**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО”**

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**До захисту допущено:**
**Завідувач кафедри**
_____ Сергій СТІРЕНКО
(підпис)
“__” _____2022 р.

# Дипломний проект

**на здобуття ступеня бакалавра**
**за освітньо-професійною програмою “Комп’ютерні системи та мережі”**
**спеціальності 123 “Комп’ютерна інженерія”**

на тему: Система контролю стану приміщення при пандемії

Виконав: студент 4 курсу, групи ІО-81
(шифр групи)

Карпенко Ярослав Русланович
(прізвище, ім’я, по батькові) (підпис)

Керівник Виноградов Юрій Миколайович
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант (нормоконтроль) проф., д. т. н. Сімоненко В. П.
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище та ініціали)(підпис)

Рецензент _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

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

Київ – 2022 р.

## АННОТАЦІЯ

У даній дипломній роботі була розроблена система контролю стану приміщення при пандемії. Даний застосунок дозволяє отримувати та аназізувати дані, отримані із спеціальних апаратних засобів та виявляти людей, які порушують масковий режим чи мають високу температуру тіла.

Система розроблена за допомогою спеціальних інструментів, та сервісів що спеціалізуються на розробці подібного програмного забезпечення, а саме мови програмування Python, її модулів та бібліотек Keras/Tensorflow, Google-Vision API, PyQt5, opencv-python, telepot у середовищі розробки PyCharm.

## ANNOTATION

In this thesis, a system for monitoring the state of the premises during a pandemic was developed. This application allows you to obtain and analyze data obtained from special hardware and detect people who violate the mask or have a high body temperature.

The system is developed using special tools and services that specialize in the development of such software, namely Python programming languages, its modules and libraries Keras / Tensorflow, Google-Vision API, PyQt5, opencv-python, telepot in the PyCharm development environment.

**Національний технічний університет України**
**«Київський політехнічний інститут»**
**ім. Ігоря Сікорського**
Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

“Комп’ютерні системи та мережі”

спеціальність 123 «Комп’ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ С.Г. Стіренко
(підпис) (ініціали, прізвище)
“___”_____2022 р.

# ЗАВДАННЯ

на бакалаврський дипломний проєкт студента

Карпенка Ярослава Руслановича
(прізвище, ім’я, по батькові)

1. Тема проекту (роботи) *Система контролю стану приміщення при пандемії* ,
керівник проекту (роботи) ст.вик. Виноградов Юрій Миколайович
( прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
Затверджені наказом по університету від “___”_____20__ р. №____

2. Термін здачі студентом закінченого проекту (роботи) _____

3. Вихідні дані до роботи *наукова та технічна документація про реалізацію окремих модулів системи для вибору найоптимальнішого рішення щодо реалізації поставленої задачі з розробки програмного забезпечення.*

4. Зміст пояснювальної записки: *опис предметної області і напрямків дослідження, аналіз та характеристика об’єкту досліджень, аналіз існуючих рішень, опис архітектури системи.*

5. Перелік графічного матеріалу (із зазначенням обов’язкових креслеників, плакатів, презентацій тощо):

*Структурна схема системи, функціональна схема класів, принципова схема системи.*

6. Консультанти розділів проекту

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| Нормоконтроль | д.т.н., проф. Сімоненко В. П. | | |

7. Дата видачі завдання _____

Календарний план

| № з/п | Назва етапів виконання дипломного проекту | Термін виконання етапів проєкту | Примітка |
|---|---|---|---|
| 1. | Затвердження теми роботи | 11.12.2021-15.12.2021 | |
| 2. | Вивчення та аналіз завдання | 15.12.2021-15.02.2021 | |
| 3. | Розробка архітектури та загальної структури системи | 15.02.2022-10.03.2022 | |
| 4. | Розробка структур окремих підсистем | 10.03.2022-11.04.2022 | |
| 5. | Програмна реалізація системи | 11.04.2022-23.04.2022 | |
| 6. | Оформлення пояснювальної записки | 30.05.2022 | |
| 7. | Захист програмного продукту | 08.06.2022-10.06.2022 | |
| 8. | Передзахист | 11.06.2022 | |
| 9. | Захист | 22.06.2022 | |

Студент _____ Карпенко Я.Р.
(підпис) (ініціали, прізвище)

Керівник проекту _____ Виноградов Ю.М.
(підпис) (ініціали, прізвище)

| № рядка | Формат | Позначення | Найменування | Кільк. | Примітка |
|---|---|---|---|---|---|
| | | | Документація загальна новорозроблена | | |
| | | | | | |
| | | | | | |
| | А4 | ІАЛЦ.467200.001 ОА | Система контролю стану приміщення при пандемії<br>Опис альбому | 2 | |
| | | | | | |
| | А4 | ІАЛЦ.467200.002 ТЗ | Система контролю стану приміщення при пандемії<br>Технічне завдання | 3 | |
| | | | | | |
| | А4 | ІАЛЦ.467200.003 ПЗ | Система контролю стану приміщення при пандемії<br>Пояснювальна записка | 61 | |
| | | | | | |
| | А4 | ІАЛЦ.467200.004 Д1 | Система контролю стану приміщення при пандемії<br>Структурна схема | 1 | |
| | | | | | |
| | А4 | ІАЛЦ.467200.005 Д2 | Система контролю стану приміщення при пандемії<br>Функціональна схема класів | 1 | |
| | | | | | |
| | А4 | ІАЛЦ.467200.006 Д3 | Система контролю стану приміщення при пандемії<br>Принципова схема | 1 | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | ІАЛЦ.467200.001 ОА |
| Зм. | Арк. | № докум. | Підпис | Дата | |
| Розробив | | Карпенко Я.Р. | | | Система контролю стану приміщення при пандемії<br>Опис альбому |
| Перевірив | | ВиноградовЮ.М | | | |
| Реценз. | | | | | |
| Н. контр. | | Сімоненко В. П. | | | |
| Затвердив | | | | | |

| Літ. | Аркуш | Аркушів |
|---|---|---|
| | 1 | 2 |

КПІ ім. Ігоря Сікорського, ФІОТ, ІО-81

| № рядка | *Формат* | Позначення | Найменування | Кільк. | Примітка |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| | | | | | |
| | А4 | ІАЛЦ.462619.007 Д4 | Система контролю стану приміщення при пандемії Програмний код | 4 | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | ІАЛЦ.467200.001 ОА | Арк |
| | | | | | | 2 |
| Зм. | Арк. | № докум. | Підпис | Дата | | |

# ТЕХНІЧНЕ ЗАВДАННЯ
## До дипломного проекту
## на тему: «Система контролю стану приміщення при пандемії»

Київ – 2022

# ЗМІСТ

## 1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання поширюється на розробку системи контролю стану приміщення при пандемії.

Метою розробки системи є її використання у повсякденному житті та створення альтернативи для схожих систем.

## 2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання бакалаврського дипломного проекту, затверджене кафедрою обчислювальної техніки Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

## 3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного проєкту є розробка системи контролю стану приміщення при пандемії.

## 4. ДЖЕРЕЛА РОЗРОБКИ

Джерелами розробки є науково-технічна література, технічна документація технологій, що використовувались для розробки системи, публікації в мережі Інтернет.

## 5. ТЕХНІЧНІ ВИМОГИ

### 5.1. Вимоги до програмного продукту, що розробляється

Система, що розроблюється, повинна відповідати поставленим вимогам, що наведені нижче:

- Простий і зрозумілий інтерфейс системи;
- Технічна коректність;
- Можливість аналізування, обробки та оперування даними, отриманими з датчиків;
- Швидкий доступ до системи;

### 5.2. Вимоги до інструментального програмного забезпечення

Для розробки проекту на локальному ПК необхідне наступне програмне забезпечення:

- Операційна система Windows;
- Мова програмування Python версії 3.8.0;
- Бібліотеки та сервіси Python Keras/Tensorflow, Google-Vision API, PyQt5, opencv-python, telepot;
- Середовище розробки PyCharm;

## 6. ЕТАПИ РОЗРОБКИ

| Назва етапів виконання | Термін виконання |
|---|---|
| Затвердження теми роботи | 11.12.2021-15.12.2021 |
| Вивчення та аналіз завдання | 15.12.2021-15.02.2021 |
| Розробка архітектури та загальної структури системи | 15.02.2022-10.03.2022 |
| Розробка структур окремих підсистем | 10.03.2022-11.04.2022 |
| Програмна реалізація системи | 11.04.2022-23.04.2022 |
| Оформлення пояснювальної записки | 30.05.2022 |
| Захист програмного продукту | 08.06.2022-10.06.2022 |
| Передзахист | 11.06.2022 |
| Захист | 22.06.2022 |

# ПОЯСНЮВАЛЬНА ЗАПИСКА
## до дипломного проекту
## на тему: «Система контролю стану приміщення при пандемії»

Київ – 2022

# ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

IoT – (Internet of things) Інтернет речей

ГП – (Edge Device) граничний пристрій

ГО – (Edge Computing) граничні обчислення

ТО – (Fog Computing) туманні обчислення

API – (Application Programming Interface) прикладний програмний інтерфейс

GPU - (Graphics Processing Unit) графічний процесор

ПЗ – програмне забезпечення

Wi-Fi – (Wireless Fidelity) стандарт бездротового підключення

HDMI – (High Definition Multimedia Interface) мультимедійний інтерфейс високої чіткості

IIoT - (Industrial IoT) Промисловий IoT, мережа об’єднаних комп’ютерних мереж та підключених до них промислових об’єктів (із датчиками та ПЗ)

ПМ – (PAN) персональна мережа, мережа побудована «навколо» людини.

ЛМ - (LAN) локальна комп’ютерна мережа, утворена об'єднанням певного числа комп'ютерів на відносно невеликій території.

ООП – об’єктно-орієнтоване програмування

GUI – (graphical user interface) Графічний інтерфеейс кориистувача

БД – база даних

# ВСТУП

На сьогоднішній день, мабуть, кожному відомо, що немає більшої цінності, ніж життя людини. На жаль, від початку поширення інфекції вірусу COVID-19 і дотепер ці життя втрачаються.

Проте, людина не належить до тих істот, що звикли здаватися. Для пошуку рішень проблеми були використані чималі сили та ресурси. Вчені наполегливо працювали над винайденням вакцин. Результати не змусили себе довго чекати: темпи захворювань скоротилися у багато разів та загроза підхоплення інфекції вже не лякає людей так, як раніше. Однак, не всі готові до вакцинації: хтось відмовляється через страх захворіти від самої вакцини, комусь щеплення забороняється віросповіданням, хтось знаходить у цьому особисті мотиви… І не дивно, адже людина тому й називається людиною, що не завжди хоче діяти за шаблоном. З іншої сторони, у багатьох країнах світу досі не відмінили масковий режим у громадських місцях. І це варте того, бо науково доведено, що носячи маску, вірогідність як заразитися штамом коронавірусу, так і заразити когось, значно зменшується. Але перевірити наявність маски у кожної людини в громадських місцях вручну стає, мабуть, невиконуваною задачею. Добре, що сьогодні майже всі проблеми можна вирішити шляхом застосування техніки та розуму людини.

У цій роботі буде розглянуто систему контролю стану приміщення при пандемії, яка працює за принципом IoT. Вона здатна робити аналіз даних, отриманих з допомогою спеціальних приладів, характерних для подібних досліджень та оперувати цими даними в режимі реального часу.

# РОЗДІЛ 1.
# АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Обґрунтування предметної області

Перша ідея створення системи розумних приладів розглядалася ще в 1982 році. В Університеті Карнегі-Меллона був розміщений торговий автомат з содовою від Coca-Cola, що був з'єднаний із мережею ARPANET [1][2]. Він вважається пристроєм-початківцем у світі інтернету речей. Пристрій давав інформацію про стан завантажених напоїв і обладнання. Сучасний задум IoT допомогли сформувати за допомогою статті Марка Вайзера «Комп'ютер 21-го століття», яку він опублікував 1991 року, та наукових платформ на кшталт UbiComp і PerCom [3][4]. У 1994 році Реза Раджі у журналі IEEE Spectrum представив замисел IoT як «переміщення невеликих пакетів даних до великого набору вузлів, задля інтеграції та автоматизації улього: від домашньої техніки до фабрик»[5]. Компанії Microsoft at Work та Novell's NEST запропонували подібні рішення протягом 1993-1997 років. Концепція розвивалася, коли Білл Джой передбачив зв'язок між пристроями в рамках своєї структури «Six Webs», яку було представлено на Всесвітньому економічному форумі в Давосі в 1999 році [6].

Поняття та ідея «Інтернету речей» вперше прозвучала на виступі Пітера Т. Льюїса 1985 року у Вашингтоні на 15-му щорічному законодавчому вікенді в Конгресі США [7]. Він описав IoT так: «це об'єднання людей, процесів і технологій з підключеними пристроями та сенсорами, задля забезпечення віддаленого спостереження, статусу, маніпулювання та оцінки тенденцій таких пристроїв».

Твердження «Інтернет речей» незалежно придумав Кевін Ештон, коли він працював із Procter & Gamble в 1999 році, хоча на його думку більшої ваги має фраза «Інтернет для речей» [8][9]. Тоді він працював над

RFID - радіочастотній ідентифікації, та розглядав її як необхідну в сфері Інтернету речей, [10] бо це дало б можливість обчислювальним машинам керувати всіма окремими пристроями [11][12][13]. Основною ідеєю Інтернету речей є інтегрування мобільних приймачів ближнього радіусу дії в різні гаджети і повсякденні потреби, задля забезпечення нових форми спілкування між людьми та пристроями і навіть між самими пристроями [14].

У 2004 році Корнеліус «Піт» Петерсон, генеральний директор NetSilicon, передбачив, що «в наступній ері інформаційних технологій будуть домінувати пристрої IoT, а мережеві пристрої в кінцевому підсумку наберуть популярності і значення в тій мірі, в якій вони набагато перевищать кількість мережевих обчислювальних машин та робочих станцій». Петерсон вважав, що медичні вироби та промисловий контроль стануть домінуючим застосуванням технології [15].

## 1.2 Огляд архітектури інтернету речей

### 1.2.1 Екосистема інтернету речей

Екосистема IoT включає такі сервіси, технології і засоби, що можуть тим чи іншим способом використовуватися в IoT. Вони діляться на такі категорії:

- Глобальна обчислювальна мережа;
- Локальні обчислювальні мережі (LAN);
- Сервіси аналізу даних;
- Сенсори, смарт датчики, давачі;
- Накопичувачі, маршрутизатори, шлюзи, ГП;
- Системи зв’язку з сенсорами;
- Хмара (Cloud);
- Безпека (security).

В цілому елементи IoT можна розділити на чотири категорії: обладнання (Hardware), правила (Rules), програмне забезпечення (Software), сервіси (Services). Ці категорії, у свою чергу, можна розділити на підкатегорії за областю використання (див. Рис. - 1.1).

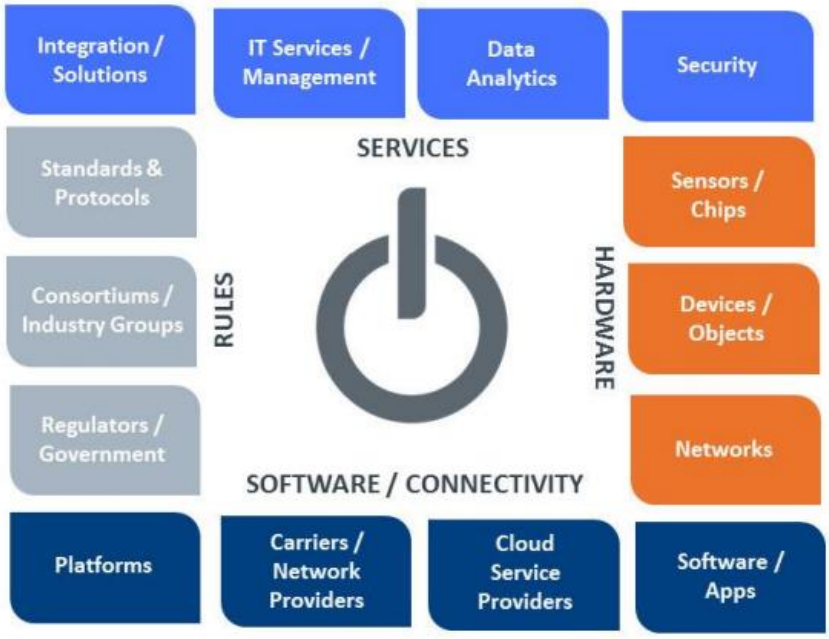



Рис. - 1.1 Зв'язок між елементами інтернету речей [14]

### 1.2.2 Огляд архітектури інтернету речей

У спрощеному варіанті архітектура системи IoT складається з трьох шарів: пристрій (Things), граничний шлюз (Edge Gateway), хмара (Cloud) (Рис1.2). [16]

Пристрої включають елементи мережі, такі як датчики (sensors) та приводи (actuators) в пристроях IoT , особливо ті, які підключені до Edge Gateway за допомогою таких протоколів, як Modbus, Bluetooth, Zigbee або власні протоколи. [16]

Рівень пограничного шлюзу складається із систем агрегації сенсорних даних, які називаються Edge Gateways, які забезпечують такі функції, як попередня обробка, підключення до хмари, системи, такі як WebSockets, центри подій, а в деяких випадках — аналітика чи

протитуманні обчислення. [16] Також існує потреба в граничних шлюзах, щоб забезпечити загальний огляд пристрою для верхніх рівнів задля легкості керування.

Останній рівень включає хмарні програми, створені для IoT з використанням архітектури мікросервісів. До нього входять різні системи баз даних, що зберігають дані отримані з датчиків. [16]

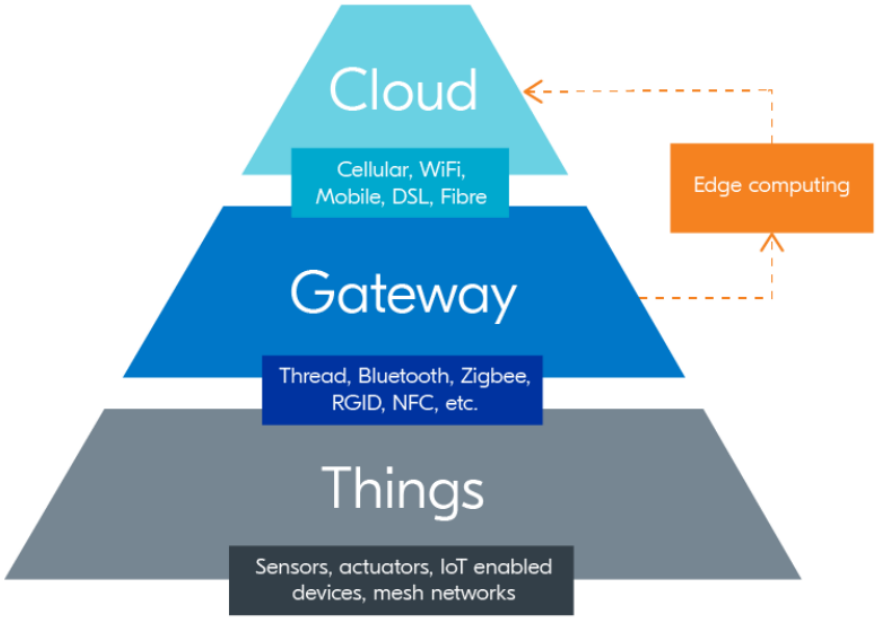



Рис. - 1.2 Приклад архітектури IoT [16]

### 1.2.3 Огляд компонентів архітектури інтернету речей

**Датчики та живлення.**

Інтернет речей починається якимсь процесом, якоюсь дією, нею і закінчується: зміною температури, простим рухом, чи навіть важелем, замикаючим замок. IoT відрізняється від безлічі існуючих сучасних пристроїв із IT-сфери саме тим, що в першу чергу стосується фізичних дій або подій. Не дивно, що це утворює реакцію на певні фактори нашого світу. Цікаво, що один сенсор може генерувати великі обсяги даних, такі як акустичні датчики обладнання профілактичних оглядів. З іншої сторони достатньо лише невеликої кількості даних, щоб послати важливу інформацію щодо здоров'я пацієнта (Рис. - 1.3). Незалежно від ситуації,

сенсорні системи розвиваються та навіть скоротилися до субнанометрів і стали дешевшими (згідно із законом Мура). Це те, що приваблює людей, які передбачають об'єднання безлічі пристроїв у Інтернету речей. Не дивно, що ці прогнози справджуються.

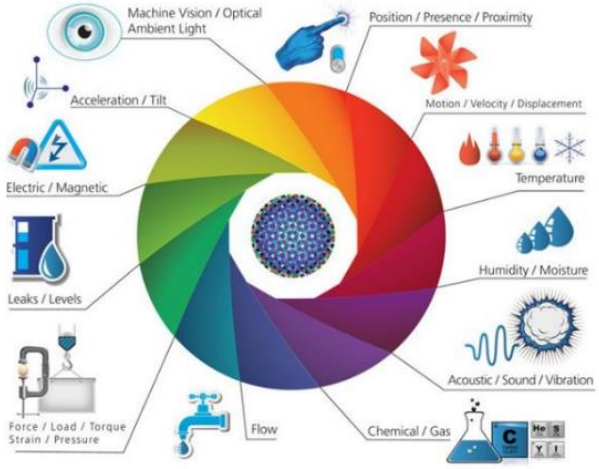



Рис. - 1.3 Схематичне зображення сил та явищ, для обробки яких може використовуватись інтернет речей [5]

Тому, досліджуючи IoT, важливим аспектом являється прийняття до уваги сенсорів, мікроелектромеханічнічних систем, та інші види дешевих ГП та електрофізичні властивості цих пристроїв. Це також стосується силових і енергетичних систем, що потрібні для роботи цих ГП. Не правильним також є твердження що ГП забезпечуються енергією за замовчуванням. Безліч датчиків маленьких розмірів досі можуть вимагати багато енергії. На питання живлення в IoT також має значний вплив організація cloud-сервісів.

**Передача даних.**

Розробляючи IoT багато уваги слід приділяти встановленню з'єднання та роботі мереж. У IoT не було б успіхів не якби не існувало надійної технології передачі даних, здатної комунікувати від найвіддаленіших і найнеблагополучніших областей до найбільших центрів обробки даних таких компаній як Google, Amazon, Microsoft та IBM.

У словосполученні «Інтернет речей» слово «Інтернет» опинилося не просто так, зважаючи на це важливим аспектом буде вивчення проблем, пов’язаних із обміном даними, мережевими технологіями, і навіть теорією сигналів. Найважливішою умовою для роботи IoT є не сенсори чи ПЗ, а можливість встановлення якісного з’єднання у системі.

Мережеві з'єднання та перенесення даних що утворені із застосуванням систем зв'язку невеликого радіусу - ПМ, в основному не створюються за правилами IP-протоколу. До них входять як мережі із дротовим зв’язком так і радіомережі. Бездротові мережі/протоколи IoT зазвичай включають Bluetooth, mesh, Zigbee, Z-Wave (Рис. - 1.4), в той час як IIoT крім цього підтримує Wireless Hart та ISA100.[17]

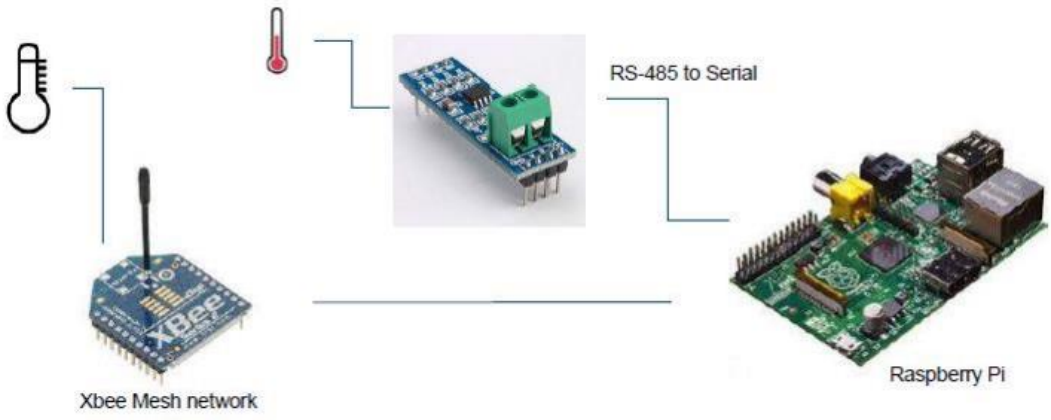



Рис. - 1.4 Приклад пристроїв передачі даних в мережі PAN [17]

В придачу до ПМ використовуються бездротові ЛМ та системи зв’язку, утворені на основі протоколу IP, включаючи різні мережі Wi-Fi на основі технологій IEEE 802.11, 6LoWPAN та Thread. Телеком оснований на базі стандартів стільникового зв’язку (3G, 4G LTE) і нових стандартів, що підтримують IoT та сумісність, таких як стандарти Cat-1 та Cat-NB, і крім цього власних протоколів Sigfox і LoRaWAN, присвячених IoT (Рис. - 1.5) застосовується досить часто. [16]

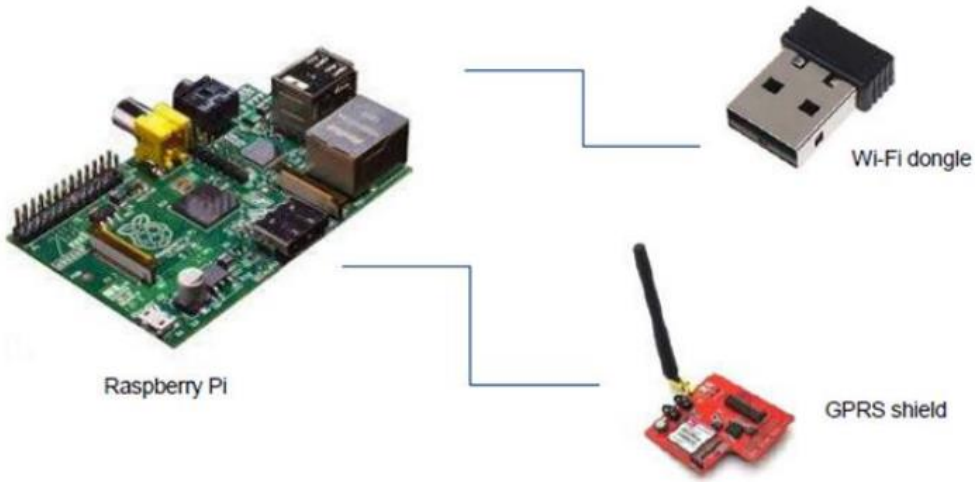



Рис. - 1.5 Приклади пристрої для передачі даних в мережі LAN [16]

**Маршрутизація.**

Щоб передавати дані, які були отримані від датчиків, до Інтернету потрібні дві технології: маршрутизатор-шлюзу та базові Інтернет-протоколи, які забезпечують ефективний обмін даними. Маршрутизатори особливо важливі з точки зору безпеки, керування та спрямування даних. Граничні маршрутизатори керують і контролюють стан пов'язаної mesh-мережі, а також налаштовують та підтримують якість даних. Конфіденційність і безпека даних також важливі. Маршрутизатори мають важливе значення при створенні віртуальних приватних, локальних та програмно-визначених глобальних мереж. Вони нерідко можуть мати тисячі вузлів, які обслуговуються граничним маршрутизатором, який являється розширенням для хмари.

На даному рівні багато протоколів використовуються для обміну даними між вузлами, маршрутизаторами та хмарними сервісами в системі IoT. IoT проклав шлях для нових протоколів IoT, які можна порівняти з традиційними протоколами HTTP і SNMP, які використовуються протягом десятиліть. Для передачі даних IoT потрібні ефективні та енергоефективні протоколи з низькою затримкою, які можуть легко та безпечно надсилати дані в хмару та з хмари. Зокрема, тут використовуються такі протоколи, як MQTT, AMPQ та CoAP.

**Туманні і граничні обчислення, аналітика і машинне навчання.**

На даному етапі вирішується, що робити з потоком даних з ГП в хмарний сервіс. Щоб зрозуміти, як правильно оцінити, як система буде розвиватися і рости, необхідно зрозуміти всі нюанси архітектури хмарної системи і як затримка впливає на систему IoT. Ба більше, не все слід відправляти в хмару. Транспортування всіх даних Інтернету речей набагато дорожче, ніж обробка даних на межі мережі (ГО) або включення граничних маршрутизаторів (ТО) у зоні хмарного обслуговування. ТО також стандартизовані, особливо за допомогою стандартів туманних обчислень, таких як архітектура OpenFog.

Дані, отримані шляхом перетворення аналогових фізичних ефектів в цифрові сигнали, можуть бути дуже важливими. Саме тут в дію вступають інструменти аналітики та процесори правил IoT. Складність впровадження системи IoT залежить від розробленого рішення. У деяких випадках це просто: наприклад, коли вам потрібно встановити простий процесор правил на регулюючий маршрутизатор, який стежить за кількома датчиками, щоб відстежувати аномальні коливання температури. Інший випадок: коли велика кількість структурованих та неструктурованих даних передається в хмарне озеро в режимі реального часу, для чого треба мати високу швидкість обробки (для прогнозного аналізу) та довгострокового прогнозування на основі високотехнологічних моделей машинного навчання, на кшталт рекурентних нейронних мереж у відповідному пакеті аналізу сигналів та згідно кореляції часу. Існують певні проблеми та труднощі з аналітикою, що можна подолати, використовуючи різні підходи, такі як формування складних обробників подій, байєсівських мереж та нейронних мереж.

**Загрози і безпека в IoT.**

Багато систем IoT не обмежуються безпечним домом або офісним приміщенням. Вони можуть бути у громадських місцях, дуже віддалених

районах, у транспортних засобах, що рухаються, і навіть всередині людей. IoT є величезною єдиною мішенню для злому всіх видів. Було виявлено численні навчальні атаки на пристрої IoT, добре організовані хакерські атаки і навіть вразливості в системах національної безпеки. Розробники рішень IoT повинні розуміти характеристики таких вразливостей і методи їх усунення, стандартні заходи захисту IoT або будь-якого компонента мережі.

### 1.2.4 Огляд технологій та протоколів передачі даних у мережі інтернету речей

Так як концепція IoT передбачає, що елементи знаходяться в безпосередній близькості і не можуть бути підключені до постійного джерела енергії, то такі пристрої вимагають специфічних технологій і певних протоколів передачі даних. Ось такі популярні на даний момент технології:

- Z-Wave - радіотехнологія, здатна працювати на частотах до 1ГГц, застосовується щоб передавати прості команди з малою затримкою;
- BLE – це варіант технології Bluetooth, але має високу енергоефективність;
- Wi-Fi HaLow - працює в частотах 900 МГц, споживає електроенергію на рівні технології Bluetooth і має вищу швидкість передачі даних, ніж Bluetooth;
- NFC - передає дані на відстань до 20 см;
- RFID - технологія ідентифікації за радіосигналами.

### 1.2.5 Протоколи для передачі повідомлень в мережі інтернету речей

Обсяг інформації, що генерується одним сенсорним вузлом, відносно невеликий, але більшість сервісів IoT влаштовано за принципом обробки інформації від великої кількості клієнтів або датчиків, що

кардинально відрізняється від архітектури, яку використовують традиційні мережі.

Тож ми зіткнулися з новою архітектурою: багато джерел - багато приймачів, крім того, трафік від сенсорних вузлів може бути дуже малим або дуже великим. У цьому випадку не можна використовувати звичайний протокол програми передачі повідомлень.

Основна топологія передачі повідомлень в IoT-мережі називається “видавець-підписник” (Publisher-Subscriber) (див. Рис. - 1.6). Ця схема вводить поняття джерела інформації-видавця та одержувача інформації-підписника. Термін підписка пов’язаний з конкретною дією, яку виконує учасник для того, щоб підписник отримав інформацію від конкретного видавця, а також спрощенням збору інформації – параметрами частоти отримання та подібними (залежно від реалізації) показниками.

Тому розглянемо випадок, коли сенсорний вузол (Node) об’єднує інформацію від кількох датчиків і надсилає її відповідно до параметрів підписки або на запит, чи самостійно через якийсь інтервал часу. Часто самі сенсори дуже примітивні і їх завдання зводиться до постійної передачі інформації про контрольований параметр. Тому необхідно об’єднати датчики у вузли, оснащені мікроконтролерами, які будуть відповідати за зчитування даних вимірювань і відправку їх далі на сервер за заздалегідь заданим алгоритмом.

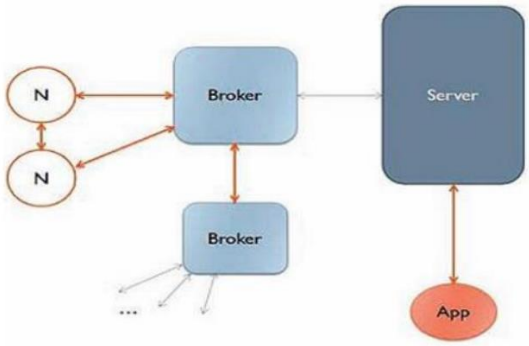



Рис. - 1.6 Топологія системи, яка виконує передачу даних між різними вузлами мережі IoT [17]

Також в більшості випадків при взаємодії клієнта з системою на персональний пристрій необхідно встановити клієнтську програму (Application), яка буде чітко відображати інформацію, отриману від датчиків, або вже оброблену сервером управління системою інформацію. Ця топологія також призначена для включення брокера (Broker). Це сервер, який отримує інформацію від видавця та доставляє її відповідним передплатникам. У складних системах брокер також можу виконувати різні операції, пов’язані з аналізом та обробкою даних, отриманих сервером. Брокери можуть визначати пріоритети з'єднань і формувати черги повідомлень. Таким чином, брокер організовує пересилання, зберігання та фільтрацію повідомлень. Черга повідомлень - контейнер або блок, в якому зберігаються повідомлення під час передачі. Якщо ресурсів каналу зв’язку недостатньо, або одержувач недоступний під час відправлення повідомлення, черга зберігає повідомлення, доки воно не буде доставлено одержувачу.

Протоколи, які використовує IoT:

- DDS – застосовуваний для систем реального часу протокол прикладного рівня M2M;
- XMPP – протокол для миттєвого обміну повідомленнями та інформацією про присутність для режиму, що близький режиму реального часу;
- CoAP – протокол для мереж та пристроїв із обмеженими ресурсами.
- MQTT – легкий, компактний, відкритий протокол обміну даними, призначений щоб передавати дані на віддалені локації;
- STOMP – формує зв'язок з брокером за методом "запитвідповідь";
- SOAP - це протокол для обміну структурованими та довільними повідомленнями XML у розподілених обчислювальних середовищах.

## 1.2 Огляд існуючих систем контролю стану приміщення

Додатки для боротьби з коронавірусом включають мобільні застосунки для цифрового моніторингу контактів – процес визначення людей, які якимось чином могли мати контакт із інфікованою людиною («контакти»), що були розгорнуті під час пандемії.

Багато програм для такого моніторингу були розроблені або запропоновані, і вони мають офіційну державну підтримку в деяких регіонах і юрисдикціях. Для створення програм відстеження контактів було розроблено кілька фреймворків. Виникали питання щодо проблеми конфіденційності, здебільшого для систем, заснованих на трекінгу географічного розташування користувачів програми.

Менш очевидні інвазивні альтернативи включають загальну версію передачі сигналу Bluetooth для запису близькості розташування користувача до інших мобільних телефонів. У квітні 2020 року Google і Apple спільно оголосили, що вони додають в свої операційні системи Android та iOS відповідно функціональні можливості для підтримки таких додатків безпосередньо на основі Bluetooth.

### 1.3.1 Aarogya Setu

Aarogya Setu (Рис. - 1.7) — це індійський цифровий сервіс «Відстеження контактів, картування симптомів та самооцінка» COVID-19, мобільний додаток, розроблений Національним центром інформатики при Міністерстві електроніки та інформаційних технологій (MeitY)[21].

Мета даного застосунка — підвищити обізнаність народу Індії про COVID-19 та інтегрувати головні медичні послуги відносно COVID-19. Цей додаток доповнює ініціативи Міністерства охорони здоров’я відносно стримування розповсюдження COVID-19 та поширює найкращі практики та поради. Це додаток для смартфона для відстеження, який реалізує можливості GPS і Bluetooth для трекінгу випадків COVID-19. Застосунок

використовується для мобільних ОС Android та iOS. Використовуючи Bluetooth, він намагається вирахувати, чи загрожує людині опинитися поблизу (у межах близько двох метрів) від пацієнта з COVID-19, аналізуючи БД випадків зараження що вже сталися на території Індії. Користуючись інформацією місцезнаходження людей, він зсилаючись на достовірні дані з'ясовує, чи являється положення свого користувача однією із заражених областей.[18]

Даний додаток є оновленням попередньо випущеного урядом Індії застосунку під назвою Corona Kavach (наразі його робота припинена)[19]. Aarogya Setu має п'ять функцій:

- Повідомляє про ризик зараження користувача на COVID-19;
- Дає інформацію про оновлені випадки зараження COVID-19 локально та по всій країні;
- Допомогає юзерам самостійно з'ясувати симптоми зараження COVID-19 і їх статус ризику;
- Додає пропуск E-pass;
- Дає можливість користувачам визначити рівень ризику зараження від своїх контактів через Bluetooth.

Додаток показує, яка кількість заражень COVID-19 може бути в радіусі від 500 метрів до 10 км від того хто користується додатком.

Додаток побудовано на платформі, яка може надавати API. Це означає, що будь-які інші мобільні чи комп'ютерні застосунки та інтернет-сервіси мають здатність користуватися даними та функціями, отриманими з Aarogya Setu.[20]

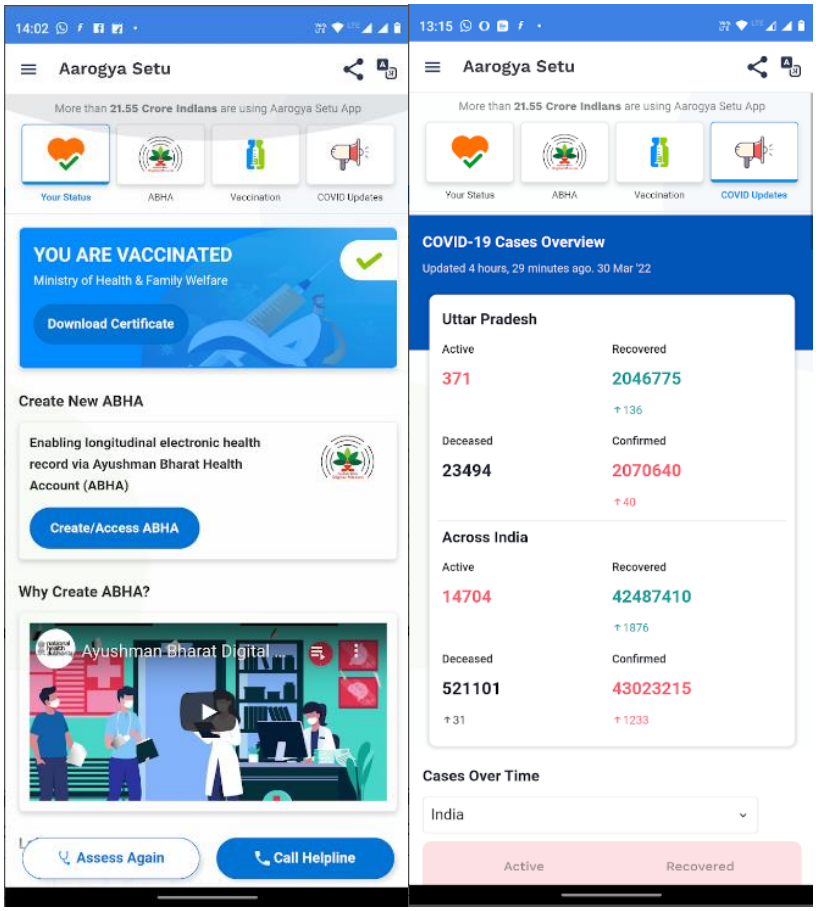



Рис. - 1.7 Інтерфейс додатку «Aarogya Setu» [20]

### 1.3.2 Act at home («Дій вдома»)

Додаток розроблено для того, щоб підтримувати зв’язок з людьми та стежити за дотриманням обов’язкової самоізоляції під пандемії (Рис 1.8). Він базується на закордонному досвіді країн, що використовують цифрові інструменти задля забезпечення здоров’я громадян під час пандемії. Застосунок надає можливість отримання пільг для громадян під час самоізоляції. Додаток безкоштовний для користування.[22]

Згідно із законом України, особи, що перетинають кордон нашої держави, мають вибрати або переміщення до спеціалізованого ізолятору, або самоізоляцію відповідно місця проживання для тих, хто підтвердив рішення пройти її використовуючи застосунок «Дій вдома» протягом чотирнадцяти днів.

Функції програми «Дій вдома» такі:

- Підтвердження домашнього карантину із з’ясуванням місцезнаходження особи;
- Фільмування перебування особи в місці самоізоляції;
- Виклик екстренної допомоги через гарячу лінію МОЗ нашої країни;
- Моніторингу розвитку симптомів хвороби.

Обравши домашній карантин з використанням застосунку «Дій вдома», особа повинна бути готова до контролю використовуючи регулярні повідомлення з необов'язковими інтервалами протягом дня (вночі таких повідомлень не буде) і фейсконтроль людини відповідно до фото, зробленого під час установки додатку, а також місцезнаходження користувача в цей момент. Якщо юзер отримує повідомлення, йому необхідно сфотографуватися фоні свого оточення протягом 15 хвилин, тому користувач завжди має тримати свій телефон поруч. [22]

Якщо ви вирішили самоізолюватися за допомогою додатку «Дій вдома», ви повинні підтвердити це коли будете проходити паспортний контроль, вказавши свої дані (телефон та адресу, де буде проходити карантин), а потім показавши відповідно скрін заяви із додатку працівнику Держприкордонслужби.[22]

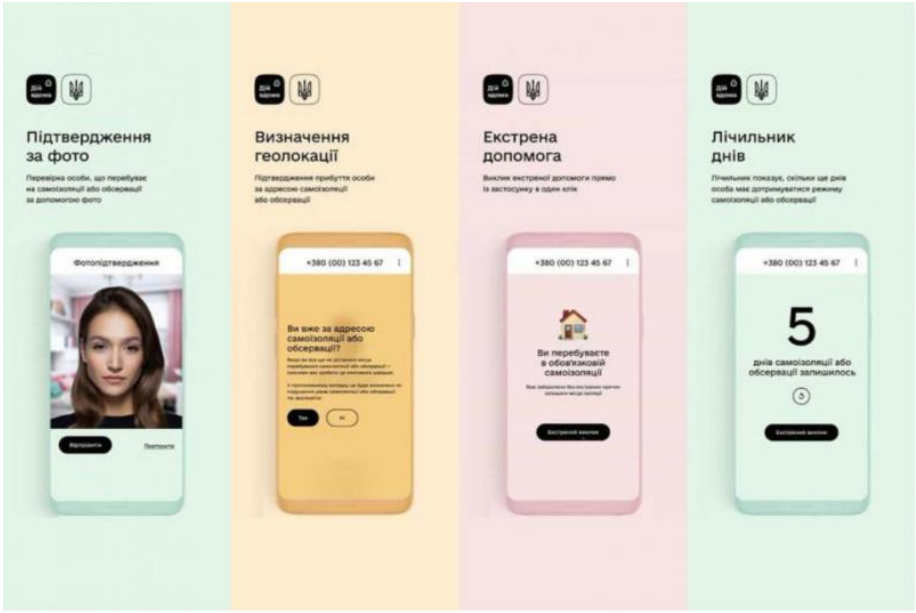



Рис 1.8 Інтерфейс та функціїї додатку «Act at home» [24]

### 1.3.3 NHS COVID-19

NHS COVID-19 - це добровільний додаток для відстеження контактів для моніторингу поширення пандемії COVID-19 в Англії та Уельсі. Він доступний з 24 вересня 2020 року для смартфонів Android та iOS, і ним може користуватися будь-хто у віці від 16 років і старше.

Було створено дві версії програми. Перша була замовлена NHSX і розроблена ключовим підрозділом американської програмної компанії VMware. Пілотне розгортання було розпочато в травні 2020 року, але 18 червня розробка програми була припинена на користь другого дизайну з використанням системи сповіщення про експозицію Apple/Google. Шотландія та Північна Ірландія мають окремі програми для відстеження контактів.[23]

Додаток використовує перевірене програмне забезпечення для конфіденційності, розроблене Apple і Google, розроблене таким чином, щоб ніхто не знав, хто юзер і де він знаходиться. Користувач може видалити свої дані або додаток у будь-який час.

Функції програми NHS COVID-19 (Рис. - 1.9):

- Сповіщення: Дізнайтеся, коли ви були поруч з іншими користувачами програми, які з тих пір дали позитивний результат на коронавірус.
- Остання інформація: Дізнайтеся останні вказівки щодо COVID-19 та знайте рівень ризику коронавірусу у вашому регіоні. [24]
- Симптоми: Перевірте, чи є у вас симптоми коронавірусу, і подивіться, чи потрібно вам замовити тест.
- Тест: Зареєструйте свої результати, якщо ви дали позитивний результат.

Додаток можна використовувати під час подорожей по Англії, Уельсу, Північній Ірландії, Джерсі та Гібралтару, виявленні користувачів програми відстеження контактів (незалежно від того, чи вони

використовують різні офіційні програми), і попереджаючи їх, якщо вони контактували з кимось, хто дав позитивний результат на COVID-19 від ПЛР-тесту.

Додаток був створений у співпраці з деякими з найбільш інноваційних організацій у світі, медичними експертами, групами конфіденційності, спільнотами ризику.

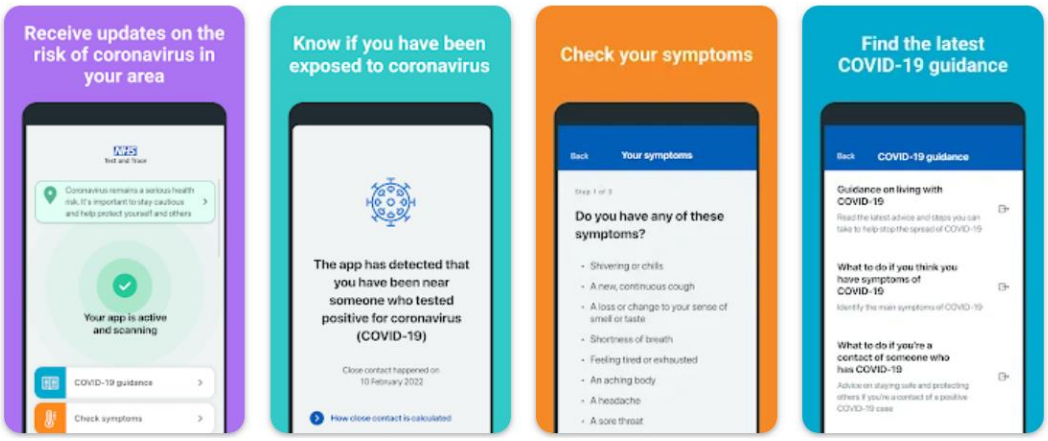



Рис. - 1.9 Інтерфейс та функціїї додатку «NHS COVID-19» [24]

## Висновок до розділу 1

У розділі було розглянуто та проаналізовано архітектуру мережі IoT. Представлено основні протоколи, які використовуються для побудови таких мереж. Виділилися техногогії BLE та NFC, що використовуються для комунікації на малій відстані, та мають найкращі показники таких характеристик як швидкість передачі даних та енергоспоживання. Щоб передавати інформацію на середніх відстанях краще скористатися Wi-Fi HaLow так як маючи енергоспоживання на рівні технології Bluetooth він має більшу швидкість обміну даними та обширніший радіус дії.

Також розглянуто існуючі системи контролю. Серед яких можна виділити NHS COVID-19, оскільки він не має вікових обмежень, підтримує велику кількість пристроїв, виконує основні необхідні функціїї, а саме: сповіщення, остання інформація, симптоми, тест.

# РОЗДІЛ 2.
# ВИБІР АПАРАТНОЇ ЧАСТИНИ ТА ЗАСОБІВ РОЗРОБКИ, ЇХ ПОРІВНЯЛЬНИЙ АНАЛІЗ

## 2.1 Опис вибраної платформи Raspberry Pi та допоміжних пристроїв

### 2.1.1 Raspberry Pi

Raspberry Pi model B - одноплатний ПК, що на початку свого існування призначався для вивчення у школах базових принципів комп'ютерних наук (Рис. - 2.1). З Прогресом у розвитку мобільних технологій, став чи не найулюбленішою платформою для втілень розумових здібностей ентузіастів, що розробляють застосунки для IoT. Платформа має значні переваги, аби користуватися саме нею: ціна, універсальність та відкритість, простота розробки під неї.[25]

Рис. - 2.1. Зовнішній вигляд плати RaspberryPi [18]

Характерні можливості вибранної платформи:

- ПК має можливість працювати на базі Unix-подібних систем;
- Підтримує розширення відео Full HD;
- Об’єм ОЗУ від 512 МБ;
- Габаритні розміри плати 85,6 * 54 * 17 мм;
- Підключення монітору, користуючись відеовиходом RCA і HDMI
- Для підключення аудіопристроїв має 3.5-міліметровий роз’єм

Живлення плати може відбуватися декількома способами. Перший - це з допомогою роз`єма micro-USB, який приймає 5В. Другий - це через універсальні піни. Третій, через адаптер живлення на 5В і 2А. Перші два способи не гарантують стабільне живлення при підключенні великої кількості переферійних пристроїв та датчиків. Третій спосіб найстабільніший, оскільки надає хорошу енергоємність для пристроїв що будуть підключатись.

Плата підтримує велику кількість інтерфейсів(рис 2.2):

- MIPI CSI-2
- UART
- 10/100Mb RJ45 Ethernet
- 16 портів input/output 3.3В
- HDMI
- 3.5мм стерео
- DSI
- Композитний відеовихід
- Слот для карт SD, MMC, SDIO
- Wi-Fi 802.11n
- I2C і SPI
- ARM JTAG
- 2x USB 2.0
- Bluetooth 4.1

Інтерфейси радіозв'язку Wi-Fi 802.11n і Bluetooth 4.1 підтримуються окремою мікросхемою Broadcom BCM43438, яка доставляється на плату через інтерфейс Shield або окремими модулями, що підключаються як периферія.

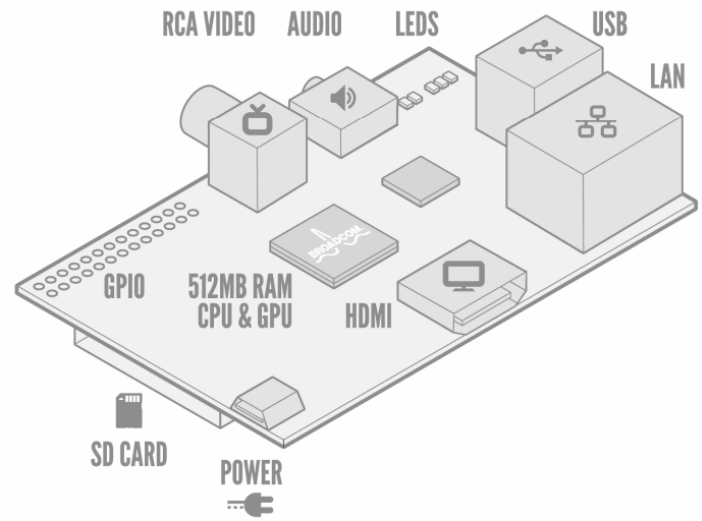



Рис. - 2.2. Схема розміщення інтерфейсів на платі [18]

**Процесор.**

Raspberry Pi B використовує 32-розрядний процесор ARM Cortex-A7 900 МГц. Він має 16 КБ кешу рівня 1 (L1) і 128 КБ кешу рівня 2 (L2). Кеш L2 в основному використовується графічним процесором. SoC знаходиться нижче чіпа RAM, тому його не видно. [18]

**Продуктивність.**

При роботі на частоті 700 МГц Raspberry Pi забезпечує реальну продуктивність, приблизно еквівалентну 0,041 ГФЛОПС.[26] На рівні ЦП швидкодія подібна Pentium II який має 300 МГц 1997-99. GPU здатний видавати 1 Гпіксель/с або 1,5 Гтексель/с обробки графіки або 24 ГФЛОПС загальної обчислювальної продуктивності.

**Програмне забезпечення.**

Raspberry Pi в основному працює на операційній системі на основі ядра Linux. Також можна встановити Windows 10 IoT. ARM11 заснований на ARM версії 6, яка завжди підтримує всі типи Linux. Для встановлення операційної системи існує інструмент NOOBS або інший подібний інструмент для запису образу операційної системи на носій, який дозволяє записати операційну систему на карту пам'яті, сумісну з Raspberry Pi.[26]

### 2.1.2 Flir Lepton 2.0 Thermal Camera

Камеру FLIR Lepton 2.0 (Рис. -2.3) можна назвати проривом у світі тепловізійних пристроїв із точки зору розмірів, енергоспоживання і вартості. На даний момент камера має роздальну здатність 80x60 пікселів у корпусі розміром 8,5 x 11,7 x 5,6 мм із номінальним енергоспоживанням 150 мВт. Частота кадрів дорівнює 9 Гц у діапазоні довжин хвиль спектрального відклику 8-14 мікрон номінально. Має горизонтальне поле зору (HFOV): 25˚, 50˚ та 50˚ із затвором. Необхідний цифровий інтерфейс I2C для установки регістру команд та управління і SPI для захвату зображення.

Завдяки даним характеристикам цей модуль може безконтактно вимірювати температуру на відносно великих відстанях, відстежувати присутність людини чи тварини в темний час доби, розпізнавати жести для безконтактного управління (використовується у ігровій індустрії та в смарт-будинках).

Рис. - 2.3 Зовнішній вигляд камери FLIR Lepton 2.0 [19]

## 2.2 Установка та налаштування Raspberry Pi

### 2.2.1 Підготовка карти пам’яті.

Так як у Raspberry Pi немає вбудованої постійної пам'яті, для роботи комп'ютера попередньо необхідно підготувати карту пам'яті – форматувати для подальшого використання.

Скористаємося програмою SD Memory Card Formatter for Windows, завантаживши із офоційного сайту. Запсутивши програму маємо отримати вікно зображене на Рис. - 2.4.

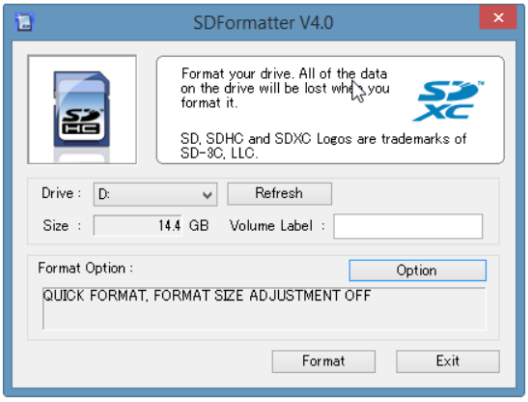



Рис. - 2.4 Інтерфейс програми SD Memory Card Formatter for Windows

Натискаємо на кнопку «Параметри» («Option») і встановлюємо для парамета «Налаштування розміру форматування» («FORMAT SIZE ADJUSTMENT») значення «ВКЛ» («ON»), потім тичнемо на «ОК» (Рис. - 2.5).

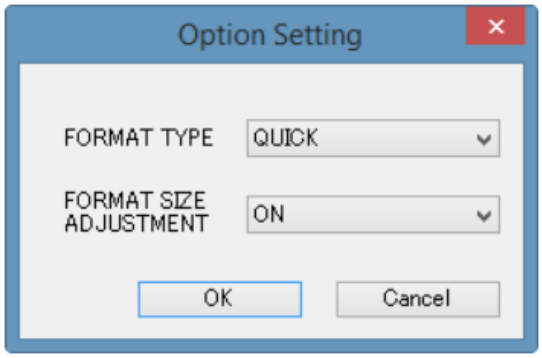



Рис. - 2.5 Опції форматування у програмі SD Memory Card Formatter

Там де написано «Drive» обираєму нашу карту пам’яті (якщо підключена лише одна, то вона вибереться автоматично).

Далі тиснемо на кнопку «Форматувати» («Format») та «ОК», після чого відбудеться саме форматування карти пам’яті і вона буде готова до використання.

Наступним кроком буде установка програмного забезпечення «NOOBS».

### 2.2.2 Установка програмного забезпечення «NOOBS».

Із офіційного сайту компанії «Raspberry Pi» із вкладки «Завантаження» («Downloads») скачуємо найновішу версію «NOOBS».

Розархівуємо та розпаковуємо вміст завантаженого на нашу відформатовану карту пам’яті. На карті пам’яті повинні опинитися приблизно такі файли як показано на Рис. - 2.6.

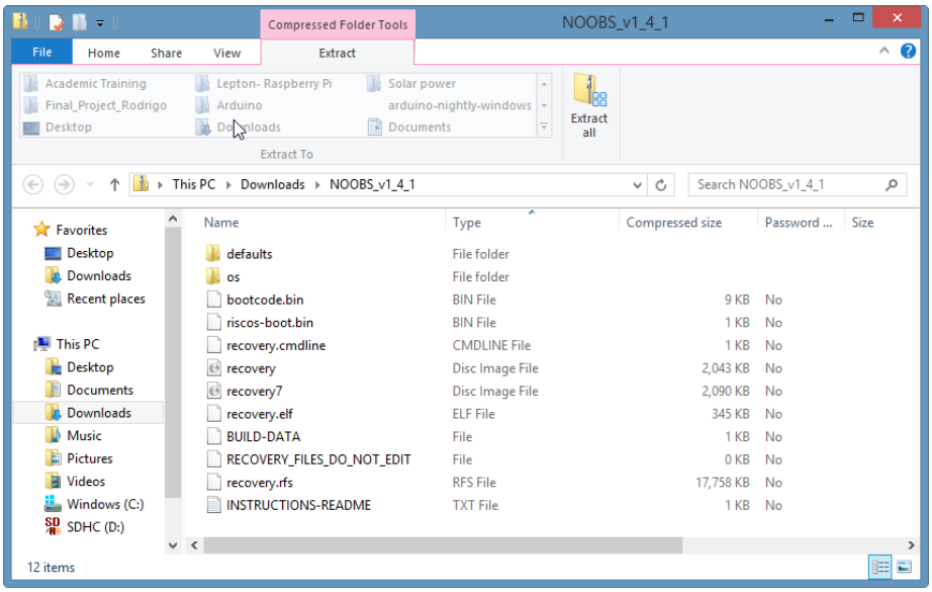



Рис. - 2.6 Розпаковані файли ПЗ «NOOBS»

Наступним кроком буде підключення необхідних для роботи системи девайсів.

### 2.2.3 Підключення потрібних для роботи системи пристроїв

### Підключення Raspberry Pi Camera Module V2

Підключаємо Raspberry Pi Camera Module V2 до Raspberry Pi вставляючи шлейф камери в необхідний порт на платі (Рис. - 2.7).

Рис. - 2.7 Raspberry Pi Camera Module V2 підключений в необхідний роз’єм [26]

**Підключення камери Lepton**

Під’єднаємо модуль камери Lepton до Raspberry Pi. З’єднуємо контакти так, як показано на Рис. - 2.8.

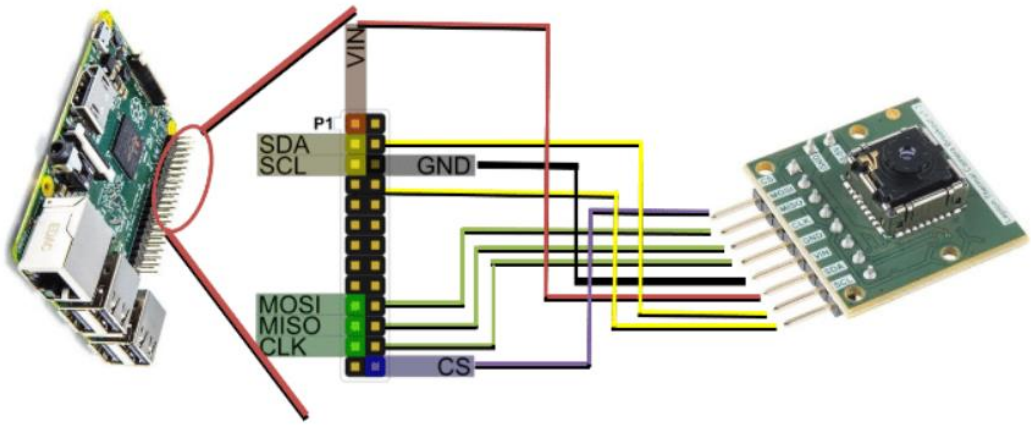



Рис. - 2.8 Правильне під’єднання камери Lepton до raspberry pi. [18]

**Підключення модуля Wi-Fi**

Для того, щоб використовувати розроблювану систему та можливості Raspberry Pi повною мірою необхідне якісне підключення до Інтернету. Оскільки дана система повинна бути зручною, було вирішено використовувати саме Wi-Fi-модуль на відміну від Ethernet-кабелю.

На офіційному сайті Raspberry Pi пропонується 3 різних способи установки Wi-Fi адаптера, та ми розглянемо тільки той що був використаний.

Відкриємо термінал LXT на робочому столі Raspberry Pi. Щоб визначити, які мережі Wi-Fi доступні для Raspberry Pi, введемо наступну команду:

*sudo iwlist wlan0 scan*

Virtual Router - це програмне забезпечення із відкритим вихідним кодом. Завантажити його можна із офіційного сайту компанії. У ньому просто вводимо ім'я та пароль мережі що нас цікавить (Рис. - 2.9).

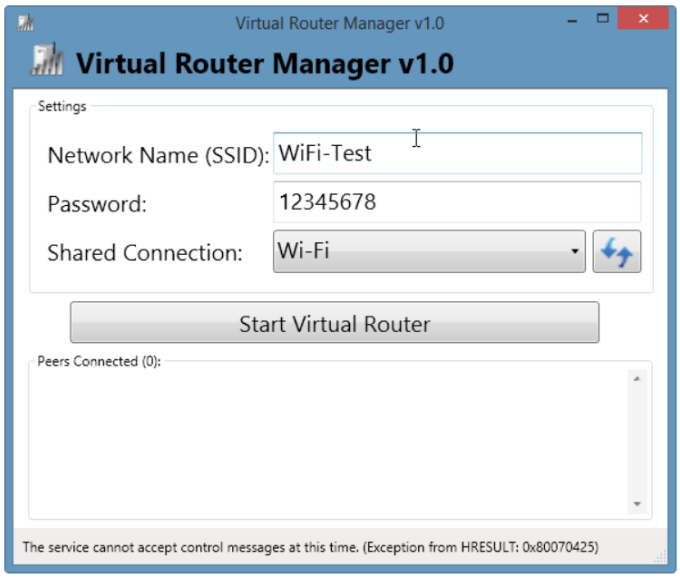



Рис. - 2.9 Інтерфейс програми Virtual Router

Буде відображено весь Доступний Wi-Fi. Ведемо SSID і пароль бажаного з'єднання Wi-Fi.

Виконаємо наступну команду:

*sudo nano /etc/wpa_supplicant/wpa_supplicant.conf*

Відкриється файл конфігурації мережі у який треба вставити код із необхідними нам значеннями назви та паролю Wi-Fi, приклад наведений нижче:

```
network={
        ssid="WiFi-Test "
        psk="12345678"
}
```

Завершивши, натиснемо Ctrl + X, погодимося, вибравши Y і натиснемо Enter для збереження внесених змін.

Для активації змін використаємо команду:

*sudo ifdown wlan0*

Потім введемо:

*sudo ifup wlan0*

Переконаємося, що Wi-Fi працює за допомогою команди:

*iwconfig*

Wlan0 буде підключений до вибраного нами Wi-Fi.

**Підключення іншої необхідної периферії.**

Підключимо монітор дисплея до Raspberry Pi за допомогою кабелю HDMI. Підключимо мишу і клавіатуру до Raspberry Pi за допомогою USB. Вставимо SD-карту в Raspberry Pi. Підключимо живлення до Raspberry Pi. Плата із під'єднаними девайсами повинна виглядати як на Рис. - 2.10.

Рис. - 2.10 Плата Raspberry Pi із підключеною апаратурою

Після всіх зроблених операцій наступним кроком буде налаштування програмного забезпечення самого Raspberry Pi.

### 2.2.4 Перший запуск Raspberry Pi, установка і налаштування ОС та програмного забезпечення.

Після підключення всіх елементів та ввімкнення живлення, можливо, із невеликою затримкою на екрані буде зображено вікно установки Raspbian через ПЗ «NOOBS» (Рис. - 2.11).

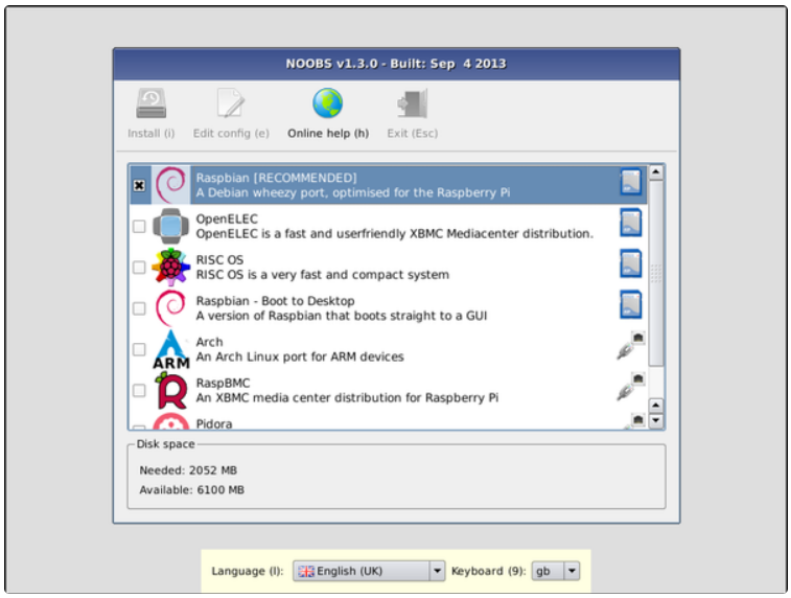



Рис. - 2.11 Меню установки ОС Raspbian з допомогою ПЗ «NOOBS»

Із усіх пунктів меню установки вибираємо перший варіант «Raspbian (RECOMMENDED)».

Тиснемо на "Встановити" («Install»). На встановлення знадобиться близько 30 хвилин, після чого ОС Raspbian буде готовою для роботи.

## 2.3 Встановлення ПЗ для камери FLIR Lepton із тепловізором.

Оскільки у нас є готова для використання ОС, наступним буде установлено та налаштовано ПЗ для тепловізійної камери.

Запустимо LXTerminal, натиснувши на значок, показаний на Рис. - 2.12 нижче, із якого далі будемо виконувати команди.

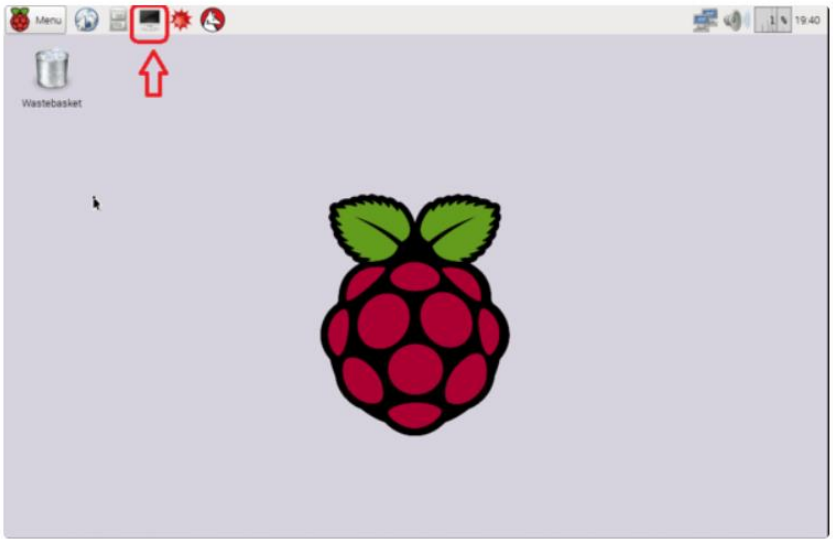



Рис. - 2.12 Інтерфейс ОС Raspbian

Виконавши команду *sudo raspi-config* , відкриємо розділ конфігурації. Повинне відкритися вікно із меню (Рис. - 2.13)

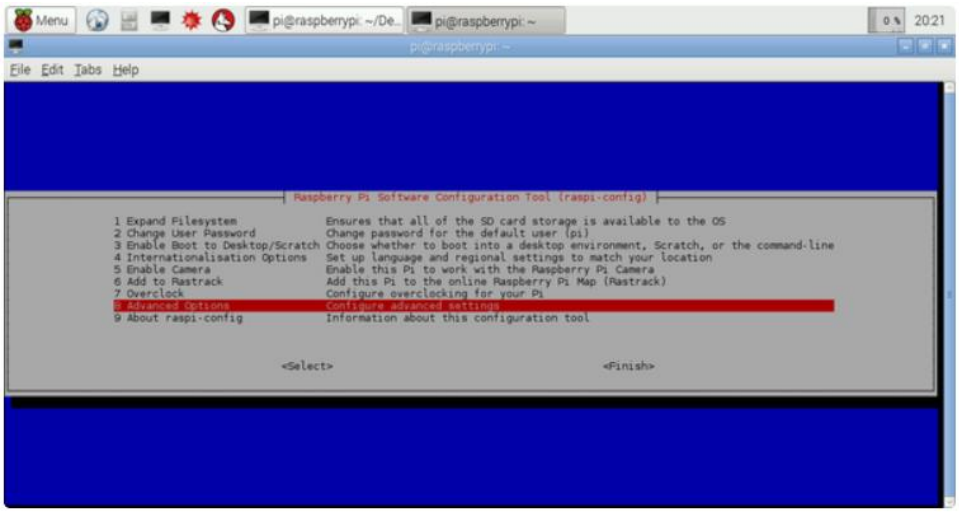



Рис. - 2.13 Меню конфігурації цифрових інтерфейсів

Виберемо «Додаткові Параметри» («Advanced Options»). Оскільки потрібно активувати цифровий інтерфейс SPI, обираємо цей пункт у меню та погоджуємося (Рис. - 2.14).

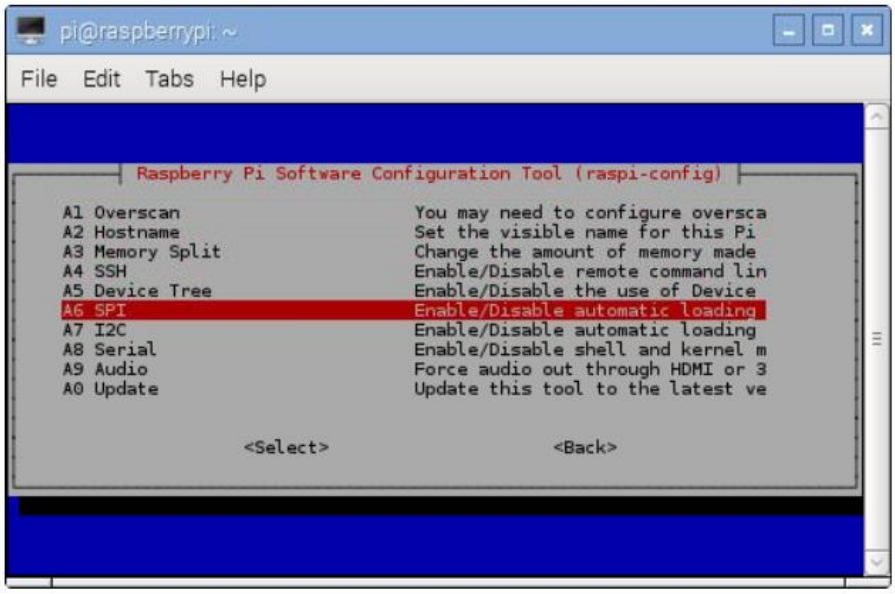



Рис. - 2.14 Процес налаштування цифрових інтерфейсів ОС Raspbian

Нам необхідно, щоб модуль ядра SPI завантажувався за замовчуванням, тому натискаємо "YES" та погоджуємося. Аналогічні операції робимо і з активацією цифрового інтерфейсу I2C. Після цього потрібно перезавантажити Raspberry Pi. Далі завантажимо із Інтернету QT-застосунок, виконавши команду:

*sudo apt-get install qt4-dev-tools*

Вводимо в термінал «y», погоджуючись.

Використовуючи інтегрований в систему браузер, перейдемо за посиланням: *https://github.com/PureEngineering/LeptonModule*, завантижимо архів, перейдемо в директорію завантажень та розпакуємо його командою:

*unzip LeptonModule-master.zip*

Перейдемо із даної директорії в папку «raspberrypi_video», ввівши в термінал команду: *cd /home/pi/LeptonModule-master/raspberrypi_video*

Перейдемо в каталог “LeptonSDKEmb32PUB”:

*cd /home/pi/LeptonModule-master/raspberrypi_video/leptonSDKEmb32PUB*

та запустимо команду *make*

Після цього перейдемо назад в директорію «raspberrypi_video» та виконаємо команду: *qmake && make***.** Тепер тепловізійна камера буде готовою до роботи (Рис. - 2.15).

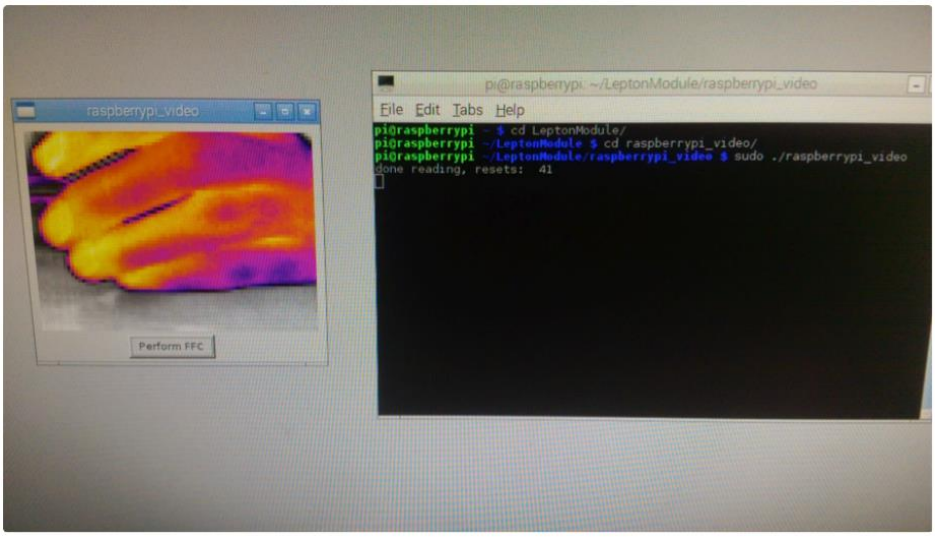



Рис. - 2.15 Тепловізійна камера Flir Lepton 2.0 Thermal Camera налаштована та працює

## 2.4 Огляд і аналіз засобів розробки

Під Raspberry Pi адаптовані майже всі популярні мови програмування. Але основними є Python та C/C++. Розглянемо їх детальніше.

### 2.4.1 Python

Python вважається універсальною інтерпретованою та компільованою (тою, якій притаманні властивості компільованої мови) мовою програмування саме завдяки його вбудованому інтерпретатору. Він вважається дуже простою у вивченні мовою, так як використовується у якості приклада навчанні у багатьох університетах світу як перша мова програмування. Проте це не заважає йому вирішувати дуже складні задачі - можна згадати хоча б машинне вивчення (machine learning) чи науку про дані (data science).

Переваг у даної мови дуже багато, але ми розглянемо ті, що виділяються на фоні інших, а саме:

- Простий синтаксис для початківців. На відміну від багатьох мов програмування, у Python фігурні дужки замінені на відступи, зайві круглі дужки просто прибрали, а змінні функцій та класів можна об'являти не вдаючись до якихось модифікаторів. Весь код Python досить читабельний, інколи його можна спутати із псевдокодом. Та що казати про його читабельність, якщо тільки це є одним із пунктів чи не всім відомого Zen of Python (Рис. - 2.16) - деякої особливості даної мови у вигляді вбудованого у інтерпретатор списка правил (викликається командою *import this*), яких краще притримуватися під час розробки.
- Має дуже велику спільноту користувачів, оскільки мова існує уже більше ніж 30 років, навколо неї, без перебільшення, сформувалася ціла культура, з'явилося чимало прихильників, послідовників та ентузіастів, які роблять чималий внесок у розвиток Python. Для звичайного розробника це означає, що чи не 99% проблем які у нього можуть виникнути уже, скоріше за все, виникали та були вирішені, і все що залишилося зробити - це правильно сформувати питання чи запит пошуковій системі.
- Динамічна типізація (надалі розглянемо цей пункт детальніше у недоліках)
- Величезна стандартна бібліотека - дуже багато рішень можна знайти використовуючи саме стандартну бібліотеку Python, тобто у багатьох випадках навіть без знання сторонніх бібліотек, що дуже рідко буває у світі сучасних мов програмування. А навіть якщо і не вийшло, то не проблема знайти рішення у сторонніх бібліотеках саме завдяки спільноті Python.

- Python можна використовувати у діалоговому режимі (може бути корисним для експериментів із кодом, перевірки себе, розв'язання деяких задач, чи просто в якості калькулятора);

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
>>>
```

Рис. - 2.16 Zen of Python

Недоліки, хоча їх можна назвати особливостями даної мови:

- Динамічна типізація. Це означає, що змінна яку ми оголошуємо у мові Python не має конкретного типу даних. Він визначається згідно із присвоєного змінній значення. Той же принцип має і оголошення сигнатур функцій та полів класів. Це дуже важливо, адже на рівні системи типів даних у розробника є набагато більше шансів допустити помилку. Цей аспект кардинально впливає на швидкість роботи (швидкість обчислення коду) програми.
- GIL (global interpreter lock) – м’ютекс, глобальний блокувальник інтерпретатора. Він не дозволяє одночасне використання декількох тредів, тобто багатопоточне виконнання коду страждає.
- У статично типізованих мовах дуже велика кількість помилок відловлюється ще на етапі компіляції, тоді як у Python це відбувається на момент виконання самої програми. Це означає,

що чим більше розрісся ваш проект - тим складніше відстежувати значення що передаються, точніше їх типи даних. Без коментування коду це стає майже невиконуваною задачею. Однак, звичайно, у Python є рішення і цієї проблеми у вигляді статичних аналізаторів, проте це неабияк впливає на естетичний вигляд вашого коду, що, по суті, нівелює ту саму перевагу Phython у вигляді красивого читабельного коду.

Важливим фактором росту Python є те що зараз дуже розвилися стрімінгові компанії типу Netflix, YouTube, Twitch. Ці компанії намагаються використовувати Phython при будь-якій нагоді. Це пов'язано з тим, що для цих сервісів важливу роль відіграють алгоритми, пов'язані із великикою кількістю оброблюваних даних, на кшталт рекомендаційних алгоритмів. У цій сфері Python дуже сильно виділяється, адже для нього не проблема працювати із великими БД користувачів.

Також Python дуже часто використовується для управління мережами розповсюдження контенту та автоматизації функцій безпеки. Він дуже гнучкий та може адаптуватися чи не під будь-яку поставлену задачу, адже має величезну кількість підтримуваних бібліотек.

Python - це скриптова мова програмування, тому дуже часто і дуже багато хто його використовує саме з ціллю написання Shell-скриптів, та і взагалі заміни Bash.

### 2.4.2 C/C++

C++ - компільвана, структурована, об'єктно орієнтована мова програмування, яка дуже сильно спрощує роботу із великими програмами, та при цьому має великий потенціал для розвитку, який продовжується досі уже протягом більше ніж 24 років. У ній зосереджено властивості як високорівневих так і низькорівневих мов програмування.

На відміну від свого попередника мови С, у С++ більше уваги приділено саме підтримці об'єктно орієнтованого (взаємодія в коді на багато вищому рівні, на рівні абстракцій) та загального (шаблонного) програмування.

Мова програмування С зробила неабиякий внесок у розвиток індустрії програмного забезпечення, а її синтаксис став базою для багатьох мов програмування (C, C#, Java, Objective-c).

Мова С++ (Рис. - 2.17) не легка у вивченні, проте після її освоєння, інші С-подібні мови даються дуже легко та швидко.

С++ можна певною мірою назвати швейцарським ножем у світі мов програмування. Під цим терміном мається на увазі, що на відміну від інших мов, які використовуються для якихось локальних потреб, із С++ можна писати програми будь-якої складності та вирішити будь-яку задачу.

```
void populate(auto &data) { // see!
    data.insert({"a",{1,4}});
    data.insert({"b",{3,1}});
    data.insert({"c",{2,3}});
}

auto merge(auto data, auto upcoming_data) { // don't write long declaration again
    auto result = data;
    for(auto it: upcoming_data) {
        result.insert(it);
    }
    return result;
}

int main() {
    std::map<std::string, std::pair<int,int>> data;
    populate(data);

    std::map<std::string, std::pair<int,int>> upcoming_data;
    upcoming_data.insert({"d",{5,3}});

    auto final_data = merge(data,upcoming_data);

    for(auto itr: final_data) {
        auto [v1,v2] = itr.second; // structured bindings discussed below
        std::cout << itr.first << " " << v1 << " " << v2 << std::endl;
    }
    return 0;
}
```

Рис. - 2.17 Синтаксис С++

Щоб перерахувати плюси даної мови можна згадати області її використання:

- Розробка ПЗ для мікроконтроллерів;
- IoT;
- Розробка ПЗ для робототехніки;
- Десктопна розробка;
- Розробка під мобільні девайси;
- Веб-розробка;
- Ігрові застосунки;
- Системи моделювання;
- Системи прогнозування;
- Обробка статистики;
- Нейронні мережі.

Тобто можна сказати, що немає області програмування, де C++ не було б застосування.

## 2.5 Огляд сторонніх бібліотек та засобів, необхідних для розробки

Для розробки даної системи контролю приміщення під Raspberry Pi було обрано мову програмування Python, тому огляд та аналіз засобів буде проводитися серед його доступних сторонніх бібліотек.

### 2.5.1 PyQt5

PyQt5 – це комплексний набір бібліотек мови програмування Python, створений на основі платформи Qt5, яка підтримується компанією Digia. Він представляє і себе дуже потужний інструмент для створення GUI.

### 2.5.2 Keras, TensorFlow

Keras – нейромережна бібліотека Python, яка створена для роботи та підтримки інших бібліотек. Працюючи з тилом у вигляді TensorFlow вона здатна покращити та оптимізувати процес обробки зображення та тексту.

### 2.5.3 Google Cloud Vision API

Google Cloud Vision – це сервіс, оснований на машинному навчанні, розроблений компанією Google. Він дає можливість програмістам розуміти та витягувати значення із захоплених зображень, вирізняти обличчя чи інші складні об’єкти, класифікувати їх.

### 2.5.4 Cv2 Python

Cv2 – це оновлена версія бібліотеки OpenCV. Вона в свою чергу являється бібліотекою зв’язок. Її робота полягає у рішенні завдань із області машинного бачення.

### 2.5.5 Telepot

Telepot – це бібліотека мови програмування Python, яка забезпечує розробку ПЗ для Telegram Bot API. Підтримується версіями Python 2.7 і вище.

### 2.5.6 Collections, os, io, numpy

Collections – це модуль бібліотеки Python який підтримує додаткові структури даних у роботі з колекціями та зберігання їх даних.

Os - це модуль бібліотеки Python який допомагає мові програмування Python працювати (здійснювати взаємодію) із ОС.

Io - це модуль бібліотеки Python який підтримує базові інструменти Python щоб працювати із вводом та виводом різних типів даних.

Numpy – модуль стандартної бібліотеки Python для роботи із масивами та матрицями на рівні складних функцій.

### 2.5.7 Flir Image Extractor

Flir Image Extractor – це невелика стороння бібліотека Python, яка знаходиться у вільному доступі. Вона призначена для роботи із зображеннями, отриманими захопленням камерами серії Flir Thermal Camera.

Цей інструмент дає можливість отримувати вихідні значення із фотографій і термосенсорів, конвертовані у температури.

## 2.6 Вибір середовища розробки

### 2.6.1 Visual Studio Code

Visual Studio Code (Рис. - 2.18) – це безкоштовне універсальне середовище для розробки програмного забезпечення для будь-яких платформ. Підтримує всі сучасні мови програмування. Створене компанією Microsoft, воно доступне для Windows, Linux та OS X базованих систем.

Із функцій даного IDE можна перерахувати наступні:

- Виділення синтаксису коду;
- Підказки згідно контексту;
- Налагодження коду;
- Підтримка платформи Git;
- Фрагментування та рефакторинг коду;
- Автодоповнення базових конструкцій.

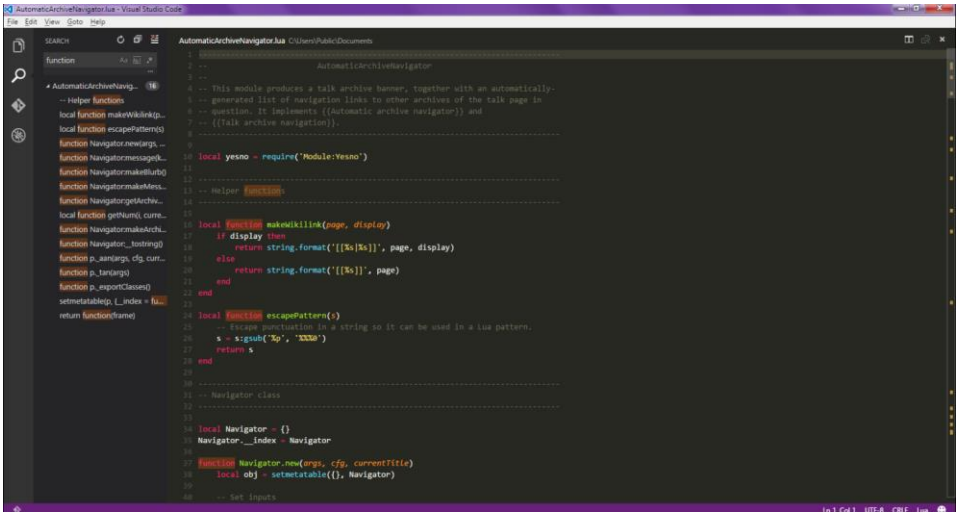


Рис. - 2.18 Інтерфейс IDE Visual Studio Code

### 2.6.2 PyCharm

PyCharm (Рис. - 2.19) – це середовище розробки для створення програмного забезпечення на мові програмування Python. Створене компанією JetBrains, воно має вбудований графічний дебагер, аналізатор коду, механізми для роботи із юніт-тестами. IDE доступне для таких платформ як Windows, Mac OS, Linux.

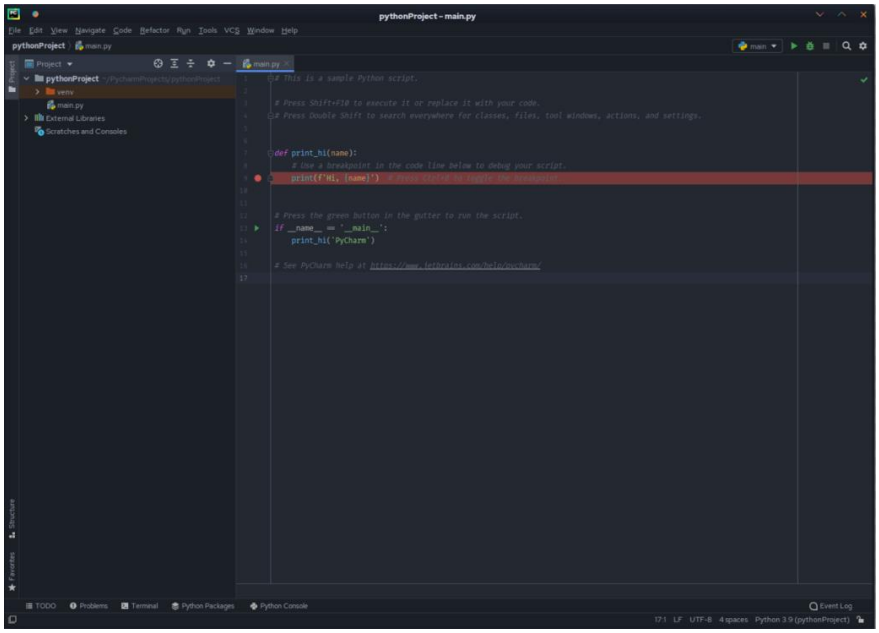


Рис. - 2.19 Інтерфейс IDE PyCharm

Дане середовище розробки має такі функції:

- Інтегрований відлагоджувач;

- Інструменти та засоби навігації по коду та рефакторингу;
- Аналізатор користувацького коду із графічним виділенням синтаксису;
- Засоби для юніт-тестів;
- Підтримка таких платформ як Git, Perforce, Mercurial, Subversion;
- Веброзробка з допомогою Django.

## Висновок до розділу 2

У даному розділі було оцінено та проаналізовано функції і можливості одноплатного ПК Raspberry Pi, та з’ясовано що розробка на його базі системи контролю стану приміщення при пандемії буде вдалою завдяки необхідним технічним характеристикам. Було проведено його налаштування а також підключення та підготовку необхідних для роботи системи пристроїв, а саме Raspberry Pi Camera Module V2, Flir Lepton 2.0 Thermal Camera, Wi-Fi модуля. Всі девайси були налаштовані та готові до роботи.

Було проаналізовано доступні найпоширеніші мови програмування для даної платформи, визначені їх переваги та недоліки. У зв’язку із вибором мови програмування Python у якості засобу розробки, було наведено та описано бібліотеки та сервіси цієї мови, котрі будуть використовуватися у процесі написання програмного коду. Інтегрованим середовищем розробки було обрано PyCharm, у зв’язку з його зручністю, лаконічністю та наявним у ньому інструментам аналізу синтаксису.

# РОЗДІЛ 3.
# РОЗРОБКА СИСТЕМИ

Оглянувши та дослідивши методи та засоби розробки можна перейти до розробки самого програмного коду. У цьому розділі буде описано основні складові програми і їх взаємодію.

## 3.1 Імпортування необхідних бібліотек та модулів

Як і кожний великий проєкт, код розроблюваної системи починається із імпортування необхідних модулів та бібліотек (Рис. - 3.1)

```
import flir_image_extractor
import cv2
import numpy as np
import os
import io
from google.cloud import vision
import telepot
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.models import load_model

from PyQt5.QtWidgets import QApplication
from PyQt5.QtWidgets import QWidget
from PyQt5.QtGui import QImage
from PyQt5.QtGui import QPixmap
from PyQt5.QtCore import QTimer

from ui_main_window import *

from collections import deque
```

Рис. - 3.1 Імпортування модулів та бібліотек

## 3.2 Створення головних глобальних змінних

Далі створюються головні змінні програми із глобальним доступом, із якими буде відбуватися взаємодія упродовж розробки (Рис. - 3.2). Присвоюються значення токена (Рис. - 3.3) та затримки відгуку бота, створеного заздалегідь в самому Telegram з допомогою «БотаБатька»

(«BotFather»), змінній *bot* присвоюється, власне, сам бот, налаштовується API Google Vision, клієнт, додаються моделі обличчя людини в масці та без маски, створюється змінна *flir* зі значенням захопленого зображення.

```
21
22  token = 'telegram-token-value'
23  mc = 'value'
24  bot = telepot.Bot(token)
25  os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = r'ServiceAccountToken.json'
26  client = vision.ImageAnnotatorClient()
27  facenet = cv2.dnn.readNet('../training custom dataset/face_detector/deploy.prototxt', '../training custom dataset/face_
28  model = load_model('../training custom dataset/mask_detector.model')
29
30  flir = flir_image_extractor.FlirImageExtractor()
31
```

Рис. - 3.2 Створення та присвоєння значень головним полям із глобальним доступом

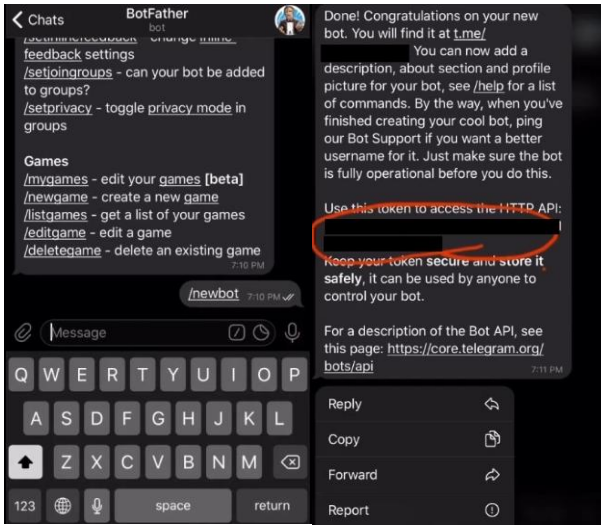



Рис. - 3.3 Отримання даних бота із BotFather

## 3.3 Клас MainWindow

### 3.3.1 Конструктор класу

Відкриваємо клас *MainWindow* (Рис. - 3.4), викликаючи конструктор класу, задаючи в ньому як поля, притаманні саме класу спадкоємцю класу *QWidget* (*ui*, *timer*, і потім налаштовуємо, встановлюючи функцію зворотного виклику таймера), так і унікальні ( поля камери *cam* та термокамери *camthermal*) та присвоюючи їм відвовідні значення, а саме: для *ui* – об'єкт вікно застосунку, для *timer* – сам об'єкт таймер, потім налаштовуючи, його встановлюючи функцію зворотного виклику таймера,

для *cam* – захоплення відео із камери, а для *camthermal* - захоплення відео із термокамери.

```
class MainWindow(QWidget):
    def __init__(self):
        super().__init__()
        self.ui = Ui_Form()
        self.ui.setupUi(self)

        self.timer = QTimer()

        self.timer.timeout.connect(self.viewCam)
        self.timer.timeout.connect(self.viewThermalCam)
        self.cam = cv2.VideoCapture(0)
        self.camthermal = cv2.VideoCapture(1)

        self.timer.start(20)
        image = cv2.imread('Logo.png')
        image = cv2.resize(image, dsize=(0, 0), fx=0.5, fy=0.7, interpolation=cv2.INTER_LINEAR)
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        self.number = 0
        self.dq = deque()
        self.textdq = deque([])
        height, width, channel = image.shape
        step = channel * width

        qImg = QImage(image.data, width, height, step, QImage.Format_RGB888)
        self.ui.label_3.setPixmap(QPixmap.fromImage(qImg))
        self.prevTime = 0
        self.temperature = 0
        self.nomask = 0
        self.response = 0
```

Рис. - 3.4 Конструктор класу *MainWindow*

Запускаємо таймер, передаючи значення 20 мілісекунд. Створюємо поле *image* та присвоюємо йому значення зображення, потім форматуючи це зображення під потрібний розмір та колір. Потім дістаємо дані зображення, у виді *height* (висота), *width* (ширина) та *channel* (канали), та створюємо поле *step* (крок) даючи йому значення *channel * width*. Створюємо нове зображення в полі *qImg* та показуэмо його в головному відео вікна. Створюємо та присвоюємо значення полям *prevTime, temperature, nomask, response.*

### 3.3.2 Метод put_img_to_labels

Цей метод виводить зображення у вікно застосунку (Рис. - 3.5). Тут час він використовується для відображення екрана захоплення на label при виявленні особливої людини.

Спочатку зберігається останнє зображення, та підганяється його розмір під розмір вікна. Встановлюється формат, та встановлюється останнє зображення для label.

```
62      def put_img_to_labels(self, dq):
63          image = self.dq[0]
64          image = cv2.resize(image, dsize=(0, 0), fx=0.2, fy=0.2, interpolation=cv2.INTER_LINEAR)
65          image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
66          height, width, channel = image.shape
67          step = channel * width
68          qImg = QImage(image.data, width, height, step, QImage.Format_RGB888)
69          self.ui.label.setPixmap(QPixmap.fromImage(qImg))
```

Рис. - 3.5 Метод *put_img_to_labels*

### 3.3.3 Метод label_2_text

Створюється локальна змінна *TEXT* типу *str*. Якщо кількість детальної інформації в поточній черзі становить 10 або більше, відкриється найстаріша інформація праворуч. Потім за допомогою циклу, ітеруючись по черзі, зберігатиметься детальна інформація в ряд один за одним у тексті, розділеному розривами рядків. Текстовий вміст у мітці *label_2* зберігається та відображається як вміст у *TEXT* .( Рис. - 3.6)

```
71      def label_2_text(self, textdq):
72          TEXT = ""
73          if len(self.textdq) > 10:
74              textdq.pop()
75          for text in textdq:
76              TEXT += text + '\n'
77          self.ui.label_2.setText(TEXT)
```

Рис. - 3.6 Метод *label_2_text*

### 3.3.4 Метод get_max_temperature

Даний метод отримує значення температури із захопленої області обличчя. За температурними даними вирізається та оглядається лише область обличчя. Якщо обличчя не виявлено, розмір кадрування дорівнює 0 та нічого не повертається. Інакше метод повертає найвищу температуру в області обличчя.( Рис. - 3.7)

```
79      def get_max_temperature(self, thermal_np, x1, y1, x2, y2):
80          crop = thermal_np[y1:y2, x1:x2]
81              if crop.size == 0:
82              return None
83
84          return np.max(crop)
```

Рис. - 3.7 Метод *get_max_temperature*

### 3.3.5 Метод viewCam

Це дуже великий метод, тому будемо розбирати його частинами.

Спочатку камера видимого світла зчитує поточний екран як зображення, потім термокамера зчитує поточний екран як зображення. Далі зображення повертається щоб нормально зчитувати інформацію (Рис. - 3.8).

```
86      def viewCam(self):
87          ret, img = self.cam.read()
88          ret, thermal_image_data = self.camthermal.read()
89          img = cv2.rotate(img, cv2.ROTATE_90_CLOCKWISE)
```

Рис. - 3.8 Початок методу *viewCam*

Дістаються значення висоти та ширини із зображення. Відбувається попередня обробка. Значення параметрів вводяться, як зазначено у FaceNet OpenCV. Зображення перетворюється у форму, яку використовують DNN. Функція *cv2.dnn.blobFromImage* (Рис. - 3.9)виконує:

- Віднімання середнього значення;

- Масштабування (зміна розміру зображення);
- Заміну каналів.

*(104.0, 177.0, 123.0)* — емпіричне оптимальне значення середнього віднімання. Тоді віднімання середнього спрощує аналіз для DNN, виключаючи деякі значення RGB.

(300, 300) - той розмір зображення, який модуль DNN може обробляти за допомогою CNN, модель фіксується на 300, 300.

```
91          h, w = img.shape[:2]
92          blob = cv2.dnn.blobFromImage(img, scalefactor=1., size=(300, 300), mean=(104., 177., 123.))
```

Рис. - 3.9 Конвертування зображення згідно стандандарту

Конвертоване зображення передається у функцію у виді параметра *blob*. Потім результати розпізнавання обличчя зберігаються (Рис. - 3.10).

```
94          facenet.setInput(blob)
95          dets = facenet.forward()
96          result_img = img.copy()
```

Рис. - 3.10 Передача конвертованого зображення та результати розпізнавання

Після виявлення обличчя обчисліимо ймовірність того, що область обличчя використала маску та додамо цю ймовірність. Це та частина коду, що обробляє зображення за допомогою значення розпізнавання обличчя.

Збережемо отримані під час циклу дані. *dets.shape[2]* – максимальна кількість захоплених рамок обличь, яку отримує модель. Оскільки значення рівно 200, то можна розпізнати до 200 облич. *Confidence* – те значення, для фкого результат виявлення буде вірним для проходження перевірки. *Dets[0,0]* – значення рамок, які ми будемо малювати. *i* – поточна рамка. *2* означає третій атрибут, який вказує на ймовірність того, що це обличчя.

При ймовірності менше *0.5* цикл продовжується.

Створимо обмежувальну рамку та будемо зберігати зображення обличчя за допомогою обмежувальної рамки (Рис. - 3.11).

```
98          for i in range(dets.shape[2]):
99              confidence = dets[0, 0, i, 2]
100
101             if confidence < 0.5:
102                 continue
103
104             x1 = int(dets[0, 0, i, 3] * w)
105             y1 = int(dets[0, 0, i, 4] * h)
106             x2 = int(dets[0, 0, i, 5] * w)
107             y2 = int(dets[0, 0, i, 6] * h)
108             face = img[y1:y2, x1:x2]
```

Рис. - 3.11 Створення рамки

Наступною буде частина коду, де перевірятиметься носіння маски.

На Рис. - 3.12 показано частину попередньої обробки. Спочатку змінюємо розмір зображення та систему кольорів зображення. Далі буде виконуватися обробка цього зображення. Якщо зробимо це, фігура вийде як *(224, 224, 3)*, але коли вставимо її, вона має бути *(1,224,224,3)*, тому додаємо один вимір.

```
111             face_input = cv2.resize(face, dsize=(224, 224))
112             face_input = cv2.cvtColor(face_input, cv2.COLOR_BGR2RGB)
113             face_input = preprocess_input(face_input)
114             face_input = np.expand_dims(face_input,axis=0)
```

Рис. - 3.12 Попередня обробка зображення

Повертаємо ймовірність носіння маски за допомогою методу прогнозування на попередньо завантаженій моделі.

```
116             mask, self.nomask = model.predict(face_input).squeeze()
```

Рис. - 3.13 Виклик методу вирахування ймовірності носіння маски

На Рис. - 3.14 зображено блок коду який відповідає за колір рамки та напис під нею залежно від того, є маска чи ні, а нижче (рядок – 128-129) – код перевірки температури тіла.

```
118         if mask > self.nomask:
119             color = (0, 255, 0)
120             label = 'Mask %d%%' % (mask * 100)
121         else:
122             color = (0, 0, 255)
123             label = 'No Mask %d%%' % (self.nomask * 100)
124
125         cv2.rectangle(result_img, pt1=(x1, y1), pt2=(x2, y2), thickness=7, color=color,
126         cv2.putText(result_img, text=label, org=(x1, y1 - 10), fontFace=cv2.FONT_HERSHEY
127
128         thermal_np = flir.process_image(thermal_image_data)
129         max_temperature = self.get_max_temperature(thermal_np, x1, y1, x2, y2)
```

Рис. - 3.14 Блок коду, що формує колір і рамки і напис під нею

Якщо ймовірність того, що маски немає перевищує певну ймовірність або максимальна температура в області обличчя висока, збережеться фотографія та отримана інформація. Розглянемо цю частину коду. До уваги беруться ті зображення, на яких ймовірність того, що маски немає більша чи дорівнює *0.75* або температура в області обличчя більша чи дорівнює *37.5* градусів по Цельсію.

Спочатку збережеться фото людини, та якщо є 4 фотографії, буде додана фотографія, зроблена останньою. Також зберігається порядок людей.

У тимчасову змінну буде збережено зображення для подальшої обробки та його назва буде у виді тексту. Потім зберігається остаточний варіант зображення. Викликаємо метод *put_img_to_labels*, описаний вище. Він покаже зображення людини без маски.

Потім у змінній *message* (повідомлення) ми формуємо дані про людину без маски. Блок перевірки закінчується частиною коду, яка відповідає за відправку повідомлення боту в Telegram.(Рис. - 3.15)

```
131         if self.nomask >= 0.75 or max_temperature >= 37.5:
132             self.dq.appendleft(face)
133             if len(self.dq) == 4:
134                 self.dq.pop()
135             self.number += 1
136             IMAGE_FILE = 'No_Mask-High_Temp/' + str(i) + '_' + str('No_Mask%d%%_' % (self.nom
137             cv2.imwrite(IMAGE_FILE, result_img)
138 
139             saved_file = 'No_Mask_File/' + str(i) + '_' + str('No_Mask%d%%_' % (self.nomask *
140             cv2.imwrite(saved_file, result_img)
141             self.put_img_to_labels(self.dq)
142 
143             message = 'temperature :' + str(max_temperature) + '\nProbability of not wearing
144 
145             f = open(IMAGE_FILE,'rb')
146             self.response = bot.sendPhoto(mc, f)
147             self.response = bot.sendMessage(mc,message)
```

Рис. - 3.15 Перевірка за температурою та присутності маски

Уже вийшовши із блоку перевірки, створюється аналогічна змінна *message2,* у змінну *textdq* вставляються значення кількості людей, яких визначили «підозрілими», та потім викликається метод *label_2_text*, що виводить ці значення у вікні додатку (Рис. -3.16).

```
149         message2 = 'temperature :' + str(self.temperature) + 'Probability of not wearing a mask
150         self.textdq.appendleft(message2)
151         self.label_2_text(self.textdq)
```

Рис. - 3.16 Вивід отриманих значень *message2* у вікно

Розглянемо останню частину методу *viewCam*.

Змінимо розмір кінцевого зображення та будемо показувати його як *main_video* використовуючи модулю *QPixmap* бібліотеки *PyQt.* Конвертуємо зображення у RGB-формат. Дістанемо значення *height* (висоти), *width* (ширини) та *channel* (каналів). Знайдемо значення *step*(кроку). Створимо зображення, відповідне до *QImage* та будемо передавати його як відео у головне вікно додатку (Рис. - 3.17).

```
153         image = cv2.resize(result_img, dsize=(0, 0), fx=0.2, fy=0.2, interpolation=cv2.INTER_LINEAR)
154         image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
155         height, width, channel = image.shape
156         step = channel * width
157         qImg = QImage(image.data, width, height, step, QImage.Format_RGB888)
158         self.ui.main_video.setPixmap(QPixmap.fromImage(qImg))
```

Рис. - 3.17 Передача зображення у головне вікно

### 3.3.6 Метод viewThermalCam

Спочатку читаємо вхідне зображення у BGR-форматі. Повернемо його та конвертуємо у RGB. Дістанемо значення *height* (висоти), *width* (ширини) та *channel* (каналів). Знайдемо значення *step* (кроку). Створимо зображення, відповідне до *QImage* та будемо передавати його як відео з термокамери у головне вікно додатку (Рис. - 3.18).

```
163     def viewThermalCam(self):
164         ret, image = self.camthermal.read()
165         image = cv2.rotate(image, cv2.ROTATE_90_CLOCKWISE)
166         image = cv2.resize(image, dsize=(0, 0), fx=0.2, fy=0.2, interpolation=cv2.INTER_LINEAR)
167
168         image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
169
170         height, width, channel = image.shape
171         step = channel * width
172
173         qImg = QImage(image.data, width, height, step, QImage.Format_RGB888)
174
175         self.ui.thermal_video.setPixmap(QPixmap.fromImage(qImg))
```

Рис. - 3.18 Метод *viewThermalCam*

## 3.4 Запуск додатку

Створюючи змінну *app*, викликаємо конструктор C++ класу *QApplication*, ініціалізуючи QT-додаток. Створюємо екземпляр класу *MainWindow* та запускаємо (Рис. - 3.19).

```
177 if __name__ == '__main__':
178     app = QApplication(sys.argv)
179
180     mainWindow = MainWindow()
181     mainWindow.show()
182
183     sys.exit(app.exec_())
```

Рис. - 3.19 Запуск додатку

## Висновок до розділу 3

У даному розділі було описано деталі розробки системи контролю стану приміщення при пандемії. Було розглянуто та описано основні класи додатку та взаємодію їх складових. Розробка була виконана на мові програмування Python з використанням доступних бібліотек, сервісів та модулів цієї мови, а саме: PyQt5, Keras.TensorFlow, Google Cloud Vision API, cv2, telepot, collections, os, io, numpy, Flir Image Extractor. Усі ці інструменти спеціалізуються на вирішенні подібних задач.

# ВИСНОВКИ

Метою даної дипломної роботи було створення системи контролю стану приміщення при пандемії.

Робота над дипломним проектом була спрямована на розробку застосунку, що дозволяє отримувати та аналізувати дані, отримані із спеціальних апаратних засобів та виявляти людей, які порушують масковий режим чи мають високу температуру тіла.

У першому розділі було проаналізовано предметну область, розглянуто та проаналізовано архітектуру мережі IoT а також представлено основні протоколи, які використовуються для побудови таких мереж. Були розглянуті існуючі аналоги розроблюваної системи, їх переваги та недоліки.

У третьому розділі було розглянуто характеристики ПК Raspberry Pi, який буде виступати базою для розробки даної системи контролю, а також апаратних засобів, необхідних для роботи розроблюваної системи. Було проведено налаштування ПК та допоміжних пристроїв, підготовка їх до використання. Були з’ясовані та описані мови програмування, бібліотеки та сервіси, що спеціалізуються на розробці подібних систем.

У третьому розділі було поетапно описано деталі розробки системи. Були розглянуті та описані основні класи застосунку, методи та поля цих класів, їх взаємодію. До кожного описуваного складового програми наводився скріншот коду із середовища розробки PyCharm.

У результаті розробки була отримана цілком функціонально коректна система. Протестувавши застосунок, можна впевнитися, що поставлених задач було досягнуто та розроблений продукт відповідає заданим вимогам.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Carnegie Mellon University, "The "Only" Coke Machine on the Internet", c.12-20. 2014p.

2. InformationWeek, "Internet of Things Done Wrong Stifles Innovation", c.49-55. 2014p.

3. Mattern Friedemann, Floerkemeier Christian, "From the Internet of Computer to the Internet of Things" c.84-90. 2014p.

4. Weiser Mark, Scientific American, "The Computer for the 21st Century" c.45-60. 2010p.

5. Raji, R.S., IEEE Spectrum, "Smart networks for control" c.32-40. 1994p

6. Pontin Jason, MIT Technology Review "ETC: Bill Joy's Six Webs" c.249-260. 2013p.

7. CHETAN SHARMA, "CORRECTING THE IOT HISTORY" c.143-150. 2021p.

8. Ashton K.,"That 'Internet of Things' Thing" c.12-20. 2017p.

9. BBC World Service, "Peter Day's World of Business" c.24-34. 2016p.

10. Magrassi P. "Why a Universal RFID Infrastructure Would Be a Good Thing" c.341-352. 2002p

11. Magrassi P., Berg T. "A World of Smart Objects" c.85-96. 2002p

12. Commission of the European Communities. "Internet of Things – An action plan for Europe" c.12-40. 2009p

13. Wood, Alex, The Guardian, "The internet of things is revolutionizing our lives, but standards are a must" c.120-130. 2015p

14. Stallings, William. «Foundations of modern networking : SDN, NFV, QoE, IoT, and Cloud. Florence Agboma, Sofiene Jelassi» c.30-40. 2016p

15. "StackPath". www.industryweek.com. 2022p.

16. Интернет вещей в научных исследованиях. 2017р.
URL: https://cyberleninka.ru/article/v/internet-veschey-v-nauchnyh-issledovaniyah

17. Обзор и сравнительний анализ технологий LPWAN сетей. 2016р. URL: www.sut.ru/doci/nauka/review/20164/33-48.pdf

18. Shimpi, Anand Lal. The iPhone 3GS Hardware Exposed & Analyzed. www.anandtech.com c.40-50 2018р.

19. RPi Performance - eLinux.org. elinux.org. c.30-40. 2018р.

20. https://www.rfwireless-world.com/Terminology/WirelessHART-vs-ISA100.html 2012р

21. nic-delhi/AarogyaSetu_Android, nic-delhi, 2020р

22. "Govt launches 'Aarogya Setu', a coronavirus tracker app: All you need to know". 2020р

23. The Indian Express."Mandating use of Aarogya Setu app illegal, says Justice B N Srikrishna" c.20-30 2020р.

24. https://www.kmu.gov.ua/en/news/yak-pracyuye-zastosunok-dij-vdoma 2020р

25. "Govt discontinues Corona Kavach, Aarogya Setu is now India's go-to COVID-19 tracking app" c.30-40. 2020р.

26. "When should I enter symptoms into the NHS COVID-19 app?". faq.covid19.nhs.uk. 2020р

27. Hern, Alex "Fault in NHS Covid app meant thousands at risk did not quarantine". The Guardian. 2020р.

28. Sidhu, Randeep; Appleton, Gaby "How the NHS COVID-19 app is making the most of cutting-edge global technology". GOV.UK. Department of Health and Social Care. 2020р.

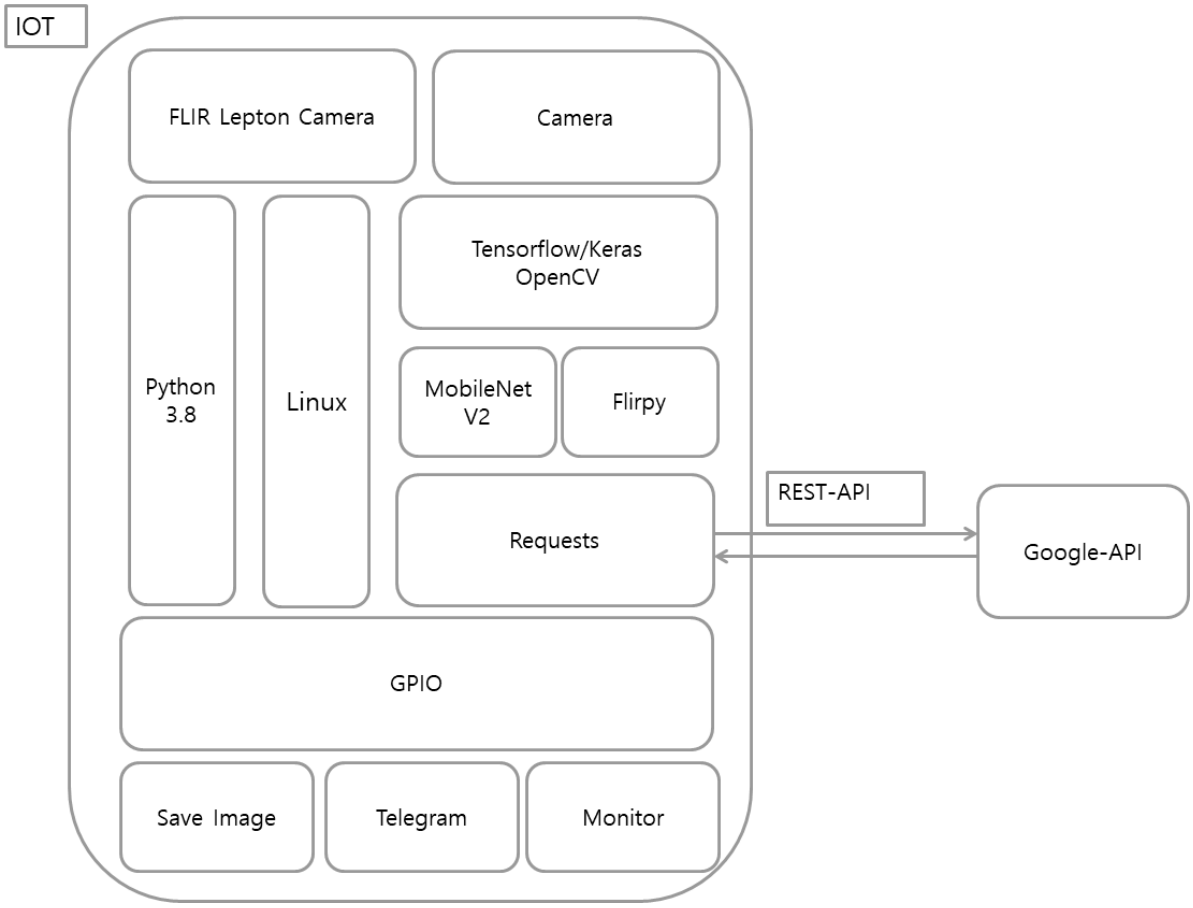
29. Жабін В. І. «Підвищення ефективності паралельної обробки даних і забезпечення відмовостійкості мультипроцесорних обчислювальних систем реального часу» с.102-107. 2015р

30 . Жабін В. І., Скоріченко О.В. «Обробка даних в неавтономному режимі на ПЛІС», с. 43-48. 2017р.

# Додаток 1

# СТРУКТУРНА СХЕМА
## до дипломного проєкту
## на тему: «Система контролю стану приміщення при пандемії»

Київ – 2022



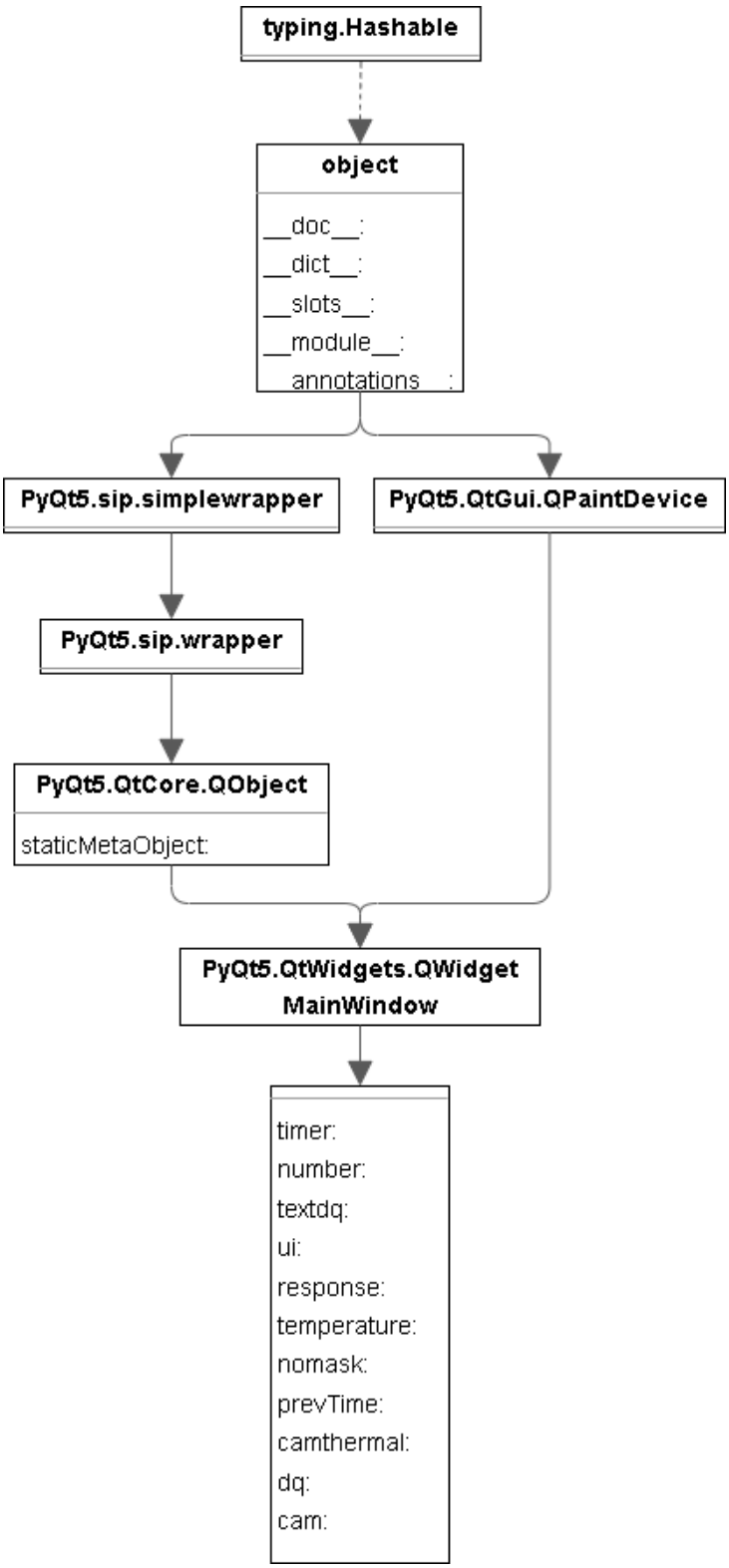

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | ІАЛЦ 467200.004 Д1 | | |
| Зм. | Арк. | № докум. | Підпис | Дата | | | |
| Розробив | | Карпенко Я.Р. | | | Система контролю стану приміщення при пандемії Структурна схема | Лит. | Аркуш 1 / Аркушів 1 |
| Перевірив | | Виноградов Ю.М. | | | | | |
| Т. Контр. | | | | | | КПІ ім. Ігоря Сікорського, ФІОТ, ІО-81 | |
| Н. Контр. | | Сімоненко В. П | | | | | |
| Затвердив | | | | | | | |

# Додаток 2

# ФУНКЦІОНАЛЬНА СХЕМА КЛАСІВ

## до дипломного проєкту на тему: «Система контролю стану приміщення при пандемії»

Київ – 2022

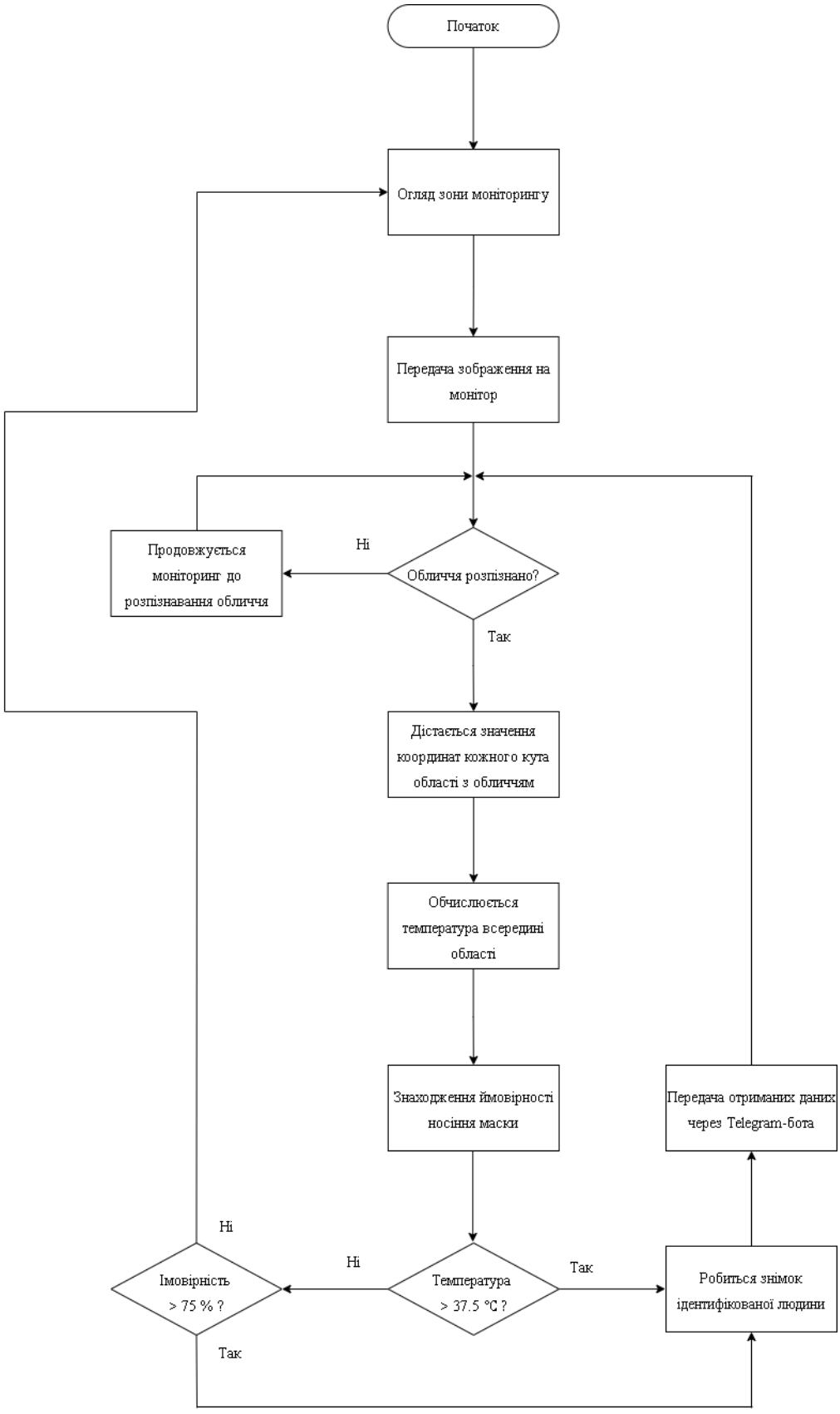



| | | | | | ІАЛЦ 467200.005 Д2 | | |
|---|---|---|---|---|---|---|---|
| Зм. | Арк. | № докум. | Підпис | Дата | | | |
| Розробив | | Карпенко Я.Р. | | | Система контролю стану приміщення при пандемії Функціональна схема класів | Лит. | Аркуш 1 / Аркушів 1 |
| Перевірив | | Виноградов Ю.М | | | | | |
| Т. Контр. | | | | | | КПІ ім. Ігоря Сікорського, ФІОТ, ІО-81 | |
| Н. Контр. | | Сімоненко В. П | | | | | |
| Затвердив | | | | | | | |

# Додаток 3

# ПРИНЦИПОВА СХЕМА РОБОТИ
## до дипломного проєкту
## на тему: «Система контролю стану приміщення при пандемії»

Київ – 2022




| Зм. | Арк. | № докум. | Підпис | Дата | ІАЛЦ 467200.006 Д3 | | |
|---|---|---|---|---|---|---|---|
| Розробив | | Карпенко Я. Р. | | | Система контролю стану приміщення при пандемії Принципова схема | Лит. | Аркуш 1 / Аркушів 1 |
| Перевірив | | Виноградов Ю.М | | | | | |
| Т. Контр. | | | | | | КПІ ім. Ігоря Сікорського, ФІОТ, ІО-81 | |
| Н. Контр. | | Сімоненко В. П | | | | | |
| Затвердив | | | | | | | |

# Додаток 4

# ТЕКСТ ПРОГРАМИ
## до дипломного проєкту
## на тему: «Система контролю стану приміщення при пандемії»

Київ – 2022

# Main.py

```
import flir_image_extractor
import cv2
import numpy as np
import os
import io
from google.cloud import vision
import telepot
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.models import load_model

from PyQt5.QtWidgets import QApplication
from PyQt5.QtWidgets import QWidget
from PyQt5.QtGui import QImage
from PyQt5.QtGui import QPixmap
from PyQt5.QtCore import QTimer

from ui_main_window import *

from collections import deque

token = 'telegram-token-value'
mc = 'value'
bot = telepot.Bot(token)
os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = r'ServiceAccountToken.json'
client = vision.ImageAnnotatorClient()
facenet = cv2.dnn.readNet('../training custom dataset/face_detector/deploy.prototxt', '../training custom dataset/face_detector/res10_300x300_ssd_iter_140000.caffemodel')
model = load_model('../training custom dataset/mask_detector.model')

flir = flir_image_extractor.FlirImageExtractor()

class MainWindow(QWidget):
    def __init__(self):
        super().__init__()
        self.ui = Ui_Form()
        self.ui.setupUi(self)

        self.timer = QTimer()

        self.timer.timeout.connect(self.viewCam)
        self.timer.timeout.connect(self.viewThermalCam)
        self.cam = cv2.VideoCapture(0)
        self.camthermal = cv2.VideoCapture(1)

        self.timer.start(20)
        image = cv2.imread('Logo.png')
        image = cv2.resize(image, dsize=(0, 0), fx=0.5, fy=0.7, interpolation=cv2.INTER_LINEAR)
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        self.number = 0
        self.dq = deque()
        self.textdq = deque([])
        height, width, channel = image.shape
        step = channel * width

        qImg = QImage(image.data, width, height, step, QImage.Format_RGB888)
        self.ui.label_3.setPixmap(QPixmap.fromImage(qImg))
```

```
    self.prevTime = 0
    self.temperature = 0
    self.nomask = 0
    self.response = 0

def put_img_to_labels(self, dq):
    image = self.dq[0]
    image = cv2.resize(image, dsize=(0, 0), fx=0.2, fy=0.2, interpolation=cv2.INTER_LINEAR)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    height, width, channel = image.shape
    step = channel * width
    qImg = QImage(image.data, width, height, step, QImage.Format_RGB888)
    self.ui.label.setPixmap(QPixmap.fromImage(qImg))

def label_2_text(self, textdq):
    TEXT = ""
    if len(self.textdq) > 10:
        textdq.pop()
    for text in textdq:
        TEXT += text + '\n'
    self.ui.label_2.setText(TEXT)

def get_max_temperature(self, thermal_np, x1, y1, x2, y2):
    crop = thermal_np[y1:y2, x1:x2]
        if crop.size == 0:
        return None

    return np.max(crop)

def viewCam(self):
    ret, img = self.cam.read()
    ret, thermal_image_data = self.camthermal.read()
    img = cv2.rotate(img, cv2.ROTATE_90_CLOCKWISE)

    h, w = img.shape[:2]
    blob = cv2.dnn.blobFromImage(img, scalefactor=1., size=(300, 300), mean=(104., 177., 123.))

    facenet.setInput(blob)
    dets = facenet.forward()
    result_img = img.copy()

    for i in range(dets.shape[2]):
        confidence = dets[0, 0, i, 2]

        if confidence < 0.5:
            continue

        x1 = int(dets[0, 0, i, 3] * w)
        y1 = int(dets[0, 0, i, 4] * h)
        x2 = int(dets[0, 0, i, 5] * w)
        y2 = int(dets[0, 0, i, 6] * h)
        face = img[y1:y2, x1:x2]


        face_input = cv2.resize(face, dsize=(224, 224))
        face_input = cv2.cvtColor(face_input, cv2.COLOR_BGR2RGB)
        face_input = preprocess_input(face_input)
        face_input = np.expand_dims(face_input,axis=0)
```

```
    mask, self.nomask = model.predict(face_input).squeeze()

        if mask > self.nomask:
            color = (0, 255, 0)
            label = 'Mask %d%%' % (mask * 100)
        else:
            color = (0, 0, 255)
            label = 'No Mask %d%%' % (self.nomask * 100)

        cv2.rectangle(result_img, pt1=(x1, y1), pt2=(x2, y2), thickness=7, color=color, lineType=cv2.LINE_AA)
        cv2.putText(result_img, text=label, org=(x1, y1 - 10), fontFace=cv2.FONT_HERSHEY_SIMPLEX, fontScale=4, color=color, thickness=6, lineType=cv2.LINE_AA)

        thermal_np = flir.process_image(thermal_image_data)
        max_temperature = self.get_max_temperature(thermal_np, x1, y1, x2, y2)

        if self.nomask >= 0.75 or max_temperature >= 37.5:
            self.dq.appendleft(face)
            if len(self.dq) == 4:
                self.dq.pop()
            self.number += 1
            IMAGE_FILE = 'No_Mask-High_Temp/' + str(i) + '_' + str('No_Mask%d%%_' % (self.nomask * 100) + str(self.number)) + 'Temp_' + str(max_temperature) + '.jpg'
            cv2.imwrite(IMAGE_FILE, result_img)

            saved_file = 'No_Mask_File/' + str(i) + '_' + str('No_Mask%d%%_' % (self.nomask * 100) + str(self.number)) + '.jpg'
            cv2.imwrite(saved_file, result_img)
            self.put_img_to_labels(self.dq)

            message = 'temperature :' + str(max_temperature) + '\nProbability of not wearing a mask : ' + str('%d%%' % (self.nomask * 100))

            f = open(IMAGE_FILE,'rb')
            self.response = bot.sendPhoto(mc, f)
            self.response = bot.sendMessage(mc,message)

    message2 = 'temperature :' + str(self.temperature) + 'Probability of not wearing a mask : ' + str('%d%%' % (self.nomask * 100))
    self.textdq.appendleft(message2)
    self.label_2_text(self.textdq)

    image = cv2.resize(result_img, dsize=(0, 0), fx=0.2, fy=0.2, interpolation=cv2.INTER_LINEAR)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    height, width, channel = image.shape
    step = channel * width
    qImg = QImage(image.data, width, height, step, QImage.Format_RGB888)
    self.ui.main_video.setPixmap(QPixmap.fromImage(qImg))




  def viewThermalCam(self):
    ret, image = self.camthermal.read()
    image = cv2.rotate(image, cv2.ROTATE_90_CLOCKWISE)
    image = cv2.resize(image, dsize=(0, 0), fx=0.2, fy=0.2, interpolation=cv2.INTER_LINEAR)

    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

```
        height, width, channel = image.shape
        step = channel * width

        qImg = QImage(image.data, width, height, step, QImage.Format_RGB888)

        self.ui.thermal_video.setPixmap(QPixmap.fromImage(qImg))

if __name__ == '__main__':
    app = QApplication(sys.argv)

    mainWindow = MainWindow()
    mainWindow.show()

    sys.exit(app.exec_())
```