості активних точок;
- ft6x06_TS_GetXY() — зчитування координат дотику;
- ft6x06_TS_EnableIT() / ft6x06_TS_DisableIT() — керування перериваннями (опційно, якщо сенсор налаштовано на генерацію переривань);

- ft6x06_TS_ResetTouchData() — очищення буфера торкань.

Ці функції можуть бути викликані у відповідному FreeRTOS-завданні, що періодично опитує сенсор або реагує на події (якщо використано апаратне переривання). При обробці координат формується подія — наприклад, натискання кнопки інтерфейсу або жест перемикання. Це дозволяє реалізувати інтуїтивно зрозумілу взаємодію з системою без потреби в додаткових пристроях введення.

2.10 Графічне виведення даних на SPI-дисплеях

У межах реалізованого проєкту для графічного виведення інформації використано кольоровий TFT-дисплей з контролером ILI9341, який характеризується високою швидкістю оновлення, широкими можливостями налаштування та сумісністю з інтерфейсом SPI.

До основних переваг ILI9341 належать:

- підтримка апаратного скролінгу;
- апаратне інверсування кольорів;
- керування яскравістю та режимами сну;
- можливість адресного малювання окремих пікселів або областей;
- збереження останнього вікна адресації для прискорення масових оновлень.

Для коректної роботи дисплея надзвичайно важливо дотримуватись послідовності ініціалізації, яка включає:

- програмний або апаратний скидання (RESET);
- надсилання команд налаштування режиму кольору, орієнтації, частоти оновлення;
- встановлення вікна адресації перед масовим виведенням зображення.

Основні функціональні блоки драйвера:

- ініціалізація дисплея (ILI9341_Init()): надсилає набір команд для налаштування дисплея у відповідності до специфікації (режим кольору, орієнтація, швидкість оновлення);
- передача команд і даних (ILI9341_WriteCommand(), ILI9341_WriteData()): відокремлює логіку команд та графічної інформації за допомогою керування піном DC;
- виведення окремих пікселів (ILI9341_DrawPixel()): дає змогу створювати кастомні графічні елементи на рівні точкової адресації;
- поворот дисплея (ILI9341_SetRotation()): задає одну з чотирьох можливих орієнтацій — портретну або ландшафтну.

Завдяки гнучкості SPI та оптимізованому драйверу, графічне оновлення екрана відбувається швидко, що є критично важливим для систем з обмеженими обчислювальними ресурсами. Зокрема, було досягнуто зменшення часу повного оновлення екрану до декількох мілісекунд.

Таким чином, дисплей виконує не лише інформаційну функцію, а й формує основу для інтуїтивної взаємодії з користувачем без потреби додаткових пристроїв.

2.11 Графічні бібліотеки для мікроконтролерних дисплеїв

Реалізація графічного інтерфейсу користувача (GUI) у вбудованих системах базується не лише на фізичному дисплеї, а й на використанні спеціалізованих графічних бібліотек, які дозволяють ефективно управляти виводом зображень, тексту та інтерактивних елементів. Такі бібліотеки виступають посередниками між прикладною логікою та низькорівневими функціями виведення, спрощуючи розробку й пришвидшуючи інтеграцію складних інтерфейсів.

У межах реалізованого програмно-апаратного комплексу було застосовано графічну бібліотеку LVGL (Light and Versatile Graphics Library).

LVGL є однією з найпопулярніших бібліотек для мікроконтролерних дисплеїв, що поєднує у собі гнучкість, продуктивність та багатий набір готових елементів.

Основні можливості бібліотеки LVGL включають:

- віджетна система: підтримка готових GUI-елементів (кнопки, слайдери, графіки, текстові поля, списки, чекбокси тощо);
- тематичне оформлення (themes): гнучка система стилів дозволяє оформлювати інтерфейс у заданій кольоровій гамі та стилі;
- масштабовані шрифти: підтримка векторних та бітмапових шрифтів, включаючи кирилицю та спеціальні символи;
- розширена система подій: обробка натискань, переміщення, тримання, фокусування, перетягування;
- анімації: плавне відображення змін, появи об'єктів, переміщення та змін кольору;
- віртуальні екрани та переходи між ними.

LVGL забезпечує компроміс між функціональністю та легкістю налаштування. Її модульна структура дозволяє використовувати лише ті компоненти, які справді потрібні. Це дає змогу зменшити footprint у пам’яті, уникнути перевантаження CPU та зберегти стабільну роботу вбудованої системи.

2.12 Аудіовивід у мікроконтролерах

Звуковий вивід у мікроконтролерних системах є простим і водночас ефективним способом забезпечення зворотного зв’язку між пристроєм і користувачем. Він дозволяє оперативно сповіщати про виникнення подій, підтвердження дій або зміну режиму роботи. Найбільш поширеним засобом генерації звукових сигналів у вбудованих системах є п’єзоелектричні бузери, які можуть бути активного або пасивного типу.

П’єзоелектричний бузер — це пристрій, що створює звукові коливання внаслідок деформації п’єзокерамічного елемента при прикладенні електричної напруги. В основі його роботи лежить п’єзоелектричний ефект: здатність певних матеріалів (зазвичай цирконату-титанату свинцю) змінювати свої геометричні розміри під дією електричного поля. Ці коливання передаються на резонатор, що створює чутний звук.

Залежно від типу внутрішньої конструкції розрізняють два основних види п’єзобузерів:

- пасивні бузери (passive buzzers) потребують зовнішнього керування частотою (зазвичай шляхом генерації PWM-сигналу), оскільки не мають вбудованого генератора;
- активні бузери (active buzzers) містять вбудований генератор частоти, що дозволяє їм видавати звук при простій подачі логічної одиниці на вхід.

У більшості мікроконтролерів, зокрема серії STM32, реалізація керування активним бузером є надзвичайно простою. Вона зводиться до керування GPIO-виводом у режимі Output Push-Pull. Подання логічної “1” на пін активує вбудований генератор у бузері, що починає видавати звуковий сигнал фіксованої частоти (зазвичай 2–4 кГц).

У випадку пасивного бузера, необхідно використовувати таймери для генерації PWM-сигналу заданої частоти. Це вимагає більш складної ініціалізації, однак відкриває ширші можливості — наприклад, генерацію мелодій або динамічне регулювання частоти звуку.

У межах дипломного проєкту використано активний бузер, що підключений до мікроконтролера через звичайний цифровий пін (GPIO). Такий підхід забезпечує мінімальне споживання апаратних ресурсів і спрощує програмну логіку керування.

Переваги використання активного бузера:

- використання активного бузера має низку переваг для мікроконтролерних систем:
- простота підключення: достатньо одного GPIO-виводу без потреби в генерації PWM;
- мінімальне споживання ресурсів: не задіюються таймери або DMA;
- швидке реагування: підходить для негайного сповіщення користувача;
- надійність: відсутність складних схем керування знижує імовірність збоїв.

2.13 Висновки до розділу

Технологічна основа програмно-апаратного комплексу була сформована на основі ретельно підібраного набору апаратних і програмних рішень, орієнтованих на стабільну роботу в умовах мікроконтролера STM32. Вибрані компоненти — сенсори руху, температури й вологості, сенсорний дисплей, графічна та звукова підсистема — забезпечують повний цикл зчитування, обробки, виводу та зворотного зв’язку з користувачем.

Використання інтерфейсів I2C та SPI дозволило організувати надійний обмін даними з периферією, тоді як реалізація графічного інтерфейсу на базі контролера ILI9341 та бібліотеки LVGL продемонструвала високу продуктивність і гнучкість при мінімальних витратах пам’яті. Додаткову функціональність забезпечила аудіосистема з активним бузером, яка слугує ефективним засобом оповіщення в режимі реального часу.

Усі ці елементи інтегруються у цілісну систему завдяки використанню FreeRTOS — операційного середовища реального часу, яке забезпечує паралельну обробку подій, точне планування задач і стійкість до затримок. Завдяки цьому комплекс набуває ознак завершеного, логічно побудованого продукту з високою адаптивністю та зручним користувацьким інтерфейсом.

Таким чином, обрані технології не лише задовольняють вимоги до функціональності, енергоефективності й надійності, а й створюють міцну основу для подальшого масштабування або модифікації системи відповідно до нових задач і сценаріїв застосування.

3. ПРОГРАМНО-АПАРАТНИЙ КОМПЛЕКС ДЕТЕКЦІЇ РУХУ НА БАЗІ МІКРОКОНТРОЛЕРА STM32

3.1 Апаратна частина

Апаратне забезпечення комплексу побудоване на основі плати STM32L475E-IOT01A (рис. 4), яка містить мікроконтролер STM32L475VG з архітектурою ARM Cortex-M4, частотою до 80 МГц та великою кількістю внутрішньої периферії.

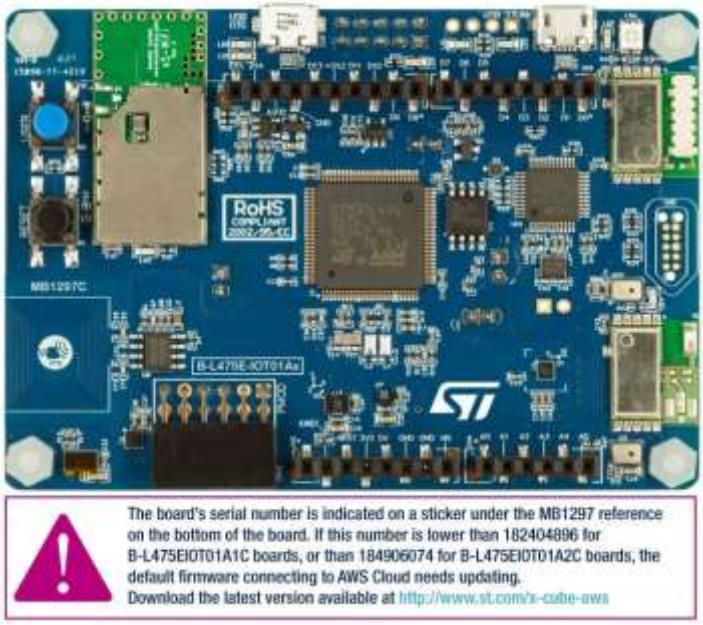


Рис. 4 – Плата STM32L475E-IOT01A

До основних підключених компонентів та інтерфейсів плати належать:

- сенсор температури та вологості HTS221 (рис. 5), підключений через інтерфейс I2C1 (лінії PB8 і PB9), який забезпечує точне вимірювання температури в діапазоні від -40 до +120°C та вологості у межах 0–100%;
- LED-індикатор, підключений до одного з виводів GPIO;

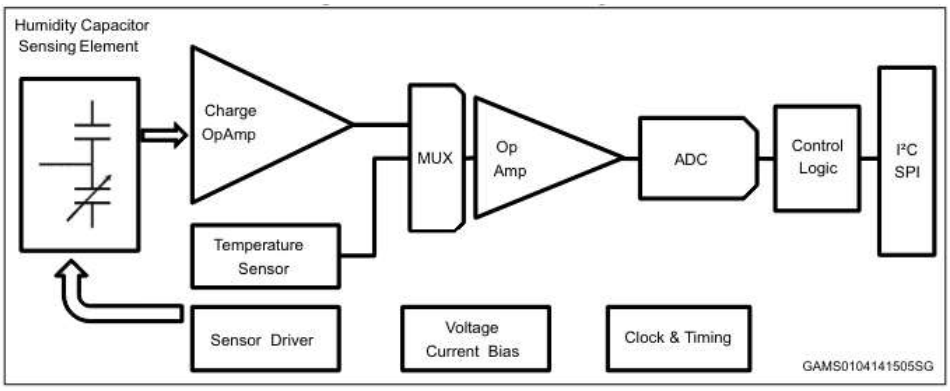


Рис. 5 – Блок-схема HTS221

- інтерфейси вводу/виводу
  - SPI1 — передача даних на дисплей ILI9341.
  - I2C1 — обмін даними з HTS221 та FT6206.
  - UART1 — для передачі логів системи
  - GPIO — загальне керування сигналами вводу/виводу
  - USB OTG — використовується для підключення до комп'ютера, живлення або налаштування плати в процесі розробки.
  - PMOD-конектор — задіяний для підключення зовнішніх модулів (PIR та бузер), через доступні GPIO-лінії.

Комплекс живиться від USB-порту або зовнішнього джерела живлення. Перетворення та стабілізація напруги до 3.3 В відбувається безпосередньо на платі. Проведені вимірювання показали, що у режимі очікування система споживає менше 15 мА, а в активному режимі з увімкненим дисплеєм — до 60 мА, що відповідає енергоефективним вимогам до вбудованих пристроїв.

Для реалізації функціональності програмно-апаратного комплексу було використано кілька зовнішніх модулів:

- інфрачервоний сенсор руху (PIR) (рис. 6). PIR-сенсор використовується для фіксації наявності руху в зоні спостереження. Замість періодичного опитування, в системі реалізовано апаратне переривання по зміні рівня сигналу на відповідному GPIO-піні. Це

дозволяє зменшити навантаження на центральний процесор і забезпечити оперативну реакцію на події;

Рис. 6 – Інфрачервоний датчик-руху

- сенсорний дисплей ILI9341 з контролером FT6206, який підключається до основної плати через виводи Arduino UNO (рис. 7-8). Виведення графічної інформації та взаємодія з користувачем реалізуються через кольоровий TFT-дисплей з роздільною здатністю 320×240 пікселів. Для передавання зображення використовується шина SPI1, налаштована на максимальну допустиму частоту передачі. Обробку дотиків здійснює контролер FT6206, підключений через I2C1, що дозволяє реалізувати інтерактивний сенсорний інтерфейс. Екран використовується як основний засіб індикації стану системи та інтерфейсу користувача;

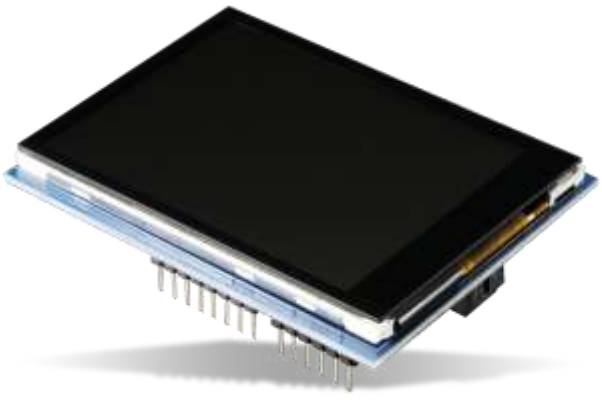


Рис. 7 – Сенсорний дисплей (вигляд зверху)

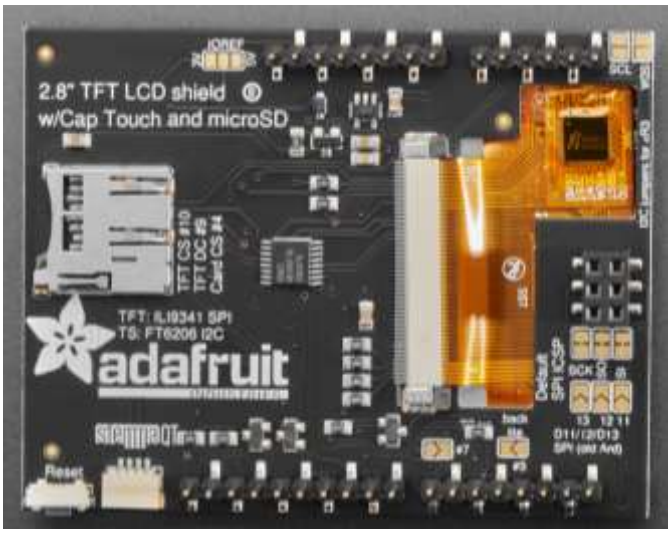


Рис. 8 – Сенсорний дисплей (вигляд знизу)

- Активний бузер, який використано для реалізації звукового сповіщення (рис. 9). Оскільки пристрій активного типу, для його керування достатньо встановлення логічного рівня на відповідному GPIO-виводі. Це дозволяє легко реалізувати звукову сигналізацію у разі спрацювання тривоги.

Рис. 9 – Активний бузер

3.2 Організація задач

Після ініціалізації мікроконтролера та запуску FreeRTOS, система переходить до запуску п’яти основних задач, кожна з яких виконує окрему функціональну роль:

- motion_task — відповідальна за моніторинг PIR-сенсора;

- sensor_task — періодично опитує сенсор температури та вологості:
- screen_task — реалізує графічний інтерфейс користувача;
- mode_manager_task — реалізує перемикання між різними режимами системи;
- event_handler – відповідальна за сигналізацію тривоги;
- logger_task — відповідає за виведення діагностичних повідомлень.

3.3. Налаштування периферії

У даному програмно-апаратному комплексі на базі мікроконтролера STM32L475 особливу увагу приділено налаштуванню периферійних інтерфейсів. Налаштування периферії здійснюється через STM32CubeMX відповідно до специфіки підключених пристроїв і режимів їх використання у FreeRTOS-задачах.

3.3.1 Інтерфейс I2C1

На рис. 10 зображено процес налаштування інтерфейсу I2C1, який використовується для зчитування даних з сенсорів, зокрема FT6206 (сенсор дотику) та HTS221 (сенсор температури і вологості). Конфігурація I2C включає налаштування ключових параметрів, необхідних для стабільної роботи вбудованих пристроїв. Було обрано стандартний режим обміну даними на частоті 100 кГц, що відповідає вимогам підключених сенсорів.

Додатково були активовані фільтри аналогових перешкод, які допомагають уникати помилок передачі в умовах електричних завад. Встановлено фіксовані значення часу підйому та спаду сигналу — по 100 нс, що є рекомендованими для стабільної роботи шини. Адресація пристроїв здійснюється у 7-бітному форматі, що є найбільш поширеним у пристроях типу slave. Загальна конфігурація забезпечує необхідну пропускну здатність для отримання виміряних значень у реальному часі без затримок.

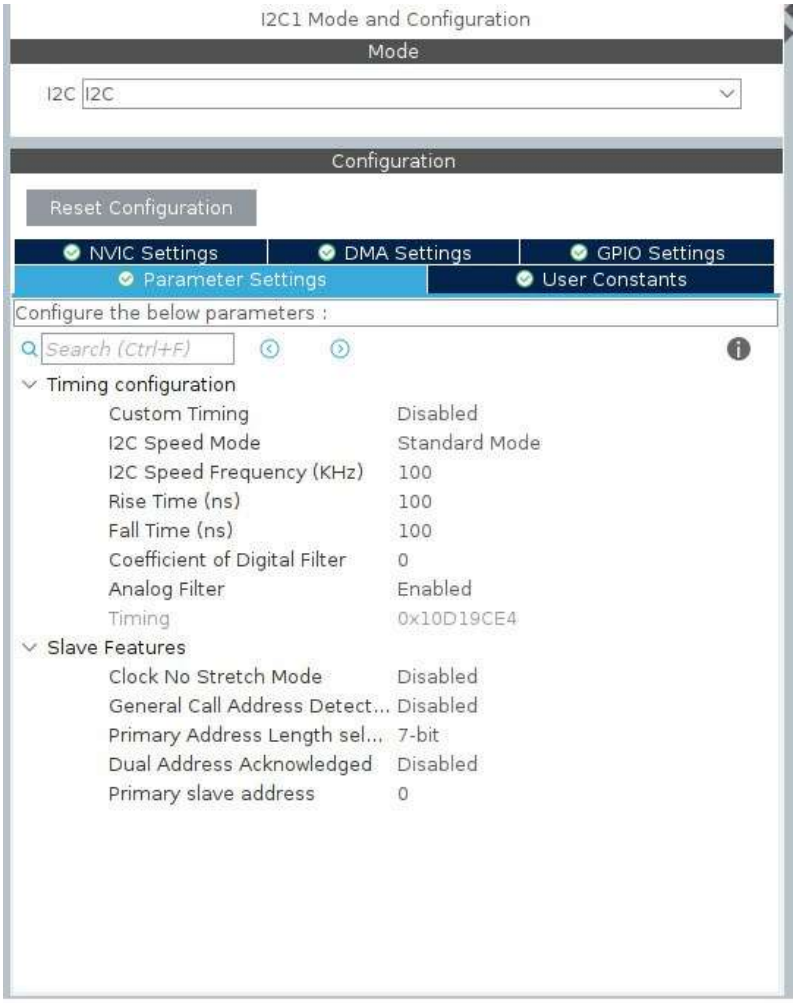


Рис. 10 – Конфігурація I2C1 у STM32CubeMX

3.3.2 Інтерфейс SPI1

Обмін з дисплеєм ILI9341 здійснюється через інтерфейс SPI1, який працює в режимі Full-Duplex Master. Це дає змогу реалізувати високошвидкісну передачу графічних даних, необхідних для швидкої візуалізації змін на екрані (наприклад, оновлення зображення або індикаторів).

На рис. 11 показано конфігурацію SPI1. Було обрано найвищу можливу швидкість передачі даних — 40 Мбіт/с (Prescaler = 2), що дозволяє дисплею працювати максимально ефективно. NSS-сигнал керується програмно, що дає змогу вручну керувати піном CS.

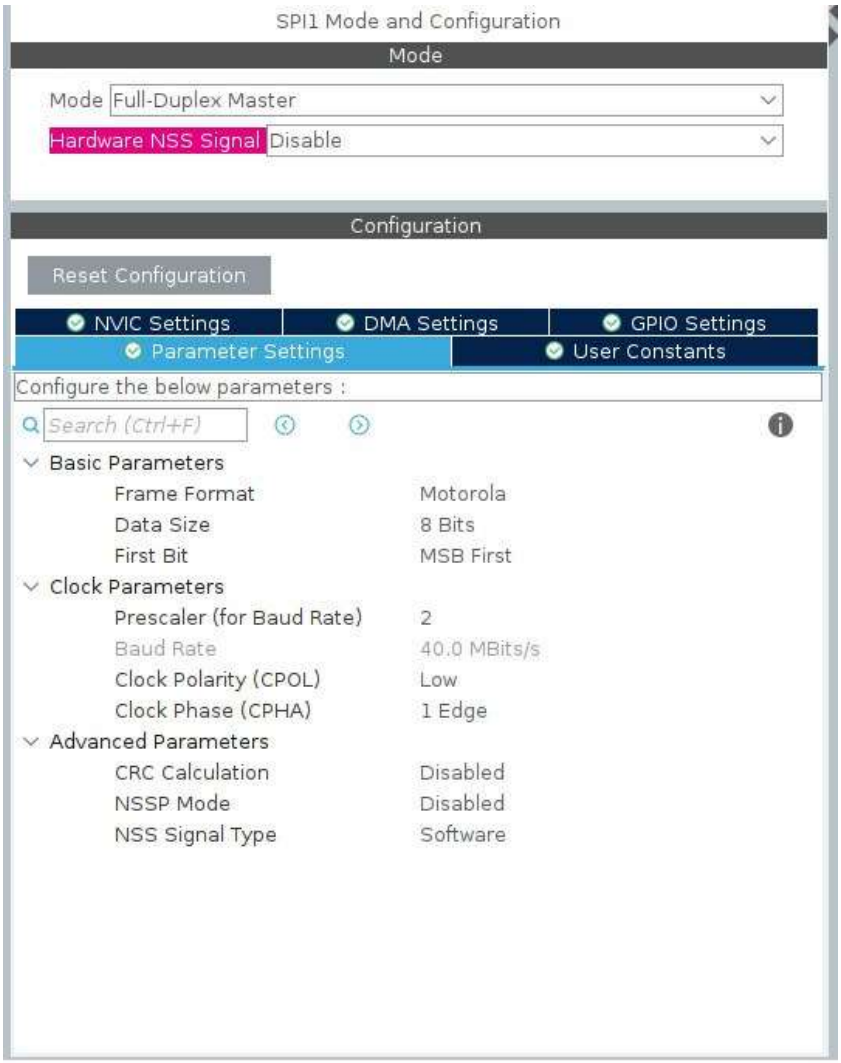


Рис. 11 – Конфігурація SPI1 для дисплея ILI9341

3.3.3 Інтерфейс USART1 з DMA

Для логування даних у режимі реального часу та обміну з комп'ютером або іншим пристроєм було налаштовано USART1 у режимі асинхронної передачі. Особливістю реалізації є використання DMA для передачі (TX) і приймання (RX) даних. Це дозволяє зняти навантаження з основного ядра, оскільки передача відбувається безперервно й неблокуюче.

На рис. 12 представлено налаштування DMA для USART1: передача виконується через DMA1 Channel 4 (Memory To Peripheral), а приймання — через DMA1 Channel 5 (Peripheral To Memory). Встановлені відповідні

пріоритети: високий для TX, середній для RX, що дозволяє швидко надсилати важливу діагностичну інформацію на ПК.

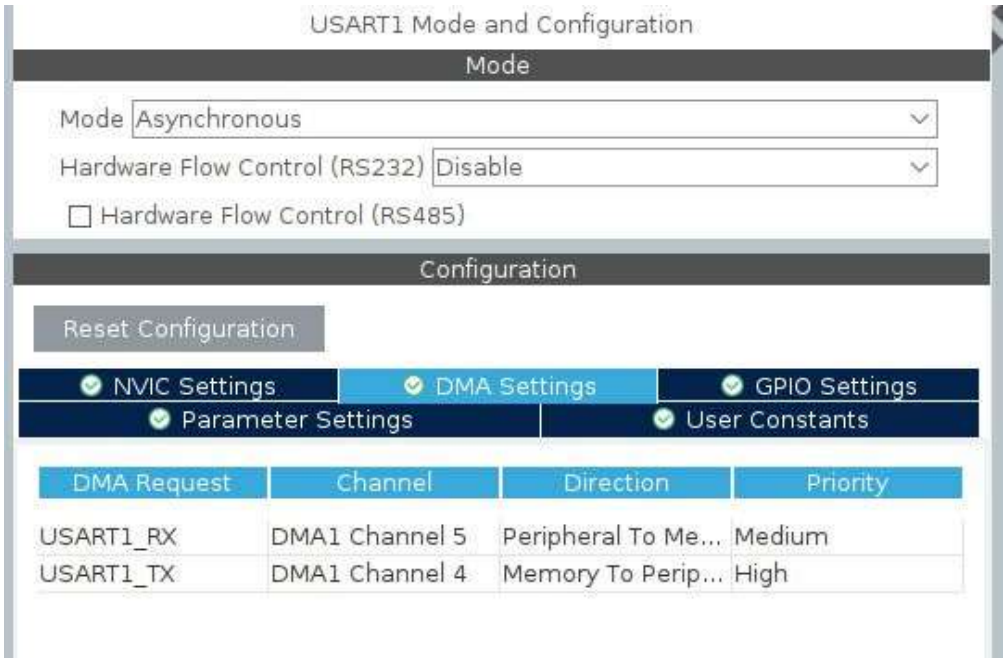


Рис. 12 – Налаштування USART1 з використанням DMA

3.4. Розробка програмного забезпечення

Програмна частина реалізованого комплексу побудована з використанням операційної системи реального часу FreeRTOS, що забезпечує ефективне керування паралельними задачами, синхронізацію доступу до апаратних ресурсів та підтримку системних механізмів, таких як семафори, черги, таймери та м’ютекси.

3.4.1 Циклічний буфер

Однією з ключових підсистем програмного забезпечення є система збереження і передавання даних у вигляді циклічного буфера. Її головна мета — забезпечення тимчасового буферизування повідомлень, отриманих від різних задач, із подальшою безпечного передачею у зовнішні модулі (наприклад, через UART або дисплей).

Циклічний буфер (кільцева черга) був реалізований у вигляді окремої структури, що зберігає покажчики на початок і кінець даних, лічильник

заповнення, а також об'єкт синхронізації (м’ютекс), необхідний для забезпечення доступу до буфера з кількох паралельних задач (рис. 13).

```
#ifndef CYCLIC_QUEUE_H
#define CYCLIC_QUEUE_H

#include "rtos_common.h"

typedef struct
{
    uint8_t *data;
    size_t capacity;
    uint8_t *write_ptr;
    uint8_t *read_ptr;
    uint8_t full_flag;
    osMutexId_t *lock;
} cyclic_queue_t;

#define DEFINE_CYCLIC_QUEUE(name, length)      \
    CONFIGURE_MUTEX(name##_lock);              \
    uint8_t name##_storage[length];            \
    cyclic_queue_t name = {                    \
        .data = name##_storage,                \
        .capacity = length,                    \
        .write_ptr = name##_storage,           \
        .read_ptr = name##_storage,            \
        .full_flag = 0,                        \
        .lock = &name##_lock,                  \
    };                                         \
    void init_##name(void)                     \
    {                                          \
        init_##name##_lock();                  \
    }

size_t queue_get_capacity(cyclic_queue_t *q);
size_t queue_peek_size(cyclic_queue_t *q);
int queue_push(cyclic_queue_t *q, uint8_t *src, size_t len);
int queue_pop(cyclic_queue_t *q, uint8_t *dst, size_t *len);
void queue_reset(cyclic_queue_t *q);

#endif // CYCLIC_QUEUE_H
```

Рис. 13 – Компонети циклічного буфера

Серед полів — покажчики на область пам’яті, поточну позицію запису та читання, а також прапорець, що сигналізує про повне заповнення буфера. Такий підхід дозволяє зберігати дані у вигляді послідовних байтів і зчитувати їх у порядку FIFO, не потребуючи динамічного виділення пам’яті чи складної логіки керування.

Ініціалізація черги здійснюється за допомогою макросу, який створює статичний масив певної довжини для зберігання даних та налаштовує початкові вказівники. Також створюється м’ютекс, який буде використовуватись для блокування доступу до буфера при паралельному використанні кількох задач.

3.4.2 Принцип роботи буфера

Запис інформації в буфер відбувається послідовно: кожне повідомлення спочатку доповнюється службовим байтом, що вказує його довжину, після чого записуються дані. У разі досягнення кінця масиву вказівники повертаються до початку, формуючи кільцеву структуру. Якщо обсяг вхідних даних перевищує наявний простір, повідомлення не записується, що дозволяє уникнути пошкодження існуючих даних.

Зчитування інформації також здійснюється поетапно. Спочатку аналізується розмір повідомлення, а потім зчитуються відповідні байти з урахуванням циклічного переходу на початок буфера. При цьому гарантується збереження порядку черги та відсутність блокувань інших задач на тривалий час.

Буфер також передбачає метод очищення, що дозволяє швидко скинути його стан, обнулити вказівники та прапорець переповнення. Це особливо корисно при скиданні системи або зміні режиму роботи.

3.4.3 Система логування та діагностики

Для зручного відлагодження програмного коду та моніторингу роботи системи в режимі реального часу у програмному забезпеченні реалізована вбудована система логування. Її основне призначення — асинхронний вивід діагностичних повідомлень, що дозволяє фіксувати інформацію про стан сенсорів, виявлені події, активацію виконавчих елементів тощо.

У якості фізичного інтерфейсу для виводу логів використано USART1, який працює у поєднанні з DMA. Це забезпечує мінімальне навантаження на процесор під час передавання великих обсягів повідомлень. Щоб уникнути втрат даних та не блокувати основні задачі, було реалізовано власний API для логування, який включає циклічний буфер та окрему задачу FreeRTOS для передавання повідомлень у фоновому режимі.

```c
#ifndef DEBUG_OUTPUT_H
#define DEBUG_OUTPUT_H

#define DEBUG_BUFFER_CAPACITY 1024
#define DEBUG_MESSAGE_LIMIT 128

void DEBUG_PRINT(const char *fmt, ...);

void debug_output_init(void);
void debug_on_tx_complete(void);
void launch_debug_task(void);

#endif // DEBUG_OUTPUT_H
```

Рис. 14 – Структура модуля логування

На рис. 14 представлено модуль оголошення основних функцій API. Зокрема, функція DEBUG_PRINT() приймає форматований рядок аналогічно до стандартного printf() і додає його у чергу повідомлень. Буферізація здійснюється у виділеній пам’яті фіксованого розміру, а передача у UART здійснюється окремою задачею.

```
[SENSOR][INFO] Temperature = 31.25
[SENSOR][INFO] Humidity = 43.59
[SENSOR][INFO] Temperature = 31.23
[SENSOR][INFO] Humidity = 43.58
[SENSOR][INFO] Temperature = 31.23
[SENSOR][INFO] Humidity = 43.53
[PIR][INFO] Motion detected!
[BUZZER][INFO] Alarm!
[SENSOR][INFO] Temperature = 31.29
[SENSOR][INFO] Humidity = 43.51
[SENSOR][INFO] Temperature = 31.25
[SENSOR][INFO] Humidity = 43.54
```

Рис. 15 – Приклад UART-логування даних сенсорів та подій у системі

Завдяки такому підходу забезпечується можливість постійного виводу логів, навіть у випадках високого навантаження або паралельного виконання задач. Це дозволяє зручно аналізувати хід роботи програми без переривання виконання основного коду.

Кожне повідомлення має маркування, що включає тип джерела (наприклад, [SENSOR], [PIR], [BUZZER]) і рівень важливості ([INFO]), що дозволяє швидко ідентифікувати джерело події (рис. 15).

3.4.4 Семплування даних

Для забезпечення надійного та інформативного збору інформації з цифрового сенсора температури та вологості (TempHum) у проєкті реалізовано алгоритм періодичного семплування даних. Семплування виконується з фіксованим інтервалом у 10 мс, що дозволяє отримувати дані з високою частотою та оперативно реагувати на зміни у навколишньому середовищі.

Ключовим елементом алгоритму є накопичення даних у циклічному буфері та подальше обчислення середнього значення після кожних 100 зчитувань. Такий підхід суттєво підвищує точність отриманої інформації, оскільки згладжує випадкові флуктуації, шум або короткочасні збої у вимірюваннях. У результаті система отримує більш стабільне та репрезентативне значення, яке вже передається на обробку або відображення.

3.4.5 Пожежні рівні небезпеки

Встановлено два рівня реагування:

- Попередження про небезпеку — температура > 60°C і вологість < 20%;
- Пожежна тривога — температура > 80°C і вологість < 10%.

Завдяки інтеграції з іншими підсистемами, зокрема логуванням, дисплеєм і звуковим сигналом, алгоритм детекції забезпечує оперативне попередження користувача про аномальні умови, покращуючи безпечність і надійність усього комплексу.

3.4.6 Робота з PIR

Виявлення руху реалізовано асинхронно, за допомогою апаратного переривання, що генерується при фіксації активності PIR-сенсором. Це дозволяє не витрачати ресурси мікроконтролера на постійне опитування стану входу, що є більш енергоефективним і продуктивним рішенням. При виникненні переривання надсилається повідомлення до відповідної задачі FreeRTOS, яка миттєво реагує на подію.

У результаті система здатна миттєво реагувати на присутність людини або інші зміни у середовищі, що робить її придатною для охоронних або моніторингових застосувань.

3.4.7 Режими охорони

Програмно-апаратний комплекс підтримує декілька режимів охорони, які відображають різні сценарії використання системи залежно від потреб користувача. Кожен режим визначає логіку реагування на події від сенсорів (руху, температури, дотику) та активацію виконавчих елементів (бузер, дисплей, пір). Перемикання між режимами виконується через централізований модуль керування — менеджер режимів (Mode manager).

No Arm (не під охороною). Цей режим використовується, коли система перебуває у пасивному стані, наприклад, у присутності користувача. У цьому стані система продовжує моніторинг температури й пожежної активності (детекція вогню), проте інші події (наприклад, рух або дотик) не викликають тривоги. Це дозволяє уникнути хибних спрацювань під час звичайної активності в приміщенні.

Arm (під охороною). У повністю охоронному стані активуються всі модулі контролю: система реагує як на детекцію руху, так і на загрозу пожежі. У разі виявлення будь-якої з цих подій активується звукова тривога, що сигналізує про небезпеку. Цей режим призначений для періодів, коли приміщення має залишатись повністю контрольованим — наприклад, уночі або під час відсутності користувача.

Silent (тихий режим). Цей режим призначений для ситуацій, коли потрібна невидима детекція. Система продовжує аналіз даних з усіх сенсорів (включно з рухом), але звуковий сигнал вимикається, і вся реакція зводиться до сповіщень на дисплеї. Це зручно, коли потрібно уникнути гучних звукових сповіщень.

Kids (дитячий режим). У цьому режимі система функціонує у найбільш м'якому варіанті. Активною залишається лише пожежна безпека (виявлення вогню), а всі інші функції охорони — зокрема реагування на рух і дотик — вимкнені або заблоковані. Додатково передбачено блокування сенсорного інтерфейсу, що унеможливлює зміну налаштувань або конфігурації системи сторонніми особами, зокрема дітьми.

Завдяки гнучкій системі режимів охорони користувач може адаптувати поведінку пристрою до різних сценаріїв реального використання, забезпечуючи баланс між безпекою, зручністю та конфіденційністю.

3.4.8 Сенсорний дисплей Adafruit 1947

Для реалізації інтуїтивного графічного інтерфейсу у програмно-апаратному комплексі використовується сенсорний дисплей з контролером ILI9341 та сенсорною панеллю FT6206. Обидва компоненти підключені до STM32L475 через SPI1 та I2C1 відповідно. Графічна частина дисплея підтримує роздільну здатність 320x240 пікселів та відображає як текстову, так і векторну графіку.

Усі елементи інтерфейсу — кнопки, рамки, індикатори — побудовані за допомогою графічної бібліотеки LVGL (Light and Versatile Graphics Library), яка спеціально адаптована для роботи в системах з обмеженими ресурсами. Вона забезпечує малювання графіки, анімацію елементів і обробку дотиків на логічному рівні.

Керування дисплеєм виконується у межах окремої FreeRTOS-задачі screen_task, яка:

- ініціалізує дисплей і сенсор дотику;

- малює основні елементи інтерфейсу;
- реагує на дотики користувача, визначаючи активну зону (наприклад, перемикання режиму, підтвердження тривоги).

Завдяки використанню LVGL інтерфейс є гнучким, адаптивним і легко масштабованим — як для виводу системної інформації, так і для інтерактивної взаємодії з користувачем.

3.4.9 Інтерфейс користувача

Головний екран: на головному екрані відображається поточна температура, статус охоронної системи та кнопки швидкого доступу (рис. 16):

- до журналу подій (Event Log),
- меню режимів охорони (Security Mode),
- панік-кнопки,
- та екрану входу в кодовий режим (розблокування/перевірка доступу).

Це забезпечує миттєвий огляд стану системи та швидку реакцію користувача.

Рис. 16 – Головний екран

Екран режимів охорони: інтерфейс режимів дозволяє користувачеві перемикати стан системи між основними сценаріями: No Arm, Silent, Kids

тощо. Кожна кнопка реалізована як окремий віджет з графічним позначенням і супровідною іконкою, що спрощує орієнтацію (рис. 17).

Рис. 17 – Екран режимів охорони

Сигнали тривоги: у разі виявлення небезпеки (руху або перегріву), інтерфейс автоматично переходить до екрану тривоги, який відображає характер події:

- «Motion Detected» для руху (рис. 18),
- «Heat Detected» для температурної небезпеки (рис. 19).

Рис. 18 – Вікно тривоги по руху

Рис. 19 – Вікно пожежної небезпеки

Журнал подій (Event Log): на окремому екрані відображається хронологічний список останніх подій, включно з повідомленнями про виявлення руху, активацію пожежної тривоги, постановка чи зняття системи з охорони (рис. 20).

Рис. 20 – Вікно журналу подій

Екран введення коду: для розблокування певних функцій або зняття з охорони передбачений екран введення PIN-коду, що містить цифрову клавіатуру з візуальним підтвердженням введених символів (рис. 21). Це підвищує захист системи від несанкціонованого доступу.

Рис. 21 – Екран блокування

3.4. Висновки до розділу

У межах третього розділу було реалізовано програмно-апаратний комплекс на базі STM32L475, що об’єднує в собі функції виявлення руху, моніторингу параметрів середовища, відображення стану системи на графічному дисплеї та генерації тривожних сповіщень.

Особливу увагу приділено правильному структуруванню задач у середовищі FreeRTOS, що забезпечує паралельну та синхронізовану роботу сенсорів, дисплея, логування та реагування на події. Усі ключові компоненти — від семплування температури до обробки переривань від PIR-сенсора — працюють узгоджено в режимі реального часу.

Інтерфейс, створений за допомогою бібліотеки LVGL, дозволяє користувачу зручно взаємодіяти із системою: переглядати журнал подій, перемикати режими охорони, підтверджувати тривоги чи вводити код доступу. Візуальна частина доповнює функціональність і значно спрощує експлуатацію.

ВИСНОВКИ

У ході виконання дипломного проєкту на тему «Програмно-апаратний комплекс детекції руху на базі STM32» було розроблено, реалізовано та протестовано повнофункціональну вбудовану систему, що поєднує апаратні засоби моніторингу, програмну обробку подій у реальному часі та інтерактивний графічний інтерфейс.

Метою проєкту було створення доступного, стабільного та зручного у використанні рішення, здатного виявляти рух, фіксувати параметри середовища, реагувати на критичні події відповідно до заданих сценаріїв та надавати користувачеві актуальну інформацію. У процесі реалізації цієї мети було досягнуто повної інтеграції між апаратними модулями (датчиками, дисплеєм, звуковим індикатором) та програмною логікою, побудованою на базі FreeRTOS.

Окрему увагу приділено створенню інтуїтивно зрозумілого інтерфейсу, який дозволяє користувачу взаємодіяти із системою без потреби в додаткових поясненнях. Реалізація сенсорного дисплея з виведенням ключової інформації, індикаторами стану та можливістю перемикання режимів охорони забезпечила високу зручність експлуатації.

Комплекс побудовано з використанням недорогої елементної бази, при цьому досягнуто достатньої енергоефективності, що робить систему доступною для широкого кола користувачів. Хоча проєкт ще не претендує на повноцінну заміну комерційних охоронних систем, закладена архітектура демонструє практичність, функціональність і перспективу подальшого вдосконалення.

У подальшій розробці система може бути значно розширена. Насамперед — за рахунок використання більшого набору вбудованих сенсорів, які вже наявні на базовій платі B-L475E-IOT01A (зокрема, акселерометр, магнітометр, мікрофон, барометр тощо). Це дозволить

перетворити комплекс на багатофункціональну платформу моніторингу, здатну працювати у ширшому спектрі задач — від безпеки до навколишнього аналізу середовища.

Другою перспективною лінією розвитку є створення мобільного застосунку для керування системою та отримання повідомлень у режимі реального часу. Інтеграція з Bluetooth або Wi-Fi-модулями дозволить користувачеві переглядати лог подій, отримувати сповіщення та змінювати режими дистанційно, що значно підвищить рівень зручності та автономності.

Таким чином, виконана робота не лише повністю реалізує поставлені цілі, а й закладає ґрунт для подальших досліджень та інженерних рішень у сфері вбудованих систем безпеки. Розроблений комплекс є прикладом ефективної інтеграції доступної апаратної бази, сучасного програмного підходу та гнучкої архітектури, що робить його перспективною основою для подальших прикладних застосувань.

СПИСОК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Adafruit. 2.8" TFT Touch Shield for Arduino w/Capacitive Touch. – URL: https://learn.adafruit.com/adafruit-2-8-tft-touch-shield-v2
2. Andrade, W.L., Machado, P.D.L., Alves, E.L.G., Almeida, D.R. (2009). Test Case Generation of Embedded Real-Time Systems with Interruptions for FreeRTOS. In: Oliveira, M.V.M., Woodcock, J. (eds) Formal Methods: Foundations and Applications. SBMF 2009. Lecture Notes in Computer Science, vol 5902. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-10452-7_5
3. FreeRTOS. Official Documentation. – 2024. – URL: https://www.freertos.org
4. Guan, F., Peng, L., Perneel, L., & Timmerman, M. (2016). Open source FreeRTOS as a case study in real-time operating system evolution. Journal of Systems and Software, 118, 19–35. https://doi.org/10.1016/j.jss.2016.04.063
5. LittlevGL Developers. LVGL – Light and Versatile Graphics Library Documentation. – 2024. – URL: https://docs.lvgl.io
6. MaJerle T. STM32 UART DMA RX and TX with circular buffer. – 2023. – URL: https://github.com/MaJerle/stm32-usart-uart-dma-rx-tx
7. Miranda, B., Oliveira, R., & Carminati, A. (2021). Analysis of FreeRTOS overheads on periodic tasks. In Proceedings of the 2021 Workshop on Performance (pp. 119–130). https://doi.org/10.5753/wperformance.2021.15728
8. Murata Manufacturing. Piezoelectric Sound Components – Technical Guide. – 2022. – URL: https://www.murata.com/en-us/products/sound/guide
9. Murty, R. Infrared Sensors and Systems. – Sensors & Transducers, 2015. – Vol. 186, Issue 3. – pp. 88–93.

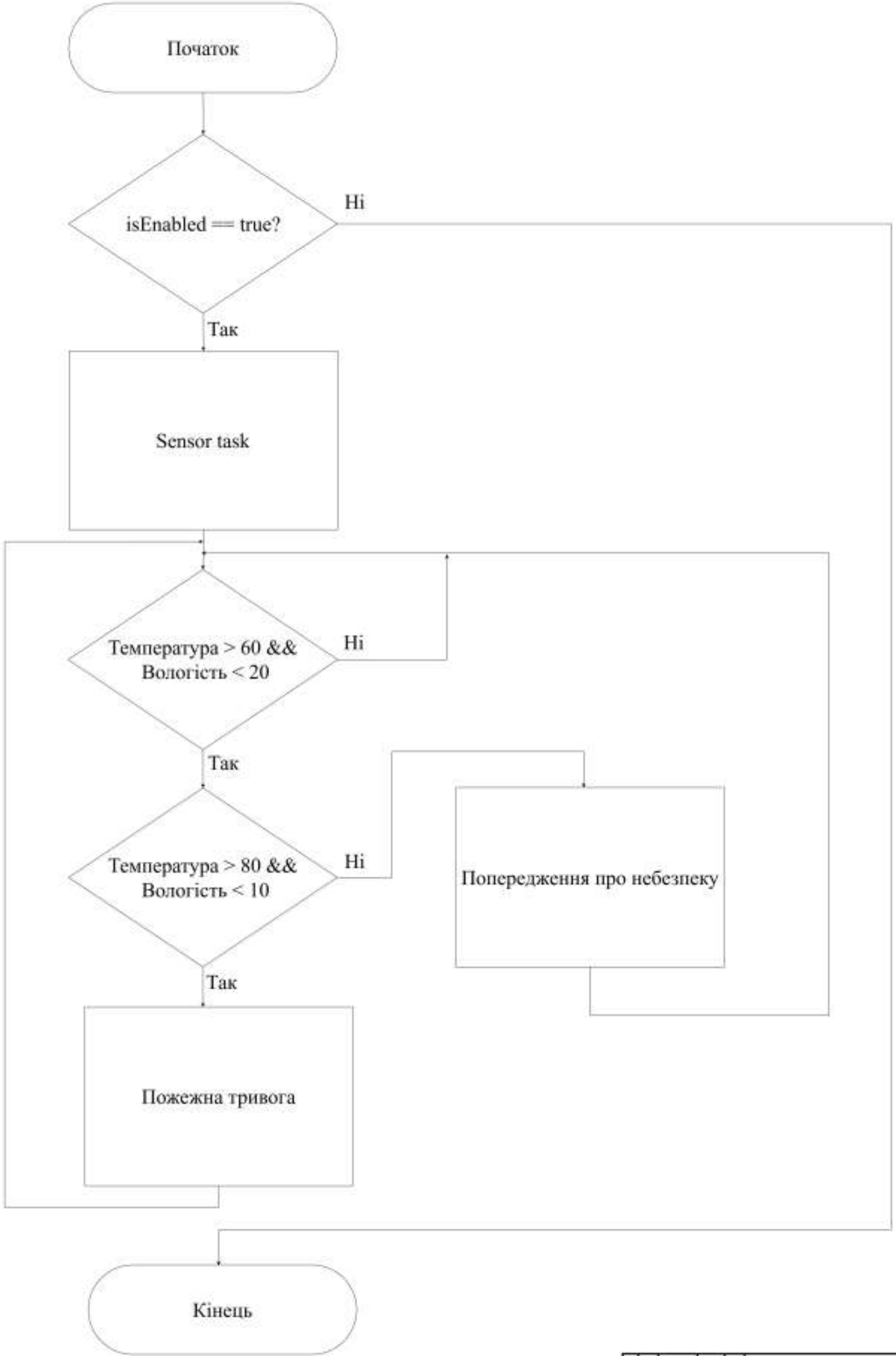
10. STMicroelectronics. AN4861: Capacitive touch panel controller FT6206 – Application Note. – 2017. – URL: https://www.st.com/resource/en/application_note/an4861-ft6x06-series-capacitive-touch-panel-controllers-stmicroelectronics.pdf
11. STMicroelectronics. HTS221: Capacitive digital sensor for relative humidity and temperature – Datasheet. – 2015. – URL: https://www.st.com/resource/en/datasheet/hts221.pdf
12. STMicroelectronics. STM32CubeMX – User Guide. – 2024. – URL: https://www.st.com/en/development-tools/stm32cubemx.html
13. STMicroelectronics. STM32L4x5 Reference Manual (RM0351). – 2023. – URL: https://www.st.com/resource/en/reference_manual/dm00151940.pdf
14. STMicroelectronics. UM2052: Getting started with the B-L475E-IOT01A Discovery kit for IoT node. – 2018. – 20 с.
15. Yiu J. The Definitive Guide to ARM Cortex-M3 and Cortex-M4 Processors. – Pearson Education, Inc., 2014. – 450 с.

ДОДАТКИ

Додаток 1.

Копії графічного матеріалу

ІАЛЦ.467200.005 Д1

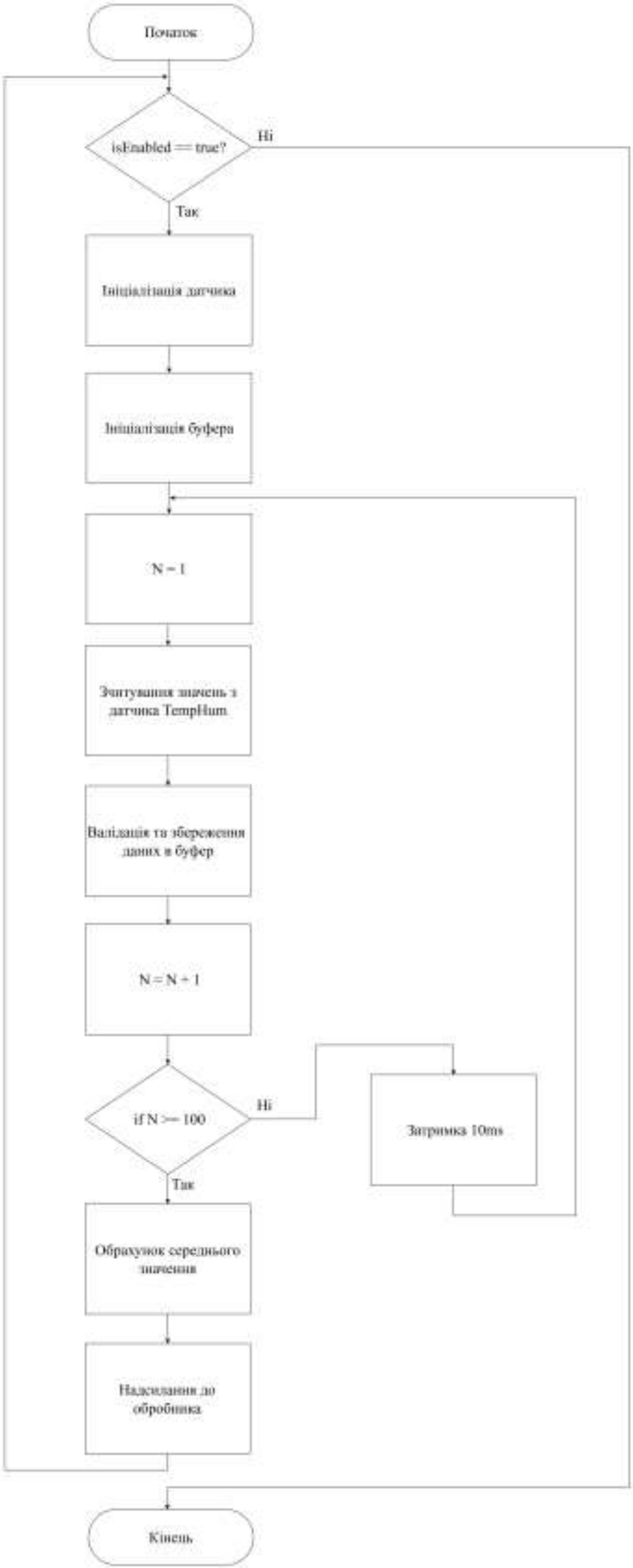


ІАЛЦ.467200.005 Д1

Блок-схема алгоритму детекції небезпеки

Схема алгоритму

ІАЛЦ.467200.006 Д2



ІАЛЦ.467200.006 Д2

Блок-схема алгоритму збору даних

Схема алгоритму

ІАЛЦ.467200.007 Д3
Main
Logging
Sensors
Motion detect
Event handler
Screen
Mode manager
ІАЛЦ.467200.007 Д3
Модель взаємодії програмного забезпечення
Схема структурна
Лит.
Маса
Масштаб
Арк. 1 Аркушів 1
КПІ ім. Ігоря Сікорського ФІОТ КВ-12
Зм. Арк. № докум. Підпис Дата
Розроб. Ступницький С.М.
Перевірив Разгонов К. О.
Н. контр.
Затв. Ковальчук Я.М. Романишин В.О.

ІА.ЛШ.467200.008 Д4

Зм.	Арк.	№ докум.	Підпис	Дата
Розробив		Сухушкін С.М.		
Перевірив		Рагімов К. О.		
Н. контр.		Квєтково Я.М.		
Затв.		Романюкич В.О.		

ІА.ЛШ.467200.008 Д4

Модель станів охорони
Схема структурна

Літ.	Маса	Масштаб
Арк. 1	Аркушів 1	

КПІ ім. Ігоря Сікорського,
ФІОМ КВ-12

Додаток 2.

Фрагменти коду

“cyclic_queue.c”

```c
#include <string.h>
#include <stdio.h>
#include <stdarg.h>
#include "cyclic_queue.h"

static void queue_insert_byte(cyclic_queue_t *q, uint8_t byte)
{
    uint8_t *next = q->write_ptr;
    *next++ = byte;
    if (next >= q->data + q->capacity)
    {
        next = q->data;
    }
    q->write_ptr = next;
}

size_t queue_get_capacity(cyclic_queue_t *q)
{
    osMutexAcquire(*q->lock, osWaitForever);

    if (q->write_ptr == q->read_ptr)
    {
        if (q->full_flag)
        {
            osMutexRelease(*q->lock);
            return q->capacity;
        }
        osMutexRelease(*q->lock);
        return 0;
    }

    size_t result;
    if (q->write_ptr > q->read_ptr)
    {
        result = (size_t)(q->write_ptr - q->read_ptr);
    }
    else
    {
        result = q->capacity - (size_t)(q->read_ptr - q->write_ptr);
    }

    osMutexRelease(*q->lock);
    return result;
}

size_t queue_peek_size(cyclic_queue_t *q)
{
    if (queue_get_capacity(q) == 0)
    {
        return 0;
    }

    size_t msg_size;
    osMutexAcquire(*q->lock, osWaitForever);
    msg_size = *q->read_ptr;
    osMutexRelease(*q->lock);
    return msg_size;
}

int queue_push(cyclic_queue_t *q, uint8_t *src, size_t len)
{
    if (q->capacity - queue_get_capacity(q) + 1 < len)
    {
        return -1; // Overflow
    }

    osMutexAcquire(*q->lock, osWaitForever);
    queue_insert_byte(q, (uint8_t)len);

    for (size_t i = 0; i < len; ++i)
```

```
    {
        queue_insert_byte(q, src[i]);
    }

    if (q->write_ptr == q->read_ptr)
    {
        q->full_flag = 1;
    }

    osMutexRelease(*q->lock);
    return 0;
}

int queue_pop(cyclic_queue_t *q, uint8_t *dst, size_t *len)
{
    if (*len == 0)
    {
        return -1;
    }
    size_t read_bytes = 0;
    size_t msg_len = 0;
    uint8_t *cursor;
    msg_len = queue_get_capacity(q);
    if (msg_len == 0)
    {
        return -1;
    }
    osMutexAcquire(*q->lock, osWaitForever);
    cursor = q->read_ptr;
    msg_len = *cursor++;
    if (cursor >= q->data + q->capacity)
    {
        cursor = q->data;
    }
    if (msg_len <= *len)
    {
        for (size_t i = 0; i < msg_len; i++)
        {
            dst[read_bytes++] = *cursor++;
            if (cursor >= q->data + q->capacity)
            {
                cursor = q->data;
            }
        }
    }
    else
    {
        for (size_t i = 0; i < *len - 1; i++)
        {
            dst[read_bytes++] = *cursor++;
            if (cursor >= q->data + q->capacity)
            {
                cursor = q->data;
            }
        }
        dst[read_bytes++] = *cursor;
        *cursor = (uint8_t)(msg_len - *len);
    }
    q->full_flag = 0;
    q->read_ptr = cursor;
    *len = read_bytes;
    osMutexRelease(*q->lock);
    return 0;
}
void queue_reset(cyclic_queue_t *q)
{
    osMutexAcquire(*q->lock, osWaitForever);
    q->write_ptr = q->data;
    q->read_ptr = q->data;
    q->full_flag = 0;
    osMutexRelease(*q->lock);
}
```

Додаток 3.

Презентація

ПРОГРАМНО-АПАРАТНИЙ КОМПЛЕКС ДЕТЕКЦІЇ РУХУ НА БАЗІ МІКРОКОНТРОЛЕРА STM32

Виконала: студентка КВ-12 Ступницька С.М.
Керівник: ст. викладач кафедри СПіСКС Радченко К.О.

МЕТА ТА АКТУАЛЬНІСТЬ РОЗРОБКИ

- **Мета проєкту** — розробка програмно-апаратного комплексу для виявлення руху з сенсорним інтерфейсом, що забезпечує реагування на загрози в реальному часі.
- **Актуальність** полягає в потребі доступних, енергоефективних та гнучких систем безпеки. В результаті даний комплекс повинен стати універсальним рішенням, яке можна буде використовувати в різних умовах та місцях.

АНАЛІЗ КОМЕРЦІЙНИХ РІШЕНЬ

Комерційні охоронні системи, доступні на ринку, забезпечують виявлення руху, сигналізацію та інформування користувача про потенційні загрози. Більшість із них використовують вбудовані PIR-сенсори, підтримують бездротовий зв'язок (Wi-Fi, GSM) і керуються через мобільні додатки або хмарні сервіси.

- Висока точність виявлення;
- Надійність і сертифікація;
- Зручність керування
- Інтеграція з іншими системами.

- Висока вартість;
- Залежність від хмарної інфраструктури;
- Обмежена можливість кастомізації.

АНАЛІЗ DIY РІШЕНЬ

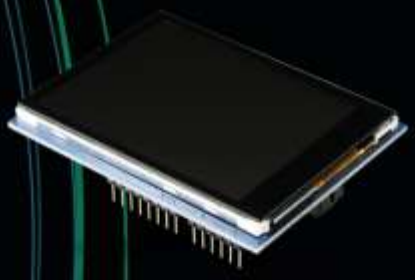
Відкриті програмні та апаратні рішення, такі як ESP32, Home Assistant, Node-RED, дають змогу самостійно створювати системи охорони, налаштовувати логіку сповіщень, збирати дані з сенсорів руху й інтегрувати їх у «розумний дім».

- Низька вартість
- Повна кастомізація
- Підтримка бездротових технологій
- Інтеграція з мобільними застосунками

- Потреба в налаштуваннях
- Обмежена функціональність

АПАРАТНА ЧАСТИНА

Плата B-L475E-IOT01A



Сенсорний дисплей
Adafruit 1947

ЗБІР ПАРАМЕТРІВ НАВКОЛИШНЬОГО СЕРЕДОВИЩА ДЛЯ ДЕТЕКЦІЇ НЕБЕЗПЕКИ

```
[SENSOR][INFO] Temperature = 31.25
[SENSOR][INFO] Humidity = 43.59
[SENSOR][INFO] Temperature = 31.23
[SENSOR][INFO] Humidity = 43.58
[SENSOR][INFO] Temperature = 31.23
[SENSOR][INFO] Humidity = 43.53
[PIR][INFO] Motion detected!
[BUZZER][INFO] Alarm!
[SENSOR][INFO] Temperature = 31.29
[SENSOR][INFO] Humidity = 43.51
[SENSOR][INFO] Temperature = 31.25
[SENSOR][INFO] Humidity = 43.54
```

НАЛАШТУВАННЯ ПЕРИФЕРІЇ

Налаштування I2C

Налаштування SPI

НАОЧНИЙ ПРИКЛАД РОБОТИ КОМПЛЕКСУ

РЕАЛІЗАЦІЯ РЕАГУВАННЯ НА РУХ

Рух виявляється за допомогою PIR-сенсора, підключеного до цифрового входу STM32. Обробка подій здійснюється через апаратне переривання — це дозволяє реагувати миттєво без постійного опитування. Після фіксації руху система активує тривогу відповідно до обраного режиму, виводить повідомлення на дисплей і зберігає подію у журналі.

РЕАЛІЗАЦІЯ РЕАГУВАННЯ НА ПОЖЕЖНУ НЕБЕЗПЕКУ

Система постійно аналізує температуру та вологість за допомогою сенсора HTS221. Зчитування відбувається з інтервалом у 10 мс, після чого значення усереднюються для зменшення шумів. Реалізовано два рівні реагування: при перевищенні 60°C та зниженні вологості нижче 20% формується попередження, а при досягненні 80°C і вологості нижче 10% — спрацьовує тривожна сигналізація.

РЕАЛІЗАЦІЯ РЕЖИМІВ ОХОРОНИ

No Arm	Arm	Silent	Kids
• лише пожежна безпека, інші сенсори неактивні.	• повна охорона: реагування на рух і перегрів.	• події фіксуються, але сигналізація вимкнена.	• активна тільки пожежна безпека, інтерфейс заблоковано.

Напрямки подальшого розвитку

- Підключення додаткових датчиків: акселерометр, мікрофон, магнітометр, барометр тощо.
- Інтеграція Bluetooth або Wi-Fi для мобільного застосунку та хмарного доступу.
- Реалізація фільтрації хибних спрацювань та розпізнавання сценаріїв.
- Оптимізація споживання енергії, використання режимів сну, живлення від акумулятора чи сонячної панелі.

ВИСНОВКИ

Розроблений комплекс детекції руху:

Забезпечує виявлення руху в реальному часі за допомогою PIR-сенсора;

Підтримує сенсор температури і вологості з періодичним семплуванням;

Реалізує інтуїтивний сенсорний інтерфейс з графічним виведенням даних;

Має вбудовану звукову та візуальну сигналізацію при спрацюванні тривоги;

Передбачає гнучке налаштування режимів охорони відповідно до потреб користувача.

ДЯКУЮ ЗА УВАГУ!