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

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних систем та мереж
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: *Комп'ютеризована система виявлення та аналізу тепловтрат будівлі*

Виконав: студент *IV* курсу групи *СІ-42*
спеціальності *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

		Семців Т. А.
	(підпис)	(прізвище та ініціали)
Керівник		*Осухівська Г. М.*
	(підпис)	(прізвище та ініціали)
Нормоконтроль		*Тиш Є. В.*
	(підпис)	(прізвище та ініціали)
Завідувач кафедри		*Осухівська Г. М*
	(підпис)	(прізвище та ініціали)
Рецензент		Млинко Б. Б.
	(підпис)	(прізвище та ініціали)

Тернопіль
2024

Факультет *комп'ютерно-інформаційних систем і програмної інженерії*
(повна назва факультету)

Кафедра *комп'ютерних систем та мереж*
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
Осухівська Г.М.
(підпис) (прізвище та ініціали)
« 25 » 04 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня *бакалавр*
(назва освітнього ступеня)

за спеціальністю *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

студенту *Семціву Тарасу Андрійовичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Комп'ютеризована система виявлення та аналізу тепловтрат будівлі*

Керівник роботи *Осухівська Галина Михайлівна, к.т.н., доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 24 » 04 2024 року № 4/7-408

2. Термін подання студентом завершеної роботи *14.06.2024*

3. Вихідні дані до роботи *Технічне завдання*

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ
Аналіз технічного завдання
Проєктна частина
Практична частина
Безпека життєдіяльності, основи охорони праці
Висновок

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Структурна схема системи
Схема електрична принципова
Блок-схема алгоритму роботи програмного забезпечення
Результати тестування

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності,	д.т.н., професор Пилипець М. І.		
основи охорони праці			

7. Дата видачі завдання *25.04.2024*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	*Розробка технічного завдання*	*01.02 – 09.02*	*Виконано*
2	*Аналіз технічного завдання*	*05.02 – 11.02*	*Виконано*
3	*Аналіз технологій побудови систем виявлення тепловтрат*	*25.04 – 03.05*	*Виконано*
4	*Обґрунтування вибору апаратного забезпечення*	*04.05 – 15.05*	*Виконано*
5	*Реалізація програмного забезпечення для системи*	*16.05 – 01.06*	*Виконано*
6	*Реалізація апаратного забезпечення*	*02.06 – 09.06*	*Виконано*
7	*Безпека життєдіяльності, основи охорони праці*	*10.06 – 15.06*	*Виконано*
8	*Оформлення кваліфікаційної роботи*	*16.06 – 20.06*	*Виконано*
9	*Попередній захист кваліфікаційної роботи*	*14.06*	*Виконано*
10	*Захист кваліфікаційної роботи*	*24.06 – 28.06*	

Студент _____ *Семців Т. А.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Осухівська Г. М.*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Комп’ютеризована система виявлення та аналізу тепловтрат будівлі // Кваліфікаційна робота на здобуття освітнього ступеня бакалавр // Семців Тарас Андрійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп’ютерно-інформаційних систем і програмної інженерії, група СІ-42 // Тернопіль, 2024 // с. – 81, рис – 32, аркушів А1 – 4, додат. – 3, бібліогр – 22.

Ключові слова: КОМП’ЮТЕРИЗОВАНА СИСТЕМА, МІКРОКОНТРОЛЕР, ТЕПЛОВІЗОР, КАМЕРА, ДИСПЛЕЙ, ТЕМПЕРАТУРА, БУДІВЛЯ.

Кваліфікаційна робота присвячена розробці комп’ютеризованої системи виявлення та аналізу тепловтрат будівлі, яка здатна виявляти аномальні зони в конструкції будівлі. Пояснювальна записка складається з чотирьох розділів.

У першому розділі проведено аналіз вимог до проектованої комп’ютеризованої системи виявлення та аналізу тепловтрат будівлі. Обґрунтовано актуальність теми та сфери використання даної системи.

У другому розділі було розроблено архітектуру комп’ютеризованої системи. Обґрунтовано вибір апаратної складової та шин передачі даних.

У третьому розділі описано реалізацію програмних функції та апаратної будови комп’ютеризованої системи виявлення та аналізу тепловтрат будівлі. Також проведено тестування системи.

В четвертому розділі розглянуті питання безпеки життєдіяльності та охорони праці.

ANOTATION

A computerized system for detecting and analyzing building heat losses // Bachelor's thesis // Semtsiv Taras Andriiovych // Ternopil Ivan Puluj National Technical University // Faculty of Computer Information Systems and Software Engineering, group SI-42 // Ternopil, 2024 // p. – 81 , fig – 32, sheets A1 – 4, supplement – 3 , Bibliog. – 22.

Keywords: COMPUTERIZED SYSTEM, MICROCONTROLLER, THERMAL IMAGE, CAMERA, DISPLAY, TEMPERATURE, BUILDING.

The qualification work is devoted to the development of a computerized system for the detection and analysis of building heat losses, which is capable of detecting abnormal zones in the building structure. The explanatory note consists of four sections.

In the first chapter, an analysis of the requirements for the designed computerized system of detection and analysis of building heat loss was carried out. The relevance of the topic and the scope of use of this system is substantiated.

In the second chapter, the architecture of the computerized system was developed. The choice of hardware components and data transmission buses is justified.

The third chapter describes the implementation of software functions and the hardware structure of the computerized system for detecting and analyzing building heat losses. The system was also tested.

The fourth chapter deals with issues of life safety and occupational health and safety.

ЗМІСТ

СПИСОК СКОРОЧЕНЬ

КСВАТБ – Комп’ютеризована система виявлення та аналізу тепловтрат будівлі

ПК – Персональний комп’ютер

I^2C – Inter Integer Circuit

UART – Universal Asynchronous Receiver-Transmitter

SPI – Serial Peripheral Interface

USB – Universal Serial Bus

ВСТУП

Сучасні тенденції енергозбереження та підвищення енергоефективності будівель диктують необхідність розробки новітніх технологій, які дозволяють знизити витрати на опалення та забезпечити комфортні умови проживання чи роботи. Ключовою проблемою цієї сфери є тепловтрати, які призводять до значних витрат енергії та фінансових ресурсів. Тому виявлення та аналіз тепловтрат будівлі стають важливими завданнями для інженерів та власників нерухомості.

У кваліфікаційній роботі розглянуто розробку та впровадження комп'ютеризованої системи для виявлення та аналізу тепловтрат будівлі. Система використовує сучасні технології тепловізійної зйомки, що дозволяє отримувати точні результати щодо стану теплової ізоляції будівлі.

Метою кваліфікаційної роботи є розроблення ефективної комп'ютеризованої системи, яка дозволить знизити тепловтрати будівель за рахунок точного виявлення проблемних зон. Досягнення цієї мети сприятиме підвищенню енергоефективності будівель та зменшенню витрат на опалення, що є актуальним завданням у контексті сучасних вимог до енергозбереження та екологічної безпеки.

Для досягнення мети буде виконано наступні завдання:

– провести аналіз наявних систем виявлення тепловтрат будівлі для ефективної реалізації власної;

– створити електричну схему для системи виявлення та аналізу тепловтрат будівлі;

– розробити алгоритм роботи та програмне забезпечення для системи;

– провести налаштування та тестування розробленої системи.

Розробка комп'ютеризованої системи виявлення та аналізу тепловтрат будівель має значний потенціал для зниження енергоспоживання та

покращення умов проживання. Використання таких систем може стати важливим кроком у напрямку сталого розвитку та збереження енергетичних ресурсів, що є критично важливим у сучасних умовах.

Використання цієї системи дозволить не лише знизити витрати на опалення, але й підвищити загальний рівень енергоефективності будівель, що є важливим фактором у досягненні сталого розвитку та економічної стабільності.

РОЗДІЛ 1. АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

1.1 Основні технічні вимоги до комп’ютеризованої системи виявлення та аналізу тепловтрат будівлі

Комп’ютеризована система виявлення та аналізу тепловтрат будівлі (КСВАТБ) призначена для виявлення місць в будівлі, де присутні тепловтрати.

Основні вимоги до комп’ютеризованої системи виявлення та аналізу тепловтрат будівлі включають в себе: виявлення тепловтрат, візуалізацію зйомки, точність, швидкість оновлення даних на дисплеї, гнучкість та налаштованість, доступний інтерфейс.

Система повинна бути готовою до використання в різних умовах, включаючи змінні погодні умови та освітлення, щоб забезпечити надійний та точний збір даних у будь-який час доби та в різних умовах експлуатації. Важливою вимогою є можливість візуального відображення результатів аналізу тепловтрат будівлі. Це може бути реалізовано через графічний інтерфейс, за допомогою відображення зображень, де показані зони з тепловтратами. Зони тепловтрат відображаються різними кольорами або за допомогою інших візуальних маркерів для визначення пріоритетних областей.

КСВАТБ повинна бути здатна виявляти аномальні області, які можуть вказувати на потенційні дефекти або прогалини в тепловому ізоляційному покритті будівлі. Це може включати виявлення дефектів у теплоізоляційних матеріалах або прорізів у конструкції будівлі.

КСВАТБ повинна надати можливість користувачам налаштовувати параметри аналізу відповідно до конкретних потреб та умов використання. Це може включати встановлення порогових значень температури для визначення зон тепловтрат та інші параметри, які впливають на точність та чутливість аналізу.

Необхідно передбачити, що КСВАТБ повинна працювати швидко та ефективно для забезпечення оперативного виявлення тепловтрат. Це включає здатність обробки даних в реальному часі та наданні результатів аналізу без затримки.

КСВАТБ повинна бути стабільною та надійною в роботі, незалежно від умов експлуатації. Зокрема, забезпечення роботи від акумулятора до 5 годин. Також потрібно передбачити можливість виправлення помилок в роботі системи.

Основні вимоги до розроблюваної комп'ютеризованої системи виявлення та аналізу тепловтрат будівлі визначають конструкцію та склад і повинні включати:

– Діапазон вимірювання температури: -40℃ – +400℃.
– Діапазон робочих температур: -30℃ – +45℃.
– Поле зору давача (FoV): не менше 45°x30°.
– Частота оновлення: 4Гц–32Гц.
– Кольорові режими роботи:

1) синій, блакитний,зелений,жовтий, червоний, пурпуровий;
2) чорний, темно-сірий, світло-сірий, білий, помаранчевий, рожевий;
3) темно-синій, оливковий, зелений, блакитний, бордовий, фіолетовий;
4) синій, зелений, темно-зелений, помаранчевий, бордовий, червоний;
5) темно-синій, темно-зелений, зелений, жовтий, помаранчевий, червоний
6) блакитний, синій, пурпуровий, жовтий, зелений, червоний;
7) білий, помаранчевий, червоний, синій, зелений, чорний;
8) фіолетовий, пурпуровий, червоний, помаранчевий, жовтий, зелений;
9) жовтий, рожевий, білий, синій, темно-блакитний, темно-зелений;
10) червоний, жовтий, зелений, блакитний, синій, пурпуровий.

– Простий інтерфейс.
– Можливість зробити та зберегти фото.
– Автономність роботи (до 5год.).
– Розміри, не більше 10x10см.

1.2 Аналіз існуючих рішень

1.2.1 Методи виявлення тепловтрат будівлі

Існує кілька способів визначення тепловтрат будівлі, але найпопулярнішими з них є технологія перевірки повітропроникності або герметичності будівлі (Blower-door test) та тепловізійний метод.

Метод Blower-door test проводиться за допомогою спеціального пристрою під назвою манометричний протягошукач або аеродвері (рис. 1.1) [1].

Рисунок 1.1 – Манометричний протягошукач

Даний прилад передбачає виявлення тепловтрат будівлі шляхом нагнітання або розрідження повітря в будівлі, щоб створити різницю тисків.

Перед тестом потрібно провести комплекс підготовчихих процедур. Для початку потрібно встановити вентилятор аеродверей в дверний отвір вхідних дверей та закрити всі інші зовнішні двері та вікна, а внутрішні – відкрити. Також необхідно щільно закрити всі вентиляційні отвори, канали витяжних вентиляторів та всі інші отвори в будівлі. Далі, для того щоб чутливі до перепаду тиску пристрої не вийшли з ладу, їх потрібно вимкнути. Також потрібно зробити заміри температури ззовні та всередині будинку, а також - тиску ззовні будинку. Якщо різниця температур складає більше 30℃ чи швидкість вітру більша ніж 8м/с, то випробування не проводиться.

Процес тестування починається з того, що включається вентилятор, який нагнітає повітря в середину, або розріджує його в середині. Вентилятор поступово набирає обертів до моменту, коли різниця зовнішнього та внутрішнього тисків сягає 50Па. Після цього спеціалісти починають записувати показники.

Герметичність будинку вимірюється такими показниками як повітрообмін та його кратність при передачі тиску в 50Па.

Другий спосіб здійснюється за допомогою методу тепловізійного обстеження [2]. Цей метод доцільніше використовувати в холодну погоду, оскільки, коли ззовні будинку тепло, то і «містки холоду» не буде виявлено. Для цього способу потрібно дешевше обладнання, ніж для попереднього і дослідження проводиться простіше.

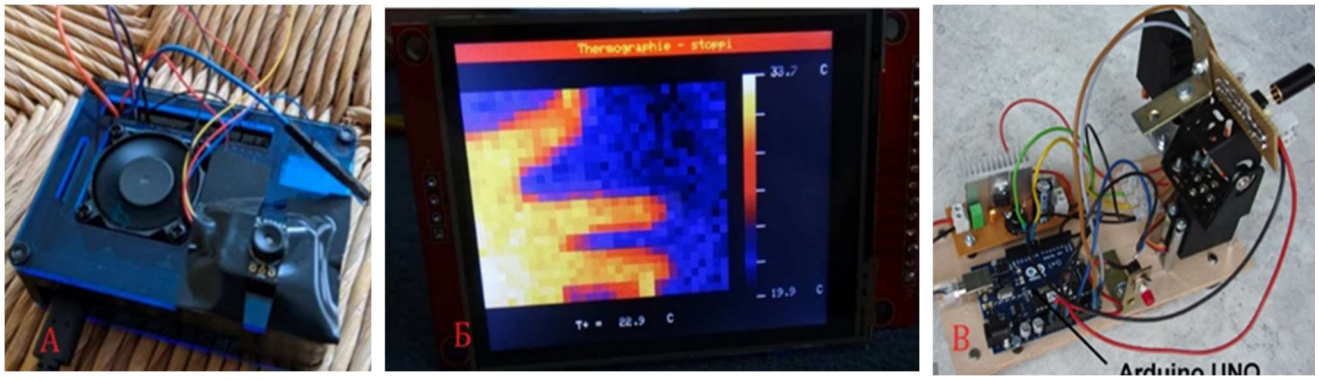
Для повної перевірки потрібно направити тепловізор на кожен кут в будівлі, там найчастіше є «містки холоду». Також не потрібно пропускати стіни, так як часто в них можуть бути тріщини чи інші механічні пошкодження.

Детальніше обстеження потрібно виконувати для вікон, оскільки це найслабша частина будинку, тому їх потрібно обстежувати найбільш ретельно. Тепловізор потрібно направляти на всі стики між стінами, а особливо важливо перевіряти місця, де склопакет стикується з рамою. Адже там часто може бути неякісний ущільнювач.

Тепловізор також дозволяє перевіряти проблеми з опалювальною системою. Наприклад, часто є проблема з розгерметизацією батареї, направивши тепловізор на них, буде видно, де присутні «повітряні пробки». Система також може відслідкувати проблеми з трубами передачі гарячої води, вони можуть бути забиті, система також може побачити це.

1.2.2 Існуючі системи виявлення та аналізу тепловтрат будівлі

КСВАТБ може бути реалізована на основі різних мікроконтролерних та мікрокомп’ютерних платформ, наприклад, Arduino, ESP32 чи Raspberry Pi[3-5]. На рис. 1.2 показано загальний вигляд тепловізійних систем розроблених на основі Raspbery Pi (а), ESP32 (б), Arduino (в).



а) б) в)

Рисунок 1.2 – Тепловізори на різних мікроконтролерах. а) – Raspberry Pi; б) – ESP32; в) - Arduino

На рис. 1.2 (а) зображена тепловізійна система на основі мікрокомп’ютера Raspberry Pi, розроблена Томом Шаффнером [3]. Тепловізійна камера, яка використовується в даній системі має назву MLX90640. В цій тепловізійній системі не передбачено екрану, а зображення передається через Wi-Fi та транслюється на монітор комп’ютера. В системі передбачено можливість збереження фото на SD-карту. Для охолодження процесора в цій системі використовується вентилятор, що створює додатковий шум. Недоліком цієї системи є її реалізація на Raspberry Pi, що здорожчує розробку.

На рис. 1.2 (б) показана система на основі мікроконтролера ESP32-WROOM [4]. Вона складається з 3ох компонентів: мікроконтролера, камери та дисплея. Камерою в пристрої виступає MLX90640, яка в даній системі використана у вигляді модуля. В якості екрану використано модуль диисплея ILI9341, з сенсорним екраном та слотом для MicroSD карти. Наявність слота дозволяє вставити в нього саму карту та зберігати фото теплового зображення. Недоліком пристрою є те, що він немає вмонтованої батареї, а живиться від ПК через USB-кабель, що робить його не таким зручним для використання. Ціна розробки приблизно 80-90 доларів США.

На рис. 1.2 (в) зображено тепловізійну систему, розроблену на основі Arduino Uno [5]. В даному пристрої використовується камера MLX90614 з розширенням 16x12 пікселів. Дисплей на цьому пристрої не передбачений, а

теплове зображення формується натисненням кнопки, після чого мікроконтролер робить фото та відправляє дані на комп'ютер через кабель. Це є вагомим недліком, через те, що пристрій постійно має бути підключеним для комп'ютера. Для візуалізації надісланих даних використовується програмне забезпечення gnuplot, яке має бути встановленим на комп'ютер. Перевагою системи є використання двох сервоприводів, що дозволяє наводити камеру на потрібне місце.

Як видно з описів, в більшості пристроїв використовується камера MLX90640, це означає що даний давач є найкращим варіантом для розробок на різних мікроконтролерах. Ціна тепловізорних систем на основі мікроконтролерів та мікрокомп'ютерів сягає від 40 до 150 доларів США, що значно дешевше професійних рішень.

Професійні системи для виявлення та аналізу тепловтрат будівлі показано на рис. 1.3.

а) б) в) г)

Рисунок 1.3 – Системи виявлення тепловтрат будівлі: а) Wintact WT3160; б) Xintest HT-A2; в) FLIR ONE PRO; г) Testo 868

На рис. 1.3 (а) зображено будівельний тепловізор від компанії Wintact, моделі WT3160 [6]. Цей тепловізор має діапазон вимірювань від -20℃ до +450℃. Дисплей WT3160 має розширення 2.8 дюймів та частоту оновлення 9Гц. Роздільна здатність тепловізійної камери - 160x120 пікселів, а теплова чутливість - 0.07 °К, спектральний діапазон - 8-14мкм. Фокус в камери фіксований, з фокусною відстанню 0.5метра. Тепловізор має можливість збереження знімків в форматі JPG, знімки можна робити в 5 різних палітрах кольорів. На внутрішньому накопичувачі тепловізора можна зберегти більш ніж 20 тисяч знімків. Живлення на тепловізор

подається з акумуляторів формату 18650, а заряджається пристрій через інтерфейс USB. Робочі температури під час яких тепловізор може коректно працювати рівні від 0℃ до +45℃, тепловізор дозволено зберігати при температурі від -20℃ до +60℃. Максимальна вологість повітря, яка допускається для роботи пристрою рівна 80%. Вага тепловізора рівна 335 грам, а габарити 225x93x69 міліметрів. Ціна даної моделі рівна приблизно 25 500 грн.

Тепловізор від компанії Xintest моделі HT-A2 [6] має діапазон вимірювання від -20℃ до +300℃. Дисплей тепловізора має розширення 320x240 дюймів та частоту оновлення 9Гц. Роздільна здатність камери - 320x240 пікселів, чутливість - 0.07 °К. Камера має спектральний діапазон від 8 до 14мкм. Фокус камери є фіксований та має мінімальну фокусну відстань 0.15м. Знімки зберігаються в внутрішню пам’ять пристрою обсягом 3ГБ. Живиться тепловізор від вбудованого акумулятора, який заряджається через інтерфейс USB. Робочі температури під час яких тепловізор може коректно працювати варіюються в діапазоні від 0℃ до +45℃, а зберігати тепловізор можна при температурі від -5℃ до +50℃. Максимальна вологість повітря, яка допускається для роботи пристрою рівна 80%. Вага тепловізора - 208 грам, а габарити - 140x80x28 міліметрів. Ціна даної моделі рівна приблизно 26000грн.

Тепловізор від компанії Flir моделі One Pro здатен вимірювати температурні показники в діапазоні від -20 до +400℃ [6]. Тепловізор під’єднується до смартфона через USB-роз’єм та передає зображення тепловтрат у спеціальну програму, оскільки власного дисплея в тепловізорі не передбачено. Тепловізійна камера має роздільну здатність 160x120 пікселів, та чутливість 0,15°К. Її спектральний діапазон може мати значення від 8 до 14 мкм. Фокус камери є фіксованим, а його мінімальна відстань рівна 0.15м. Знімки можна робити у програмі на смартфоні та зберігати їх у його внутрішню пам’ять, їх кількість обмежується тільки пам’яттю смартфона. Живиться тепловізор від смартфона через USB-роз’єм. Тепловізор може виконувати свої функції при температурі навколишнього середовища від 0℃ до +35℃, температура його зберігання від -20℃ до +60℃. Вологість повинна бути не більшою 80%. Вага тепловізора 36.5г, а розміри 68x34x14мм.

Професійний будівельний тепловізор від компанії Testo моделі 868 [7] здатен проводити виявлення та аналіз тепловтрат будівлі в двох діапазонах вимірювання, які можна переключати. Перший діапазон може вимірювати температуру від -30℃ до +100℃, а другий - від 0℃ до +650℃. Дисплей тепловізора має розширення 320x240 дюймів та частоту оновлення 9Гц.Роздільна здатність камери - 160x120 пікселів та чутливість - 0.08°К. Фокус камери є фіксований та має мінімальну фокусну відстань 0.1м. Знімки зберігаються у внутрішню пам'ять пристрою, яка має обсяг 2.8ГБ. Живиться тепловізор від вбудованого акумулятора, який заряджається через інтерфейс USB. Тепловізор може працювати при температурі від -15℃ до +50℃, температура його зберігання від -20℃ до +65℃. Вологість повинна бути не меншою 80%. Тепловізор має вбудований Bluetooth та може обмінюватися даними зі смартфоном, також при використанні цього тепловізора потрібен спеціальний додаток Thermography. Система може автоматично розпізнавати найбільш гарячу та найбільш холодну точки, також може розраховувати різниці температур. Розміри тепловізора рівні 219x96x95мм, а вага - 510грам. Ціна такого пристрою рівна приблизно 110 000 гривень.

Таблиця 1.1 – Порівняння тепловізорів

Модель	Wintact WT3160	Hintest HT-A2	Flir One Pro	Testo 868
Діапазон вимірювання	-20 до +450	-20 до 400	-20 до 400	-30 до +100 0 до +600
Роздільна здатність камери	160x120	320x240	160x120	160x120
Розширення екрану	320x240	320x240	–	320x240
Вбудована пам'ять	2 ГБ	3 ГБ	–	2.8 ГБ
Габарити	225x93x69	68x34x14	140x80x28	219x96x95
Ціна	25 500	26 000	16 000	110 000

Як видно з таблиці 1.1, більшість тепловізорів є схожими за основними характеристиками, проте, чим дорожчий тепловізор, тим більше різних додаткових функцій вони мають.

1.3 Аналіз можливих рішень

Для розробки комп'ютеризованої системи виявлення та аналізу тепловтрат будівлі буде використано кілька основних компонентів, це дисплей, камера та мікроконтролер.

Запропоновано використання мікроконтролера від компанії Espressif Systems під назвою ESP32-Wrover-E, оскільки він передбачає додаткових 16Мб Flash-пам'яті. Не менш важливим аспектом в виборі мікроконтролера був і його розмір, а дана плата розміром всього 31х18мм, дуже добре підходить для невеликої системи. Для можливого майбутнього розширення можливостей системи мікроконтролер має вбудовані Wi-Fi та Bluetooth модулі.

Під час обирання камери було розглянуто кілька камер від різних виробників, а саме AMG8833, MLX90640 та MLX90641 [8]. Ціни на них приблизно однакові, але все ж найдешевшою з них є AMG8833, вона має роздільну здатність лише 8х8, тоді як не сильно дорожча MLX90640 – 32х24. Остання камера MLX90641 має роздільну здатність 16х12. Тому, так як ця камера має найбільшу роздільну здатність в ціновому діапазоні, було вибрано її. Камера добре підходить для розробки тим, що її під'єднання здійснюється через протокол I^2C, для чого потрібно тільки 2 виводи, та виводи під живлення та землю.

Для візуалізації зображення тепловтрат запропоновано використання TFT-дисплею ILI9341, з розширенням 320х240 пікселів та шириною екрану в 2.4 дюйми. Дисплей є кольоровим з кількістю кольорів 262 тисячі в палітрі RGB. Він є якісним для своєї ціни у 8$ США. Підключатиметься дисплей через інтерфейс SPI. Даний дисплей буває як і з сенсорним екраном, так і без нього. В розроблюваній системі буде використовуватися версія без сенсорного екрану.

Також комп’ютеризована система виявлення та аналізу тепловтрат будівлі буде мати можливість збереження даних на MicroSD-карту. Джерелом живлення буде виступати літій-полімерний акумулятор на 300 мА. Для його заряджання буде використовуватися роз’єм USB-C з мікросхемою-контролером заряду.

Загальна ціна системи буде складати не більше 4 000 тисяч гривень, що в порівнянні з професійними варіантами є досить дешево.

РОЗДІЛ 2. ПРОЄКТНА ЧАСТИНА

2.1 Розробка структури комп'ютеризованої системи виявлення та аналізу тепловтрат будівлі

Відповідно до технічного завдання, система виявлення та аналізу тепловтрат будівлі повинна відповідати таким основним вимогам: діапазон вимірювання температури:-40℃ – +400℃, діапазон робочих температур:-30℃ – +45℃, поле зору давача (FoV): не менше 45°x30°, частота оновлення: 4Гц–32Гц, 10 кольорових режимів роботи, простий інтерфейс, можливість зробити та зберегти фото, автономність роботи (до 5 годин), розміри не більше 10x10см.

Ці вимоги розроблюваної комп'ютеризованої системи виявлення та аналізу тепловтрат будівлі визначають структуру системи, яка складається з камери MLX90640 з інфрачервоним тепловим давачем, дисплея ILI9341, SD-карти для збереження фото, які можуть бути зроблені, та мікроконтролера ESP32. Також пристрій має бути автономним, тому в конструкції повинен бути акумулятор.

За цими вимогами було розроблено структурну схему для комп'ютеризованої системи виявлення та аналізу тепловтрат будівлі (рис. 2.1).

Структурна схема комп'ютеризованої системи виявлення та аналізу тепловтрат будівлі побудована таким чином, щоб задовольнити вимоги, що зазначені в технічному завданні. Всі компоненти є загальнодоступними та бюджетними.

Комп'ютеризована система виявлення та аналізу тепловтрат будівлі включає в себе такі основні блоки:

– Камера з інфрачервоним тепловим давачем MLX90640.
– Плата з мікроконтролером ESP32-WROVER-E.
– SD-карта з файловою системою FAT32.

– TFT-дисплей ILI9341.
– Автономне джерело живлення.
– TFT-дисплей ILI9341.
– Автономне джерело живлення.

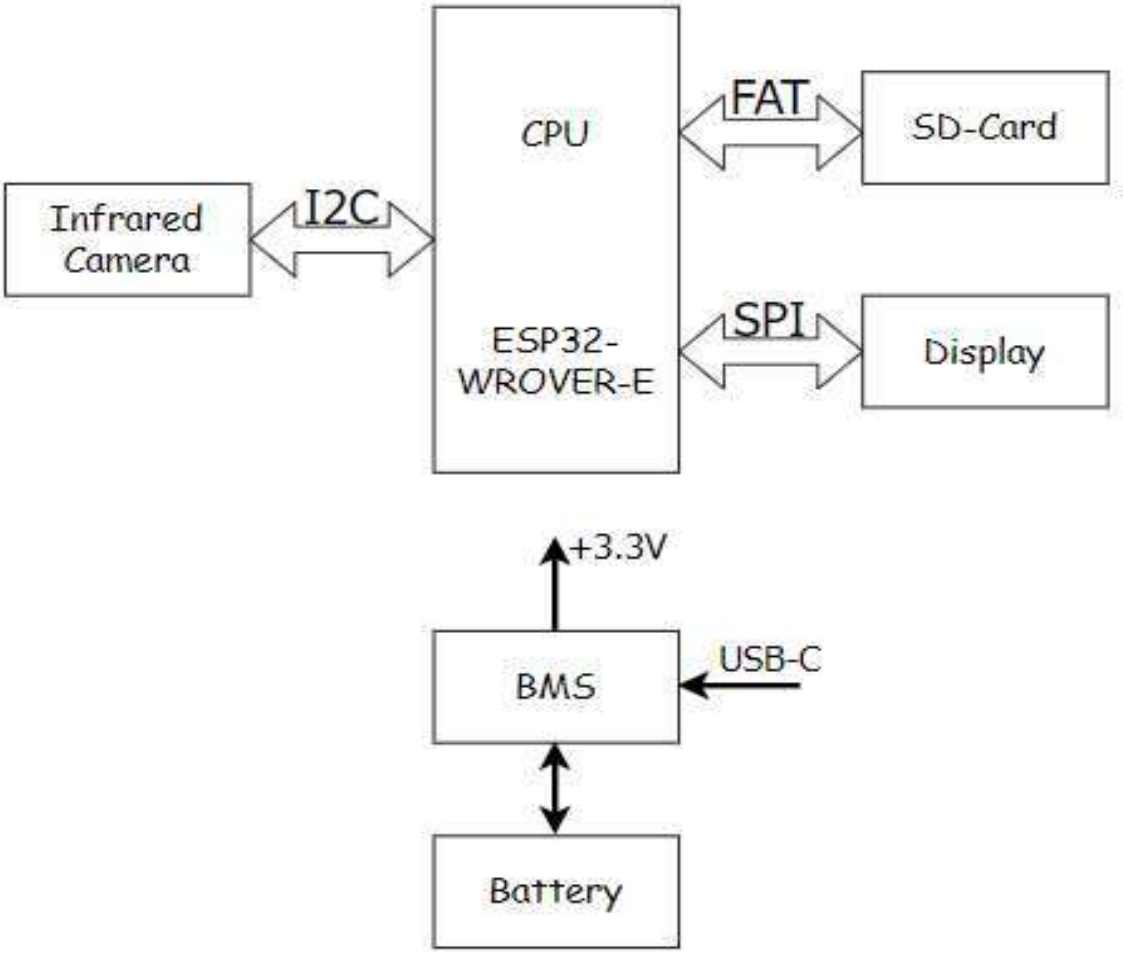


Рисунок 2.1 – Структурна схема комп’ютеризованої системи виявлення та аналізу тепловтрат будівлі

КСВАТБ функціонує таким чином: камера з інфрачервоним сенсорами обмінюється даними з мікроконтролером ESP32-WROVER-E через шину даних I^2C, після чого дані пересилаються на дисплей, для цього застосовується інтерфейс SPI. Коли потрібно зробити знімок, мікроконтролер перевіряє чи наявна SD-карта, яка прошита під файлову систему FAT32. Після натиснення кнопки, фото, зроблене з екрану, відправляється на карту пам’яті, також, при необхідності, можна переглянути це фото через меню пристрою. Зарядка акумулятора відбувається

через порт USB-C, контролює рівень заряду система BMS (система управління батареєю). Живлення мікроконтролера здійснюється напругою 3.3В.

2.2 Обґрунтування вибору апаратного забезпечення проектованої системи

2.2.1 Опис основних компонентів

Для розроблюваної КСВАТБ було вибрано плату з мікроконтролером ESP32-WROVER-E [9] (рис. 2.2).

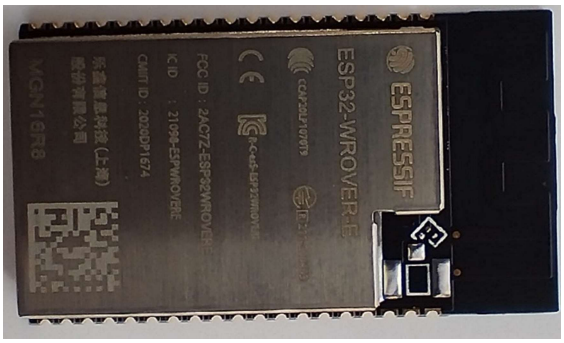


Рисунок 2.2 – Плата ESP32-WROVER-E

Мікроконтролером в ESP32-WROVER-E виступає ESP32-DOWD-V3, це 32-бітний MCU з підтримкою Wi-Fi та Bluetooth. В середині наявні два процесорні ядра XTensa LX6 з регульованою тактовою частотою від 80 до 240МГц. Wi-Fi та Bluetooth працюють на частоті 2.4ГГц. Wi-Fi працює за стандартами 802.11 b/g/n та має підтримку HT40 зі швидкістю до 150 Мбіт/с. Також є підтримка мережевих протоколів TCP/IP. Технологія Bluetooth в цьому мікроконтролері підтримує застарілі типи з’єднань, такі як L2CAP, GAP, A2DP та інші. Підтримуються профілі Bluetooth LE, які для підключення передають маячки з низьким енергоспоживанням для легкого виявлення. Всередині мікроконтролера є вбудований давач Холла.

Мікроконтролер має 34 програмованих GPIO, 12-біний АЦП на 18 каналів, два 8-бітних ЦАП, 10 ємнісних сенсорних входів IOs. Мікроконтролер має один хост-контролер, який підтримує інтерфейси SD, eMMC, SDIO, також наявний один

інтерфейс в режимі slave, який підтримує інтерфейси SDIO та SPI. Мікроконтролер підтримує такі інтерфейси:

- 4 x SPI.
- 2 x I2C, 2 x I2S.
- 3 x UART.
- ADC/DAC.
- Motor PWM та LED PWM на 16 каналів.
- Ethernet з виділеною пам’яттю DMA.
- CAN 2.0.
- JTAG для розробки та тестування.

ESP32-DOWD-V3 має 448КБ ROM, 520КБ SRAM та 16КБ SRAM для RTC, проте вбудованої пам’яті Flash на чіпі немає.

Чіп має криптографічні апаратні прискорювачі для AES, SHA-2, ECC та генератора випадкових чисел RNG. Також є вбудований пристрій цифрового підпису для зберігання особистих ключів та генерації підписів RSA.

Живитися чіп ESP32-DOWD-V3 може від напруги в діапазоні від 2.3 до 3.6В, а в режимі сну споживання складає менше 5мкА, що дозволить використовувати його пристроями з тривалим терміном автономного функціонування.

Крім мікроконтролера на платі ESP32-WROVER-E присутні мікросхеми XMC 25QH128 та ESP PSRAM64H. XMC 25QH128 є мікросхемою flash-пам’яті об’ємом 16МБ. Дана мікросхема компенсує відсутність flash-пам’яті на самому мікроконтролері. ESP PSRAM64H це мікросхема псевдо-статичної оперативної пам’яті об’ємом 8МБ, яка забезпечує розширення SRAM яка вбудована в мікроконтролер. Живиться мікросхема від напруги в 1.8В.

Плата ESP32-WROVER-E має розміри 31.3x18мм. Малі розміри дозволяють використовувати плату в невеликих за розміром пристроїв.

Другим основним компонентом пристрою є камера з інфрачервоним тепловим сенсором MLX90640 (рис. 2.3).

В камері MLX90640 використовується матриця тепловізійних інфрачервоних давачів з розширенням 32x24 пікселів. Підключається камера через інтерфейс I^2C, використовуючи адресу 0x33, швидкість передавання даних досягає 1МГц.

Рисунок 2.3 – Камера з тепловим сенсором MLX90640

Діапазон вимірюваної температури камерою сягає від -40°С до приблизно 85°С. Кут огляду камери - 55°x35°, а частота оновлення - 0.5-64Гц.

Зображення з камери буде виводитися на TFT-дисплей ILI9341 [11] (рис. 2.4) в режимі реального часу.

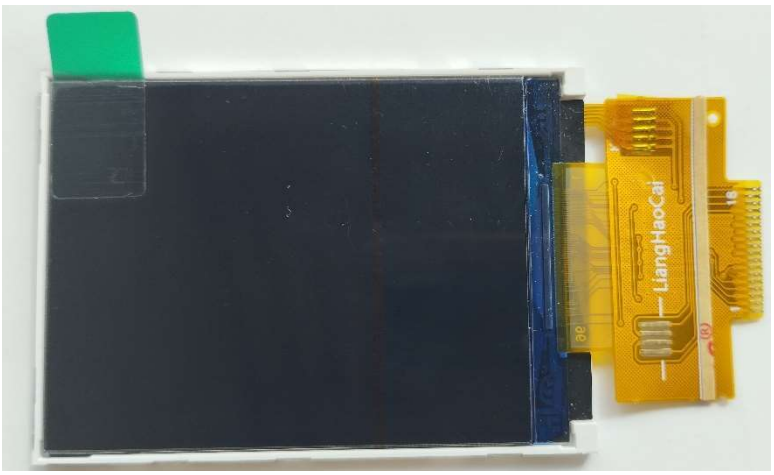


Рисунок 2.4 – TFT-дисплей ILI9341

Даний дисплей має роздільну здатність 240x320px та розмір 2.4 дюйми (дисплей також може мати 2.2,2.8, 3.2 дюйми), що дозволяє його використовувати в невеликих пристроях з досить чіткою картинкою. В дисплеї використовується

кольорова матриця типу TFTLCD, що дає змогу відтворювати до 262 тисяч кольорів палітри RGB.

Дисплей використовується без сенсорного екрану, хоча є версія з сенсорним екраном. Підсвітка дисплею біла з можливістю керування яскравістю за допомогою ШІМ.

Дисплей має шлейф, який складається з 18 контактів для під'єднання до плати з мікроконтролером. Під'єднання здійснюється через шину SPI, для цього призначені контакти 3-SCL, 5-CS, 6-SDA(MISO), 7-SDO(MOSI). Живиться дисплей від напруги 3.3В, тому при підключенні до плат, які мають живлення 5В, потрібно під'єднювати дисплей через пристрій для пониження напруги, по іншому дисплей може вийти з ладу. Струм, від якого працює даний дисплей, рівний 300мА.

2.2.2 Опис вузлів комп'ютеризованої системи виявлення та аналізу тепловтрат будівлі

Повна електрична принципова схема системи наведена у графічних матеріалах, на креслені КС КРБ 123.131.00.00 Е3. На рис. 2.5 показано вузол схеми для зарядки та прошивки системи [12]. На рис. 2.5 (а) показано USB-порт типу С, він використовується для заряджання та для прошивки системи. Живлення підключено до контролера живлення, який побудовано на основі Р-канального мосфет-транзистора AO3401 позначеного як U2 та діода Шотткі SS34 (D1). Коли живлення від USB доступне, то система буде живитися від нього, одночасно заряджаючи батарею. А коли зарядний пристрій буде відключено, пристрій автоматично перейде на живлення від акумулятора.

На рис. 2.5 (б) зображено регулятор напруги та кнопку живлення. Для регулювання напруги було використано мікросхему MIC5219 на 3.3 В від компанії Microchip, вона здатна забезпечувати до 500мА струму з падінням напруги 500мВ. Повзунковий перемикач, позначення якого SW1, підключено за допомогою резистора підтяжки до контакту EN мікросхеми MIC5219. Перемикач використовується для вмикання та вимикання пристрою. Коли контакт перемикача

замкнеться на землю, то MIC5219 вимкнеться, а за нею і інші компоненти, крім акумулятора.

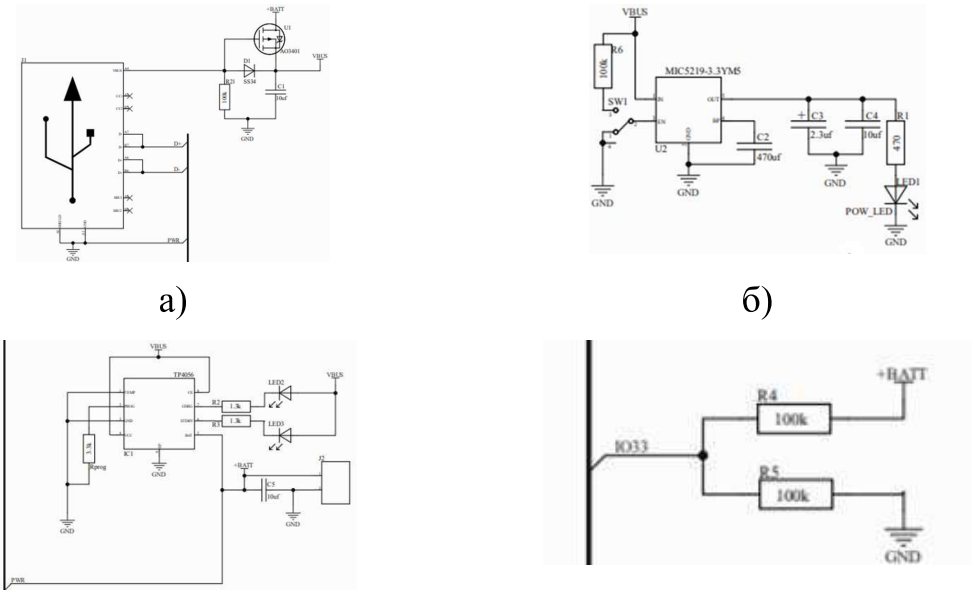


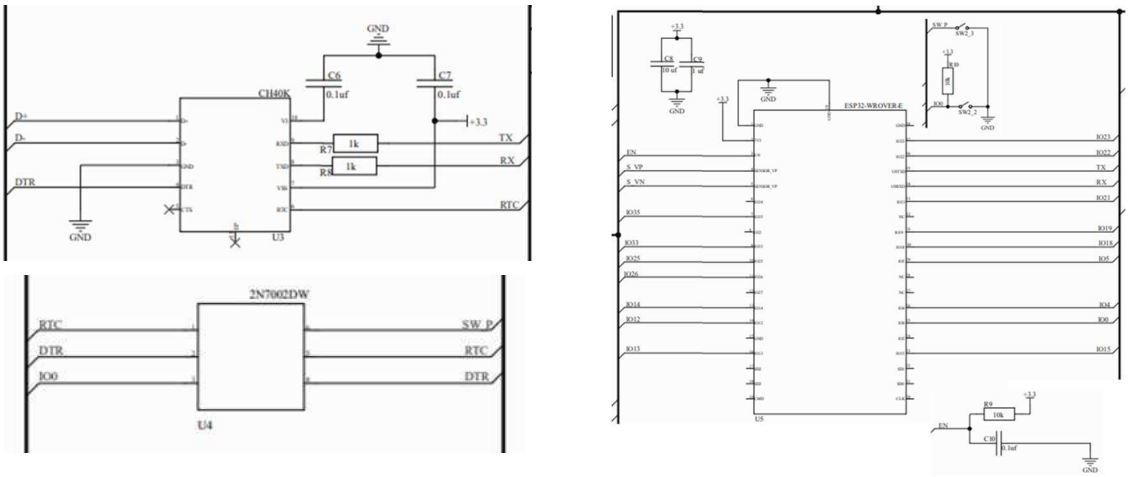
Рисунок 2.5 – Вузли схеми електричної принципової: а) USB-вхід та контроль живлення; б) кнопка живлення та регулятор напруги; в) контролер заряду; г) давач напруги батареї

Контролер TP4056, позначений як IC1, використовується для контролю заряду батареї, він витримує струм заряду до 1А. Для визначення напруги батареї використовується подільник напруги, він зменшує напругу батареї до безпечного рівня для вимірювання.

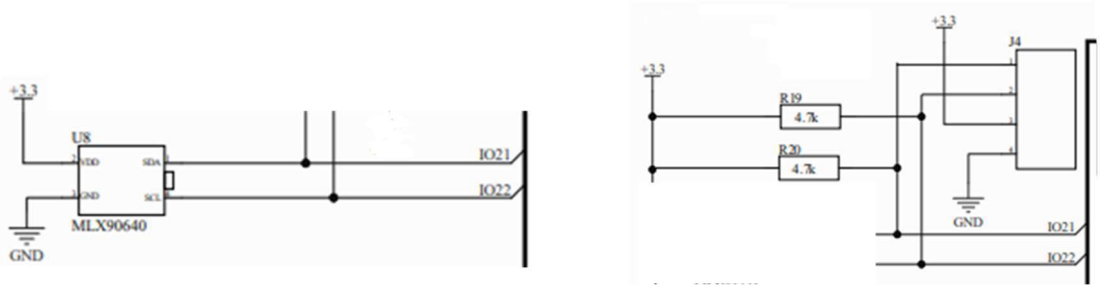
На рис. 2.6 зображено мікроконтролер ESP32-WROVER-E, вузол, за допомогою якого відбувається його програмування, та камеру з матрицею теплових давачів далекого інфрачервоного діапазону, також на рис. показано додатковий роз’єм I^2C.

Вузол, за допомогою якого програмується мікроконтролер, скадається з контролера CH340K USB-UART та подвійного N-N-канального мосфет-транзистора 2N7002DW від ON Semi, під позначеннями U4 та U5 відповідно. CH340 було вибрано через його невеликий розмір, що для даної системи є важливим, та низьку вартість. Мосфет-транзистор 2N7002DW в даній схемі

використовується як схема автоматичного скидання ESP32-WROVER-E, це прибирає потребу в ручному перериванні чи виборі завантаження під час програмування. Також передбачено кнопку для заванатаження, позначену як SW2_2, та для скидання – позначена як SW2_3.



а) б)



в) г)

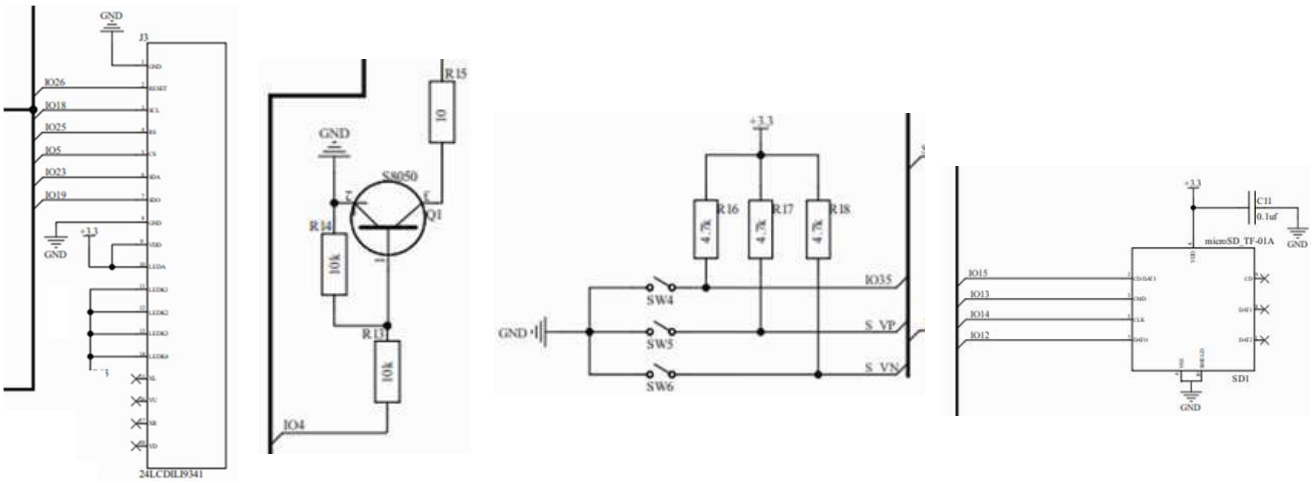
Рисунок 2.6 – Вузли схеми електричної принципової. а) USB-UART програматор, мікросхема перезагрузки; б) мікроконтролер; в) тепловізіцна камера; г) зовнішній роз’єм I^2C

Схема для мікроконтролера ESP32-WROVER-E, є стандартною, тільки необхідні конденсатори та резистори підтяжки. Також представлена схема для фільтрації зайвих шумів, побудована на основі двох конденсаторів, підключених паралельно.

Інфрачервона теплова камера розміщена безпосередньо на друкованій платі, для того щоб пристрій був компактнішим. Давач під’єднано до мікроконтролера за допомогою шини I^2C, він має 4 контакти: SDA, SCL, живлення та земля. За

необхідності розширення функціоналу в схемі передбачено додатковий роз'єм I^2C, для під'єднання додаткового модуля камери. Роз'єм має позначення J4.

На рис. 2.7 показано схему під'єднання TFT-дисплею ILI9341, контроль його підсвітки, кнопки навігації та слот для SD-карти.



а) б) в) г)

Рисунок 2.7 – Вузли схеми електричної принципової: а) TFT-дисплей; б) контроль підсвітки; в) кнопки навігації; г) SD-карта

Для візуалізації зображення буде використовуватися 2,4-дюймовий TFT-дисплей, який має роздільну здатність 320x240 пікселів. В дисплеї використовується мікросхема драйвер ILI9341. Підключення дисплею здійснюєтьсяза допомогою шини SPI та має швидкість передачі до 65МГц. Він безпосередньо розташований на платі.

Контроль підсвітки дисплею здійснюється за допомогою біполярного транзистора S8050, який позначений як Q1. Керування яскравістю здійснюється за допомогою ШІМ. Дисплей підключений до мікроконтролера через інтерфейс VSPI.

Слот для Micro SD карти підключений до мікроконтролера ESP32-WROVER-E через інтерфейс HSPI, це гарантує що мікроконтролер може мати доступ та керувати, не тільки дисплеєм чи слотом, а одночасно і тим, і тим, якщо є така необхідність.

Кнопки навігації під позиційними номерами SW4, SW5, SW6 одним контактом підключені до землі, а іншим - до мікроконтролера. Також контакти, що підключені до мікроконтролера, під'єднанні до живлення 3.3В через резистори 4.7кОм. Кожна кнопка має два режими: дія за замовчуванням та дія в налаштуваннях. Використання кнопок наступне:

– Верхня кнопка/+:

За замовчуваням: зміна режиму інтерполяції.

У налаштуваннях: змінити вибране значення.

– Середня кнопка/ОК:

За замовчуванням:

а) Коротке натиснення – збереження зображеня на SD-карту.

б) Довге натиснення – увійти в меню налаштувань.

У налаштуваннях:

а) Коротке натиснення – змінити вибір меню.

б) Довге натиснення – вийти з меню налаштувань.

– Нижня кнопка/-:

За замовчуванням: зміна палітри кольорів.

У налаштуваннях: змінти вибране значення.

2.3 Розробка блок-схему алгоритму роботи та опис бібліотек комп'ютеризованої системи виявлення та аналізу тепловтрат будівлі

Алгоритм роботи програми для комп'ютеризованої системи виявлення та аналізу тепловтрат будівлі (рис. 2.8) [13] складається з двох етапів, ініціалізації периферії та основного циклу програми.

Перший етап починається з ініціалізації серійного з'єднання, під час цього процесу встановлюється з'єднання зі всіма шинами, які присутні в пристрої, після чого починається ініціалізація дисплея. По закінченню ініціалізації починається налаштування п'яти кнопок та переривань, також виконується перевірка чи встановлено показники мінімальної та максимальної вимірюваних температур.

Якщо показники встановлено, то програма продовжує ініціалізацію інших пристрої, а якщо ні, то перед цим йде встановлення мінімальної та максимальної вимірюваних температур. Після встановлення мінімальної та максимальної температур починається перевірка увімкнення підсвітки дисплею, якщо підсвітка увімкнена, то програма йде далі, а якщо ні, то підсвітка вмикається. Останнім ініціалізується слот для MicroSD-карти, в незалежності від того чи карта пам'яті встановлена програма йде далі, виконуючи попередні обчислення для визначення однієї з 10 кольорових палітр та одного з п'яти режимів інтерполяції.

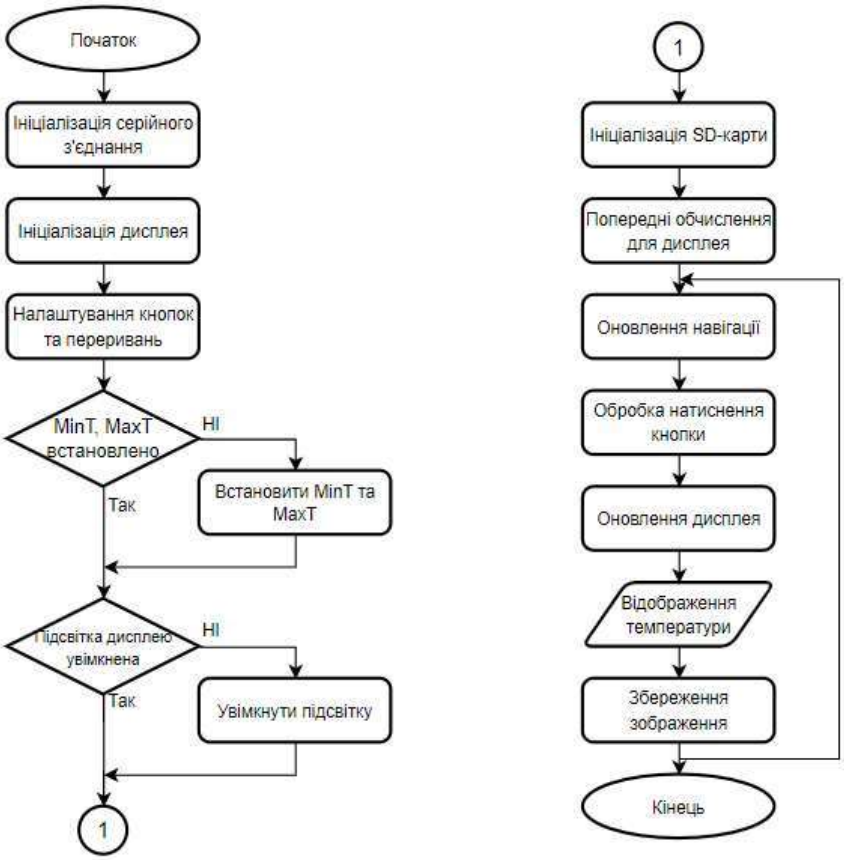


Рисунок 2.8 – Блок-схема алгоритму роботи системи виявлення та аналізу тепловтрат будівлі

Основний цикл програми, тобто створення та зберігання фото, починається з оновлення навігації. Далі обробляється натиснення кнопки, за допомогою переривань. Після того як фото було зроблено, дисплей оновлюється, щоб фото не залишалося на екрані, а можна було і далі бачити температуру предметів в режимі

реального часу, також після того як дисплей оновиться температурний показник починає знову відображатися на екрані. Останньою процедурою, що виконується в основному циклі програми є збереження зображення на MicroSD-карту, якщо та наявна.

Для написання коду для комп'ютеризованої системи виявлення та аналізу тепловтрат будівлі використано 10 бібліотек:

– Preferences.h – це вбудована бібліотека для ESP32, яка призначена для зберігання налаштувань. Бібліотека записує дані в енергонезалежну пам'ять, тому налаштування не зникають після вимкнення та перезавантаження пристрою.

– Wire.h – бібліотека, яка використовується для реалізації комунікації за протоколом I2C. Дана бібліотека надає можливість мікроконтролеру обмінюватися даними з периферією, яка підтримує протокол I^2C.

– SPI.h – бібліотека, яка дозволяє мікроконтролеру взаємодіяти з пристроями що підтримують комунікацію по протоколу SPI. Також ця бібліотека дозволяє мікроконтролеру виступати в якості ведучого пристрою.

– SD.h – бібліотека призначена для зчитування та запису даних на підключену до мікроконтролера SD-карту.

– FS.h – бібліотека, яка надає інтерфейси для роботи ESP32 з файловими системами, що дозволяє зберігати та читати файли з флеш-накопичувачів.

– Adafruit_MLX90640.h – це бібліотека, яка призначена для того, щоб мікроконтролер міг взаємодіяти з інфрачервоним тепловізійним сенсором MLX90640.

– TFT_eSPI.h – бібліотека, яка призначена для роботи з TFT-дисплеями які використовують драйвер ILI9341, та підключені по SPI.

– AnimatedGIF.h – дана бібліотека призначена для оптимізації відтворення багатокадрових GIF-фалів збережених в пам'яті.

– Font/GFXFF/gfxfont.h – бібліотека, яка дозволяє використання різних шрифтів. Бібліотека є частиною графічної бібліотеки Adafruit GFX.

– Open_Sans_ExtraBold_10.h – частина бібліотеки Adafruit GFX, яка дозволяє використання шрифту Open Sans ExtraBold з висотою символів 10 пікселів.

2.4 Обґрунтування вибору мови програмування

Для програмування мікроконтролера ESP32 є декілька варіантів вибору мови програмування, це C/C++, Python, Lua, Assembler. Для КСВАТБ мовою програмування було вибрано С. Такий вибір було зроблено з кількох причин. Основною є те, що С є мовою низького рівня, що дає більше можливостей для контролю апаратного забезпечення.

Широке поширення С дає змогу легше знайти потрібний матеріал та для освоєння та використання мови програмування, також через її поширення існує безліч бібліотек написаних для спрощення написання коду на С[7].

Ефективність даної мови є важливою в роботі пристрою, адже чим швидше обробляються дані, тим швидше працює система. Так як С є мовою низького рівня та має прямий доступ до апаратного забезпечення, то їй не потрібно використовувати безліч системних викликів, вона обробляє дані набагато швидше інших мов.

2.5 Опис шин обміну даними

Шина даних UART (Universal Asynchronous Receiver-Transmitter) [14] – це універсальна асинхронна шина, яка призначена для передавання даних між пристроями. Дані передаються по двох лініях, також, наявна лінія GND. Так як шина є асинхронною, то дані ніяк не тактуються, через що відсутня синхронізація вихідних бітів. В даному протоколі дані передаються послідовно, а різниця одиниці від нуля лише через час перепадів рівня сигналу. Сигнал задається швидкістю передачі. UART може працювати з одним з трьох видів зв’язку: симплексний – дані надсилаються в одному напрямку; напівдуплексний – дані передаються в будь-яку

сторону, але в різні моменти часу; повнодуплексний – дані можуть передаватися в обидві сторони одночасно.

Інтерфейс UART складається з передавача та приймача. Це два окремі мікроконтролери чи мікросхеми. Сигнали в UART називаються Tx – передавач та Rx – приймач. При підключені дані сигнали потрібно з’єднати хрест-навхрест.

UART також може забезпечувати контроль цілісності даних за допомогою біта парності, що дозволяє перевіряти правильність переданих даних. У разі виявлення помилки приймач може вимагати повторної передачі даних, що підвищує надійність зв'язку.

В протоколі UART був вироблений і прижився короткий спосіб запису параметрів. Короткий запис параметрів UART зазвичай включає такі основні параметри, як швидкість передачі (baud rate), біт парності, кількість біт даних та кількість стопових бітів.

UART використовується в багатьох застосунках, включаючи зв'язок між мікроконтролерами, комп'ютерами та периферійними пристроями. Одним із прикладів є підключення GPS-модулів, модулів Bluetooth або Wi-Fi до мікроконтролера для обміну даними. Цей інтерфейс широко застосовується завдяки простоті реалізації та низькій вартості.

Хоча UART є асинхронним, у деяких системах можлива додаткова синхронізація за допомогою спеціальних сигналів, що дозволяє підвищити надійність передачі даних. Це робить UART гнучким інструментом для різних видів електронних проєктів і пристроїв.

Шина I^2C(Inter-Integrated Circuit) [15] – це послідовна шина для обміну даними між інтегральними схемами, яка використовує дві двонаправлені лінії:SDA і SCL. Вона застосовується для підключення низькошвидкісних периферійних пристроїв до материнської плати, а також у вбудованих системах і мобільних телефонах.

I^2C використовує дві двонапрямлені лінії які підключені до напруги живлення резисторами (рис 2.10) і керуються через відкритий колектор чи стік. Завдання кожної лінії шини:

– SDA (Serial Data) – це послідовна лінія, яка використовується для передавання даних.

– SCL (Serial Clock) – це послідовна лінія, яка використовується для тактування.

I^2C використовує напругу живлення рівну +5В та +3.3В, але зрідка використовуються й інші напруги. Адресація шини використовує адресний простір що складається з 7-біт, тобто є 128 адрес, але 16 з них є зарезервованих, тому до однієї шини можна підключити до 112 пристроїв. I^2C може працювати в двох швидкісних режимах 10кбіт/с та 100кбіт/с.

У протоколі I^2C адресація визначає, які пристрої будуть «спілкуватися» в даний час. Існують два типи адрес: стандартна 7-бітна та розширена 10-бітна. Зазвичай використовується 7-бітна адреса, яка є унікальною для кожного пристрою на шині. Перед тим, як передавати дані, ця адреса відправляється на шину. Якщо молодший біт адреси дорівнює «1» - то пристрій «читає» дані, а якщо «0» — «пише». Розширена 10-бітна адреса використовується рідше, в основному у великих системах з понад 127 пристроями. Вона містить два біти для режиму розширеної адресації і 8 біт для реальної адреси.

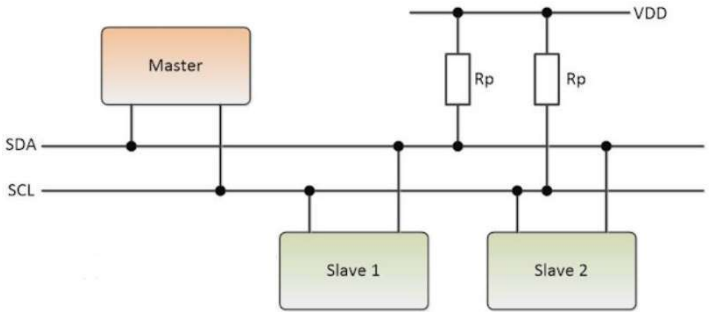


Рисунок 2.9 – Під’єднання пристроїв через I2C

Інтерфейс I^2C працює за схемою Master-Slave (ведучий-ведений). Ведучий пристрій, або master, керує шиною, тоді як підлеглі пристрої, або slave, виконують команди ведучого. Обмін даними починається зі стану START, коли сигнал SDA

змінюється з високого рівня на низький при високому рівні SCL. Це сприймається як початок передачі. Далі ведучий надсилає адресу підлеглого з бітами читання/запису. Для запису даних ведучий відправляє байти на SDA, а підлеглий приймає. Для читання підлеглий відправляє дані, а ведучий приймає. Після кожного байта даних отримувач надсилає підтвердження (ACK) або відмову (NAK). Процедура завершується станом STOP, коли сигнал SDA змінюється з низького рівня на високий при високому рівні SCL.

Синхронізація даних здійснюється через сигнал SCL. Частота передачі визначає швидкість пересилання бітів. В I^2C вона контролюється ведучим пристроєм через делімітр у контрольному регістрі. Стартові та стопові сигнали координують початок і кінець передачі. Узгодженість сигналів дозволяє пристроям бути в правильному стані для обміну даними. Час передачі кожного біта має бути стабільним. Пристрої повинні синхронізувати свої годинники для відповідності встановленій частоті передачі. Підтвердження або відмова визначається сигналом на SDA, який має бути стабільним. Для асинхронної роботи потрібна узгодженість і мінімальні затримки між пристроями.

Тактування в I^2C визначається тактовими сигналами для синхронізації передачі даних. Сигнал SCL генерується ведучим і використовується для синхронізації даних на SDA. Кожен біт даних передається або приймається при зміні сигналу SCL.

Шина даних SPI(Serial Peripheral Interface) [16] – це синхронний послідовний інтерфейс передачі даних, що підтримує повний дуплексний режим і призначений для з'єднання мікроконтролера з периферійними пристроями [8].

Шина SPI, як і I^2C працює за принципом Master-Slave (ведучий-ведений). В даній шині всі дані передаються синхронно разом з тактовим сигналом, цей сигнал генерує ведучий пристрій, а переферія вже синхронізується зі згенерованим сигналом. До одного інтерфейсу може бути приєднано один ведучий і кілька ведених пристрої. При цьому одночасно спілкується тільки ведучий і один ведений пристрій, решта в цей момент не бере участі в передаванні даних.

Шина SPI використовує чотири цифрові сигнали, які можуть змінювати імена

портів залежно від виробника апаратних засобів, але найчастіше їх називають MISO, MOSI, SCLK та CS(SS). Функції кожної лінії:

– MOSI – вихід ведучого, вхід веденого. Дана лінія передає дані від ведучого мікроконтролера до периферії.

– MISO – вхід ведучого, вихід веденого. Лінія, яка служить для передачі даних від переферії до мікроконтролера.

– SCLK – послідовний тактовий сигнал. Лінія, яка передає тактовий сигнал згенерований ведучим пристроєм для ведених.

– CS (SS) – вибір веденого. Лінія служить для того щоб мікроконтролер сповіщав пристрій з яким буде спілкуватися в даний момент.

Синхронізація в SPI відбувається таким чином, що коли ведучий пристрій генерує тактуючий сигнал, ведучі використовують його для визначення моментів зміни бітів на лінії даних. У всіх пристроїв, які входять в шину SPI є лічильники імпульсів синхронізації, у ведених пристроїв лічильник дозволяє визначити момент, в який він завершує передачу пакета. Лічильник скидається при вимкненні підсистеми SPI, проте, у ведених пристроїв лічильник може скидатися коли ведучий пристрій перемикається на наступний з ведених.

Передавання даних в SPI здійснюється пакетами, довжина яких становить 8біт. Master-пристрій починає цикл зв’язку встановлюючи низький рівень сигналу на виводі SS веденого пристрою з яким хоче з’єднатися. Після чого дані які мають бути передані обома пристроями поміщаються в регістри. Далі ведучий пристрій генерує імпульси синхронізації по лінії SCLK. Дані передаються по біту послідовно, від master-пристрою по лінії MOSI, а від переферії - по MISO. Після закінчення передачі даних мікроконтролер переводить лінію SS на переферії у високий стан.

Шина SPI має два регістри, керування та статусу. Значення кожного біта регістру керування від нульового до сьомого:

– SPR0 – вибір частоти тактового сигналу (тільки для ведучого пристрою);

– SPR1 – вибір частоти тактового сигналу (тільки для ведучого

пристрою);

– CPHA – фаза тактового сигналу;

– SPOL – полярність тактового сигналу;

– MSTR – вибір режиму ведучий/ведений;

– DORD – даний біт керує порядком передачі даних;

– SPE – сигнал, який вмикає, чи вимикає роботу інтерфейсу SPI (1 – працює, 0 – не працює).

Значення перших шести бітів регістру є зарезервованими та встановлені в значення «0». Біт 6 (WCOL) відповідає за встановлення прапора, коли виникає помилка при записі даних. Біт 7 (WCOL), це прапор переривання, встановлюється в режим «1», коли завершується обмін даними, а в режим «0» - при читанні з регістру.

Отже, в цьому розділі було проведено проектування комп'ютеризованої системи виявлення та аналізу тепловтрат будівлі. В результаті було спроектовано загальну структурну схему, схему електричну принципову та блок схему алгоритму.

РОЗДІЛ 3. ПРАКТИЧНА ЧАСТИНА

3.1 Реалізація програмної частини комп'ютеризованої системи виявлення та аналізу тепловтрат будівлі

Для комп'ютеризованої системи виявлення та аналізу тепловтрат будівлі було розроблено програмний код в середовищі Arduino IDE [17]. Спочатку було підключено бібліотеки, визначено глобальні змінні, пізніше для кожної змінної було створено екземпляр окремо. Скрін частини коду, де відбувається підключення бібліотек показано на рис. 3.1.

```
#include <Preferences.h>
#include <Wire.h>
#include <SPI.h>
#include <SD.h>
#include "FS.h"
#include <Adafruit_MLX90640.h>
#include <TFT_eSPI.h>
#include <AnimatedGIF.h>
#include <Fonts/GFXFF/gfxfont.h>
#include "Open_Sans_ExtraBold_10.h"
SPIClass spiSD(HSPI);
#define SD_CS 15
//#define USE_DMA
#define NORMAL_SPEED
#define TFT_WIDTH 320
#define TFT_HEIGHT 240
#define BUFFER_SIZE 320
uint16_t usTemp[BUFFER_SIZE];
#define VBAT_PIN 33
#define BATTV_MAX 4.2
#define BATTV_MIN 3.2
#define GIF_IMAGE BootAnimationIMG
#define SGIF_IMAGE success_GIF
#define EGIF_IMAGE Error_GIF
bool dmaBuf = 0;
Adafruit_MLX90640 mlx;
Preferences preferences;
AnimatedGIF gif;
TFT_eSPI tft = TFT_eSPI();
File bmpFile;
float frame[32 * 24];
float batv;
const int upButton = 35;
const int middleButton = 36;
const int downButton = 39;
const int backlightPin = 4;
int interpolationMode = 0;
int AutoScale = 0;
volatile bool MenuChange = false;
volatile bool _upShort = false;
volatile bool _downShort = false;
volatile bool middleShort = false;
volatile bool middlePressed = false;
unsigned long middlePressStartTime = 0;
int Menu = 0, Menuitem = 0;
int BLPWM = 0;
int RefreshRate = 0;
float MinT, MaxT;
String menuItems[7] = { "Auto Scale
String ASindex[2] = { "Off", "On" };
String IPindex[6] = { "Nearest Neighbor", "Average", "Bilinear", "Bilinear Fast", "Triang
String RRindex[4] = { "4 Hz", "8 Hz", "16 Hz", "32 Hz" };
float xRatios[320];
float yRatios[240];
float xOppositeRatios[320];
float yOppositeRatios[240];
#define PALETTE_COUNT 10
uint16_t colorPalettes[PALETTE_COUNT][6] = {
  { TFT_BLUE, TFT_CYAN, TFT_GREEN, TFT_YELLOW, TFT_RED, TFT_MAGENTA },
  { TFT_BLACK, TFT_DARKGREY, TFT_LIGHTGREY, TFT_WHITE, TFT_ORANGE, TFT_PINK },
  { TFT_NAVY, TFT_OLIVE, TFT_DARKGREEN, TFT_DARKCYAN, TFT_MAROON, TFT_PURPLE },
  { TFT_BLUE, TFT_GREEN, TFT_DARKGREEN, TFT_ORANGE, TFT_MAROON, TFT_RED },
  { TFT_NAVY, TFT_DARKGREEN, TFT_GREEN, TFT_YELLOW, TFT_ORANGE, TFT_RED },
  { TFT_CYAN, TFT_BLUE, TFT_MAGENTA, TFT_YELLOW, TFT_GREEN, TFT_RED },
  { TFT_WHITE, TFT_ORANGE, TFT_RED, TFT_BLUE, TFT_GREEN, TFT_BLACK },
  { TFT_PURPLE, TFT_MAGENTA, TFT_RED, TFT_ORANGE, TFT_YELLOW, TFT_GREEN },
  { TFT_YELLOW, TFT_PINK, TFT_WHITE, TFT_BLUE, TFT_DARKCYAN, TFT_DARKGREEN },
  { TFT_RED, TFT_YELLOW, TFT_GREEN, TFT_CYAN, TFT_BLUE, TFT_MAGENTA }
};
int paletteIndex = 0;

float tempMin = 20.0;
float tempMax = 32.0;

TFT_eSprite sprite = TFT_eSprite(&tft);
```

Рисунок 3.1 – Частина коду де відбувається підключення бібліотек та визначення глобальних змінних

Функції upButton_ISR, downButton_ISR та midleButton_ISR використовуються для виявлення та оброблення натискань верхньої, нижньої та середньої кнопок відповідно (рис. 3.2). Для виявлення натискань використовуються апаратні переривання. Також в даному коді є усунення дебаунсу, для правильного виявлення натискань кнопок. Залежно від того, яку кнопку було натиснути та часу

натискання, функція встановлює для відповідної змінної значення true, після чого воно буде оброблено в головній програмі.

```
void IRAM_ATTR upButton_ISR() {
  static unsigned long last_interrupt_time = 0;
  unsigned long interrupt_time = millis();
  if (interrupt_time - last_interrupt_time > 200) {
    _upShort = true;  }
  last_interrupt_time = interrupt_time;  }
void IRAM_ATTR downButton_ISR() {
  static unsigned long last_interrupt_time = 0;
  unsigned long interrupt_time = millis();
  if (interrupt_time - last_interrupt_time > 200) {
    _downShort = true;  }
  last_interrupt_time = interrupt_time;  }
void IRAM_ATTR middleButton_ISR() {
  if (digitalRead(middleButton) == LOW) {
    if (!middlePressed) {
      middlePressed = true;
      middlePressStartTime = millis();  }
  } else {
    if (middlePressed) {
      if (millis() - middlePressStartTime < 1000) {
        middleShort = true;  }
      middlePressed = false;
      middlePressStartTime = 0;  }  }  }
```

Рисунок 3.2 – Функції оброблення натиснень кнопок

Функція MLXInit використовується для ініціалізації давача MLX90640, наступна функція Configresreshrate призначена для налаштування частоти оновлення давача. Функції ReadConfig та WriteConfig зчитую та записую конфігурації у регістри камери. Частина коду в середовищі Arduino IDE, де реалізовано функції зв’язку та налаштування давача MLX90640 показано на рис. 3.3.

```
void MLXInit() {
  Wire.begin();
  Wire.setClock(1000000);
  if (!mlx.begin()) {
    Serial.println("Failed to initialize MLX90640!");
  }
  mlx.setResolution(MLX90640_ADC_18BIT);
  ConfigRefreshrate();
}

void ConfigRefreshrate() {
  switch (RefreshRate) {
    case 1:
      mlx.setRefreshRate(MLX90640_4_HZ);
      break;
    case 2:
      mlx.setRefreshRate(MLX90640_8_HZ);
      break;
    case 3:
      mlx.setRefreshRate(MLX90640_16_HZ);
      break;
    case 4:
      mlx.setRefreshRate(MLX90640_32_HZ);
      break;
    default:
      break;
  }
}
```

```
void ReadConfig() {
  preferences.begin("Config", false);
  String temp;
  temp = preferences.getString("AutoScale", "");
  AutoScale = temp.toInt();
  temp = preferences.getString("MinTe", "");
  MinT = temp.toFloat();
  temp = preferences.getString("MaxTe", "");
  MaxT = temp.toFloat();
  temp = preferences.getString("Imode", "");
  interpolationMode = temp.toInt();
  temp = preferences.getString("PIndex", "");
  paletteIndex = temp.toInt();
  temp = preferences.getString("RefreshRate", "");
  RefreshRate = temp.toInt();
  temp = preferences.getString("BLPWM", "");
  BLPWM = temp.toInt();
  preferences.end();  }
void WriteConfig() {
  preferences.begin("Config", false);
  preferences.putString("AutoScale", String(AutoScale));
  preferences.putString("MinTe", String(MinT));
  preferences.putString("MaxTe", String(MaxT));
  preferences.putString("Imode", String(interpolationMode));
  preferences.putString("PIndex", String(paletteIndex));
  preferences.putString("RefreshRate", String(RefreshRate));
  preferences.putString("BLPWM", String(BLPWM));
  preferences.end();  }
```

Рисунок 3.3 – Реалізація функцій ініціалізації на налаштування давача MLX90640

Функція initSDcard використовується для ініціалізації SD-карти та для визначення її наявності перед збереженням зображення. Якщо SD-карта не виявлена то функція повертає помилку. Функція generateFilename виконує генерування імен для зображень. Вона перевіряє останнє збережене зображення та присвоює наступному більший номер. Функція WriteBMP зберігає зображення в форматі BMP на SD-карту після натиснення кнопки збереження зображення. Частина коду в середовищі Arduino IDE, де реалізовано функції ініціалізації слота для SD-карти та запису зображення на неї показано на рис. 3.3.

```
void initSDcard() {
  //spiSD.begin(14, 12, 13, SD_CS);  //CLK,MISO,MOIS,SS
  if (!SD.begin(SD_CS, spiSD)) {
    Serial.println("Card Mount Failed");
    return;
  } else {
    Serial.println("Card Mount Successful");
  }
  uint8_t cardType = SD.cardType();

  if (cardType == CARD_NONE) {
    Serial.println("No SD card attached");
    return;
  }
}
String generateFilename(fs::FS &fs) {
  for (int i = 0; i <= 9999; i++) {
    char filename[23];
    sprintf(filename, "/ThermalCamera%04d.bmp", i);

    if (!fs.exists(filename)) {
      return String(filename);
    }
  }

  return "";  /
}
int writeBMP(fs::FS &fs, const char *path, TFT_eSprite *sprite) {
  const int width = sprite->width();
  const int height = sprite->height();

  uint8_t bmpFileHeader[14] = { 'B', 'M', 0, 0, 0, 0, 0, 0, 0, 0, 54, 0, 0, 0 };

  uint32_t fileSize = 54 + width * height * 2;
  bmpFileHeader[2] = (uint8_t)(fileSize);
  bmpFileHeader[3] = (uint8_t)(fileSize >> 8);
  bmpFileHeader[4] = (uint8_t)(fileSize >> 16);
  bmpFileHeader[5] = (uint8_t)(fileSize >> 24);

  uint8_t bmpInfoHeader[40] = { 40, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 16, 0 };

  bmpInfoHeader[4] = (uint8_t)(width);
  bmpInfoHeader[5] = (uint8_t)(width >> 8);
  bmpInfoHeader[6] = (uint8_t)(width >> 16);
  bmpInfoHeader[7] = (uint8_t)(width >> 24);

  bmpInfoHeader[8] = (uint8_t)(height);
  bmpInfoHeader[9] = (uint8_t)(height >> 8);
  bmpInfoHeader[10] = (uint8_t)(height >> 16);
  bmpInfoHeader[11] = (uint8_t)(height >> 24);

  File file = fs.open(path, FILE_WRITE);
  if (!file) {
    Serial.println("Failed to open file for writing");
    return 0;
  } else {

    file.write(bmpFileHeader, 14);
    file.write(bmpInfoHeader, 40);

    for (int y = height - 1; y >= 0; y--) {
      for (int x = 0; x < width; x++) {
        uint16_t pixel = sprite->readPixel(x, y);
        uint16_t r = (pixel >> 11) & 0x1F;
        uint16_t g = (pixel >> 5) & 0x3F;
        uint16_t b = pixel & 0x1F;
        pixel = (b << 11) | (r << 5) | g;
        file.write(pixel >> 8);
        file.write(pixel);
      }
    }
    file.close();
    return 1;
  }
}
```

Рисунок 3.4 – Реалізація функцій ініціалізації слота для SD-карти та запису зображення на неї

Функція navigationUpdate використовується для оброблення всіх процедур пов’язаних з кнопками та навігацією по меню. Функція обробляє натиснення залежно від того, на якому екрані перебуває користувач (екран зйомки чи екран налаштувань), так вона обробляє натиснення кнопок.

Функція displayUpdate використовується для оброблення процедур графіки та оброблення даних з давача MLX90640. Дана функція зчитує дані з давача, збільшує масштаб та викликає функцію drawPixel (рис. 3.5) для малювання зображення піксель за пікселем.

```
590  void drawPixel(int x, int y, float val) {
591    int colorIndex;
592    if (AutoScale == 1) {
593      if (val <= tempMin) {
594        colorIndex = 0;
595      } else if (val >= tempMax) {
596        colorIndex = 5;
597      } else {
598        colorIndex = int(map(val, tempMin, tempMax, 0, 5));
599      }
600    } else {
601      if (val <= MinT) {
602        colorIndex = 0;
603      } else if (val >= MaxT) {
604        colorIndex = 5;
605      } else {
606        colorIndex = int(map(val, MinT, MaxT, 0, 5));
607      }
608    }
609    int color = colorPalettes[paletteIndex][colorIndex];
610    sprite.drawPixel(x, y, color);
611  }
```

Рисунок 3.5 – Код функції для малювання картинки на екрані

Функція drawPixel використовує вибрану палітру для малювання зображення. Також функція displayUpdate відповідає за друк мінімальної, максимальної та середньої температур за допомогою функції drawBatttery (рис. 3.6). Ще ця функція малює та обробляє меню налаштувань.

```
612  void drawBattery(int batpc) {
613    int x = 290;
614    int y = 10;
615    int w = 20;
616    int h = 10;
617    int color = TFT_RED;
618    int fillLevel = 0;
619    if (batpc > 75) {
620      fillLevel = w;
621      color = TFT_GREEN;
622    } else if (batpc > 50) {
623      fillLevel = w * 3 / 4;
624      color = TFT_GREEN;
625    } else if (batpc > 25) {
626      fillLevel = w / 2;
627      color = TFT_BLUE;
628    } else if (batpc > 0) {
629      fillLevel = w / 4;
630      color = TFT_BLUE;
631    }
632    sprite.drawRect(x, y, w, h, TFT_WHITE);
633    sprite.drawRect(x + w, y + h / 4, 2, h / 2, TFT_WHITE);
634    if (fillLevel > 0) {
635      sprite.fillRect(x + 1, y + 1, fillLevel - 2, h - 2, color);
636    }
637  }
```

Рисунок 3.6 – Код функції для відображення заряду батареї

Останні функції це функції налаштування та циклу. Функція setup використовується для того, щоб ініціалізувати та налаштувати всі бібліотеки під час запуску системи. Функція циклу викликає функції navigationUpdate та displayUpdate. Частина коду в середовищі Arduino IDE, де реалізовано функції налаштування та циклу показано на рис. 3.7.

```
774  void setup() {
775    Serial.begin(115200);
776    tft.init();
777  #ifdef USE_DMA
778    tft.initDMA();
779  #endif
780    tft.setRotation(1);
781    tft.fillScreen(TFT_BLACK);
782    gif.begin(BIG_ENDIAN_PIXELS);
783    sprite.createSprite(320, 240);
784    pinMode(upButton, INPUT_PULLUP);
785    attachInterrupt(digitalPinToInterrupt(upButton), upButton_ISR, CHANGE);
786    pinMode(downButton, INPUT_PULLUP);
787    attachInterrupt(digitalPinToInterrupt(downButton), downButton_ISR, CHANGE);
788    pinMode(middleButton, INPUT_PULLUP);
789    attachInterrupt(digitalPinToInterrupt(middleButton), middleButton_ISR, CHANGE);
790    ReadConfig();
791    MLXInit();
792    if (MinT == 0 && MaxT == 0) {
793      MinT = 20.0;
794      MaxT = 32.0;
795      WriteConfig();  }
796    if (BLPWM == 0) {
797      BLPWM = 100;
798      WriteConfig();  }
799    pinMode(backlightPin, OUTPUT);
800    analogWrite(backlightPin, map(BLPWM, 0, 100, 0, 255));
801    if (gif.open((uint8_t *)GIF_IMAGE, sizeof(GIF_IMAGE), GIFDraw)) {
802      Serial.printf("Successfully opened GIF; Canvas size = %d x %d\n",
803      while (gif.playFrame(true, NULL)) {
804        sprite.pushSprite(0, 0);
805        yield(); }
806      gif.close();  }
807    delay(1000);
808    initSDcard();
809    for (int x = 0; x < 320; x++) {
810      xRatios[x] = (x % 10) / 10.0f;
811      xOppositeRatios[x] = 1 - xRatios[x];  }
812    for (int y = 0; y < 240; y++) {
813      yRatios[y] = (y % 10) / 10.0f;
814      yOppositeRatios[y] = 1 - yRatios[y];  }  }
815
816  void loop() {
817    navigationUpdate();
818    displayUpdate();  }
```

Рисунок 3.8 – Код функцій налаштування та циклу

3.2 Реалізація апаратної частини комп’ютеризованої системи виявлення та аналізу тепловтрат будівлі

На основі розробленої схеми електричної принципової було розведено друкований вузол [18]. Плата є двосторонньою з розмірами 80х50мм. На платі було передбачено посадкові місця для всіх компонентів відповідно до вимог. По краях плати наявні технологічні отвори діаметром 2.2міліметра. Крім посадкових місць штатних компонентів на платі було передбачено місце під зовнішній роз’єм I^2C для під’єднання модуля камери MLX90640. Також на платі є додаткові посадкові місця під плату ESP32-WROOM, її можна припаяти замість ESP32-WROVER-E. Зовнішній вигляд плати зображено на рис. 3.8

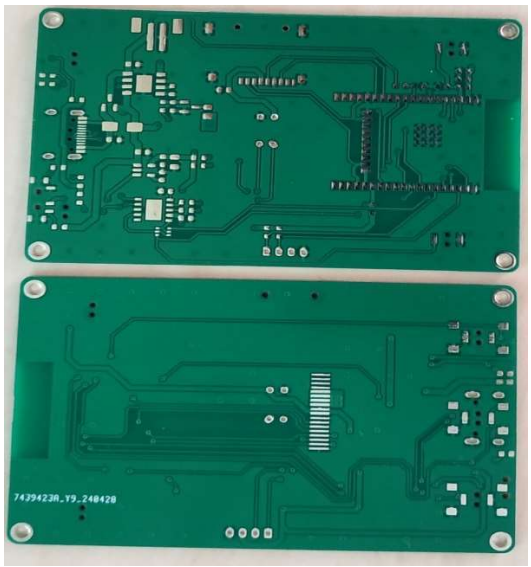


Рисунок 3.8 – Вигляд плати знизу на зверху

Після того як плати було виготовлено потрібно запаяти всі компоненти на неї. Спочатку було припаяно компоненти, які знаходяться знизу плати, це мікроконтролер, кнопки живлення та перезапуску, роз'єм USB-C, контролер заряду, USB-UART перетворювач, роз'єм під SD-карту та ще кілька деталей. Після того, як основні компоненти були на платі, потрібно додати до них обв'язку, це різні резистори, конденсатори, діоди та інше. На задній стороні було зібрано все крім камери та батареї (рис. 3.9).

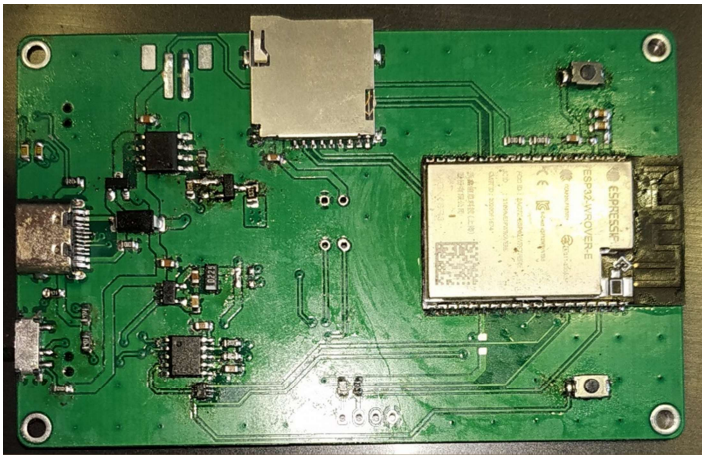


Рисунок 3.9 – Частково зібрана задня сторона плати

На верхній стороні плати менше компонентів за рахунок того, що дисплей займає приблизно 70% плати. Тому зверху потрібно припаяти лише три кнопки, та три світлодіода (рис. 3.10), дисплей буде припаяний згодом.

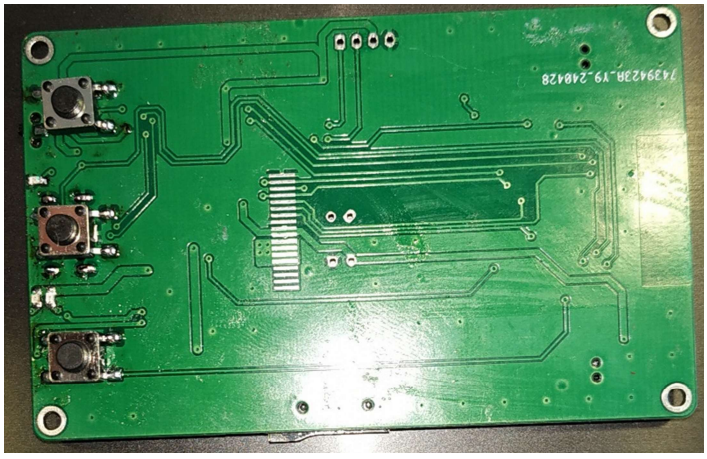


Рисунок 3.10 - Частково зібрана передня сторона плати

Наступним, що потрібно припаяти - це камера, після того як її було припаяно на верхній стороні плати її виводи потрібно заізолювати, наприклад каптоновим скотчем, щоб виводи камери не коротили, так як на задній стороні екрану є металева кришка. Далі паяємо дисплей на передню частину плати, в нього немає гребінки, тому до плати потрібно припаяти вісімнадцяти-контактний шлейф. На цьому фронтальна сторона закінчена (рис. 3.11).

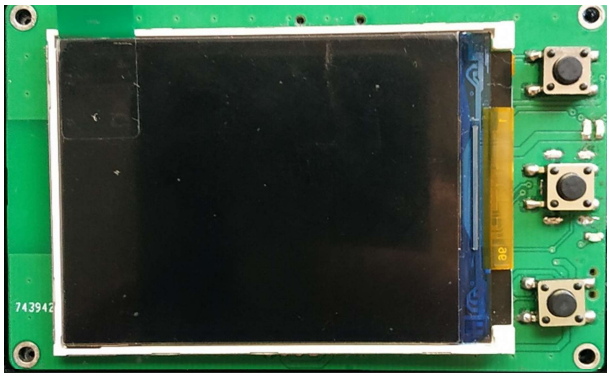


Рисунок 3.11 – Зібрана фронтальна сторона

Для того щоб нічого не замкнуло, останньою було припаяно батарею, вона кріпиться до задньої сторони плати, до спеціально виведених контактів (рис. 3.12).

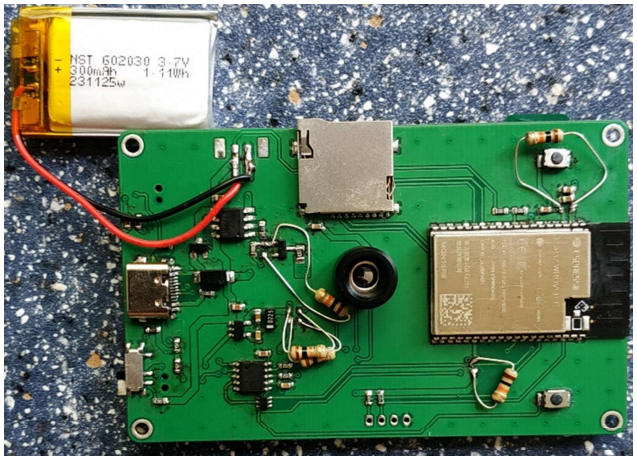


Рисунок 3.12 – Зібрана задня сторона плати

Для комп'ютеризованої системи виявлення та аналізу тепловтрат будівлі було розроблено корпус. Він складається з двох частин, верхньої, з вирізом під екран та кнопки, та нижньої, з вирізом під камеру, слот microSD, кнопку

ввімкнення та роз’єм USB-C. Спроектований в програмі корпус показано на рис. 3.13.

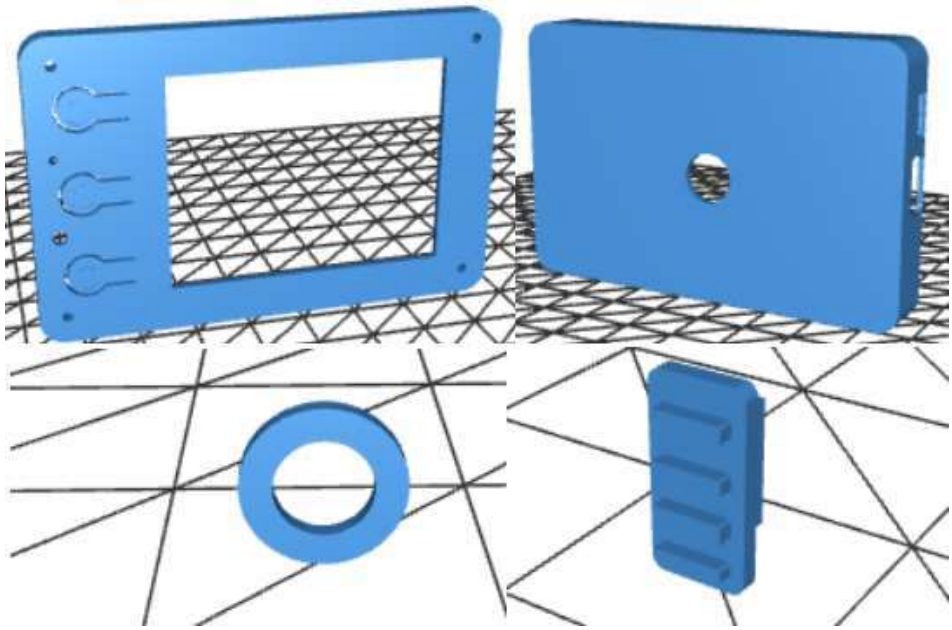


Рисунок 3.13 – Корпус в програмі для 3D-моделювання

Після проектування корпус було надруковано на 3D-принтері та поміщено в нього раніше зібраний пристрій (рис. 3.14).

Рисунок 3.14 – Повністю готовий пристрій

3.3 Налаштування та тестування системи

Перед тим, як використовувати систему, потрібно завантажити програмний код. Перед завантаженням коду в середовищі Arduino IDE необхідно встановити підтримку модулів ESP32, для цього в середовищі є розділ «Менеджер плат». В пошуку потрібно ввести назву платформи, після чого в списку з'явиться кілька варіантів вибору. Необхідно вибрати розробку від Espressif Systems (рис. 3.14) та натиснути кнопку ВСТАНОВИТИ.

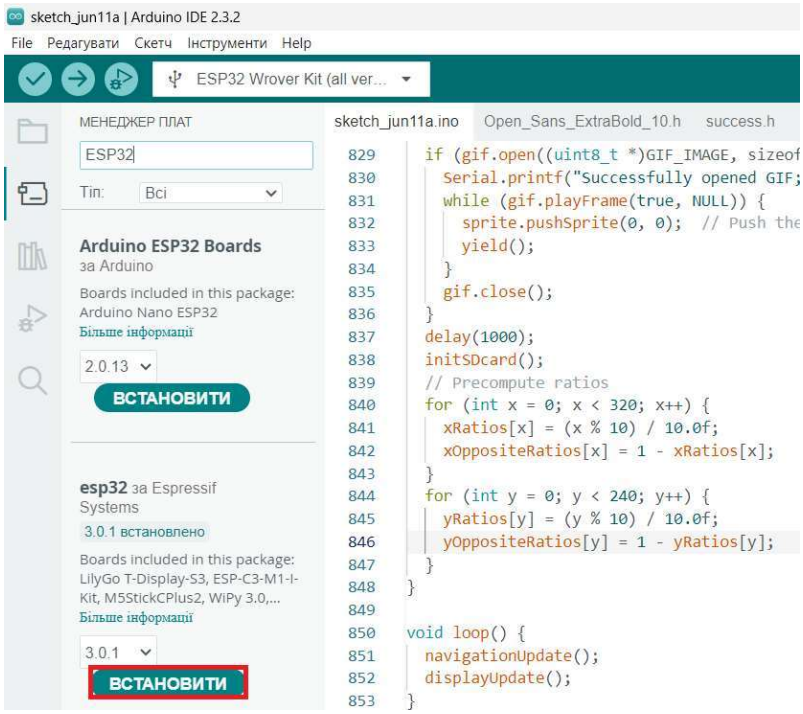


Рисунок 3.14 – Встановлення підтримки для модулів ESP32 в Arduino IDE

Коли модулі будуть встановлені, у програмі з'являться відповідні повідомлення. Щоб код коректно завантажився на плату, необхідно вибрати тип плати та COM-порт, в цьому випадку в полі «Плата» має бути вибрано ESP32 Wrover Kit (all version), а в полі COM-порт необхідно вибрати той, який з'являється після підключення плати.

Для завантаження коду на плату в верхньому меню є стрілка вправо, після її натиснення починається процес завантаження. Він триває декілька хвилин, після чого з'являється повідомлення про успішне завантаження (рис. 3.15).

```
Вивід
Writing at 0x00117578... (96 %)
Writing at 0x0011cc0e... (98 %)
Writing at 0x00122113... (100 %)
Wrote 1139136 bytes (945473 compressed) at 0x00010000 in 12.3 seconds (effective 742.6 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

Рисунок 3.15 – Сповіщення про успішне завантаження прошивки для плати

Коли на платі буде прошивка, можна вмикати пристрій та переходити до налаштування мінімальної та максимальної яскравості. Для цього необхідно затиснути середню кнопку на пристрої, після чого здійсниться вхід у налаштування системи, кнопками вверх на вниз можна міняти значення.

У зв’язку з тим що система була розроблена у теплу пору року, виміряти тепловтрати в повній мірі немає можливості, тому було виміряно інші предмети не пов’язані з будівлями.

Коли система є повністю налаштованої можна переходити до тестування. Наведемо камеру КСВАТБ на стик між стінами кімнати (рис. 3.15).

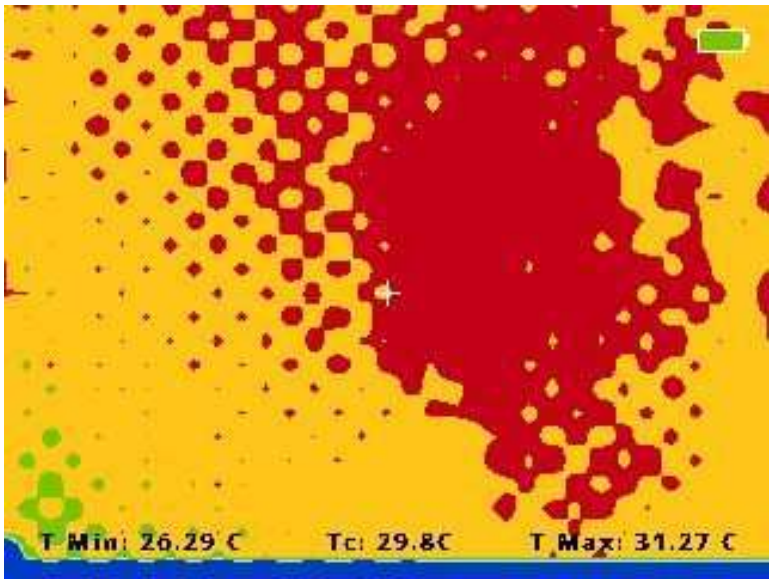


Рисунок 3.15 – Перевірка тепловтрат між стінами будівлі

Як видно, з рис. 3.15, тепловтрати дійсно є, вони позначені червоним кольором, а решта стіни жовтим.

Перевірити чи дійсно система може виміряти низькі показники температури, для цього направимо камеру на відкритий холодильник (рис. 3.16).

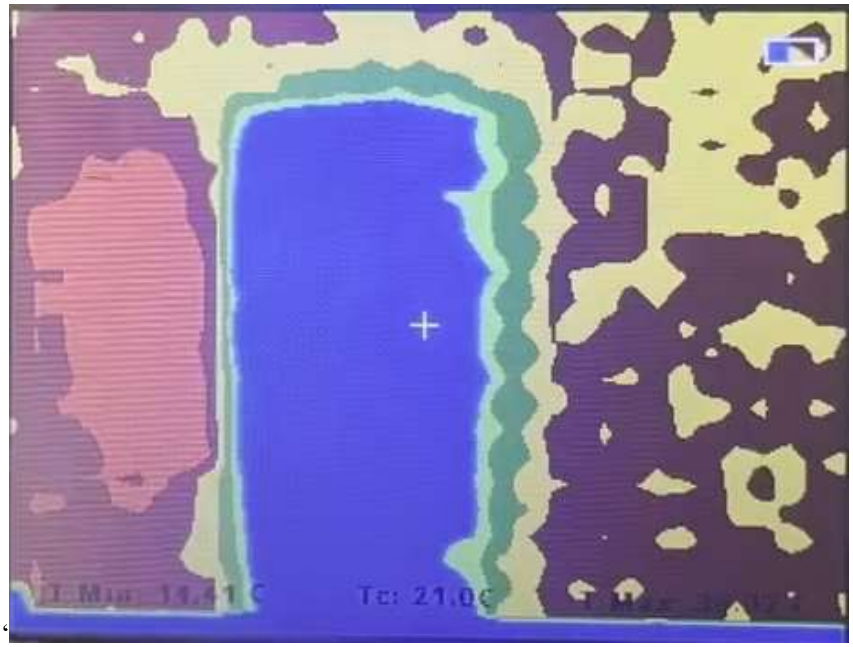


Рисунок 3.16 – Перевірка вимірювання низьких температур КСВАТБ

Щоб перевірити чи система дійсно може виявити температури до 400°С, наведемо її камеру на вогонь від газової плити (рис. 3.17).

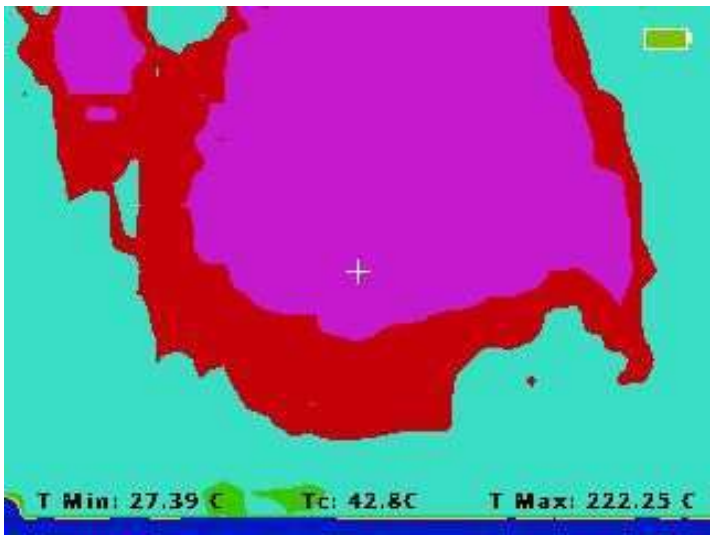


Рисунок 3.16 – Перевірка вимірювання високих температур КСВАТБ

Як видно з рис. 3.16 камера може розпізнавати високу температуру.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Долікарська допомога при обмороженні.

Обмороження – це серйозне ушкодження тканин, спричинене впливом низьких температур. Правильна та своєчасна перша допомога може запобігти серйозним наслідкам та сприяти швидшому одужанню.

Обмороження буває чотирьох стадій (рис. 4.1). Стадію обмороження можна визначити за такими ознаками [19]:

– перша стадія:

а) бліда або червонувата шкіра;

б) відчуття поколювання;

в) оніміння або легкого болю;

– друга стадія:

а) бліда або сірувата шкіра;

б) пухирі, заповнені рідиною які можуть з’явитися після відігрівання;

в) відчуття сильного болю або оніміння;

– третя стадія:

а) синьо-біла шкіра або сіра шкіра;

б) великі пухирі, заповнені кров’янистою рідиною;

в) тверда шкіра та тканини під нею;

– четверта стадія:

а) темно-синя або чорна шкіра;

б) трата чутливості в уражених ділянках;

в) некроз тканин.

Кожна стадія обмороження вимагає різного підходу до надання першої допомоги, яка повинна бути надана негайно для мінімізації пошкоджень.

Рисунок 4.1 – Стадії обмороження

При будь-якій стадії обмороження, перше що треба зробити, це максимально швидко доправити постраждалого до найближчого теплого приміщення. Далі потрібно зняти з постраждалої людини весь верхній одяг, взуття, шкарпетки, рукавиці та оглянути її, щоб виявити яку саме стадію обмороження отримала людина. Залежно від цього потрібно надати медичну допомогу.

При першій стадії обмороження потрібно зігріти обморожені ділянки шкіри теплими руками, розтиранням, теплим диханням, або теплою водою, але ні в якому разі не гарячою оскільки це може викликати опіки, температура води має бути приблизно 37-39 градусів Цельсія.

При стадіях від другої до четвертої швидке зігрівання чи розтирання протипоказане. На місце обмороження слід накласти теплоізоляційну пов’язку яка складається з таких шарів [19]:

– марля;
– вата;
– марля;
– клейонка.

Далі необхідно зафіксувати уражені кінцівки в стані спокою, за допомогою шини яку можна виготовити з щільного картону чи дошки.

При будь-якій стадії обмороження категорично заборонено розтирати уражену частину тіла снігом, застосовувати швидке зігрівання біля багаття, безконтрольно застосовувати грілки та інші джерела тепла, оскільки це погіршує перебіг обмороження. Також при глибокому обморожені заборонено застосовувати олії чи розтирати кінцівки спиртом.

Після виконаних маніпуляцій постраждалому необхідно дати випити теплої рідини та чогось теплого поїсти. Також можна дати невелику кількість якогось алкоголю.

Постраждалому потрібно дати яке небудь обезболююче, щоб зменшити біль. З ліків, що можна знайти в будь-якій аптечці, підійдуть одна таблетка анальгіну, одна таблетка аспірину, а також дві таблетки папаверину і но-шпи.

Але все ж найкращим варіантом буде доставити людину в лікарню.

4.2 Заходи, що забезпечують оптимальні метеорологічні умови в санітарно-побутових приміщеннях.

Показники мікроклімату приміщень характеризуються трьома наступними показниками [20]:

– температура повітря;
– відносна вологість повітря;
– швидкість руху повітря.

Для приміщень встановлені оптимальні та допустимі мікрокліматичні умови, з урахуванням періоду року та важкості виконуваної роботи.

Оптимальні величини температури повітря, відносної вологості та швидкості руху повітря в холодний період року наведені у таблиці 4.1.

Таблиця 4.1 – Умови мікроклімату в холодний період року

Категорія робіт	Температура повітря, °С	Відносна вологість, %	Швидкість руху повітря, м/сек.
Легка Іа	22-24	40-60	0,1
Легка Іб	21-23	40-60	0,1
Середньої важкості ІІа	19-21	40-60	0,2
Середньої важкості ІІа	17-19	40-60	0,2
Важка ІІІ	16-18	40-60	0,3

Оптимальні величини температури повітря, відносної вологості та швидкості руху повітря в теплий період року наведені у таблиці 4.2.

Таблиця 4.2 – Умови мікроклімату в теплий період року

Категорія робіт	Температура повітря, °С	Відносна вологість, %	Швидкість руху повітря, м/сек.
Легка Іа	23-25	40-60	0,1
Легка Іб	22-24	40-60	0,2
Середньої важкості ІІа	21-23	40-60	0,3
Середньої важкості ІІа	20-22	40-60	0,3
Важка ІІІ	18-20	40-60	0,4

Щоб дані параметри оцінювались як оптимальні, потрібно щоб середнє значення та результати вимірювань відрізнялися не більше ніж на 5% від зазначених значень. Щоб дані параметри оцінювались як допустимі, потрібно щоб середнє значення та результати вимірювань відрізнялися не більше ніж на 15% від зазначених значень. Значення вважаються як такі що не задовольняють умовам санітарних норм коли середнє значення та результати вимірювань відрізнялися від зазначених більше ніж на 15% [20].

Для забезпечення даних параметрів потрібно використовувати різні методи, один з них це вентиляція. Це можна зробити природнім та механічним способами. Природня вентиляція здійснюється за допомогою забезпечення припливу та відпливу повітря через вентиляційні канали, кватирки та фрамуги. Коли природної вентиляції не вистачає застосовується механічна, її реалізують за допомогою встановлення витяжних вентиляторів та систем примусової вентиляції для забезпечення постійного обміну повітря.

Системи опалення та кондиціонування є важливим аспектом для забезпечення комфортної температури. В холодний період року застосовують

системи опалення, вона може бути як централізованою, так і індивідуальною. Щоб не допустити великих втрат тепла застосовують заходи ізоляції приміщення, для цього утепляють стіни, вікна та двері за допомогою спеціальних матеріалів. В теплий період року використовують кондиціонери які охолоджують повітря до комфортних температур, також для забезпечення оптимальних значень вологості повітря в 40-60% використовують зволожувачі повітря.

Контроль освітлення та звукоізоляція є не менш важливим ніж попередні заходи для комфортного знаходження та виконання роботи в приміщенні. Найпростіший спосіб забезпечити оптимальний рівень освітлення це при плануванні встановити потрібну кількість вікон. Але якщо цього не було зроблено, то використовують штучне освітлення, яке повинно бути в межах 200-300 люкс. Зменшення рівня шуму в приміщенні забезпечується шляхом встановлення звукоізоляційних матеріалів, таких як мінеральна вата чи спеціалізовані поролонові панелі, на стіни, підлогу та стелю [20].

Без регулярного прибирання даних параметрів досягнути буде неможливо. Важливо регулярно проводити вологе прибирання, це дозволить запобігти накопиченню пилу та бруду. Для забезпечення чистоти в санітарних зонах важливо регулярно проводити дезінфекцію та очищення поверхонь, щоб зменшити кількість шкідливих бактерій.

Для багатьох приміщень проблемою є погана якість повітря. Для того щоб забезпечити якість повітря потрібно використовувати повітряні фільтри, які очищують повітря від пилу, алергенів та бактерій. Також важливим є моніторинг вмісту вуглекислого газу, вологи, температури за допомогою спеціальних пристроїв. При перевищені допустимих параметрів потрібно застосовувати заходи що забезпечують оптимальні метеорологічні умови в санітарно-побутових приміщеннях.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи бакалавра було розроблено комп’ютеризовану систему виявлення та аналізу тепловтрат будівлі на основі мікроконтролера ESP32-WROOVER-E, яка здатна виявляти аномалії в спорудах.

В роботі здійснено аналіз наявних систем виявлення тепловтрат, розроблено структурну схему системи, схему електричну принципову КСВАТБ, алгоритм роботи та програмне забезпечення, а також налаштування і тестування системи.

Використання сучасних технологій тепловізійної зйомки дозволило отримати точні результати щодо стану теплової ізоляції будівель, що, в свою чергу, сприяло ідентифікації найбільш проблемних зон. Завдяки цьому стало можливим прийняття обґрунтованих рішень щодо проведення необхідних заходів з теплоізоляції будівлі.

Розроблена комп'ютеризована система виявлення та аналізу тепловтрат будівель має значний потенціал для зниження енергоспоживання, покращення умов проживання та зменшення негативного впливу на навколишнє середовище.

Таким чином, результати цієї роботи сприятимуть підвищенню енергоефективності будівель, що є важливими факторами у досягненні екологічної стабільності та економічного розвитку, через зменшення використання ресурсів на опалення і збереження природніх ресурсів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Blower Door Test | ТОВ «Сахара». URL: https://caxapa.ua/posluhy-test-na-hermetychnist (дата звернення: 07.03.2024).
2. Тепловізійне обстеження – кращий метод діагностики якості утеплення будівель та споруд. URL: https://ppu.org.ua/uk/teplovizijne-obstezhennya/ (дата звернення: 09.03.2024).
3. How to Build Your Own Thermal Camera With a Raspberry Pi. URL: https://petapixel.com/2021/03/25/how-to-build-your-own-thermal-camera-with-a-raspberry-pi/ (дата звернення: 26.04.2024).
4. Infrared Thermal Imaging Camera With MLX90640 and ESP32 for Less Than 70 USD. URL: https://www.instructables.com/Infrared-Thermal-Imaging-Camera-With-MLX90640-and-/ (дата звернення: 27.04.2024).
5. DIY Thermal Imaging Infrared Camera. URL: https://www.instructables.com/DIY-thermal-imaging-infrared-camera/ (дата звернення: 02.29.2024).
6. Q-Techno online. URL: https://q-techno.com.ua/catalogsearch/ (дата звернення: 01.05.2024).
7. Testo be sure. URL: https://www.testo.kiev.ua/ua/testo-868.html (дата звернення: 02.05.2024).
8. What’s the difference between MLX90640 and MLX90641 thermal camera. URL: https://www.seeedstudio.com/blog/2020/06/11/whats-the-difference-between-mlx90640-and-mlx90641-thermal-camera/ (дата доступу: 03.05.2024).
9. Wi-Fi модуль ESP32-WROVER. URL: https://www.mini-tech.com.ua/ua/esp32-wrover-modul (дата звернення: 05.05.2024).
10. Матрична тепловізійна ІЧ камера MLX90640-D55. URL: https://uamper.com/Матрична_тепловізійна_ІЧ камера-MLX90640-D55 (дата звернення: 07.05.2024).
11. ILI9341 Datasheet. URL: https://www.alldatasheet.com/datasheet-pdf/pdf/1131760/ETC2/ILI9341.html (дата звернення: 10.05.2024).

12. Технології проєктування комп’ютерних систем. URL: https://dl.tntu.edu.ua/content.php?cid=298343 (дата звернення: 11.05.2024).

13. Швачич Г. Г., Пасинков В.М., Павленко Г.А. Побудова блок-схем. Дніпропетровськ: НМетФУ, 2004. – 24с.

14. Velychko D., Osukhivska H., Palaniza Y., Lutsyk N., Sobaszek L. Artificial Intelligence Based Emergency Identification Computer System. Advances in Science and Technology Research Journal, 18 no. 2, 2024, P. 296-304.

15. I2C інтерфейс. URL: https://itmaster.biz.ua/directory/standarts/i2c.html (дата звернення: 20.05.2024).

16. SPI protocol. URL: https://www.arduino.cc/reference/en/language/functions/communication/spi/ (дата звернення: 22.05.2024).

17. ESP32 Arduino (урок 2). URL: https://itmaster.biz.ua/electronics/esp32/esp32-arduino.html (дата звернення: 27.05.2024).

18. Palamar A., Karpinski M., Palamar M., Osukhivska H., Mytnyk M. Remote Air Pollution Monitoring System Based on Internet of Things. CEUR Workshop Proceedings, 2nd International Workshop on Information Technologies: Theoretical and Applied Problems (ITTAP 2022), Ternopil, Ukraine, November 22–24, 2022. Vol. 3309. P. 194-204.

19. Бедрій І.Я., Нечай В.Я. Безпека життєдіяльності. Навчальний посібник. Львів: Манголія 2006, 2007. 499 с.

20. ДСН 3.3.6.042 – 99 „Санітарні норми мікроклімату виробничих приміщень”.

21. Осухівська Г. М., Тиш Є. В., Луцик Н. С., Паламар А. М. Методичні вказівки до виконання кваліфікаційних робіт здобувачів першого (бакалаврського) рівня вищої освіти спеціальності 123 «Комп’ютерна інженерія» усіх форм навчання. Тернопіль, ТНТУ. 2022. 28 с.

22. Микитишин А. Г., Митник М. М., Стухляк П. Д., Пасічник В. В. Комп’ютерні мережі. Книга 2. [навчальний посібник]. Львів : "Магнолія 2006", 2014. 312 с.

Додаток А

Технічне завдання

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Тернопільський національний технічний університет імені Івана Пулюя

Кафедра комп’ютерних систем та мереж

«ЗАТВЕРДЖУЮ»

Завідувач кафедри КС

_____ Осухівська Г.М.

“_09_” _____02_____ 2024 р.

КОМП’ЮТЕРНА СИСТЕМА ВИЯВЛЕННЯ ТА АНАЛІЗУ ТЕПЛОВТРАТ БУДІВЛІ

ТЕХНІЧНЕ ЗАВДАННЯ

на __8__ листках

Вид робіт: Кваліфікаційна робота

На здобуття освітнього ступеня «Бакалавр»

Спеціальність 123 «Комп’ютерна інженерія»

«УЗГОДЖЕНО»

Керівник кваліфікаційної роботи

_____ к.т.н., доц., Осухівська Г. М.

“__09__” _____02_____ 2024 р.

«ВИКОНАВЕЦЬ»

Студент групи СІ-42

_____ Семців Т. А.

“__07__” ____02____ 2024 р.

Тернопіль 2024

1 Загальні відомості

1.1 Повна назва та її умовне позначення

Повна назва теми кваліфікаційної роботи бакалавра: «Комп’ютеризована система виявлення та аналізу тепловтрат будівлі».

Умовне позначення кваліфікаційної роботи бакалавра: КС КРБ 123.131.00.00

1.2 Виконавець

Студент групи СІ-42, факультету комп’ютерно-інформаційних систем і програмної інженерії, кафедри комп’ютерних систем та мереж, Тернопільського національного технічного університету імені Івана Пулюя, Семців Тарас Андрійович.

1.3 Підстава для виконання роботи

Підставою для виконання кваліфікаційної роботи бакалавра є наказ по університету №4/7-237 від 24 квітня 2024 року.

1.4 Планові терміни початку та завершення роботи

Плановий термін початку виконання кваліфікаційної роботи бакалавра – 29 січня 2024 року.

Плановий термін завершення виконання кваліфікаційної роботи бакалавра – 23 червня 2024 року.

1.5 Порядок оформлення та пред’явлення результатів роботи

Оформлення пояснювальної записки та графічного матеріалу для кваліфікаційної роботи бакалавра виконується згідно з діючими стандартами ДСТУ, ГОСТ, ISO, ЄСКД та ЕСПД. Технічна документація повинна містити пояснювальну записку і креслення, які чітко і стисло відображають основні результати роботи.

Проміжні результати роботи пред'являються відповідно до графіку, затвердженого керівником. Попередній захист проводиться, коли робота готова на 90% і є пояснювальна записка та графічний матеріал. Остаточний захист кваліфікаційної роботи відбувається на засіданні екзаменаційної комісії, де основні досягнення представляються за допомогою графічного матеріалу та готового пристрою.

2 Призначення і цілі створення системи

2.1 Призначення системи

Комп’ютеризована система виявлення та аналізу тепловтрат будівлі призначена для проведення тепловізійного моніторингу будівель різного призначення, таких як житлові будинки, офісні приміщення, промислові об'єкти, громадські установи тощо.

Система складається з мікроконтролера, дисплея та камери, які використовується для збору зображень теплового випромінювання від поверхонь будівлі. Зібрані дані дозволяють виявити зони з підвищеною тепловою емісією, що вказує на можливі проблеми з теплоізоляцією.

Користувачі можуть переглядати теплові зображення у реальному часі та фіксувати їх для подальшого аналізу і прийняття рішень щодо ремонту або модернізації будівлі з метою зниження тепловтрат.

2.2 Мета створення системи

Метою роботи є розроблення ефективної комп'ютеризованої системи, яка дозволить знизити тепловтрати будівель за рахунок точного виявлення проблемних зон. Досягнення цієї мети сприятиме підвищенню енергоефективності будівель та зменшенню витрат на опалення, що є актуальним завданням у контексті сучасних вимог до енергозбереження та екологічної безпеки.

Для досягнення мети буде виконано наступні завдання:

– провести аналіз наявних систем виявлення тепловтрат будівлі для ефективної реалізації власної;
– створити електричну схему для системи виявлення та аналізу тепловтрат будівлі;
– розробити алгоритм роботи та програмне забезпечення для системи;
– провести налаштування та тестування розробленої системи.

2.3 Характеристика об’єкту

Комп’ютеризована система проектується для можливості виявлення та аналізу тепловтрат будівлі, для розробки необхідно забезпечити:

– структурну схему;
– схему електричну принципову;
– алгоритм роботи коду програми;
– лістинг коду програми;
– електричні елементи та друковану плату

3 Вимоги до системи

3.1 Вимоги до системи в цілому

Комп’ютеризована система виявлення та аналізу тепловтрат будівлі повинна задовольняти таким основним умовам:

– Діапазон вимірювання температури: -40℃ – +400℃;
– Діапазон робочих температур: -30℃ – +45℃;
– Поле зору давача (FoV): не менше 45° ×30°;
– Частота оновлення: 4Гц – 32Гц
– Кольорові режими роботи:

1) синій, блакитний, зелений, жовтий, червоний, пурпуровий
2) чорний, темно-сірий, світло-сірий, білий, помаранчевий, рожевий
3) темно-синій, оливковий, зелений, блакитний, бордовий, фіолетовий
4) синій, зелений, темно-зелений, помаранчевий, бордовий, червоний
5) темно-синій, темно-зелений, зелений, жовтий, помаранчевий, червоний
6) блакитний, синій, пурпуровий, жовтий, зелений, червоний
7) білий, помаранчевий, червоний, синій, зелений, чорний
8) фіолетовий, пурпуровий, червоний, помаранчевий, жовтий, зелений
9) жовтий, рожевий, білий, синій, темно-блакитний, темно-зелений
10) червоний, жовтий, зелений, блакитний, синій, пурпуровий

– Простий інтерфейс;
– Можливість зробити та зберегти фото;
– Автономність роботи (до 5 годин);
– Розміри, не більше 10х10см.

3.2 Вимоги до структури та функціонування схеми

В загальному випадку, модель проектованої комп’ютеризованої системи виявлення та аналізу тепловтрат будівлі повинна забезпечувати реалізацію основних функцій щодо виявлення аномальних зон в будівлі та транслювати їх на екран пристрою.

До функцій комп’ютеризованої системи виявлення та аналізу тепловтрат будівлі належать:

– здатність отримувати дані з тепловізійної камери;
– отримання теплового зображення на дисплей;

– зміна режимів інтерполяції та кольорових режимів роботи;
– зберігання зображення на SD-карту.

3.3 Вимоги до способів та засобів зв’язку між компонентами система

Взаємодія камери та мікроконтролера повинна здійснюватися через протокол передачі даних I^2C. Взаємодія дисплея мікроконтролера повинна здійснюватися через протокол передачі даних SPI. Прошивка мікроконтролера повинна здійснюватися через протокол UART.

3.4 Вимоги по діагностуванню системи

Діагностування комп’ютеризованої системи виявлення та аналізу тепловтрат будівлі проводиться у відповідності до регламенту, визначеного організацією, що впроваджуватиме дану систему.

3.5 Перспективи розвитку, модернізації системи

До перспектив розвитку проектованої системи належить можливість надсилання даних на сервер через Wi-Fi, також пересилання даних на інший пристрій через Bluetooth.

3.6 Вимоги до надійності системи

Комп’ютеризована система виявлення та аналізу тепловтрат будівлі повинна забезпечувати працездатність своїх функцій при виникненні наступних ситуацій:
– при відсутності джерела зарядки батареї до 5 годин;
– при відсутності карти пам’яті;
– при температурах від -30℃ до + 45℃;

3.7 Вимоги до апаратного забезпечення

Основні вимоги до апаратних засобів системи:

– камера з матрицею тепловізійних інфрачервоних давачів роздільною здатністю не менше 24×24 пікселів.

– кольоровий дисплей з розширенням не менше 2 дюймів;

– можливість під’єднання накопичувача.

3.8 Вимоги до програмного забезпечення

Програмне забезпечення повинно включати функції для виконання таких дій:

– виведення зображення на екран;

– можливість отримання фото з екрану;

– зміна режимів інтерполяції та кольорових режимів роботи;

– можливість налаштування мінімальної та максимальної вимірюваних температур.

4 Вимоги до документації

Документація повинна відповідати вимогам ЄСДК та ДСТУ

Комплект документації повинен складатися з:

– пояснювальної записки;

– графічного матеріалу:

а) структурна схема,

б) схема електрична принципова,

в) блок-схема алгоритму роботи програми,

г) результати тестування

*Примітка: У комплект документації можуть вноситися зміни та доповнення в процесі розробки.

5. Стадії та етапи проектування

Таблиця 1 – Стадії та етапи виконання кваліфікаційної роботи бакалавра

№ з/п	Назва етапів роботи	Термін виконання етапів роботи
1	*Розробка технічного завдання*	01.02 – 09.02
2	*Аналіз технічного завдання*	05.02 – 11.02
3	*Аналіз технологій побудови систем виявлення та аналізу тепловтрат*	25.04 – 03.05
4	*Обґрунтування вибору апаратного забезпечення*	04.05 – 15.05
5	*Реалізація програмного забезпечення для системи*	16.05 – 01.06
6	*Реалізація апаратного забезпечення*	02.06 – 09.06
7	*Безпека життєдіяльності, основи охорони праці*	10.06 – 15.06
8	*Оформлення кваліфікаційної роботи*	16.06 – 20.06
9	*Попередній захист кваліфікаційної роботи*	14.06
10	*Захист кваліфікаційної роботи*	24.06 – 28.06

6 Додаткові умови виконання кваліфікаційної роботи

Під час виконання кваліфікаційної роботи у дане технічне завдання можуть вноситися зміни та доповнення.

Додаток Б

Перелік елементів

Поз. позначення	*Найменування*	*Кіл.*	*Примітка*
	Мікросхеми		
IC1	TopPower TP4056 4.2В 1А (ESOP-8)	1	
U1	AOS AO3401 4A 30В (SOT-23)	1	
U2	Microchip MIC5219-3.3YM5 1A (SOT23-5)	1	
U3	WCH CH340K (ESSOP-10)	1	
U4	ON Semi 2N7002DW (SOT-363)	1	
U5	Espressif ESP32-WROVER-E	1	
U6	Melexis MLX90640	1	
	Конденсатори		
C1, C4, C5, C8	AVX SMD 10uF 10В (A), 10%	4	
C2	SAMSUNG SMD 470pF 10 В (0603), 5%	1	
C3	YAGEO SMD 2.2uF 25В (0805), 10%	1	
C6, C7, C10, C11	YAGEO SMD 0.1uF 50В (0603), 10%	4	
C9	YAGEO SMD 1uF 25В (0603), 10%	1	
	Діоди		
D1	YANGJIE SS34 3А 40В (SMC)	1	
	Роз’єми		
J1	USB-C_Receptacle_USB2.0	1	
J2	Battery 602030 Li-pol 3.7В 300мА/год	1	
J3	TFT-LCD ILI9341 2.4`	1	
J4	ST_PH_B4B-PH-K_1x04	1	
SD1	MicroSD TF-01A	1	
	Світлодіоди		
LED1	Everlight SMD 20мА 3В (0603), Green	1	
LED2, LED3	Everlight SMD 20мА 3В (0603), Red	2	
	Транзистори		
Q1	CJ-Electronics S8050 1.5А 40В (SOT-23)	1	

					КС КРБ 123.131.00.00 ПЕ	
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*		
Розроб.		Семців Т. А			*Комп’ютеризована система виявлення та аналізу тепловтрат будівлі Перелік елементів*	*Літ.* / *Арк.* 1 / *Акрушів*
Перевір.		Осухівська Г.М				
Реценз.		Млинко Б. Б.				*ТНТУ, каф. КС, гр. СІ-42*
Н. контр.		Тиш Є. В.				
Зав. каїлф.		Осухівська Г. М				

	Резистори		
R1	ELZET SMD 470Ом 0.1Вт (0603), 5%	1	
R2, R3	ELZET SMD 1.3kОм 0.1Вт (0603), 5%	2	
R4, R5, R6, R21	YAGEO SMD 100kОм 0.1Вт (0603), 1%	4	
R7, R8, R13	YAGEO SMD 1kОм 0.1Вт (0603), 1%	3	
R9, R10, R11	YAGEO SMD 10kОм 0.1Вт (0603), 1%	3	
R12, R14	YAGEO SMD 10kОм 0.1Вт (0603), 1%	2	
R15	Viking SMD 10Ом 0.125Вт (0803), 5%	1	
R16, R17, R18	YAGEO SMD 4.7kОм 0.1Вт (0603), 1%	3	
R19, R20	YAGEO SMD 4.7kОм 0.1Вт (0603), 1%	2	
Rprog1	YAGEO SMD 3.3kОм 0.1Вт (0603), 1%	1	
	Перемикачі		
SW1	K3-1280S-F1	1	
SW2_2, SW2_3	TS-1088R-02526	2	
SW3, SW4, SW5	SW SPST	3	

Додаток В

Лістинг коду програми

```
#include <Preferences.h>
#include <Wire.h>
#include <SPI.h>
#include <SD.h>
#include "FS.h"
#include <Adafruit_MLX90640.h>
#include <TFT_eSPI.h>
#include <AnimatedGIF.h>
#include <Fonts/GFXFF/gfxfont.h>
#include "Open_Sans_ExtraBold_10.h"
SPIClass spiSD(HSPI);
#define SD_CS 15
#define NORMAL_SPEED
#define TFT_WIDTH 320
#define TFT_HEIGHT 240
#define BUFFER_SIZE 320
uint16_t usTemp[BUFFER_SIZE];
#define VBAT_PIN 33
#define BATTV_MAX 4.2
#define BATTV_MIN 3.2
bool dmaBuf = 0;
Adafruit_MLX90640 mlx;
Preferences preferences;
AnimatedGIF gif;
TFT_eSPI tft = TFT_eSPI();
File bmpFile;
float frame[32 * 24];
float batv;
const int upButton = 35;
const int middleButton = 36;
const int downButton = 39;
const int backlightPin = 4;
int interpolationMode = 0;
int AutoScale = 0;
volatile bool MenuChange = false;
volatile bool _upShort = false;
volatile bool _downShort = false;
volatile bool middleShort = false;
volatile bool middlePressed = false;
unsigned long middlePressStartTime = 0;
int Menu = 0, Menuitem = 0;
int BLPWM = 0;
int RefreshRate = 0;
float MinT, MaxT;
String menuItems[7] = { "Auto Scale    : ", "Min Temp      : ", "Max
Temp      : ", "Interpolation : ", "Palette       : ", "Refresh
Rate  : ", "BackLight     : " };
String ASindex[2] = { "Off", "On" };
```

```
String IPindex[6] = { "Nearest Neighbor", "Average", "Bilinear",
"Bilinear Fast", "Triangle" };
String RRindex[4] = { "4 Hz", "8 Hz", "16 Hz", "32 Hz" };
float xRatios[320];
float yRatios[240];
float xOppositeRatios[320];
float yOppositeRatios[240];
#define PALETTE_COUNT 10
uint16_t colorPalettes[PALETTE_COUNT][6] = {
  { TFT_BLUE, TFT_CYAN, TFT_GREEN, TFT_YELLOW, TFT_RED, TFT_MAGENTA
},
  { TFT_BLACK, TFT_DARKGREY, TFT_LIGHTGREY, TFT_WHITE, TFT_ORANGE,
TFT_PINK },
  { TFT_NAVY, TFT_OLIVE, TFT_DARKGREEN, TFT_DARKCYAN, TFT_MAROON,
TFT_PURPLE },
  { TFT_BLUE, TFT_GREEN, TFT_DARKGREEN, TFT_ORANGE, TFT_MAROON,
TFT_RED },
  { TFT_NAVY, TFT_DARKGREEN, TFT_GREEN, TFT_YELLOW, TFT_ORANGE,
TFT_RED },
  { TFT_CYAN, TFT_BLUE, TFT_MAGENTA, TFT_YELLOW, TFT_GREEN, TFT_RED
},
  { TFT_WHITE, TFT_ORANGE, TFT_RED, TFT_BLUE, TFT_GREEN, TFT_BLACK
},
  { TFT_PURPLE, TFT_MAGENTA, TFT_RED, TFT_ORANGE, TFT_YELLOW,
TFT_GREEN },
  { TFT_YELLOW, TFT_PINK, TFT_WHITE, TFT_BLUE, TFT_DARKCYAN,
TFT_DARKGREEN },
  { TFT_RED, TFT_YELLOW, TFT_GREEN, TFT_CYAN, TFT_BLUE, TFT_MAGENTA
}
};
int paletteIndex = 0;
float tempMin = 20.0;
float tempMax = 32.0;
TFT_eSprite sprite = TFT_eSprite(&tft);
void IRAM_ATTR upButton_ISR() {
  static unsigned long last_interrupt_time = 0;
  unsigned long interrupt_time = millis();
  if (interrupt_time - last_interrupt_time > 200) {
    _upShort = true;
  }
  last_interrupt_time = interrupt_time;
}

void IRAM_ATTR downButton_ISR() {
  static unsigned long last_interrupt_time = 0;
  unsigned long interrupt_time = millis();
  if (interrupt_time - last_interrupt_time > 200) {
    _downShort = true;
  }
  last_interrupt_time = interrupt_time;
}

void IRAM_ATTR middleButton_ISR() {
```

```
  if (digitalRead(middleButton) == LOW) {
    if (!middlePressed) {
      middlePressed = true;
      middlePressStartTime = millis();
    }
  } else {
    if (middlePressed) {
      if (millis() - middlePressStartTime < 1000) {
        middleShort = true;
      }
      middlePressed = false;
      middlePressStartTime = 0;
    }
  }
}

void MLXInit() {
  Wire.begin();
  Wire.setClock(1000000);
  if (!mlx.begin()) {
    Serial.println("Failed to initialize MLX90640!");
  }
  mlx.setResolution(MLX90640_ADC_18BIT);
  ConfigRefreshrate();
}

void ConfigRefreshrate() {
  switch (RefreshRate) {
    case 1:
      mlx.setRefreshRate(MLX90640_4_HZ);
      break;
    case 2:
      mlx.setRefreshRate(MLX90640_8_HZ);
      break;
    case 3:
      mlx.setRefreshRate(MLX90640_16_HZ);
      break;
    case 4:
      mlx.setRefreshRate(MLX90640_32_HZ);
      break;
    default:
      break;
  }
}

void ReadConfig() {
  preferences.begin("Config", false);
  String temp;
  temp = preferences.getString("AutoScale", "");
  AutoScale = temp.toInt();
  temp = preferences.getString("MinTe", "");
  MinT = temp.toFloat();
  temp = preferences.getString("MaxTe", "");
```

```
  MaxT = temp.toFloat();
  temp = preferences.getString("Imode", "");
  interpolationMode = temp.toInt();
  temp = preferences.getString("PIndex", "");
  paletteIndex = temp.toInt();
  temp = preferences.getString("RefreshRate", "");
  RefreshRate = temp.toInt();
  temp = preferences.getString("BLPWM", "");
  BLPWM = temp.toInt();
  preferences.end(); }
void WriteConfig() {
  preferences.begin("Config", false);
  preferences.putString("AutoScale", String(AutoScale));
  preferences.putString("MinTe", String(MinT));
  preferences.putString("MaxTe", String(MaxT));
  preferences.putString("Imode", String(interpolationMode));
  preferences.putString("PIndex", String(paletteIndex));
  preferences.putString("RefreshRate", String(RefreshRate));
  preferences.putString("BLPWM", String(BLPWM));
  preferences.end();
}

void initSDcard() {
  if (!SD.begin(SD_CS, spiSD)) {
    Serial.println("Card Mount Failed");
    return;
  } else {
    Serial.println("Card Mount Successful");
  }
  uint8_t cardType = SD.cardType();
  if (cardType == CARD_NONE) {
    Serial.println("No SD card attached");
    return;
  }
}
String generateFilename(fs::FS &fs) {
  for (int i = 0; i <= 9999; i++) {
    char filename[23];
    sprintf(filename, "/ThermalCamera%04d.bmp", i);
    if (!fs.exists(filename)) {
      return String(filename);
    }
  }
  return "";
}
int writeBMP(fs::FS &fs, const char *path, TFT_eSprite *sprite) {
  const int width = sprite->width();
  const int height = sprite->height();
  uint8_t bmpFileHeader[14] = { 'B', 'M', 0, 0, 0, 0, 0, 0, 0, 0,
54, 0, 0, 0 };
  uint32_t fileSize = 54 + width * height * 2;
  bmpFileHeader[2] = (uint8_t)(fileSize);
  bmpFileHeader[3] = (uint8_t)(fileSize >> 8);
```

```
  bmpFileHeader[4] = (uint8_t)(fileSize >> 16);
  bmpFileHeader[5] = (uint8_t)(fileSize >> 24);
  uint8_t bmpInfoHeader[40] = { 40, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 0, 16, 0 };
  bmpInfoHeader[4] = (uint8_t)(width);
  bmpInfoHeader[5] = (uint8_t)(width >> 8);
  bmpInfoHeader[6] = (uint8_t)(width >> 16);
  bmpInfoHeader[7] = (uint8_t)(width >> 24);
  bmpInfoHeader[8] = (uint8_t)(height);
  bmpInfoHeader[9] = (uint8_t)(height >> 8);
  bmpInfoHeader[10] = (uint8_t)(height >> 16);
  bmpInfoHeader[11] = (uint8_t)(height >> 24);
  File file = fs.open(path, FILE_WRITE);
  if (!file) {
    Serial.println("Failed to open file for writing");
    return 0;
  } else {
    file.write(bmpFileHeader, 14);
    file.write(bmpInfoHeader, 40);
    for (int y = height - 1; y >= 0; y--) {
      for (int x = 0; x < width; x++) {
        uint16_t pixel = sprite->readPixel(x, y);
        uint16_t r = (pixel >> 11) & 0x1F;
        uint16_t g = (pixel >> 5) & 0x3F;
        uint16_t b = pixel & 0x1F;
        pixel = (b << 11) | (r << 5) | g;
        file.write(pixel >> 8);
        file.write(pixel);
      }
    }
    file.close();
    return 1;
  }
}

void navigationUpdate() {
  if (_upShort == true) {
    _upShort = false;
    if (Menu == 0) {
      interpolationMode++;
      if (interpolationMode > 4) interpolationMode = 0;
    } else {
      if (Menuitem == 0) {
        AutoScale++;
        if (AutoScale > 1) AutoScale = 1;
      } else if (Menuitem == 1) {
        MinT++;
        if (MinT > 300) MinT = 300;
      } else if (Menuitem == 2) {
        MaxT++;
        if (MaxT > 300) MaxT = 300;
      } else if (Menuitem == 3) {
        interpolationMode++;
```

```
      if (interpolationMode > 4) interpolationMode = 4;
    } else if (Menuitem == 4) {
      paletteIndex++;
      if (paletteIndex > 9) paletteIndex = 9;
    } else if (Menuitem == 5) {
      RefreshRate++;
      if (RefreshRate > 3) RefreshRate = 3;
      ConfigRefreshrate();
    } else if (Menuitem == 6) {
      BLPWM = BLPWM + 10;
      if (BLPWM > 100) BLPWM = 100;
      analogWrite(backlightPin, map(BLPWM, 0, 100, 0, 255));
    }
    MenuChange = true;
  }
  WriteConfig();
  Serial.println("Up Short Press");
}
if (_downShort == true) {
  _downShort = false;
  if (Menu == 0) {
    paletteIndex = (paletteIndex + 1) % PALETTE_COUNT;
  } else {
    if (Menuitem == 0) {
      AutoScale--;
      if (AutoScale < 0) AutoScale = 0;
    } else if (Menuitem == 1) {
      MinT--;
      if (MinT < 0) MinT = 0;
    } else if (Menuitem == 2) {
      MaxT--;
      if (MaxT < 5) MaxT = 5;
    } else if (Menuitem == 3) {
      interpolationMode--;
      if (interpolationMode < 0) interpolationMode = 0;
    } else if (Menuitem == 4) {
      paletteIndex--;
      if (paletteIndex < 0) paletteIndex = 0;
    } else if (Menuitem == 5) {
      RefreshRate--;
      if (RefreshRate < 0) RefreshRate = 0;
      ConfigRefreshrate();
    } else if (Menuitem == 6) {
      BLPWM = BLPWM - 10;
      if (BLPWM < 10) BLPWM = 10;
      analogWrite(backlightPin, map(BLPWM, 0, 100, 0, 255));
    }
    MenuChange = true;
  }
  WriteConfig();
  Serial.println("down Short Press");
}
if (middleShort == true) {
```

```
    middleShort = false;
    if (Menu == 0) {
      if (!SD.begin(SD_CS, spiSD)) {
        Serial.println("Card Mount Failed");
        if (gif.open((uint8_t *)EGIF_IMAGE, sizeof(EGIF_IMAGE),
GIFDraw1)) {
          Serial.printf("Successfully opened GIF; Canvas size = %d x
%d\n", gif.getCanvasWidth(), gif.getCanvasHeight());
          while (gif.playFrame(true, NULL)) {
            sprite.pushSprite(0, 0);
            yield();
          }
          gif.close();
        }
        return;
      } else {
        Serial.println("Card Mount Successful");
      }
      uint8_t cardType = SD.cardType();
      if (cardType == CARD_NONE) {
        Serial.println("No SD card attached");
        if (gif.open((uint8_t *)EGIF_IMAGE, sizeof(EGIF_IMAGE),
GIFDraw1)) {
          Serial.printf("Successfully opened GIF; Canvas size = %d x
%d\n", gif.getCanvasWidth(), gif.getCanvasHeight());
          while (gif.playFrame(true, NULL)) {
            sprite.pushSprite(0, 0);
            yield();
          }
          gif.close();
        }
        return;
      } else {
        String filename = generateFilename(SD);
        if (filename != "") {
          if (writeBMP(SD, filename.c_str(), &sprite)) {
            if (gif.open((uint8_t *)SGIF_IMAGE, sizeof(SGIF_IMAGE),
GIFDraw1)) {
              Serial.printf("Successfully opened GIF; Canvas size =
%d x %d\n", gif.getCanvasWidth(), gif.getCanvasHeight());
              while (gif.playFrame(true, NULL)) {
                sprite.pushSprite(0, 0);
                yield();
              }
              gif.close();
            }
          } else {
            if (gif.open((uint8_t *)EGIF_IMAGE, sizeof(EGIF_IMAGE),
GIFDraw1)) {
              Serial.printf("Successfully opened GIF; Canvas size =
%d x %d\n", gif.getCanvasWidth(), gif.getCanvasHeight());
              while (gif.playFrame(true, NULL)) {
                sprite.pushSprite(0, 0
```

```
              yield();
            }
            gif.close();
          }
        }
      } else {
        Serial.println("Failed to create filename.");
        if (gif.open((uint8_t *)EGIF_IMAGE, sizeof(EGIF_IMAGE),
GIFDraw1)) {
          Serial.printf("Successfully opened GIF; Canvas size = %d
x %d\n", gif.getCanvasWidth(), gif.getCanvasHeight());
          while (gif.playFrame(true, NULL)) {
            sprite.pushSprite(0, 0);
            yield();
          }
          gif.close();
        }
      }
    }
  } else {
    Menuitem++;
    if (Menuitem > 6) {
      Menuitem = 0;
    }
    MenuChange = true;
  }
  Serial.println("Middle Short Press");
  Serial.print(Menu);
  Serial.print("");
  Serial.println(Menuitem);
 }
 if (middlePressed && !middleShort && millis() -
middlePressStartTime > 1000 && middlePressed && !middleShort &&
millis() - middlePressStartTime < 1500) {
  Serial.println("Middle Long Press");
  middlePressed = false;
  middlePressStartTime = 0;
  if (Menu == 0) {
    Menu = 1;
    MenuChange = true;
    Menuitem = 0;
  } else {
    Menu = 0;
  }
 }
}

void displayUpdate() {
 if (!mlx.getFrame(frame)) {
  MLXInit();
 }
 float tempMinRead = 1000;
 float tempMaxRead = -1000;
```

```
  float tempCenter = 0.0;
  for (int i = 0; i < 32 * 24; i++) {
    if (frame[i] < tempMinRead) {
      tempMinRead = frame[i];
    }
    if (frame[i] > tempMaxRead) {
      tempMaxRead = frame[i];
    }
    if (i == 32 * 12 + 16) {
      tempCenter = frame[i];
    }
  }
  sprite.fillSprite(TFT_BLACK);
  if (interpolationMode == 0) {
    for (int y = 0; y < 240; y++) {
      int yIndex = (y / 10) * 32;
      for (int x = 0; x < 320; x++) {
        float val = frame[yIndex + (x / 10)];
        drawPixel(319 - x, y, val);
      }
    }
  } else if (interpolationMode == 1) {
    for (int y = 0; y < 240; y++) {
      int yIndex = (y / 10) * 32;
      int yNextIndex = ((y / 10) + 1) * 32;
      for (int x = 0; x < 320; x++) {
        int xIndex = x / 10;
        float val = (frame[yIndex + xIndex] + frame[yIndex + xIndex
+ 1] + frame[yNextIndex + xIndex] + frame[yNextIndex + xIndex + 1])
/ 4.0;
        drawPixel(319 - x, y, val);
      }
    }
  } else if (interpolationMode == 2) {
    for (int y = 0; y < 240; y++) {
      int yIndex = (y / 10) * 32;
      float y_ratio = (y % 10) / 10.0;
      float y_opposite_ratio = 1 - y_ratio;
      for (int x = 0; x < 320; x++) {
        float x_ratio = (x % 10) / 10.0;
        float x_opposite_ratio = 1 - x_ratio;
        int x_over_10 = x / 10;
        float val = y_opposite_ratio * (x_opposite_ratio *
frame[yIndex + x_over_10] + x_ratio * frame[yIndex + x_over_10 + 1])
+ y_ratio * (x_opposite_ratio * frame[(yIndex + 32) + x_over_10] +
x_ratio * frame[(yIndex + 32) + x_over_10 + 1]);
        drawPixel(319 - x, y, val);
      }
    }
  } else if (interpolationMode == 3) {
    int yIndex;
    int xIndex;
    int yNextIndex;
```

```
    for (int y = 0; y < 240; y++) {
      yIndex = (y / 10) * 32;
      yNextIndex = yIndex + 32;
      for (int x = 0; x < 320; x++) {
        xIndex = x / 10;
        float val = yOppositeRatios[y] * (xOppositeRatios[x] *
frame[yIndex + xIndex] + xRatios[x] * frame[yIndex + xIndex + 1]) +
yRatios[y] * (xOppositeRatios[x] * frame[yNextIndex + xIndex] +
xRatios[x] * frame[yNextIndex + xIndex + 1]);
        drawPixel(319 - x, y, val);
      }
    }
  }
  else if (interpolationMode == 4) {
    for (int y = 0; y < 240; y++) {
      int yIndex = (y / 10) * 32;
      int yNextIndex = ((y / 10) + 1) * 32;
      for (int x = 0; x < 320; x++) {
        int xIndex = x / 10;
        float t = (x % 10) / 10.0f;
        float u = (y % 10) / 10.0f;
        float val;
        if (t > u) {
          val = (1 - t) * frame[yNextIndex + xIndex] + (1 - u) *
frame[yIndex + xIndex + 1] + (t + u - 1) * frame[yNextIndex + xIndex
+ 1];
        } else {
          val = (1 - t) * frame[yIndex + xIndex] + (1 - u) *
frame[yNextIndex + xIndex + 1] + (t + u - 1) * frame[yIndex + xIndex
+ 1];
        }
        drawPixel(319 - x, y, val);
      }
    }
  }

  sprite.setTextColor(TFT_BLACK);
  sprite.setTextSize(1);
  sprite.setFreeFont(&Open_Sans_ExtraBold_10);
  sprite.drawString("T Min: " + String(tempMinRead) + " C", 15,
220);
  sprite.drawString("T Max: " + String(tempMaxRead) + " C", 220,
220);
  sprite.drawString("Tc: " + String(tempCenter, 1) + "C", 135, 220);
  sprite.drawLine(155, 120, 165, 120, TFT_WHITE);
  sprite.drawLine(160, 115, 160, 125, TFT_WHITE);
  detachInterrupt(digitalPinToInterrupt(downButton));
  batv = ((float)analogRead(VBAT_PIN) / 4095) * 2 * 1.07 * 3.3;
  attachInterrupt(digitalPinToInterrupt(downButton), downButton_ISR,
CHANGE);
  int batpc = (uint8_t)(((batv - BATTV_MIN) / (BATTV_MAX -
BATTV_MIN)) * 100);
  drawBattery(batpc);
```

```
  if (Menu == 1) {
    MenuChange = false;
    sprite.fillRect(60, 40, 200, 120, TFT_BLACK);
    sprite.fillRect(60, 40, 200, 10, TFT_WHITE);
    sprite.setTextFont(0);
    sprite.setTextSize(1);
    sprite.setTextColor(TFT_BLACK);
    sprite.drawString("Settings", 140, 41);
    sprite.setTextColor(TFT_WHITE);
    for (int i = 0; i < 7; i++) {
      if (i == Menuitem) sprite.setTextColor(TFT_GREEN);
      int y = 55 + (15 * i);
      sprite.drawString(menuItems[i], 65, y);
      if (i == 0) {
        sprite.drawString(ASindex[AutoScale], 160, y);
      } else if (i == 1) {
        sprite.drawString(String(MinT) + " C", 160, y);
      } else if (i == 2) {
        sprite.drawString(String(MaxT) + " C", 160, y);
      } else if (i == 3) {
        sprite.drawString(IPindex[interpolationMode], 160, y);
      } else if (i == 4) {
        sprite.drawString("Palette " + String(paletteIndex), 160,
y);
      } else if (i == 5) {
        sprite.drawString(RRindex[RefreshRate], 160, y);
      } else if (i == 6) {
        sprite.drawString(String(BLPWM) + " %", 160, y);
      }
      sprite.setTextColor(TFT_WHITE);
    }  }
  sprite.pushSprite(0, 0);
}

void drawPixel(int x, int y, float val) {
  int colorIndex;
  if (AutoScale == 1) {
    if (val <= tempMin) {
      colorIndex = 0;
    } else if (val >= tempMax) {
      colorIndex = 5;
    } else {
      colorIndex = int(map(val, tempMin, tempMax, 0, 5));
    }
  } else {
    if (val <= MinT) {
      colorIndex = 0;
    } else if (val >= MaxT) {
      colorIndex = 5;
    } else {
      colorIndex = int(map(val, MinT, MaxT, 0, 5));
    }
  }
```

```
  int color = colorPalettes[paletteIndex][colorIndex];
  sprite.drawPixel(x, y, color);
}
void drawBattery(int batpc) {
  int x = 290;
  int y = 10;
  int w = 20;
  int h = 10;
  int color = TFT_RED;
  int fillLevel = 0;
  if (batpc > 75) {
    fillLevel = w;
    color = TFT_GREEN;
  } else if (batpc > 50) {
    fillLevel = w * 3 / 4;
    color = TFT_GREEN;
  } else if (batpc > 25) {
    fillLevel = w / 2;
    color = TFT_BLUE;
  } else if (batpc > 0) {
    fillLevel = w / 4;
    color = TFT_BLUE;
  }
  sprite.drawRect(x, y, w, h, TFT_WHITE);
  sprite.drawRect(x + w, y + h / 4, 2, h / 2, TFT_WHITE);
  if (fillLevel > 0) {
    sprite.fillRect(x + 1, y + 1, fillLevel - 2, h - 2, color);
  }  }
void setup() {
  Serial.begin(115200);
  tft.init();
#ifdef USE_DMA
  tft.initDMA();
#endif
  tft.setRotation(1);
  tft.fillScreen(TFT_BLACK);
  gif.begin(BIG_ENDIAN_PIXELS);
  sprite.createSprite(320, 240);
  pinMode(upButton, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(upButton), upButton_ISR,
CHANGE);
  pinMode(downButton, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(downButton), downButton_ISR,
CHANGE);
  pinMode(middleButton, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(middleButton),
middleButton_ISR, CHANGE);
  ReadConfig();
  MLXInit();
  if (MinT == 0 && MaxT == 0) {
    MinT = 20.0;
    MaxT = 32.0;
    WriteConfig();
```

```
  }
  if (BLPWM == 0) {
    BLPWM = 100;
    WriteConfig();  }
  pinMode(backlightPin, OUTPUT);
  analogWrite(backlightPin, map(BLPWM, 0, 100, 0, 255));
  if (gif.open((uint8_t *)GIF_IMAGE, sizeof(GIF_IMAGE), GIFDraw)) {
    Serial.printf("Successfully opened GIF; Canvas size = %d x
%d\n", gif.getCanvasWidth(), gif.getCanvasHeight());
    while (gif.playFrame(true, NULL)) {
      sprite.pushSprite(0, 0);
      yield();  }
    gif.close();
  }
  delay(1000);
  initSDcard();
  for (int x = 0; x < 320; x++) {
    xRatios[x] = (x % 10) / 10.0f;
    xOppositeRatios[x] = 1 - xRatios[x];
  }
  for (int y = 0; y < 240; y++) {
    yRatios[y] = (y % 10) / 10.0f;
    yOppositeRatios[y] = 1 - yRatios[y];
  }
}

void loop() {
  navigationUpdate();
  displayUpdate();
}
```