МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ПрАТ «ПРИВАТНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД «ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ»

Кафедра Інформаційних технологій

ДО ЗАХИСТУ ДОПУЩЕНА

Зав.кафедрою _____
д.е.н.,доцент Левицький С.І.
«___» _____ 2023 р.

МАГІСТЕРСЬКА ДИПЛОМНА РОБОТА

РОЗРОБКА СИСТЕМИ ПІДВИЩЕННЯ БЕЗПЕКИ ПІШОХОДА У ТЕМНИЙ ЧАС ДОБИ

Виконав
ст. гр. ІПЗ-211м _____ В.А. Денисюк
(підпис)

Керівник
к.т.н. _____ О.А. Хараджян
(підпис)

Запоріжжя
2023

ПРАТ «ПВНЗ «ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ

ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ»

Кафедра Інформаційних технологій

ЗАТВЕРДЖУЮ
Зав. кафедрою

_____
д.е.н.,доцент Левицький С.І.
«___» _____ 2023 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ ДИПЛОМНУ РОБОТУ

Студенту гр. *ІПЗ–211м*, спеціальності *«Інженерія програмного забезпечення»*

Денисюк Вікторія Андріївна

1.Тема: *Розробка системи підвищення безпеки пішохода у темний час доби.*

затверджена наказом по інституту «___»_____ 2022 р. № ____

2. Термін здачі студентом закінченої роботи: «___»_____ 2023 р.

3. Перелік питань, що підлягають розробці:

1. Аналіз основних факторів, що впливають на безпеку пішохода.
2. Вплив освітленості пішохода на його безпеку.
3. IoT, як компонент персональної системи безпеки пішохода.
4. Апаратна система підвищення безпеки пішохода.
5. Аналіз особливостей операційної системи платформи Arduino.
6. Драйвер та бібліотека інтерфейсу світлодіоднинної матриці 8x8.

7. Розробка програми пристрою підвищення безпеки пішохода.
8. Тестування пристрою підвищення безпеки пішохода.

4. Календарний графік підготовки кваліфікаційної роботи

№ етапу	Зміст	Терміни виконання	Готовність по графіку %, підпис керівника	Підпис керівника про повну готовність етапу, дата
1	Збір практичного матеріалу за темою кваліфікаційної бакалаврської роботи	04.09.23-17.10.23		
2	І атестація І розділ кваліфікаційної бакалаврської роботи	23.10.23-28.10.23		
3	ІІ атестація ІІ розділ кваліфікаційної бакалаврської роботи	20.11.23-25.11.23		
4	ІІІ атестація ІІІ розділ кваліфікаційної бакалаврської роботи, висновки та рекомендації, додатки, реферат	18.12.23-23.12.23		
5	Перевірка кваліфікаційної бакалаврської роботи на оригінальність	18.12.23-23.12.23		
6	Доопрацювання кваліфікаційної бакалаврської роботи, підготовка презентації, отримання відгуку керівника і рецензії	25.12.23-06.01.24		
7	Попередній захист кваліфікаційної бакалаврської роботи	08.01.24-13.01.24		
8	Подача кваліфікаційної бакалаврської роботи на кафедру	за 3 дні до захисту		
9	Захист кваліфікаційної бакалаврської роботи	15.01.24-20.01.24		

Дата видачі завдання: 16.01.2023 р.

Керівник кваліфікаційної бакалаврської роботи _____ (підпис) О.А. Хараджян (прізвище та ініціали)

Завдання отримав до виконання _____ (підпис) В.А. Денисюк (прізвище та ініціали)

РЕФЕРАТ

Дипломна робота містить 65 стор., 8 рис., 7 таблиць, 2 додаток, 8 використаних джерел.

Об’єкт роботи: системи забезпечення безпеки пішохода.

Предмет роботи: система підвищення безпеки пішохода в темний час доби.

Мета роботи: розробка апаратної платформи та програмного забезпечення системи підвищення безпеки пішохода в темний час доби.

Задачі роботи: аналіз основних факторів, що впливают на безпеку пішохода; вплив освітленості пішохода на його безпеку; IoT, як компонент персональної системи безпеки пішохода; апаратна система підвищення безпеки пішохода; аналіз особливостей операційної системи платформи Arduino; драйвер та бібліотека інтерфейсу світлодіоднинної матриці 8x8; розробка програми пристрою підвищення безпеки пішохода; тестування пристрою підвищення безпеки пішохода.

Безпека пішоходів нещодавно стала проблемою громадського здоров’я в усьому світі. Люди зазнають фізичної шкоди через втрату уваги до свого оточення та загрозу безпеці. Хоча безпека пішоходів є спільною відповідальністю, встановлено, що відволікання різними пристроями та зниження когнітивних навичок є основними причинами аварій.

В роботі розроблено систему, яка базується на IoT технологіях для підвищення безпеки пішоходів у темний час доби.

ARDUINO, IOT, БЕЗПЕКА ПІШОХОДА, ДРАЙВЕР МАТРИЦІ СВІТЛОДІОДІВ, ПАСИВНА БЕЗПЕКА

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Слово / словосполучення	Скорочення	Умови використання
А		
Аналого-цифровий перетворювач	АЦП	
Д		
Дорожньо-транспортна пригода	ДТП	
I		
Industrial internet of things	IIOT	
Internet of things	IoT	
Integrated development environment	IDE	
P		
Processing Development Environment	PDE	
R		
Radio Frequency IDentification	RFID	
V		
Visibility level	VL	

ВСТУП

Безпека пішоходів нещодавно стала проблемою громадського здоров’я в усьому світі. Люди зазнають фізичної шкоди через втрату уваги до свого оточення та загрозу безпеці. Хоча безпека пішоходів є спільною відповідальністю, встановлено, що відволікання різними пристроями та зниження когнітивних навичок є основними причинами аварій. Існує можливість допомогти пішоходам шляхом посилення когнітивних навичок за допомогою різнорідного пристроями IoT і датчиків. Ці технології можуть виявляти та попереджати користувачів про непередбачені події, такі своєчасні попередження про небезпеки, відволікання, екстремальні погодні катаклізми та потенційні небезпеки, що насуваються. Автоматизований персоналізований агент допомагає контролювати, діагностувати проблеми та захищати людей у міському середовищі. Різні розробники запропонували різні системи та впровадили їх у багатьох областях.

В епоху стрімкого розвитку технологій інтернету речей важливим завданням є забезпечення безпеки пішоходів у темний час доби за допомогою інтеграції цих передових технологій в системи безпеки. Сучасний прогрес у галузі комп'ютерної інженерії відкриває широкі можливості в цьому напрямку, дозволяючи розробляти та впроваджувати інноваційні рішення для підвищення безпеки пішоходів, особливо у ситуаціях обмеженої видимості.

Ця робота має на меті розробку системи, яка базується на IoT технологіях для підвищення безпеки пішоходів у темний час доби. Звернення до IoT дозволяє створити інтелектуальну систему, яка поєднує в собі сенсори, засоби зв'язку та обробку даних для ефективного виявлення та реагування на потенційні небезпеки для пішоходів.

Розробка цієї системи передбачає використання передових IoT пристроїв, здатних інтегруватися з оточуючим середовищем та забезпечувати

надійний механізм захисту для пішоходів, навіть у низькоосвітленому оточенні.

Дослідження та розробка системи підвищення безпеки пішоходів у темний час доби на основі IoT технологій є кроком до створення інтелектуальних систем, спроможних ефективно захищати громадськість в умовах обмеженої видимості.

Системи підвищення безпеки пішоходів можно розділити на три великі категорії: активне втручання, розумні транспортні засоби та інфраструктура дорожнього руху.

Зіткнення між пішоходами та транспортними засобами може статися в будь-якій точці проїжджої частини та може бути результатом поведінки як пішохода, так і поведінки водія.

Видимість пішоходів у темну частину доби відіграє суттеву роль в забезпеченні безпеки пішохода.

Основне завдання проектувальників, як пішохідних переходів, їх освітлення так і пасивних засобів у пішоходів, це максимізувати контраст між пішоходами на пішохідному переході або поблизу нього та візуальним фоном позаду цих пішоходів з точки зору водіїв.

Системи підвищення безпеки пішоходів можно розділити на три великі категорії: активне втручання, розумні транспортні засоби та інфраструктура дорожнього руху.

Об’єкт роботи: системи забезпечення безпеки пішохода.

Предмет роботи: система підвищення безпеки пішохода в темний час доби.

Мета роботи: розробка апаратної платформи та програмного забезпечення системи підвищення безпеки пішохода в темний час доби.

Задачі роботи:

- аналіз основних факторів, що впливають на безпеку пішохода;
- вплив освітленості пішохода на його безпеку;
- IoT, як компонент персональної системи безпеки пішохода;
- апаратна система підвищення безпеки пішохода;
- аналіз особливостей операційної системи платформи Arduino;
- драйвер та бібліотека інтерфейсу світлодіоднинної матриці 8x8;
- розробка програми пристрою підвищення безпеки пішохода;
- тестування пристрою підвищення безпеки пішохода.

РОЗДІЛ 1
БЕЗПЕКА ПІШОХОДА

1.1. Вплив освітленості пішохода на його безпеку

Зіткнення між пішоходами та транспортними засобами може статися в будь-якій точці проїжджої частини та може бути результатом поведінки пішохода, поведінки водія або обох. У звіті Велосипедної та пішохідної програми зазначається, що під час розслідування ДТП пішоходів і транспортних засобів, 51.6 відсотка ДТП цього класу сталися вночі з рівномірним розподілом між освітленими та неосвітленими дорогами. Однак для смертельних випадків ДТП, 58.6 відсотка сталися вночі на неосвітлених дорогах і 25.3 відсотка сталися вночі на освітлених дорогах. Основною причиною цих ДТП стала недостатня видимість пішоходів під час переходу дороги.

Розглянемо параметри освітлення, які впливають на здатність водіїв бачити пішоходів на пішохідних переходах посередині кварталу, а також дозволити агентствам оцінити потенційну ефективність дизайну освітлення.

У початковому статичному експерименті в якості показника видимості використовувався час, потрібний спостерігачеві для виявлення пішохода або сурогатної цілі, тоді як у динамічному експерименті в якості метрики використовувалася відстань, на якій пішоходи або сурогатні цілі були ідентифіковані. Змінні умов експерименту включали тип лампи (натрієва високого тиску, металогалогенна), рівень вертикальної освітленості (6, 10, 20

і 30 лк), колір одягу пішоходів (білий, чорний і джинсовий), положення пішоходів і сурогатів. , і наявність відблисків.

Видимість пішоходів. Відстань видимості пішоходів – це відстань, на якій водій може бачити пішохода достатньо добре, щоб мати можливість належним чином реагувати на присутність пішохода. Чим більша відстань видимості, тим більше часу буде у водія, щоб відреагувати на пішохода, перш ніж виникне конфлікт. Контраст — це різниця між візуальним виглядом цікавого об’єкта та візуальним фоном, на якому цей об’єкт спостерігається, а контраст є основою видимості об’єкта. По суті, завдання дизайнера освітлення пішохідного переходу полягає в тому, щоб максимізувати контраст між пішоходами на пішохідному переході або поблизу нього та візуальним фоном позаду цих пішоходів з точки зору водіїв, що наближаються.

Загалом існує два аспекти контрасту — контраст кольорів і контраст яскравості. Колірний контраст заснований на різниці в кольорі між об’єктом інтересу та його фоном. Наприклад, при денному світлі блакитний об’єкт можна легко побачити на зеленому фоні, навіть якщо дві області мають однакову яскравість. Контраст яскравості, з іншого боку, базується на різниці в виміряній яскравості об’єкта інтересу та його фону. Два об’єкти одного кольору можуть різко контрастувати, якщо один має більшу яскравість, ніж інший. Вночі яскравий контраст є основним засобом виявлення об’єкта. Тому основою проектування освітлення дорожнього полотна є забезпечення адекватного контрасту яскравості.

Кілька факторів впливають на контраст яскравості між пішоходами та їхнім візуальним фоном: фіксоване освітлення проїжджої частини, освітлення фар, одяг пішохода та характеристики візуального фону. З них лише освітленням проїжджої частини можуть керувати світлодизайнери. Фари визначають виробники транспортних засобів, одяг вибирають пішоходи, а

об’єкти та простори, які складають візуальний фон для водіїв, які наближаються до пішохідного переходу, як правило, статичні. Дизайнер освітлення повинен реагувати, але не може змінити ці основні характеристики.

Верхнє освітлення доріг, встановлене на пішохідних переходах, зазвичай забезпечує більшу відстань видимості, ніж одні фари для освітлення сцени. Ефективність верхнього освітлення у збільшенні відстані видимості — за рахунок збільшення контрастності яскравості — залежить від кількох змінних: розташування та орієнтації світильника, інтенсивності випромінюваного світла та кольору джерела світла.

Концепції видимості. Освітленість (E) — це міра кількості світла, яке падає на поверхню на одиницю площі. Одиницею освітленості є люкс. З практичних міркувань освітленість на площині, перпендикулярній до напрямку поширення світла, дорівнює силі світла (I), поділеній на квадрат відстані.

Коли цікава поверхня, наприклад профіль пішохода, скошена відносно нормалі напрямку розповсюдження, доступне світло поширюється на більшу (проектовану) площу. Освітленість на площі проектування дорівнює освітленості на площині, перпендикулярній до напрямку поширення, помноженому на косинус кута між напрямком поширення та нормаллю до досліджуваної поверхні. Вертикальна освітленість. Вертикальна освітленість (E_{Vert}) визначається як освітленість вертикальної поверхні. Вертикальна освітленість пішохода — це сила світла, випромінювана світильником у напрямку пішохода, помножена на косинус кута між напрямком поширення світла та горизонтальною лінією, паралельною поверхні дороги, поділена на відстань між світильником і пішоходом.

Для доріг відстань між світильником і поверхнею дороги зазвичай виражається як висота встановлення (h) світильника, поділена на синус кута між горизонтальною поверхнею дороги та лінією від точки вимірювання до світильника. Висота 1,5 м над поверхнею дороги була обрана як точка, в якій вимірювалася вертикальна освітленість пішохода, таким чином, відстань між світильником і точкою вимірювання для освітлення пішохідного переходу дорівнює висоті монтажу мінус 1,5 м, поділений на синус кута між горизонтальною поверхнею та лінією до світильника. На рис. 1.1 показані компоненти, які використовуються для розрахунку вертикальної освітленості пішохода.

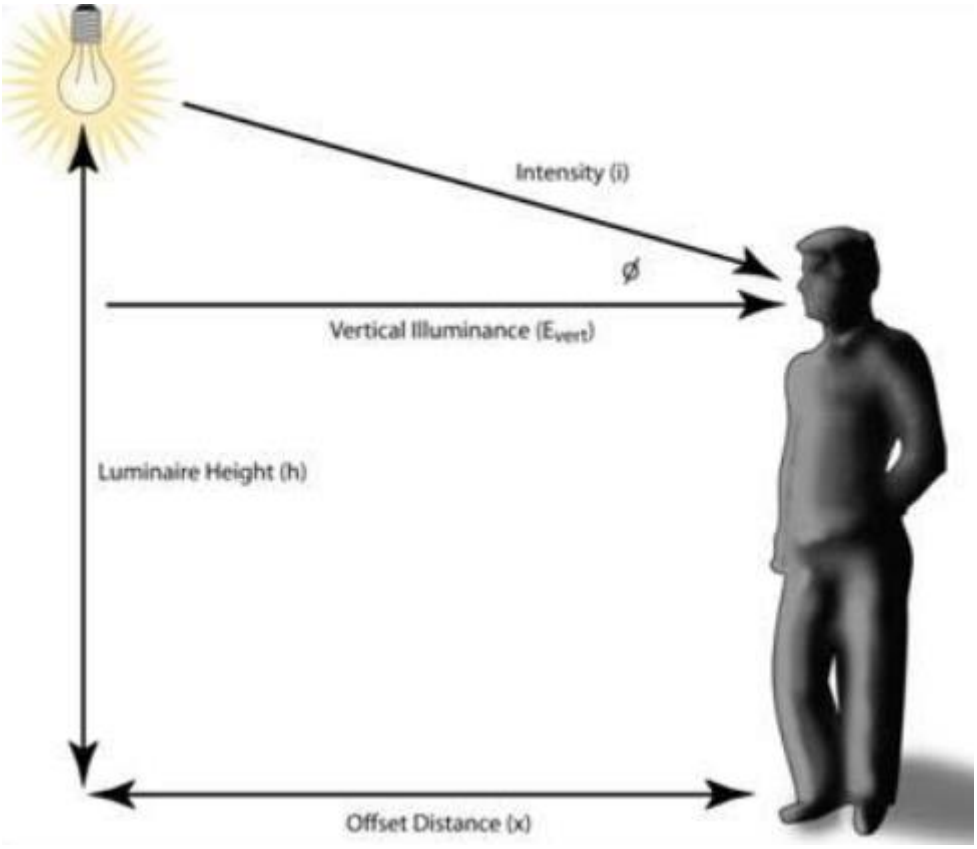

Рис. 1.1 Вертикальна освітленість пішохода

Як показано на рис. 1.1, той самий кут φ використовується для розрахунку вертикальної освітленості пішохода та відстані між світильником і точкою вимірювання. Нижче наведено рівняння для вертикальної освітленості пішохода від одного світильника.

$$E_{Vert} = \frac{I\cos\phi}{D^2} = (I\cos\phi)\frac{\sin^2\phi}{(h-1.5)^2} = \frac{I\cos\phi\sin^2\phi}{(h-1.5)^2} \quad (1.1)$$

Яскравість (L) — це світло, випромінюване поверхнею в певному напрямку на одиницю площі поверхні. Одиницею яскравості є кандела на квадратний метр. З точки зору візуального сприйняття, спостерігач сприймає яскравість. Це приблизний опис того, наскільки «яскравим» виглядає об’єкт, якщо дивитися з певного напрямку. Яскравість — це психофізичний феномен, який включає вигляд навколишнього поля зору та рівень адаптації спостерігача до яскравості об’єкта. Наприклад, повний місяць має однакову виміряну яскравість вдень і вночі, але місяць виглядає яскравішим на темному нічному небі. Однак яскравість не використовується для розрахунку візуального вигляду об’єкта; замість них використовуються яскравість і контраст.

Дифузне відбиття. Світло, що падає на поверхню, поглинається або відбивається. Розрізняють три види відбиття: дзеркальне, дифузне і світловідбивання. При дзеркальному відбиванні світло відбивається під рівним і протилежним кутом від кута падіння (обидва кути вимірюються від нормалі до поверхні); дифузне відбивання відбувається, коли світло відбивається в усіх напрямках, при цьому кількість відбитого світла в будь-якому заданому напрямку пропорційна косинусу кута між напрямком відбиття та нормаллю до поверхні; і світловідбивання виникає, коли світло відбивається назад до джерела освітлення. Хоча пішохід не є ідеальним

дифузним відбивачем, одяг, як правило, створює більш розсіяне відбиття. Таким чином, властивості дифузного відбивача використовуються для розрахунку освітленості пішохода при освітленні проїжджої частини. Загальне рівняння для яскравості дифузного відбивача показано нижче

$$L = \frac{\rho \times E}{\pi} \tag{1.2}$$

Контраст. Як показано вище, контраст є мірою візуальної різниці об’єкта та його фону. Найпоширенішим рівнянням для опису контрасту, як його сприймають спостерігачі, є Weber Contrast — різниця двох яскравостей, поділена на меншу яскравість. У системах освітлення проїжджої частини використовується модифікована версія Weber Contrast, яка розраховується як різниця в яскравості об’єкта інтересу (L_{Object}) і фоновій яскравості ($L_{Background}$), поділеній на фонову яскравість.

$$C = \frac{(L_{Object} - L_{Background})}{L_{Background}} \tag{1.3}$$

Зауважте, що контраст може бути як позитивним, так і негативним у діапазоні від позитивної нескінченності до -1. Негативний контраст — це стан, коли об’єкт темніший за фон, тому об’єкт виглядає силуетом, тоді як позитивний контраст — це стан, коли об’єкт яскравіший за фон.

Замінивши загальні терміни для розрахунку яскравості об’єкта на конкретну відбивну здатність (ρ_{Object}) і вертикальну освітленість (E_{Vert}), контраст можна представити наступним чином

$$C = \frac{\left(\frac{\rho_{Object} \cdot E_{Vertical}}{\pi} - L_{Background}\right)}{L_{Background}} \quad (1.4)$$

У цій формулі відбивна здатність об’єкта (ρ_{Object}) визначається вибором одягу пішохода, вертикальне освітлення пішохода (E_{Vert}) забезпечується пішохідним світильником, а фонова освітленість ($L_{Bockground}$) визначається навколишнім середовищем і розміщенням пішохідного переходу на проїжджій частині. Вертикальна освітленість – це єдиний параметр, який безпосередньо вибирає світлодизайнер. Хоча державні установи мало впливають на вибір одягу пішоходів, високий коефіцієнт відображення одягу, безумовно, покращить виявлення. Дослідження показали, що на видимість пішохода, одягненого в білий одяг, не впливала зміна рівня освітлення, тоді як на видимість людини, одягненої в джинсовий одяг, впливали різні умови освітлення. Інші дослідження показали такі ж результати.

Рівень видимості. Об’єкт у полі зору має теоретичний пороговий контраст (контраст яскравості, при якому об’єкт може бути просто виявлений). Пороговий контраст є функцією візуального розміру об’єкта (зазвичай описується кутом зору об’єкта α), тривалості часу спостереження, адаптаційної яскравості ($L_{Adaptation}$) спостерігача та віку спостерігача. Імовірність виявлення об’єкта з пороговою контрастністю становить 50%. Тому, щоб забезпечити надійне виявлення об’єктів або пішоходів на або вздовж проїжджої частини, умови освітлення повинні забезпечувати фактичний контраст, який перевищує пороговий контраст. Відношення фактичного контрасту до порогового контрасту визначається як рівень видимості (visibility level, VL). VL забезпечує вимірювання умов видимості та використовується як засіб для оцінки продуктивності освітлювальних

установок. У випадку пішохідного переходу вищий VL для даної умови вказує на те, що пішохід легше видно, ніж установка з нижчим VL.

$$VL = \frac{C_{Actual}}{C_{Threshold}} = \frac{C_{Actual}}{f\left(ObjectSize(\alpha), Age, L_{Adaptation}, ObservationTime(t)\right)} \quad (1.5)$$

VL можна змінити шляхом зміни порогового контрасту або фактичного контрасту. Однак багато факторів, з яких визначається пороговий контраст, не піддаються контролю. Вік спостерігача і час спостереження не можуть бути змінені, як і візуальний розмір об'єкта, який визначається відстанню від спостерігача до об'єкта інтересу. Це означає, що адаптована яскравість і фактичний контраст є єдиними значеннями, якими можна маніпулювати для зміни VL.

Для застосувань на дорозі фонова яскравість, в якій домінує яскравість поверхні дороги, вважається адаптаційною яскравістю спостерігача. Таким чином, маніпулювання яскравістю фону змінює як пороговий контраст, так і фактичний контраст. У ситуації, коли об’єкт видно з позитивним контрастом (яскравість об’єкта більша за яскравість фону), збільшення фону яскравість збільшить пороговий контраст, а фактичний зменшить. Це призводить до зниження VL. Для пішохідних переходів, де система освітлення, що використовується для освітлення пішоходів, також освітлює поверхню дороги, необхідно досягти балансу між рівнем освітленості проїжджої частини (фону), який створюється, і вертикальним освітленням, що забезпечує видимість пішоходів.

Полярність контрасту та дисперсія контрасту. Два інших аспекти контрасту, які необхідно враховувати під час розрахунку видимості об’єкта на проїжджій частині або вздовж неї, – це полярність контрасту та постійність контрасту на об’єкті. Як зазначалося вище, контраст може бути як

негативним, так і позитивним. Як правило, об’єкт легше виявити на негативному контрасті, а на відстанях понад 100 м яскравість дороги (фон) зазвичай вища, ніж яскравість більшості об’єктів на дорозі або вздовж неї. Однак на відстані менше 30 м збільшення вертикальної освітленості, що забезпечується фарами ближнього світла транспортних засобів, призведе до того, що об’єкт матиме вищу яскравість, ніж проїжджа частина. Для діапазону 30–100 м полярність контрасту залежатиме від кількох факторів, у тому числі від наявності фіксованого освітлення проїжджої частини, відбиття поверхні дороги, відбиття об’єкта та профілю інтенсивності фари автомобіля. Оскільки системі освітлення важко або неможливо підтримувати заданий рівень негативного контрасту, зазвичай вважається хорошою практикою встановити систему освітлення, яка забезпечує достатній позитивний контраст для виявлення пішоходів на відстані, що забезпечить адекватний час реакції. Другим аспектом контрасту, який слід враховувати, є постійність контрасту в об’єкті. У реальних ситуаціях на дорозі певний об’єкт зазвичай має різний контраст. Контраст може змінюватися від негативного до позитивного, а також змінюватися за величиною. На малюнку 7 пішохід у темному одязі має негативний контраст від стоп до колін, після чого контраст змінюється на позитивний — чорний одяг виглядає світлішим за фон. Для порівняння, пішохід у світлішому одязі виглядає в негативному контрасті лише біля ніг і в позитивному контрасті для решти тіла. Обидва ці аспекти контрасту пішоходів слід враховувати при проектуванні системи освітлення пішохідного переходу.

Візуальний фон. Фон для більшості пішохідних переходів складається з проїжджої частини та навколишнього середовища. Яскраві поверхні проїжджої частини або яскраві позашляхові об’єкти, як-от АЗС чи торговельні зони, збільшують яскравість фону та зменшують контрастність,

ускладнюючи виявлення пішоходів. Одна з труднощів виявлення пішоходів полягає в тому, що яскравість фону є непостійною. Фон також залежить від точки зору водія під час наближення до пішохідного переходу. На малюнку 8 пішохідний перехід показаний на відстані 60 м і 300 м від транспортного засобу. Червоні стрілки позначають пішохода на пішохідному переході. Тут можна побачити, що фон, на якому видно пішохода, змінюється, коли транспортний засіб наближається до пішохідного переходу. На відстані 300 м пішохода в основному видно на тлі навколишнього середовища (небо та неосвітлений віддалений фон), тоді як на висоті 60 м нижню половину пішохода видно на яскравому тлі освітленої проїзної частини.

Рівень дизайну освітлення слід вибирати так, щоб забезпечити адекватну продуктивність навіть на яскравому фоні. Як правило, чим яскравіший фон, тим більша вертикальна освітленість потрібна водієві, щоб чітко бачити пішохода на пішохідному переході.

1.2. IoT, як компонент персональної системи безпеки пішохода

Інтернет речей (IoT) відноситься до мережі фізичних пристроїв, транспортних засобів, приладів та інших фізичних об’єктів, у які вбудовано датчики, програмне забезпечення та підключення до мережі, що дозволяє їм збирати та обмінюватися даними. Ці пристрої — також відомі як «розумні об’єкти» — можуть варіюватися від простих пристроїв «розумного дому», таких як розумні термостати, до носіїв, таких як розумні годинники та одяг із підтримкою RFID, до складних промислових машин і транспортних систем. Технологи навіть уявляють собі цілі «розумні міста», засновані на технологіях IoT.

IoT дозволяє цим розумним пристроям спілкуватися один з одним та з іншими пристроями з підтримкою Інтернету, такими як смартфони та шлюзи, створюючи величезну мережу взаємопов’язаних пристроїв, які можуть обмінюватися даними та виконувати різноманітні завдання автономно. Це може включати все: від моніторингу умов навколишнього середовища на фермах до керування транспортними потоками за допомогою інтелектуальних автомобілів та інших інтелектуальних автомобільних пристроїв, керування машинами та процесами на заводах до відстеження запасів і поставок на складах.

Потенційні сфери застосування IoT величезні та різноманітні, і його вплив уже відчувається в багатьох галузях промисловості, включаючи виробництво, транспорт, охорону здоров’я та сільське господарство. Оскільки кількість пристроїв, підключених до Інтернету, продовжує зростати, IoT, ймовірно, відіграватиме все більш важливу роль у формуванні нашого світу та зміні того, як ми живемо, працюємо та взаємодіємо один з одним.

У корпоративному контексті пристрої IoT використовуються для моніторингу широкого діапазону параметрів, таких як температура, вологість, якість повітря, споживання енергії та продуктивність машини. Ці дані можна аналізувати в режимі реального часу, щоб виявити закономірності, тенденції та аномалії, які можуть допомогти підприємствам оптимізувати свою діяльність і підвищити прибутковість.

IoT важливий для бізнесу з кількох причин. Наведемо основні переваги IoT.

Використовуючи пристрої IoT для автоматизації та оптимізації процесів, підприємства можуть підвищити ефективність і продуктивність. Наприклад, датчики IoT можна використовувати для моніторингу продуктивності обладнання та виявлення або навіть вирішення потенційних

проблем до того, як вони спричинять простой, зменшуючи витрати на обслуговування та покращуючи час безвідмовної роботи.

Прийняття рішень на основі даних. Пристрої IoT генерують величезні обсяги даних, які можна використовувати для прийняття більш обґрунтованих бізнес-рішень і нових бізнес-моделей. Аналізуючи ці дані, підприємства можуть отримати уявлення про поведінку клієнтів, ринкові тенденції та операційну ефективність, дозволяючи їм приймати більш обґрунтовані рішення щодо стратегії, розробки продукту та розподілу ресурсів.

Економія коштів. Зменшуючи ручні процеси та автоматизуючи повторювані завдання, IoT може допомогти підприємствам зменшити витрати та підвищити прибутковість. Наприклад, пристрої IoT можна використовувати для моніторингу споживання енергії та оптимізації споживання, зниження витрат на енергію та підвищення екологічності.

Покращений досвід роботи з клієнтами. Використовуючи технологію IoT для збору даних про поведінку клієнтів, підприємства можуть створювати більш персоналізовані та привабливі умови для своїх клієнтів. Наприклад, роздрібні торговці можуть використовувати датчики Інтернету речей, щоб відстежувати рух клієнтів у магазинах і надавати персоналізовані пропозиції на основі їх поведінки.

Технології, які роблять IoT можливим. Кілька технологій поєднуються, щоб зробити IoT можливим.

Датчики та виконавчі механізми. Датчики – це пристрої, які можуть виявляти зміни в навколишньому середовищі, такі як температура, вологість, світло, рух або тиск. Актуатори — це пристрої, які можуть спричиняти фізичні зміни в навколишньому середовищі, наприклад відкривати чи закривати клапан або вмикати двигун. Ці пристрої є основою IoT, оскільки

вони дозволяють машинам і пристроям взаємодіяти з фізичним світом. Автоматизація можлива, коли датчики та виконавчі механізми працюють для вирішення проблем без втручання людини.

Технології підключення: щоб передавати дані IoT від датчиків і приводів до хмари, пристрої IoT мають бути підключені до Інтернету. В IoT використовується кілька технологій підключення, зокрема Wi-Fi, Bluetooth, стільниковий зв'язок, Zigbee і LoRaWAN.

Хмарні обчислення: у хмарі зберігаються, обробляються та аналізуються величезні обсяги даних, створених пристроями Інтернету речей. Платформи хмарних обчислень забезпечують інфраструктуру та інструменти, необхідні для зберігання та аналізу цих даних, а також для створення та розгортання додатків IoT.

Аналітика великих даних: щоб зрозуміти величезні обсяги даних, створені пристроями Інтернету речей, компаніям потрібно використовувати розширені інструменти аналітики для отримання інформації та виявлення закономірностей. Ці інструменти можуть включати алгоритми машинного навчання, інструменти візуалізації даних і моделі прогнозної аналітики.

Технології безпеки та конфіденційності: оскільки розгортання IoT стає все більш поширеним, безпека та конфіденційність IoT стають все більш важливими. Такі технології, як шифрування, контроль доступу та системи виявлення вторгнень, використовуються для захисту пристроїв IoT і даних, які вони генерують, від кіберзагроз.

Приклади додатків IoT. У галузі охорони здоров'я пристрої IoT можна використовувати для віддаленого спостереження за пацієнтами та збору даних у режимі реального часу щодо їхніх життєво важливих показників, таких як частота серцевих скорочень, артеріальний тиск і насичення киснем. Дані датчиків можна аналізувати, щоб виявити закономірності та потенційні

проблеми зі здоров’ям, перш ніж вони стануть серйознішими. Пристрої IoT також можна використовувати для відстеження медичного обладнання, управління запасами та моніторингу відповідності ліків.

Промислові пристрої IoT можна використовувати на виробництві для моніторингу продуктивності машин, виявлення несправностей обладнання та оптимізації виробничих процесів. Наприклад, датчики можна використовувати для моніторингу температури та вологості на виробничому підприємстві, забезпечуючи оптимальні умови для виробництва чутливої продукції. Пристрої IoT також можна використовувати для відстеження запасів, управління ланцюжками поставок і моніторингу якості готової продукції. Промисловий IoT — це такий великий простір нових технологій, що його іноді називають власною абревіатурою: IIOT (Industrial IoT).

У роздрібній торгівлі пристрої IoT можна використовувати для відстеження поведінки клієнтів, моніторингу рівня запасів і оптимізації планування магазинів. Наприклад, датчики можна використовувати для відстеження відвідувачів у магазині та аналізу поведінки покупців, дозволяючи роздрібним торговцям оптимізувати розміщення товарів і покращити взаємодію з клієнтами. Пристрої IoT також можна використовувати для моніторингу ланцюжків поставок, відстеження поставок і керування рівнем запасів.

Пристрої IoT можна використовувати в сільському господарстві для моніторингу стану ґрунту, погодних умов і росту врожаю. Наприклад, датчики можна використовувати для вимірювання вмісту вологи в ґрунті, гарантуючи, що посіви зрошуються в оптимальний час. Пристрої IoT також можна використовувати для моніторингу здоров’я худоби, відстеження обладнання та керування ланцюгами поставок. Пристрої з низькою

потужністю або сонячною енергією часто можна використовувати з мінімальним наглядом у віддалених місцях.

У транспортній галузі пристрої IoT можна використовувати для моніторингу роботи транспортних засобів, оптимізації маршрутів і відстеження відправлень. Наприклад, датчики можна використовувати для моніторингу паливної ефективності підключених автомобілів, зменшуючи витрати на пальне та покращуючи екологічність. Пристрої IoT також можна використовувати для моніторингу стану вантажу, гарантуючи, що він прибуде до пункту призначення в оптимальному стані.

Ризики та виклики в IoT. IoT пропонує багато переваг, але також створює кілька ризиків і проблем. Ось деякі з найбільш значущих з них.

Ризики для безпеки та конфіденційності: оскільки пристрої IoT стають все більш поширеними, безпека та конфіденційність стають дедалі важливішими. Багато пристроїв IoT вразливі до хакерів та інших кіберзагроз, які можуть поставити під загрозу безпеку та конфіденційність конфіденційних даних. Пристрої IoT також можуть збирати величезні обсяги персональних даних, що викликає занепокоєння щодо конфіденційності та захисту даних.

Проблеми сумісності: пристрої IoT від різних виробників часто використовують різні стандарти та протоколи, що ускладнює для них здійснення так званого зв’язку «машина-машина». Це може призвести до проблем сумісності та створити накопичені дані, які важко інтегрувати та аналізувати.

Перевантаження даних: пристрої IoT генерують величезні обсяги даних, які можуть перевантажити підприємства, які не готові їх обробляти. Аналіз цих даних і отримання значущої інформації може бути серйозною

проблемою, особливо для компаній, які не мають необхідних інструментів аналітики та досвіду.

Вартість і складність: Впровадження системи IoT може бути дорогим і складним, вимагаючи значних інвестицій в апаратне забезпечення, програмне забезпечення та інфраструктуру. Управління та підтримка системи Інтернету речей також може бути складною справою, що вимагає спеціальних навичок і досвіду.

Регуляторні та юридичні проблеми: Оскільки пристрої IoT стають все більш поширеними, виникають нормативні та правові проблеми. Підприємствам необхідно дотримуватися різноманітних нормативних актів щодо захисту даних, конфіденційності та кібербезпеки, які залежать від країни.

Як бізнес повинен підходити до Інтернету речей. Управління пристроями IoT може бути складним і важким завданням, але є кілька найкращих практик, яких компанії можуть дотримуватися, щоб переконатися, що їхні пристрої IoT безпечні, надійні та оптимізовані для продуктивності. Ось кілька порад щодо керування пристроями IoT.

Планування стратегії Інтернету речей: перед розгортанням будь-яких пристроїв Інтернету речей компанії повинні мати чітке розуміння своїх цілей, випадків використання та бажаних результатів. Це може допомогти їм вибрати правильні пристрої, платформи й технології IoT, а також переконатися, що їх стратегія IoT узгоджується з їхніми бізнес-цілями.

Вибирайте безпечні продукти Інтернету речей: Безпека є критично важливим фактором для рішень Інтернету речей, оскільки вони можуть бути вразливими до кібератак. Компанії повинні вибирати пристрої, розроблені з урахуванням безпеки, і впроваджувати відповідні системи безпеки, такі як шифрування, автентифікація та контроль доступу.

Контролюйте та обслуговуйте пристрої: пристрої IoT необхідно регулярно контролювати та обслуговувати, щоб переконатися, що вони працюють оптимально та не вразливі до загроз безпеці. Це може включати моніторинг стану та продуктивності пристрою, оновлення мікропрограми та програмного забезпечення, а також проведення регулярних перевірок безпеки та прогнозного обслуговування.

Ефективне керування даними: пристрої IoT генерують величезні обсяги реальних даних, якими може бути складно керувати та аналізувати. Підприємства повинні мати чітку стратегію управління даними, включаючи зберігання, аналіз і візуалізацію даних, щоб гарантувати, що вони можуть отримати значущу інформацію з даних, створених їхніми пристроями IoT.

Створення екосистеми: пристрої IoT часто є частиною більшої екосистеми, яка включає інші пристрої, платформи та технології. Компанії повинні мати чітке розуміння цієї екосистеми та переконатися, що їхні пристрої IoT можуть ефективно інтегруватися з іншими системами та технологіями.

РОЗДІЛ 2
АПАРАТНА СИСТЕМА ПІДВИЩЕННЯ БЕЗПЕКИ ПІШОХОДА

2.1. Особливості операційної системи платформи Arduino

Arduino — це електронна платформа з відкритим вихідним кодом, яка використовує прості плати введення/виведення та середовище розробки для взаємодії з платою. Плати Arduino здатні виконувати функцій на основі набору інструкцій, які надсилаються мікроконтролеру. Програмне забезпечення Arduino з відкритим кодом (Arduino IDE) використовується для написання коду (C і C++) і завантаження його на плату.

Високорівнева архітектура Arduino IDE складається з основних модулів arduino-core, app і build (рис. 2.1).

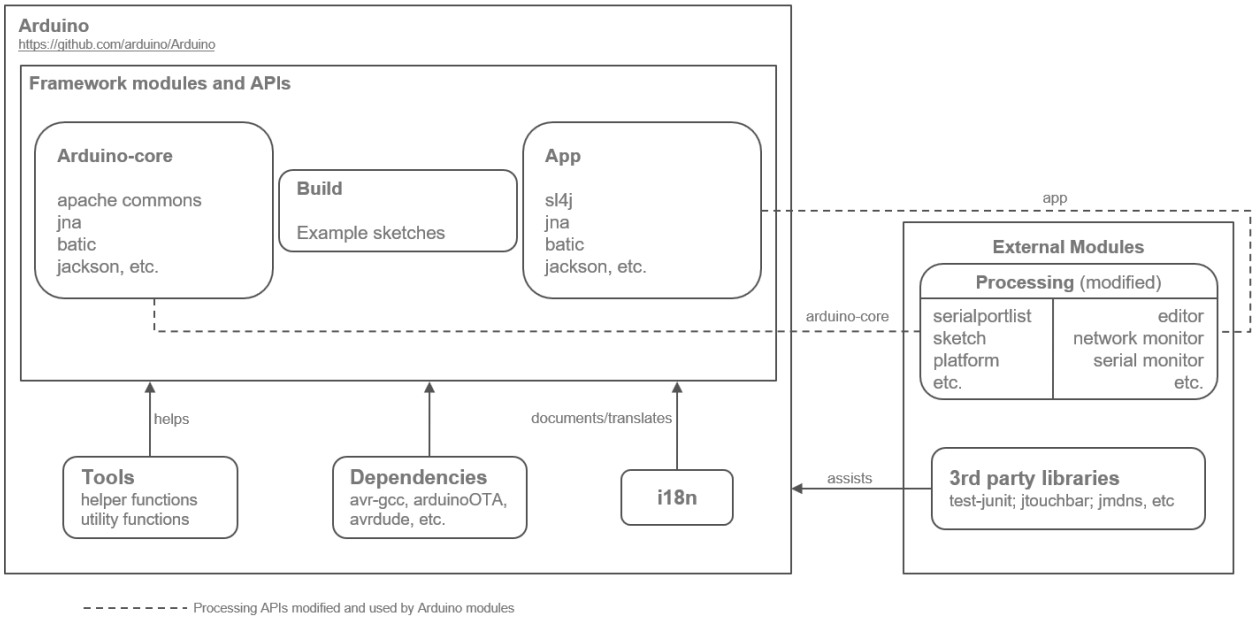

Рис. 2.1 Модульна організація Arduino IDE.

Модуль arduino-core (Java). Цей модуль містить код, необхідний для основних функцій IDE. Він не містить графічного інтерфейсу і може бути виконаний з командного рядка. Найважливішими функціями є забезпечення з'єднання через USB послідовний порт з мікропроцесором Arduino, об'єднання ескізів коду C із використаними бібліотеками, об'єднання скомпільованих ескізів із правильним завантажувачем для використовуваного типу мікропроцесора Arduino та забезпечення системи компіляції.

Модуль app (Java). Цей модуль побудований на основі модуля arduino-core і забезпечує графічний інтерфейс, який ми знаємо як Arduino IDE. Найважливішими компонентами цього графічного інтерфейсу є редактор коду, послідовний монітор, послідовний плоттер і менеджер бібліотеки.

Модуль build (binaries/images/text). Цей модуль містить додаткові файли, відмінні від Java, які використовуються в поєднанні з arduino-core та додатком для створення розповсюджуваного архіву Arduino IDE, який можуть завантажити кінцеві користувачі. Найважливіші файли:

- The Apache Ant build file: цей файл містить конфігурацію збірки всього проекту.
- Example sketch references: це список прикладів репозиторіїв ескізів для програмування мікропроцесорів Arduino та хешів sha для підтвердження того, що ці автоматично клоновані сховища справді містять очікуваний вміст (версію).
- OS-specific drivers: драйвери для ОС, які дозволяють підключатися до мікропроцесорів Arduino через USB.
- Native libraries: бібліотеки, які потрібні arduino-core або програмам, таким як компілятор GCC C і файли .dll у Windows для отримання певних каталогів за замовчуванням.
- Art: зображення, які використовує додаток.

Проект написаний на Java і має такі залежності збірки:

- Apache Ant: використовується для створення проекту.
- Java Development Kit (JDK): використовується для компіляції джерел Java.
- Arduino C libraries: це бібліотеки, які можуть використовуватися кінцевими користувачами для написання коду для мікропроцесорів Arduino. Це: Ethernet, GMN, Stepper, TFT, WiFi, arduino, Bridge, Robot_Control, Robot_Motor, RobotIRremote, SpacebrewYun, Temboo, Esplora, Mouse, Keyboard, SD, Servo, LiquidCrystal і Adafruit_CircuitPlayground. По суті, усі бібліотеки, крім бібліотеки arduino, можна опустити. Бібліотека Arduino містить функції, які дозволяють користувачеві отримати доступ до функцій мікроконтролера Arduino без необхідності виконувати прямі маніпуляції з регістрами.
- GCC compiler: користувачі програмують мікроконтролери Arduino на C. Компілятор GCC використовується для компіляції коду C у машинний код, який працює на різних архітектурах мікропроцесорів Arduino.
- launch4j: використовується для обгортання проекту Java у виконуваний файл Windows.
- Open source Java libraries: використовується кілька більш загальних бібліотек Java з відкритим кодом. Банки з їх ліцензіями можна переглянути в репозиторії за адресами app/lib і arduino-core/lib.

Зовнішні залежності. Модулі arduino-core і app містять пакет processing.app, що містить частково модифіковані класи із Processing Development Environment (PDE). Час від часу, коли проект PDE оновлюється, ці оновлення також вручну об’єднуються в скопійовані класи. Намагаючись зробити цей процес злиття максимально простим, розробники Arduino IDE

намагаються зберегти сигнатуру цих класів якомога подібнішою до оригіналу. Зміни до цих класів вносяться для задоволення потреб специфічних функцій Arduino IDE.

Загальна модель дизайну Common Design Model. Кожна версія Arduino виявляється схожою через фактор спільності, який забезпечується обмеженнями розробки Arduino IDE. Основною причиною цього є зменшення ризику та дублювання зусиль у поєднанні зі збільшенням загальної узгодженості системи.

Загальна обробка. Модулі arduino-core та app складаються з унікальних пакетів, включаючи адаптовані пакети із середовища розробки обробки (PDE), які були змінені відповідно до потреб Arduino IDE. Немає спільних процесів.

Налаштування бібліотек: деякі важливі бібліотеки підтримуються командою розробників, оскільки вони є важливою частиною коду. Ці бібліотеки створені спільнотою, і їх можна побачити в списку. Будь-які налаштовані бібліотеки можна надіслати на форум розробників Arduino для будь-яких пропозицій або можливого включення як нового запису.

Ядро Arduino — це місце, де можна знайти вихідні файли всіх вбудованих функцій. В таблицях нижче наведені основних вихідних файлів, згрупованих відповідно до функціональності.

Таблиця 2.1

Загальні вихідні файли

Вихідний файл	Основне призначення
Arduino.h	1- Оголошення більшості вбудованих функцій Arduino, таких як pinMode(), digitalWrite() тощо.

Вихідний файл	Основне призначення
	2- Макроси деяких констант, таких як HIGH, LOW, INPUT, OUTPUT тощо. 3- Макрофункції для побітових операцій та деяких інших загальних операцій, таких як: min(), rand(), ..і т.д. 4- Оголошення масивів відображення контактів і портів (перевірте наш мікроблог про відображення масивів у Arduino) у флеш-пам'яті, щоб створити карту між номером контакту Arduino та фізичним номером. тобто зіставлення виводу 13 із відповідним портом і виводом у регістрах MCU. Значення цих масивів можна знайти тут: hardware\arduino\avr\variants\standard\pins_arduino.h

main.cpp	Тут оголошено базову структуру програми Arduino. Ось актуальна програма:
	`int main(void)`
	`{`
	`init();`
	`initVariant();`
	`setup();`
	`for (;;) {`
	`loop(); }`
	`return 0;`
	`}`

Таблиця 2.2

Вихідні файли, пов'язані з цифровими та аналоговими портами

Вихідний файл	Основне призначення
wiring_digital.c	Defines digitalWrite(), digitalRead(), pinMode(), ..etc.

Вихідний файл	Основне призначення
wiring_analog.c	Defines analogRead(), analogWrite() ..etc.
wiring_shift.c	Two functions to read in/out serial bits-streams.
wiring_pulse.c	Two functions to read input PWM signals
WInterrupts.c	Defines some functions to handle external interrupts.

Таблиця 2.3

Вихідні файли, пов’язані з синхронізацією та таймерами

Вихідний файл	Основне призначення
wiring.c	1- Визначає функції синхронізації, такі як millis(), delay() тощо. 2- Timer0 переповнення ISR 3- Функція ініціалізації, яка використовується для ініціалізації деяких периферійних пристроїв перед викликом функції налаштування в main.cpp.

Таблиця 2.4

Вихідні файли, пов’язані з послідовним інтерфейсом

Вихідний файл	Основне призначення
Stream.h, Stream.cpp	Визначає клас Stream, які його методи використовуються (за успадкуванням) у класі HardwareSerial, наприклад readUntil(), find(), ..і т.д.
Print.cpp, Print.h	Визначає клас Print із методами друку повідомлень (print & println). Це успадковується класом Stream.
Printable.h	Цей заголовок визначає клас, який може бути успадкований іншими, щоб зробити будь-який новий тип об’єкта

Вихідний файл	Основне призначення
	доступним для друку за допомогою print() і println().
HardwareSerial.h, HardwareSerial.cpp, HardwareSerial_private.h	1- Визначає клас HardwareSerial. 2- Інші методи, такі як print(), find(), .. тощо, успадковані від класу Stream
HardwareSerialx.cpp	1- Визначає UARTx ISR. 2- Оголошує об’єкт класу HardwareSerial Serialx

Таблиця 2.5

Додаткові файли

Вихідний файл	Основне призначення
New.cpp and new.h	Нові визначення функцій динамічного розподілу пам'яті.
binary.h	Макроси нотації Bxxx в Arduino
Tone.cpp	Деякі функції для створення тонів на деяких контактах
WCharacter.h	Визначає кілька простих функцій для обробки символів, як-от: toAscii() за допомогою функцій із заголовка ctype
Wmath.cpp	Деякі математичні функції, як-от функція map()
wiring_private.h	Деякі розрізнені декларації
WString.cpp & WString.h	Визначає клас String в Arduino

Огляд файлової системи. По-перше, весь вихідний файл ядра Arduino розміщується в наступному каталозі

`\Arduino\hardware\arduino\{the MCU architecture}\cores\arduino. Another place for Arduino third`

party cores (i.e. ESP8266 core) is in \Users\{username}\ AppData\Local \Arduino15 \packages

і ці ядра завантажені з менеджера board (Tools>Boards>Board Manager). Інструменти Tool-chain (і навіть файли заголовків avr-gcc) знаходяться в

Arduino\hardware\tools\{the MCU architecture}

а також

Users\{username}\AppData\Local\Arduino15\packages\arduino\tools.

2.2. Драйвер інтерфейсу світлодіоднинної матриці 8x8

MAX7219/MAX7221 — це компактні драйвери дисплеїв із послідовним входом/виводом із загальним катодом, що з’єднують мікропроцесори та 7-сегментні числові світлодіодні дисплеїв до 8 цифр, гістографічних дисплеїв або 64 окремих світлодіодів. У мікросхемі є декодер коду BCD, схема мультиплексного сканування, драйвери сегментів і цифр, а також статична оперативна пам’ять 8x8, яка зберігає кожну цифру. Для встановлення струму сегмента для всіх світлодіодів потрібен лише один зовнішній резистор. MAX7221 сумісний із SPI, QSPI та MICROWIRE і має сегментні драйвери з обмеженою швидкістю наростання для зменшення електромагнітних перешкод.

Зручний 4-провідний послідовний інтерфейс підключається до всіх поширених мікроконтролерів. Окремі цифри можна адресувати та оновлювати без перезапису всього дисплея. MAX7219/MAX7221 також дозволяє користувачеві вибирати декодування з кодом B або без декодування для кожної цифри.

Мікросхема має режим відключення з низьким енергоспоживанням 150 мкА, аналоговий і цифровий контроль яскравості, регістр обмеження сканування, який дозволяє користувачеві відображати від 1 до 8 цифр, і тестовий режим, який примусово включає всі світлодіоди.

Типові застосування: гістограми, промислові контролери, панельні лічильники, світлодіодні матричні дисплеї.

Основні особливості: послідовний інтерфейс зі швидкістю 10 МГц, індивідуальне керування сегментами світлодіодів, вибір цифри декодування/недекодування, режим з низьким енергоспоживанням 150 мкА, цифрове та аналогове керування яскравістю, дисплей гасне під час увімкнення, світлодіодний дисплей із загальним катодом, драйвери обмеженого сегмента швидкості наростання для зниження рівня електромагнітних перешкод, послідовний інтерфейс SPI, QSPI, MICROWIRE.

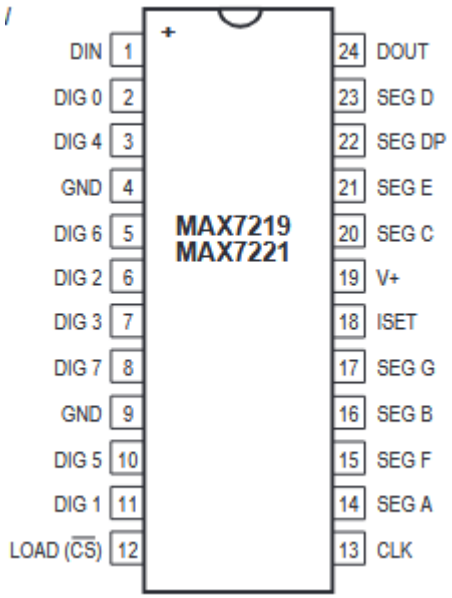

Рис. 2.2. Виводи мікросхеми MAX7219

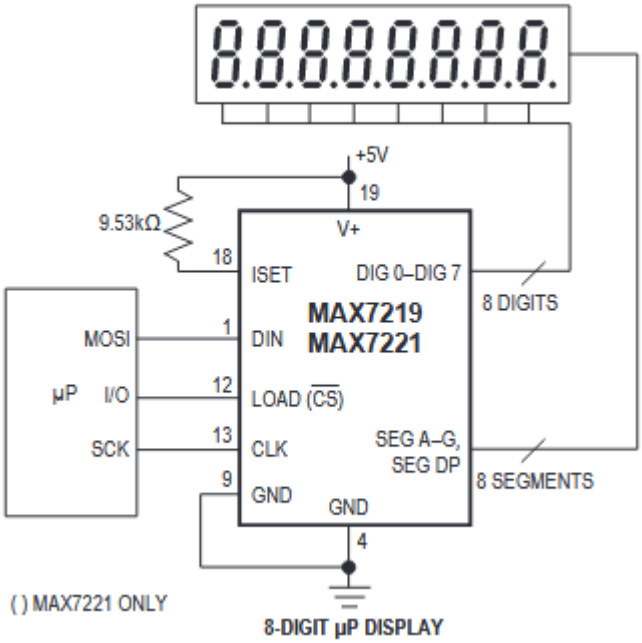

Рис. 2.3. Зовнішні з’єднання мікросхеми MAX7219

Таблиця 2.6

Призначення виводів

Вивод	Назва	Функція
1	DIN	Послідовний вхід даних Дані завантажуються у

		внутрішній 16-розрядний регістр зсуву по наростаючому фронту CLK.
2, 3, 5–8, 10, 11	DIG 0–DIG 7	Восьмирозрядні лінії приводу, які споживають струм із загального катода дисплея. MAX7219 підтягує цифрові виходи до V+, коли вимкнено. Цифрові драйвери MAX7221 мають високий опір у вимкненому стані.
4, 9	GND	Обидва контакти GND повинні бути підключені.
12	LOAD (MAX7219)	Завантаження даних. Останні 16 біт послідовних даних фіксуються на передньому фронті LOAD.
4, 9	CS (MAX7221)	Вхід вибору мікросхеми. Послідовні дані завантажуються в регістр зсуву, поки CS низький. Останні 16 біт послідовних даних фіксуються на передньому фронті CS.
13	CLK	Послідовний тактовий вхід. Максимальна швидкість 10 МГц. На передньому фронті CLK дані зміщуються у внутрішній регістр зсуву. На спадному фронті CLK дані тактуються з DOUT. На MAX7221 вхід CLK активний лише тоді, коли CS низький.
14–17, 20–23	SEG A–SEG G, DP	Сім сегментів та десяткова крапка. Передають джерело струму на дисплей. На MAX7219, коли сегментний драйвер вимкнено, він підтягується до GND. Сегментні драйвери MAX7221 мають високий опір у вимкненому стані.
18	ISET	Підключіться до VDD через резистор (R_{SET}), щоб встановити піковий струм сегмента.

19	V+	Позитивна напруга живлення.
24	DOUT	Послідовний вихід даних. Дані в DIN дійсні через 16,5 тактів DOUT. Цей вихід використовується для послідовного підключення кількох MAX7219/MAX7221 і ніколи не має високого опору.

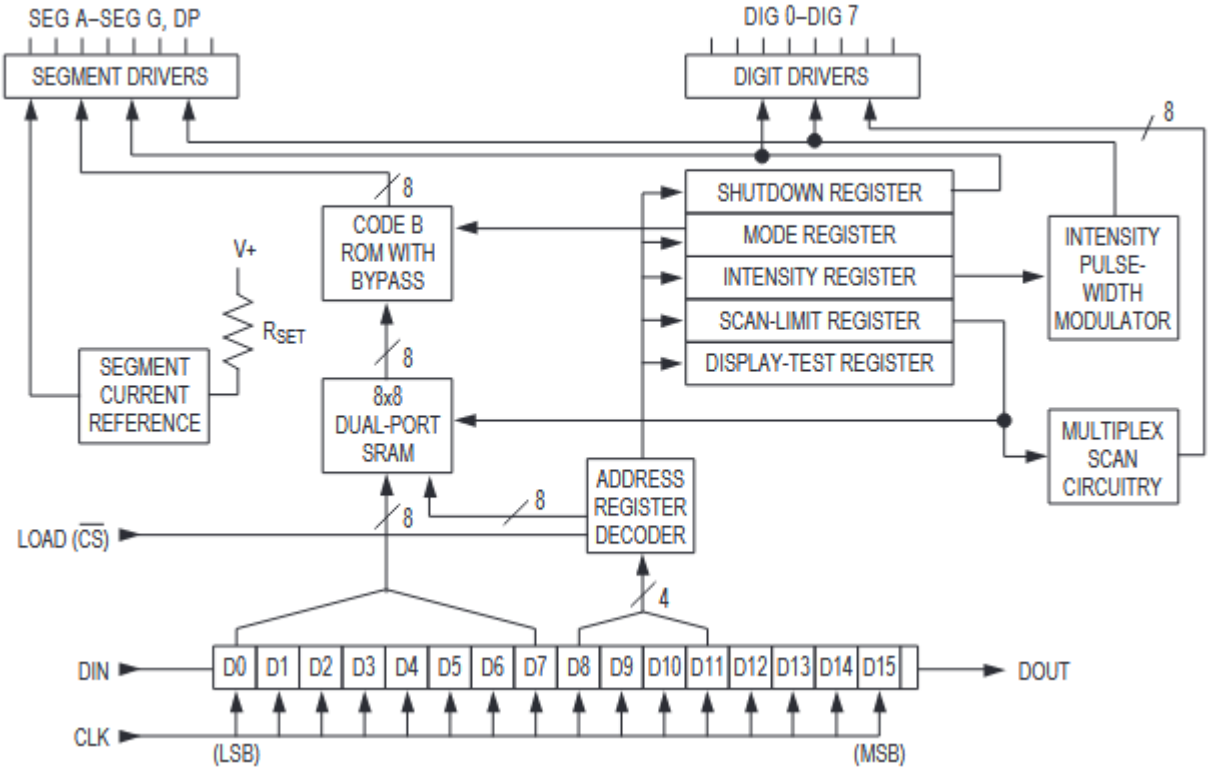

Рис. 2.4 Внутрішня структура драйвера

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	ADDRESS				MSB	DATA						LSB

Рис. 2.5. Формат послідовних даних 16 біт.

Мікросхеми MAX7219 і MAX7221 ідентичні, за винятком двох параметрів: драйвери сегментів MAX7221 обмежені швидкістю наростання

для зменшення електромагнітних перешкод (EMI), а його послідовний інтерфейс повністю сумісний з SPI.

Режими послідовної адресації. Для MAX7219 послідовні дані в DIN, надіслані в 16-бітних пакетах, зсуваються у внутрішній 16-бітний регістр зсуву з кожним наростаючим фронтом CLK незалежно від стану LOAD. Для MAX7221 CS має бути низьким для синхронізації даних. Потім дані фіксуються або в цифровому, або в регістрі керування на передньому фронті LOAD/CS. LOAD/CS має досягати високого рівня одночасно з або після 16-го наростаючого фронту тактової частоти, але до наступного наростаючого фронту тактової частоти, інакше дані будуть втрачені.

Дані в DIN поширюються через регістр зсуву і з'являються на DOUT на 16,5 тактів пізніше. Дані синхронізуються на спадному фронті CLK. Біти даних позначені D0–D15. Біти D8–D11 містять адресу реєстру. D0–D7 містять дані, а D12–D15 – незадіяні. Першим отриманим є D15, старший біт (MSB).

У таблиці 2.7 наведено 14 адресованих цифрових і контрольних регістрів.

Цифрові регістри реалізовані за допомогою двопортової SRAM 8x8. Вони адресовані безпосередньо, щоб окремі цифри могли оновлюватися та зберігати дані, поки V+ перевищує 2 В. Регістри керування складаються з режиму декодування, інтенсивності відображення, обмеження сканування (кількість сканованих цифр), вимкнення та перевірки дисплея (всі світлодіоди увімкнені).

Режим енергозбереження. Коли MAX7219 перебуває в режимі енергозбереження, генератор сканування зупиняється, усі сегментні джерела струму заземлюються, а всі цифрові драйвери переходять до V+, тим самим вимикаючи дисплей. MAX7221 ідентичний, за винятком драйверів з високим опором. Дані в цифрових і контрольних регістрах залишаються незмінними.

Вимкнення можна використовувати для економії енергії або як сигнал тривоги для блимання дисплея шляхом послідовного входу та виходу з режиму енергозбереження. Для мінімального струму живлення в режимі енергозбереження логічні входи мають бути заземлені або V+ (логічні рівні CMOS).

Зазвичай для виходу MAX7219/MAX7221 з режиму вимкнення потрібно менше 250 мкс. Драйвер дисплея можна запрограмувати в режимі енергозбереження, а режим енергозбереження можна перевизначити функцією тестування дисплея.

Таблиця 2.7

Карта адрес регістрів

REGISTER	ADDRESS					HEX CODED
	D15–D12	D11	D10	D9	D8	
No-Op	X	0	0	0	0	0xX0
Digit 0	X	0	0	0	1	0xX1
Digit 1	X	0	0	1	0	0xX2
Digit 2	X	0	0	1	1	0xX3
Digit 3	X	0	1	0	0	0xX4
Digit 4	X	0	1	0	1	0xX5
Digit 5	X	0	1	1	0	0xX6
Digit 6	X	0	1	1	1	0xX7
Digit 7	X	1	0	0	0	0xX8
Decode Mode	X	1	0	0	1	0xX9
Intensity	X	1	0	1	0	0xXA

Scan Limit	X	1	0	1	1	0xXB
Shutdown	X	1	1	0	0	0xXC
Display Test	X	1	1	1	1	0xXF

2.3. Бібліотека драйвера матриці світлодіодів

Бібліотека LedControl Arduino може управляти більш ніж 8 сегментами (обмежується лише пам'яттю). Це бібліотека для світлодіодних матричних дисплеїв на основі MAX7219.

Цей драйвер використовує апаратний SPI, що робить його набагато швидшим (за винятком програмного SS).

Функції Display або displayRow() потрібно використовувати для оновлення дисплея і вони підтримують до 255 сегментів.

Може використовувати зовнішню пам'ять або саморозподілений буфер.

Вибір виводів. Кожен сегмент дисплея вимагає 16 біт, і всі вони можуть бути зміщені одночасно. Щоб зробити передачу даних більш швидкою використовується апаратний SPI замість програмного опитування.

Виводи підключаються наступним чином:

MOSI підключається до DIN,

SCK підключається до CLK,

MISO не використовується,

SS не використовується.

Користувач повинен вибрати, який контакт використовувати для програмного забезпечення SS, але не використовувати апаратні контакти SPI.

Рекомендовані контакти для Arduino Uno:

DIN підключається до 11;

CLK підключений до 13;

CS підключений до 9.

Бібліотека представлена класом LEDMatrixDriver

```
class LEDMatrixDriver
{
```

Команди, які визначено в таблиці для мікросхеми

```
const static uint16_t ENABLE =          0x0C00;
const static uint16_t TEST =       0x0F00;
const static uint16_t INTENSITY =  0x0A00;
const static uint16_t SCAN_LIMIT = 0x0B00;
const static uint16_t DECODE =          0x0900;
```

Константи для інвертування

```
public:
     const static uint8_t INVERT_SEGMENT_X = 1;
     const static uint8_t INVERT_DISPLAY_X = 2;
     const static uint8_t INVERT_Y = 4;
```

Створення об’єкту з N сегментами та ssPin як SS, прапорці описують орієнтацію сегмента, можна також надати вже виділений буфер (необов'язково)

```
     LEDMatrixDriver(uint8_t N, uint8_t ssPin, uint8_t flags = 0,
uint8_t* frameBuffer = nullptr);
     ~LEDMatrixDriver();
```

Створений об’єкт неможна копіювати, тому відповідні конструктори та функції забороняються

```
LEDMatrixDriver(const LEDMatrixDriver& other) = delete;
LEDMatrixDriver(LEDMatrixDriver&& other) = delete;
LEDMatrixDriver& operator=(const LEDMatrixDriver& other) = delete;
```

Функції для піксельних дісплеїв

```
//Команда розблокування
void setEnabled(bool enabled);
//Яскравість дисплея: 0 - 15
void setIntensity(uint8_t level);
//Встановити піксель
void setPixel(int16_t x, int16_t y, bool enabled);
```

Отримати значення пікселу

```
bool getPixel(int16_t x, int16_t y) const;
```

Встановлює пікселі в стовпці відповідно до значення (LSB => y=0)

```
void setColumn(int16_t x, uint8_t value);
uint8_t getSegments() const {return N;}
```

Встановити адресу буфера

```
uint8_t* getFrameBuffer() const {return frameBuffer;}
```

Функції для 7-сегментних дисплеїв. Встановлення кількості цифр, що відображаються (0 -> 1 цифра, 7 -> 8 цифр)

```
void setScanLimit(uint8_t level);
```

Встановлення декодеру

```
void setDecode(uint8_t mask);
```

Запис числа для відображення

```
void setDigit(uint16_t digit, uint8_t value, bool dot = false);
```

Вивести дані на дисплей для відображення

```
void display();
```

Скинути один рядок на дисплей

```
void displayRow(uint8_t row) {_displayRow(row);}
```

Очистити кадровий буфер

```
void clear() {memset(frameBuffer, 0, 8*N);}
```

Функція прокручування кадрового буферу на 1 піксель у вказаному напрямку

```
enum class scrollDirection
{
     scrollUp = 0,
     scrollDown,
     scrollLeft,
     scrollRight
};
void scroll( scrollDirection direction );
```

Таблиця кодування BCD Code B

```
const static uint8_t BCD_DASH =     0x0A;
const static uint8_t BCD_E =        0x0B;
const static uint8_t BCD_H =        0x0C;
const static uint8_t BCD_L =        0x0D;
const static uint8_t BCD_P =        0x0E;
const static uint8_t BCD_BLANK =    0x0F;
```

Внутрішні приватні параметри класу

```
private:
     uint8_t* _getBufferPtr(int16_t x, int16_t y) const;
     void _sendCommand(uint16_t command);
     void _displayRow(uint8_t row);

     const uint8_t N;
     SPISettings spiSettings;
     uint8_t flags;
```

```
        uint8_t* frameBuffer;
        bool selfAllocated;
        uint8_t ssPin;
};
```

Для відображення інформації в цьому проекті використовується світлодіодний матричний дисплей 8 x 8. Світлодіодні матриці доступні в різних стилях, наприклад, одноколірні, двоколірні, багатоколірні або RGB LED матриці. Вони також доступні в різних розмірах, таких як 5 x 7, 8 x 8, 16 x 16, 32 x 32 тощо.

Точкова матриця, яку ми збираємося використовувати, — це матриця 8×8, що означає, що вона має 8 стовпців і 8 рядків, тож загалом вона містить 64 світлодіоди.

Мікросхема MAX7219 полегшує керування точковою матрицею, використовуючи лише 3 цифрові контакти плати Arduino.

Рис. 2.6. Зовнішній вигляд світлодіодної матриці 8x8

Можна контролювати більше ніж одну матрицю одночасно. Для цього вам просто потрібно з’єднати їх один з одним, оскільки вони мають шпильки з обох сторін, щоб розширити точкову матрицю.

Як було зазначено раніше, ця матриця має 8 стовпців і 8 рядків. Кожен з них має індекс від 0 до 7. Ось цифра для кращого розуміння:

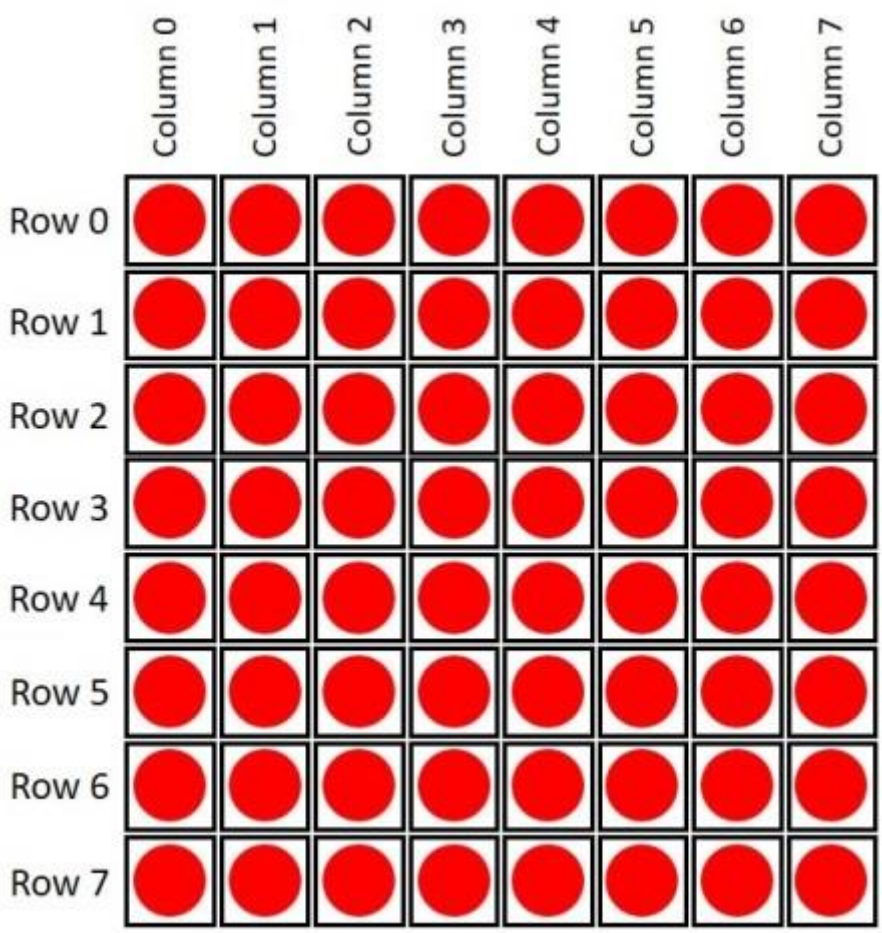

Рис. 2.7. Нумерація рядків та стовпців

Якщо необхідно відобразити щось у матриці, вам просто потрібно знати, увімкнено чи вимкнено світлодіоди в певному рядку чи стовпці.

Наприклад, якщо необхідно показати щасливе обличчя, ось що необхідно зробити:

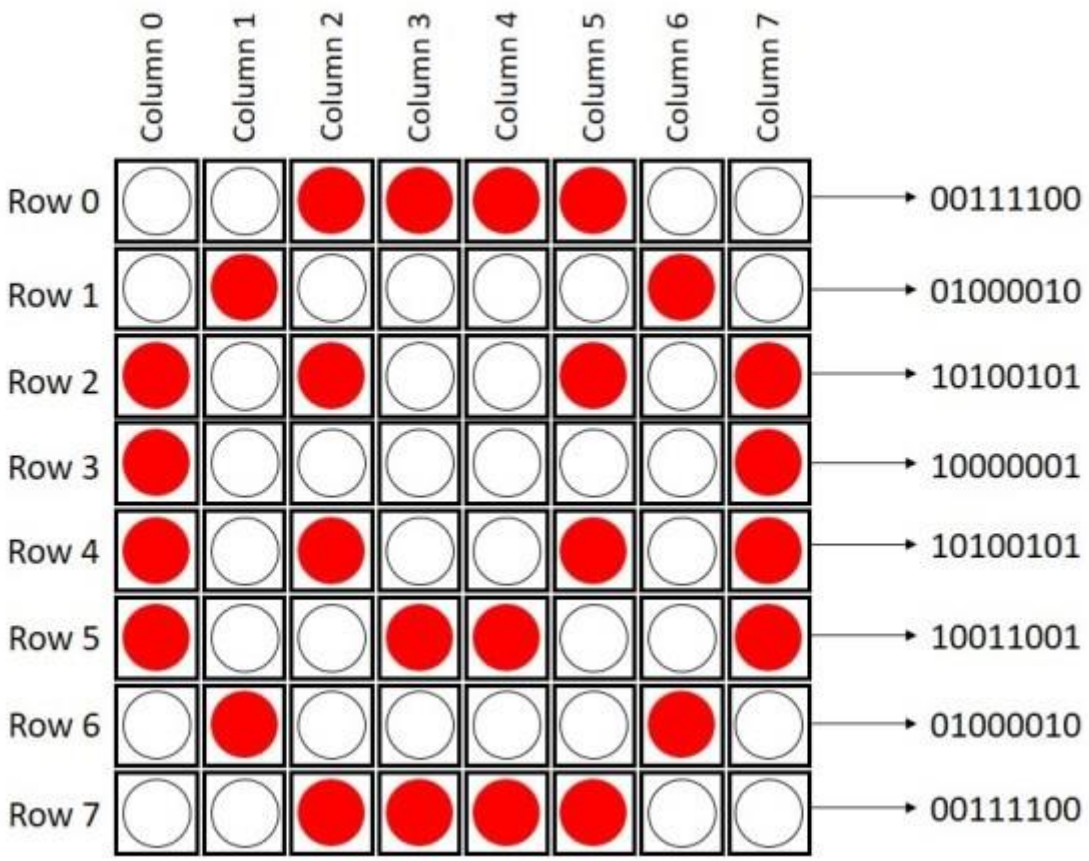

Рис. 2.8. Відображення смайліка

РОЗДІЛ 3
ОПИС ПРОГРАМИ ПРИСТРОЮ ПІДВИЩЕННЯ БЕЗПЕКИ ПІШОХОДА

3.1. Функції ініціалізації та основного циклу

Розглянемо структуру програми.

Для взаємодії з мобільним телефоном для налаштування та формування повідомлень використовується інтерфейс Bluetooth.

Для використання інтерфейсу Bluetooth необхідно підєднати файл заголовків та визначити необхідні константи та об’єкти

```
#include "BluetoothSerial.h"
#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
#endif
BluetoothSerial SerialBT;
```

Для взаємодії з драйверами світлодіодних індикаторів використовується шина SPI.

```
#include <SPI.h>
```

Методи та об’єкти для реалізації відображення зображень та їх анімація завантажуються натупним чином

```
#include "LedMatrix.h"
#include "LED_View.h"
```

Визначимо кількість драйверів світлодіодних матриць та номера виводів мікроконтролера, до яких під’єднані драйвери світлодіодних матриць.

```
#define NUMBER_OF_DEVICES 4
#define CS_PIN 15
#define CLK_PIN 14
#define MISO_PIN 12
#define MOSI_PIN 13
```

Визначимо кількість сегментів розміром 8x8 та довжину відображення

```
const int LEDMATRIX_SEGMENTS = 4;
const int LEDMATRIX_WIDTH = LEDMATRIX_SEGMENTS * 8;
```

Створення об’єкта керування світлодіодними матрицями

```
LedMatrix ledMatrix = LedMatrix(LEDMATRIX_SEGMENTS, CLK_PIN, MISO_PIN, MOSI_PIN, CS_PIN);
```

Для зручної роботи з перериваннями таймеру мікроконтролера ESP32 використаємо бібліотеку ESP32_New_TimerInterrupt. Для її корректної роботи спочатку необхідно визначити параметр _TIMERINTERRUPT_LOGLEVEL_

```
#define _TIMERINTERRUPT_LOGLEVEL_     4
#include "ESP32_New_TimerInterrupt.h"
ESP32Timer ITimer0(0);
#define TIMER0_INTERVAL_MS 1
```

Визначення акустичного фону виконується за допомогою перетворення аналогового сигналу від мікрофону за допомогою АЦП та подальшої його обробки цифровими фільтрами. Налаштування параметрів фільтрів виконується наступним чином.

```
#define PIN_AIN 34
#define PIN_INP_SW 32
#define Freq_LO 200
#define Freq_HI 400
#define Freq_a 4000
#include "Filter_FIR.h"
Filtr_FIR filtr(7);
LP_filter_t filtr_LOW1;
int16_t sqr1=0;
LP_filter_t filtr_LOW2;
int16_t sqr2=0;
```

Обробник переривань від таймеру виконує зчитування сигналу з АЦП та його цифрову фільтрацію двома фільтрами нижньої частоти.

```
bool IRAM_ATTR TimerHandler0(void * timerNo)
{
   int16_t dd;
  int16_t d = analogRead(PIN_AIN)-2048;
  dd = filtr.step(d);
  if( sqr1 == 0) sqr1=1;
  sqr1 = lpfiltering(((int32_t)d*d)/sqr1, &filtr_LOW1);
  if( sqr2 == 0) sqr2=1;
  sqr2 = lpfiltering(((int32_t)dd*dd)/sqr2, &filtr_LOW2);
  return true;
```

```
}
```

Переривання від датчику руху забезпечує запис лічильника тактів початковим значенням.

```
int volatile move_cnt = 0;
void IRAM_ATTR change_sw_isr()
{
  move_cnt = 20;
}
```

Функція setup() інфраструктури Arduino виконує налаштування обробників переривань, ініціалізацію драйверів світлодіодних матриць та інше.

```
void setup()
{
  shift_view = 0;
  move_cnt = 0;
  pinMode(PIN_INP_SW, INPUT);
  attachInterrupt(PIN_INP_SW, change_sw_isr, FALLING);
  Serial.begin(115200);
  delay(200);
  ledMatrix.init();
  ledMatrix.setIntensity(0);
  ledMatrix.clear();
  ledMatrix.commit();
  SerialBT.begin("ESP32test");  //Визначення ім’я пристрою Bluetooth
  filtr.calc_coef_LH(Freq_LO, Freq_HI, Freq_a);
  LPFILTER_INIT( 1.0/(Freq_a*1.0), 2.0, filtr_LOW1);
```

```
      LPFILTER_RESET(filtr_LOW1);
      LPFILTER_INIT( 1.0/(Freq_a*1.0), 2.0, filtr_LOW2);
      LPFILTER_RESET(filtr_LOW2);
      if  (ITimer0.attachInterruptInterval(  1000000L  /  Freq_a,
TimerHandler0))
      {
        Serial.print(F("Starting   ITimer0  OK,  millis()  =  "));
Serial.println(millis());
      }
      else
        Serial.println(F("Can't  set  ITimer0.  Select  another
freq. or timer"));
      Serial.flush();
      Led_View(0, &ledMatrix);
    }
```

Головний цикл програми виконує повільні обчислення зчитування даних з інтерфейсу Bluetooth та відображення на світлодіодних матрицях.

```
    void loop() {
      delay(100);
      Serial.println((int16_t)((sqr2*16384L)/sqr1));
      int16_t kg = (int16_t)((sqr2*16384L)/sqr1);
      if( kg > 20000)
      {
        move_cnt = 20;
      }
      if(move_cnt > 0)
      {
        char out_bt = 'H';
        SerialBT.write(out_bt);
        move_cnt--;
```

```
    shift_view++;
  }
  else
  {
    shift_view=0;
  }
  Led_View( shift_view % 16 , &ledMatrix);
}
```

3.2. Функції обчислення аналогового сигналу

Реалізація FIR фільтра виконана у класі Filtr_FIR.

```
class Filtr_FIR
{
  int N;
  int16_t Coef[KK_MAX];
  int16_t buf[2*KK_MAX+1];
  int16_t Gain;
public:
```

Клас надає інтерфейс для створення фільтра, визначення коеффіцієнтів фільтрів та обчислення одного шагу фільтру

```
  Filtr_FIR(int n){N=n;}
  float calc_coef_LOW(float Fg, float Fa, float *coef);
  void calc_coef_LH(float Fg1, float Fg2, float Fa);
  float calc_gain(float *c);
```

```
  int16_t step(int16_t v);
};
```

Метод визначення часткових коефіцієнтів фільтра

```
  float Filtr_FIR::calc_coef_LOW(float Fg, float Fa, float
*coef)
  {
    int i;
    float kk = 2*M_PI*Fg/Fa;
    for(i=1; i<N; i++)
    {
      coef[i] = (kk/M_PI)*sin(i*kk)/(i*kk);
    }
    coef[0] = (kk/M_PI);
    return calc_gain(coef);
  }
```

Метод визначення основних коефіцієнтів фільтра

```
  void Filtr_FIR::calc_coef_LH(float Fg1, float Fg2, float
Fa)
  {
    float g;
    float c1[KK_MAX], c2[KK_MAX];
    g = -calc_coef_LOW(Fg1, Fa, c1);
    g += calc_coef_LOW(Fg2, Fa, c2);
      int i;
    for(i=0; i<N; i++)
    {
      Coef[i] = (c2[i]-c1[i]) * 32767;
```

```
  }
  Gain = g * 32767;
}
```

Метод визначення загального коефіцієнта підсилення

```
float Filtr_FIR::calc_gain(float *c)
{
  float g = c[0];
  int i;
  for(i=1; i<N; i++)
  {
    g += 2*c[i];
  }
  return g;
}
```

Метод визначення розрахункового значення вихідного вигналу фільтра для поточного вхідного значення

```
int16_t Filtr_FIR::step(int16_t v)
{
  int i;
  for(i=0; i<2*N; i++)
  {
    buf[i] = buf[i+1];
  }
  buf[2*N] = v;

  int32_t d=buf[N-1];
  for(i=1; i<N; i++)
```

```
          {
            d+= (Coef[i]*((int32_t)buf[N-1 + i] + buf[N-1 - i] ) )
/ 32767;
          }
          return d;
        }
```

ВИСНОВКИ

У роботі проведено аналіз основних факторів, які впливають на безпеку пішоходів, включаючи аналіз дорожньо-транспортних пригод, психофізіологічні аспекти та інші важливі чинники. Результати цього аналізу підкреслюють необхідність розвитку ефективних систем безпеки для пішоходів у темний час.

В роботі показано, що освітленість грає ключову роль у забезпеченні безпеки пішоходів. Особливо важливим є використання технологій, які можуть покращити видимість та забезпечити пішоходам додатковий рівень захисту у темний час.

Використання IoT технологій в реалізації систем безпеки пішоходів виявилося ключовим фактором у підвищенні ефективності та функціональності таких систем. Наші дослідження підтвердили перспективи використання IoT для створення персональних систем підвищення безпеки пішоходів.

Розроблена апаратна система, спрямована на підвищення безпеки пішоходів, виявилася ефективною та надійною. Використання датчику руху, мікрофону та інших периферійних пристроїв сприяло створенню комплексної системи безпеки.

Дослідження особливостей операційної системи платформи Arduino показало її великий потенціал для створення систем безпеки, забезпечуючи швидкість та ефективність у реалізації програмного забезпечення.

Система розроблена на базі платформи Arduino та системи світлодіодних матричних дисплеїв з драйверами MAX7219.

Розроблена структура показав ефективність та легкість використання для реалізації візуальних елементів у системі безпеки.

Результати тестування свідчать про високу ефективність пристрою у підвищенні безпеки пішоходів у темний час.

РЕКОМЕНДАЦІЇ

Для підвищення ефективності системи безпеки пішоходів у темний час доби, рекомендується акцентувати увагу на використанні IoT технологій, що можуть сприяти створенню інтелектуальної системи, яка виявляє та запобігає небезпекам.

Дослідженням та розробкою систем, які забезпечують безпеку пішоходів, рекомендується надавати пріоритетне значення розумним інтерфейсам та апаратній платформі, орієнтованій на підвищення безпеки в умовах обмеженої видимості.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Arduino Documentation [Електронний ресурс]. – Режим доступу: www. URL: https://docs.arduino.cc/ (дата звернення: 10.09.23)
2. ESP32 External Interrupts using Arduino IDE [Електронний ресурс]. – Режим доступу: www. URL: https://microcontrollerslab.com/esp32-external-interrupts-tutorial-arduino-ide/ (дата звернення: 02.10.23)
3. LED DOT MATRIX BL-M07X881 [Електронний ресурс]. – Режим доступу: www. URL: https://cdn-shop.adafruit.com/datasheets/BL-M07C881.PDF (дата звернення: 15.10.23)
4. MAX7219/MAX7221 Serially Interfaced, 8-Digit LED Display Drivers [Електронний ресурс]. – Режим доступу: www. URL: https://www.analog.com/media/en/technical-documentation/data-sheets/max7219-max7221.pdf (дата звернення: 13.10.23)
5. Reference > Libraries > Max7219 [Електронний ресурс]. – Режим доступу: www. URL: https://www.arduino.cc/reference/en/libraries/max7219 (дата звернення: 21.09.23)
6. Безпека дорожнього руху: навчальний посібник / А. А. Кашканов, О. Г. Грисюк, І. І. Гуменюк. – Вінниця : ВНТУ, 2017. – 90 с.
7. Основи та методи цифрової обробки сигналів: від теорії до практики: навч. посібник / уклад. : Ю.О. Ушенко, М.С. Гавриляк, М.В. Талах, В.В. Дворжак. – Чернівці : Чернівецький нац. ун-т ім. Ю. Федьковича, 2021. 308 с.
8. Татарчук Д. Д., Діденко Ю. В. Програмування мовами С та С++: навч. посіб. / Д.Д. Татарчук, Ю.В. Діденко. – К.: , 2012. – 112 с.

ДОДАТКИ

ДОДАТОК А

Вихідний код програми ledaudio.ino

```
#include "BluetoothSerial.h"

#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
#endif

BluetoothSerial SerialBT;

#include <SPI.h>
#include "LedMatrix.h"
#include "LED_View.h"
#define NUMBER_OF_DEVICES 4 //number of led matrix connect in series
#define CS_PIN 11
#define CLK_PIN 13
#define MISO_PIN 17
#define MOSI_PIN 3
// Number of 8x8 segments you are connecting
const int LEDMATRIX_SEGMENTS = 4;
const int LEDMATRIX_WIDTH    = LEDMATRIX_SEGMENTS * 8;
LedMatrix ledMatrix = LedMatrix(LEDMATRIX_SEGMENTS, CLK_PIN, PIN, MOSI_PIN, CS_PIN);

#include "ESP32_New_TimerInterrupt.h"
ESP32Timer ITimer0(0);
#define TIMER0_INTERVAL_MS 1
#define PIN_AIN 34
#define PIN_INP_SW 32
#define Freq_LO 100
#define Freq_HI 500
```

```
#define Freq_a 6000

#include "Filter_FIR.h"

Filtr_FIR filtr(7);
LP_filter_t filtr_LOW1;
int16_t sqr1=0;
LP_filter_t filtr_LOW2;
int16_t sqr2=0;

bool IRAM_ATTR TimerHandler0(void * timerNo)
{
   int16_t dd;
  int16_t d = analogRead(PIN_AIN)-2048;
  dd = filtr.step(d);
  if( sqr1 == 0) sqr1=1;
  sqr1 = lpfiltering(((int32_t)d*d)/sqr1, &filtr_LOW1);
  if( sqr2 == 0) sqr2=1;
  sqr2 = lpfiltering(((int32_t)dd*dd)/sqr2, &filtr_LOW2);
  return true;
}

void setup()
{
  shift_view = 0;
  move_cnt = 0;
  pinMode(PIN_INP_SW, INPUT);
  attachInterrupt(PIN_INP_SW, change_sw_isr, FALLING);

  Serial.begin(115200);
  ledMatrix.init();
  ledMatrix.setIntensity(0);//0-15
  ledMatrix.clear();
  ledMatrix.commit();

  SerialBT.begin("ESP32test"); //Bluetooth device name
```

```
      filtr.calc_coef_LH(Freq_LO, Freq_HI, Freq_a);
      LPFILTER_RESET(filtr_LOW1);
      LPFILTER_RESET(filtr_LOW2);

      if (ITimer0.attachInterruptInterval( 1000000L / Freq_a,
TimerHandler0))
      {
        Serial.print(F("Starting  ITimer0 OK, millis() = "));
Serial.println(millis());
      }
      else
        Serial.println(F("Can't  set  ITimer0.  Select  another
freq. or timer"));

      Led_View(0, &ledMatrix);
    }

    void loop() {
      int16_t kg = (int16_t)((sqr2*16384L)/sqr1);
      if( kg > 20000)
        move_cnt = 20;
      if(move_cnt > 0)
      {
        char out_bt = 'H';
        SerialBT.write(out_bt);
        move_cnt--;
        shift_view++;
      }
      else
      {
        shift_view=0;
      }
      Led_View( shift_view % 16 , &ledMatrix);
    }
```

ДОДАТОК Б

Вихідний код програми Filter_FIR.cpp

```
#include <arduino.h>
#include "Filter_FIR.h"
  float Filtr_FIR::calc_coef_LOW(float Fg, float Fa, float
*coef)
  {
    int i;
    float kk = 2*M_PI*Fg/Fa;

    for(i=1; i<N; i++)
    {
      coef[i] = (kk/M_PI)*sin(i*kk)/(i*kk);
    }
    coef[0] = (kk/M_PI);
    return calc_gain(coef);
  }
  void Filtr_FIR::calc_coef_LH(float Fg1, float Fg2, float
Fa)
  {
    float g;
    float c1[KK_MAX], c2[KK_MAX];
    g = -calc_coef_LOW(Fg1, Fa, c1);
    g += calc_coef_LOW(Fg2, Fa, c2);
      int i;
    for(i=0; i<N; i++)
    {
      Coef[i] = (c2[i]-c1[i]) * 32767;
    }
    Gain = g * 32767;
  }
  float Filtr_FIR::calc_gain(float *c)
  {
    float g = c[0];
```

```
  int i;
  for(i=1; i<N; i++)
  {
    g += 2*c[i];
  }
  return g;
}

int16_t Filtr_FIR::step(int16_t v)
{
  int i;
  for(i=0; i<2*N; i++)
  {
    buf[i] = buf[i+1];
  }
  buf[2*N] = v;

  int32_t d=buf[N-1];
  for(i=1; i<N; i++)
  {
    d+= (Coef[i]*((int32_t)buf[N-1 + i] + buf[N-1 - i] ) )
/ 32767;
  }
  return d;
}
```