

**Міністерство освіти і науки України**  
**Національний університет «Львівська політехніка»**  
**Кафедра програмного забезпечення**

**Звіт**  
про переддипломну практику за темою бакалаврської кваліфікаційної роботи

---

студента IV курсу групи \_\_\_\_\_

\_\_\_\_\_  
(прізвище, ініціали)

**база практики** \_\_\_\_\_

\_\_\_\_\_  
(назва підприємства)

**термін практики**    з « \_\_\_\_\_ » до « \_\_\_\_\_ »

**Керівники практики**

від кафедри \_\_\_\_\_  
(підпис)

від бази практики \_\_\_\_\_  
(підпис)

**Керівник бакалаврської  
кваліфікаційної роботи**

\_\_\_\_\_  
(підпис)    (наук.ст., вч.звання, ППП)

**Оцінка** \_\_\_\_\_ **Дата** \_\_\_\_\_

Львів 2020

## ЗМІСТ

1. КОРОТКИЙ ОПИС БАЗИ ПРАКТИКИ.....	4
2. ЗАВДАННЯ, ОТРИМАНЕ НА БАЗІ ПРАКТИКИ.....	5
3. РЕЗУЛЬТАТИ ВИКОНАННЯ ЗАВДАННЯ НА БАЗІ ПРАКТИКИ.....	6
3.1. Операційні системами реального часу.....	6
3.2. FreeRTOS.....	6
3.3. Практична частина.....	9
4. ЗАВДАННЯ ДИПЛОМНОГО ПРОЕКТУ .....	13
5. ОГЛЯД ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ У ПРОЦЕСАХ ЛОГУВАННЯ ДАНИХ .....	14
5.1. Опис програмної області використання інформаційних технологій у процесах логування даних.....	14
5.2. Загальні відомості про реєстратори даних і область їх застосування ....	15
5.3. Аналіз існуючих програмних засобів логування даних у низькорівневих системах.....	18
5.4. Висновки до розділу .....	21
6. ПОСТАНОВКА ЗАВДАННЯ ДЛЯ РОЗРОБКИ ПРИСТРОЮ ДЛЯ ЛОГУВАННЯ ДАНИХ СЕНСОРІВ НА ОСНОВІ МІКРОКОНТРОЛЕРА STM32 .....	22
6.1. Постановка завдання .....	22
6.2. Специфікація вимог до програмного продукту .....	22
6.3. Опис технологій та засобів розробки .....	28
6.4. Висновки до розділу .....	32
7. ПРОЕКТУВАННЯ ЛОГГЕРА ДАНИХ СЕНСОРІВ НА ОСНОВІ МІКРОКОНТРОЛЕРА STM32.....	33
7.1. Налаштування системи мікроконтролера.....	33
7.2. Структура збереження даних в пам'яті.....	37
7.3. Взаємодія з флеш пам'яттю.....	38

7.4. Застосування FreeRTOS .....	41
7.5. Висновки до розділу.....	42
8. ВИСНОВКИ ПРО ОТРИМАНІ ПІД ЧАС ПРАКТИКИ РЕЗУЛЬТАТИ.....	43
СПИСОК ЛІТЕРАТУРИ .....	45
ДОДАТКИ .....	47
Додаток А. Порівняння логгерів.....	47
Додаток Б. Діаграма прецедентів.....	48
Додаток В. Флеш пам'ять STM32F303RE.....	49
Додаток Г. Блок-схема алгоритму запису у сторінку флеш пам'яті .....	50
Додаток Д. Діаграма послідовності.....	51

## **1. КОРОТКИЙ ОПИС БАЗИ ПРАКТИКИ**

GlobalLogic Україна (англ. GlobalLogic Ukraine) — IT-компанія повного циклу розробки програмних продуктів. Компанія є українським підрозділом GlobalLogic і має офіси в Києві, Харкові, Львові, Миколаєві, співпрацює з більш ніж 4000 розробниками в Україні. За даними щопіврічного рейтингу ДООУ "ТОП-50", GlobalLogic входить до трійки найбільших IT-компаній України. GlobalLogic Україна входить до GlobalLogic Inc., яка налічує понад 13000 спеціалістів в Аргентині, Великій Британії, Ізраїлі, Індії, Китаї, Польщі, Словаччині, США, Україні, Хорватії та Чилі. Головний офіс GlobalLogic знаходиться у Сан-Хосе (Каліфорнія, США). Компанія займається розробкою програмних продуктів у таких сферах як цифрові медіа та телекомунікації, автомобільні технології, охорона здоров'я, фінанси, роздрібна торгівля та електронна комерція.

ФОП Дячок Р.В., який є керівником практики, створює власні продукти та надає послуги GlobalLogic Україна. ФОП Дячок Р.В. допомагає компанії створювати, та підтримувати цінність протягом усього життєвого циклу продукту. Він має близько 5 років досвіду у галузі розробки програмного забезпечення для вбудованих систем. За цей час був задіяний у 3 проектах, різними за своїми напрямками. Зараз він задіяний на проектах, що створюють новації в галузі медицини та охорони здоров'я. Також бере участь у менторській діяльності – готує людей без комерційного досвіду до роботи у компанії.

## 2. ЗАВДАННЯ, ОТРИМАНЕ НА БАЗІ ПРАКТИКИ

На базі практики було поставлено завдання освоїти операційну систему реального часу FreeRTOS та за допомогою функціоналу, що дає ця технологія розробити власний алгоритм динамічного виділення пам'яті. Вимогами до даного завдання були застосування мікроконтролера STM32, як пристрій, на якому буде виконуватися програма. Обмежень в середовищі розробки не було, тому спільно було погоджено використання STM32CubeIDE, як найбільш сумісної платформи з мікроконтролерами STM32.

Дане завдання було структуроване та розбите на такі частини:

1. Опрацювання теоретичного матеріалу про операційні системами реального часу. Визначення основних особливостей та галузей застосування.
2. Опрацювання теоретичного матеріалу про ОСРЧ FreeRTOS. Визначення основних особливостей та галузей застосування.
3. Використання набутих знань на практиці:
  - 3.1. Запуск FreeRTOS на мікроконтролері.
  - 3.2. Опрацювання наявних алгоритмів динамічного виділення пам'яті.
  - 3.3. Розробка власного алгоритму для динамічного виділення пам'яті.

### **3. РЕЗУЛЬТАТИ ВИКОНАННЯ ЗАВДАННЯ НА БАЗІ ПРАКТИКИ**

#### **3.1. Операційні системами реального часу**

За допомогою електронних ресурсів, наукових статей та книжок з інформаційних технологій було проведено ознайомлення з операційними системами реального часу. Операційні системи реального часу (ОСРЧ (RTOS)) призначені для забезпечення інтерфейсу до ресурсів, критичних за часом систем реального часу. Основним завданням в таких системах є своєчасність (timeliness) виконання обробки даних. Також було проведено порівняння між системами жорсткого та м'якого реального часу та визначено їх основні відмінності. Головною ознакою між цими типами є те, що в ОС жорсткого реального часу кожна задача повинна виконуватися за відведений квант часу, не виконання цієї умови веде до краху цілої системи.

#### **3.2. FreeRTOS**

За допомогою електронних ресурсів і статей ознайомився з одною із ОСРЧ, а саме FreeRTOS. За рекомендаціями керівника практики багато корисної і структурованої інформації було взято із самого електронного ресурсу FreeRTOS-<https://www.freertos.org/>. Під час вивчення нової технології ознайомився з можливостями, що дає FreeRTOS для роботи з мікроконтролерами:

- реалізація багатозадачності для вбудованих систем;
- вирішення проблем синхронізації;
- спрощення коду для задач з нетривіальною логікою;
- виконання вимог жорсткого реального часу.

Також ознайомився з основним набором функціоналу, що дає FreeRTOS для виконання завдань:

- таски – для реалізації багатозадачності;
- черги – для міжпроцесної взаємодії;
- мютекси, семафори, критичні секції – для синхронізації ресурсів.

### 3.3. Практична частина

#### 3.3.1. Запуск FreeRTOS на мікроконтролері

Було виконано інтеграцію FreeRTOS у тестовий проект в середовищі STM32CubeIDE. Та створено найпростіше завдання для блимання діодом на мікроконтролері, щоб перевірити, що інтеграція пройшла успішно та FreeRTOS запусився на мікроконтролері.

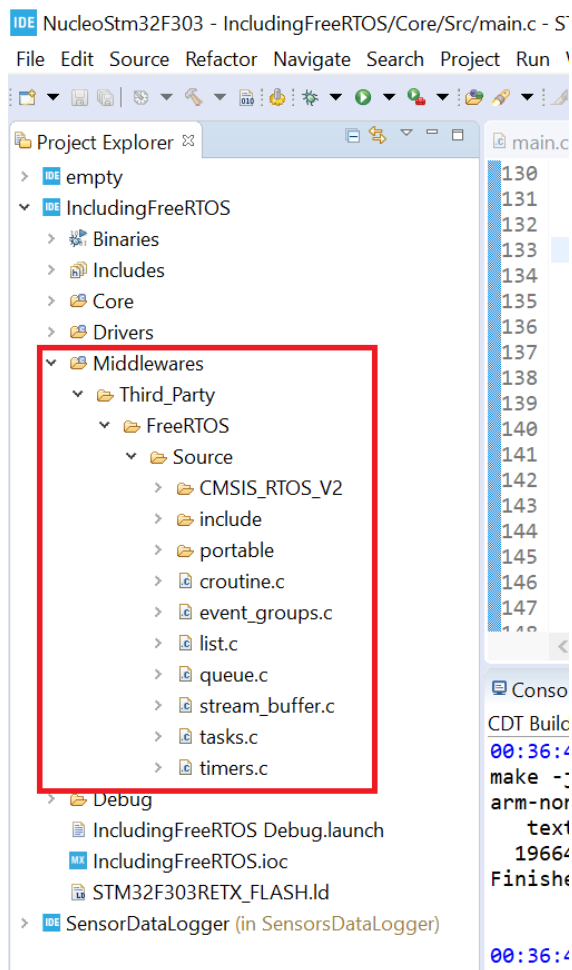


Рис. 3.1. Інтеграція FreeRTOS у проект

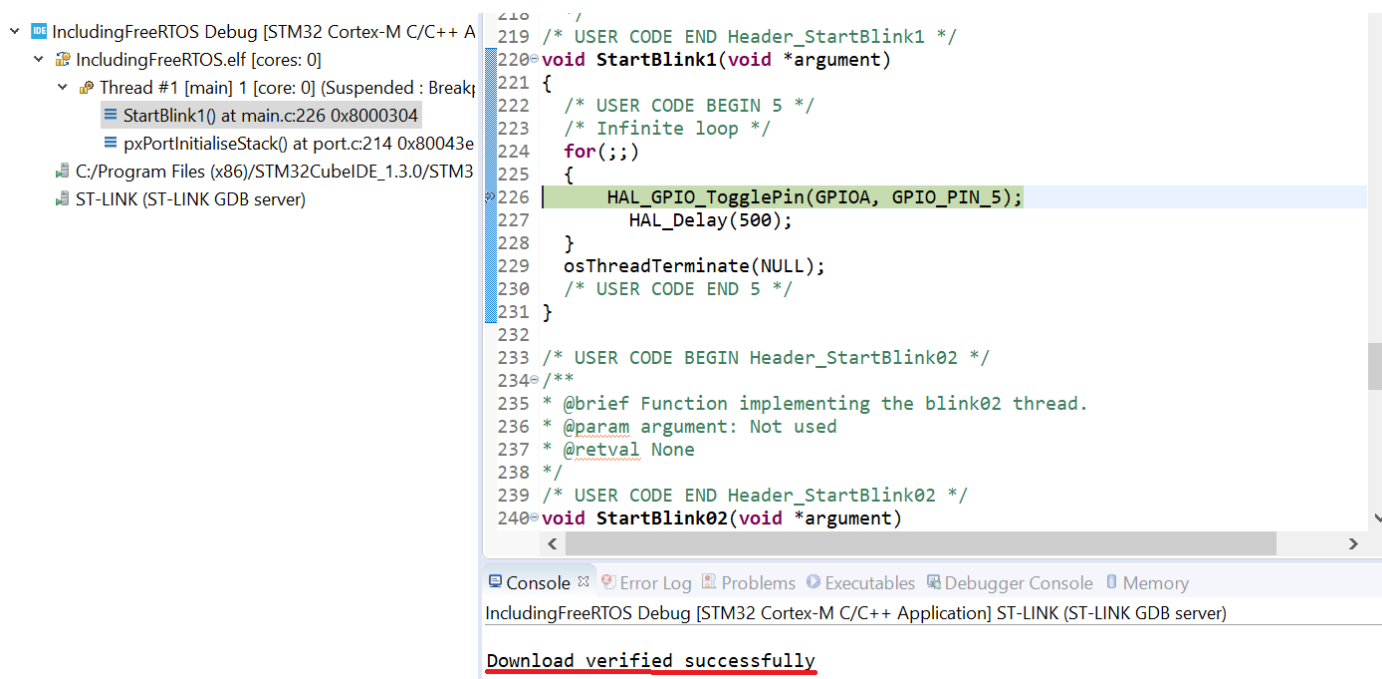


Рис. 3.2. Тестовий запуск FreeRTOS

### 3.3.2. Динамічне виділення пам'яті у FreeRTOS

Проведено опрацювання матеріалів щодо динамічного виділення пам'яті та аналіз алгоритмів, які вже реалізовані та можуть бути використані при роботі з FreeRTOS. Таких алгоритмів є 5. Щоб скористатись одним з них треба підключити файл з обраним алгоритмом. Назви файлів, що містять потрібні алгоритми: heap\_1.c, heap\_2.c, heap\_3.c, heap\_4.c, heap\_5.c. Для демонстраційного запуску було використано heap\_4.c.

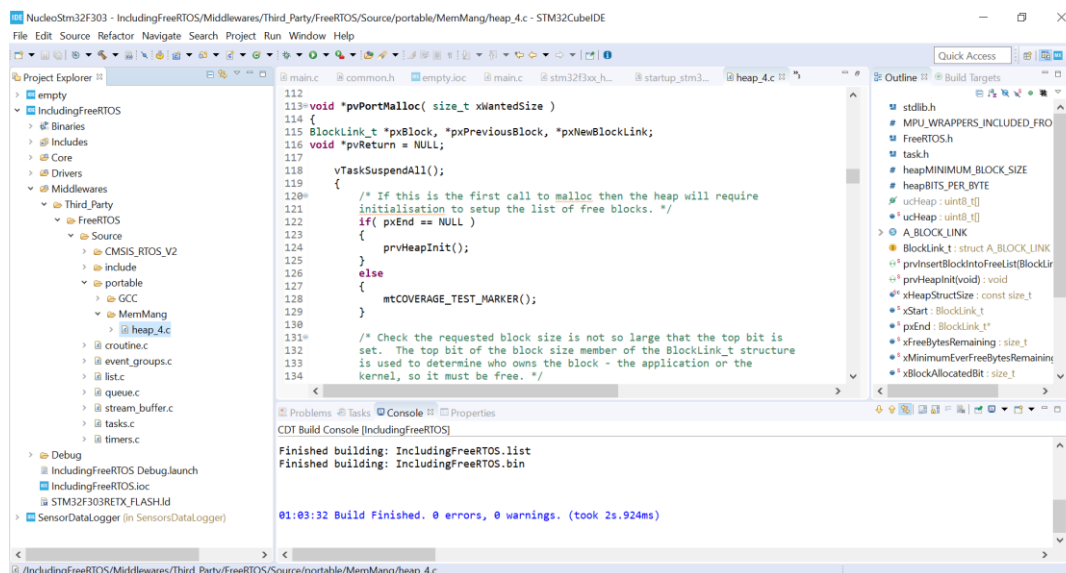


Рис. 3.3. Використання heap\_4.c



### 3.3.3. Розробка власного алгоритму для динамічного виділення пам'яті

Хоч FreeRTOS має власні алгоритми для динамічного видалення пам'яті, вони не завжди будуть оптимальними для вирішення різних типів задач, а в деяких випадках взагалі неприпустимими. Тому є сенс навчитися створювати власні засоби для динамічного виділення пам'яті для вбудованих систем. Розглянувши доступний функціонал, що пропонує FreeRTOS, було узгоджено використати реалізацію `heap_5.c`, як базову.

Для виділення блоку пам'яті є визначена структура `HeapRegion_t`, яка містить адресу початку блоку і його розмір. У своїй реалізації буде використано масив таких структур для виділення декількох блоків різного розміру. Проте цей масив має виконувати 2 вимоги:

- ділянки пам'яті мають бути в порядку адрес (від найменшої до найбільшої);
- масив мусить мати символ закінчення – наприклад, `NULL`.

Для того, щоб інкапсулювати функціональність `malloc_addblock`, додано керування внутрішнім масивом `HeapRegion_t` для відстеження блоків пам'яті. Також знадобиться кілька змінних стану, які допоможуть керувати процесом ініціалізації. Додано константу для максимального ліміту для записів регіону кучі. Оскільки для таблиці `HeapRegion_t` потрібно додатковий елемент, що буде вказувати на закінчення, виділено додатковий запис, щоб гарантувати, що завжди буде місце стоп-елемент. Також додано змінні для максимальної кількості записів і відсідковування поточної використаної пам'яті. Оскільки заборонено виділяти пам'ять, поки куча не ініціалізована, створено булеву змінну для перевірки стану ініціалізації.

Оскільки потрібно підтримувати можливість використання декількох блоків пам'яті в кучі, нам потрібно розділити функції `malloc_addblock` та `malloc_init`. Ця спрощена реалізація `malloc_addblock` просто додає новий запис до таблиці регіону кучи.

Як тільки додано кілька блоків пам'ят Простота і, можна ініціалізувати кучу за допомогою виклику `vPortDefineHeapRegions`. Перед тим, як викликати цю функцію, потрібно відсортувати ділянки пам'яті, щоб переконатися, що вони в правильному порядку. Після ініціалізації купи, булева змінна стану ініціалізації кучі встановлюється істинну. Тепер майбутні виклики `malloc()` зможуть завершуватись успішно.

Після завершення процесу ініціалізації виділення пам'яті реалізується досить просто (`custom_malloc()`). Здійснюється це через виклик `pvPortMalloc`, але з блокуванням всіх викликів `malloc` до завершення ініціалізації.

Код реалізованого програмного рішення:

```
#ifndef FREERTOS_HEAP_REGION_CNT
#define FREERTOS_HEAP_REGION_CNT 2
#endif

/// Maximum number of heap regions that can be specified
static const uint8_t heap_region_max = FREERTOS_HEAP_REGION_CNT;

/// Current number of allocated heap regions
static volatile uint8_t heap_region_cnt = 0;

/**
 * FreeRTOS internal memory pool structure when using heap_5.c
 *
 * The block with the lowest starting address should appear first in the array
 *
 * An additional block is allocated to serve as a NULL terminator
 */
static HeapRegion_t heap_regions[FREERTOS_HEAP_REGION_CNT + 1];

/**
 * Flag that is used in malloc() to cause competing threads to wait until
 * initialization is completed before allocating memory.
 */
static volatile bool initialized_ = false;

static int cmp_heap(const void* a, const void* b)
{
    const HeapRegion_t* ua = a;
    const HeapRegion_t* ub = b;

    return ((ua->pucStartAddress < ub->pucStartAddress)
        ? -1
        : ((ua->pucStartAddress != ub->pucStartAddress)));
}

/**
 * malloc_addblock must be called before memory allocation calls are made.
 * In this FreeRTOS implementation, malloc() calls will block until memory
```

```

    * has been allocated
    */
void malloc_addblock(void* addr, size_t size)
{
    assert(addr && (size > 0));
    assert((heap_region_cnt < heap_region_max) && "Too many heap regions!");

    // Increment the count early to claim a spot in case of multi-threads
    uint8_t cnt = heap_region_cnt++;

    if(cnt < heap_region_max)
    {
        heap_regions[cnt].pucStartAddress = (uint8_t*)addr;
        heap_regions[cnt].xSizeInBytes = size;
    }
    else
    {
        // Decrement the count if we don't have space
        heap_region_cnt--;
    }
}

void malloc_init()
{
    assert((heap_region_cnt > 0) && !initialized_);

    if(heap_region_cnt > 0 && !initialized_)
    {
        // Sort the heap regions so addresses are in the correct order
        qsort(heap_regions, heap_region_cnt, sizeof(HeapRegion_t), cmp_heap);

        // Pass the array into vPortDefineHeapRegions() to enable malloc()
        vPortDefineHeapRegions(heap_regions);

        initialized_ = true;
    }
}

void* custom_malloc(size_t size)
{
    void* ptr = NULL;

    while(!initialized_)
    {
        // Thread blocks until application malloc has been correctly initialized
        vTaskDelay(1);
    }

    if(size > 0)
    {
        // wrap the FreeRTOS call into a standard form
        ptr = pvPortMalloc(size);
    }

    return ptr;
}

void custom_free(void* ptr)
{

```

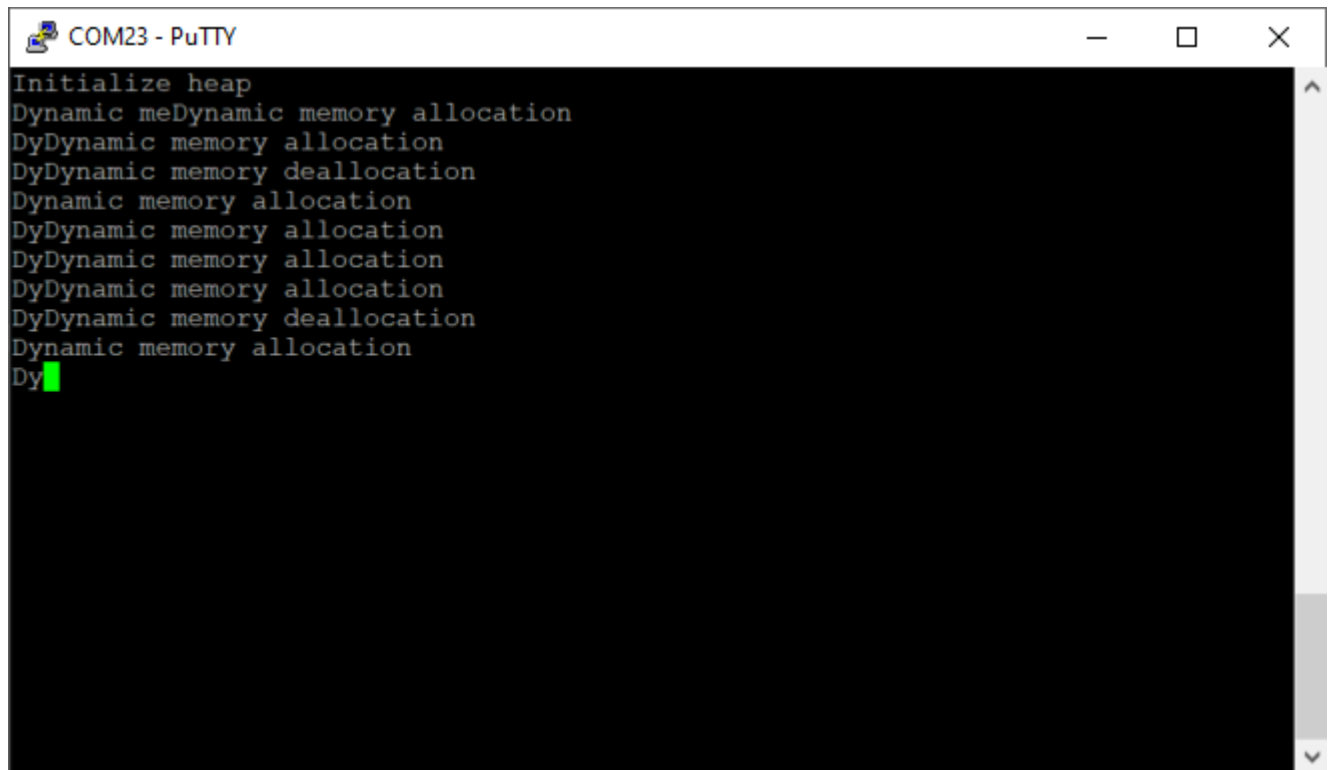
```

    /// free should NEVER be called before malloc is init
    assert(initialized_);

    if(ptr)
    {
        // wrap the FreeRTOS call into a standard form
        vPortFree(ptr);
    }
}

```

На рис. 3.4 зображено консольний вивід тестового запуску програми.



```

COM23 - PuTTY
Initialize heap
Dynamic meDynamic memory allocation
DyDynamic memory allocation
DyDynamic memory deallocation
Dynamic memory allocation
DyDynamic memory allocation
DyDynamic memory allocation
DyDynamic memory allocation
DyDynamic memory deallocation
Dynamic memory allocation
Dy

```

Рис. 3.4. Перевірка роботи алгоритму

Таким чином було створено власний алгоритм для виділення динамічної пам'яті, який дозволяє використовувати декілька вільних ділянок з різних місць (адрес), що дозволить збільшити розмір кучі.

#### 4. ЗАВДАННЯ ДИПЛОМНОГО ПРОЕКТУ

Завданням даної бакалаврської кваліфікаційної роботи є створення логгера даних сенсорів на основі мікроконтролера STM32. Така система передбачає виконання двох основних завдань — зчитування показників датчиків і зберігання отриманої інформації в пам'яті пристрою.

Даний документ містить опис таких розділів дипломного проекту:

1. Огляд використання інформаційних технологій у процесах логування даних.

Цей розділ описує предметну область, задачі, які поставлений вирішувати пристрій, що розробляється, приклади застосування логгерів даних. Також у ньому знаходиться дослідження пристроїв аналогів та вказано деякі переваги і недоліки проаналізованих пристроїв.

2. Постановка завдання для розробки пристрою для логування даних сенсорів на основі мікроконтролера stm32.

Тут структурована задача, яка має бути вирішена під час дипломної роботи. Розроблено специфікацію вимог. Також даний розділ містить опис засобів розробки, що будуть використані під час розробки.

3. Проектування логгера даних сенсорів на основі мікроконтролера stm32.

Цей розділ — один з найважливіших у дипломній роботі. Він описує підхід розробника до вирішення задачі. Тобто містить опис прототипу пристрою, який покликаний вирішувати поставлені завдання, пояснення архітектури застосунку, визначення алгоритмів, що спроектовані для виконання поставлених цілей.

## **5. ОГЛЯД ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ У ПРОЦЕСАХ ЛОГУВАННЯ ДАНИХ**

### **5.1. Опис програмної області використання інформаційних технологій у процесах логування даних**

Коли справа стосується аналізу ефективності та зручності використання інформаційних технологій, важливим є збір необхідних даних для прийняття рішень. Однак ці дані потрібно десь знайти та зберегти, щоб створити змістовні набори інформації. Цей збір необхідної інформації здійснюється за допомогою процесу, який називається реєстрацією або логуванням даних.

Логування даних — це запис даних за деякий проміжок часу або окремим пристроєм, або комп'ютерною системою, створеною для збору інформації. Тип записаних даних залежить від типу середовища, в якому знаходиться реєстратор даних [1]. Наприклад, окремі пристрої можуть накопичувати інформацію про такі речі, як дослідження дикої природи, погодні умови, умови дорожнього руху, дії транспортного засобу, стан робочого приладу, та багато інших [1]. Комп'ютерні системи, створені для ведення реєстру, як правило, збирають дані про події, що відбуваються на комп'ютері, мережі, сервері або цілій ІТ-системі.

Після запису дані можуть бути отримані та проаналізовані вручну або за допомогою спеціального програмного забезпечення, яке допомагає в аналізі даних. Таке програмне забезпечення, як правило, пропонує огляд зібраних даних у візуальній формі (наприклад, графік чи гістограма) для швидкого огляду та кращого сприйняття, а також більш детальні звіти, що дозволяють глибше зрозуміти ключові показники, які фіксує пристрій, що записує дані.

Для цілей моніторингу програм логгер даних записує інформацію, що стосується обслуговування інфраструктури, необхідної для запуску програми. Це можуть бути різні типи даних, такі як дії, які виконуються користувачем на комп'ютерній або програмній системі, трафік на сервер і з нього, або дії, здійснені

в цілій ІТ-мережі [1]. Є багато різних подій, що можна зареєструвати, і вся ця інформація може бути корисною для подальшого аналізу, для покращення роботи своїх конкретних систем [1].

Що стосується реєстрації даних у додатках, то це може бути розширено для запису інформації не тільки на веб-сайті, але й на інші фрагменти, такі як бази даних, сервери додатків, цілі мережі та інші частини ІТ-системи, пов'язані з будь-яким числом програм [1].

## **5.2. Загальні відомості про реєстратори даних і область їх застосування**

Реєстратор даних (також логгер даних) — це електронний пристрій, який записує у внутрішню пам'ять, на зовнішнє сховище або передає на хмарний сервіс дані протягом певного періоду часу [3]. Сама інформація також зберігається через певні проміжки часу або по відношенню до зовнішніх подій, що є тригерами здійснення записів. Дані можуть надходити від вбудованого в прилад сенсора або від зовнішніх приладів і датчиків. На даний час майже всі реєстратори даних створюються на базі цифрових процесорів (або комп'ютерів). Вони, як правило, невеликі, на батарейках, портативні, і забезпечені мікропроцесором, внутрішньою пам'яттю для зберігання даних і різними вбудованими датчиками для вимірювань [3]. Деякі реєстратори даних мають спеціальний інтерфейс для підключення до персонального комп'ютера, таким чином користувач може використовувати програмне забезпечення, щоб активувати прилад, переглядати і аналізувати зібрані дані. У той же час деякі логгери можуть мати локальний інтерфейс (клавіатура, дисплей) і можуть бути використані як автономні пристрої [3].

Реєстратори даних варіюються від приладів загального призначення для вимірювальних застосувань до дуже специфічних реєстраторів для вимірювання показників в одному середовищі, або тільки одного параметра [1]. Можливість програмування є спільною особливістю для логгерів загального призначення, в

той час як багато інших реєстраторів залишаються статичними машинами з обмеженою кількістю або повною відсутністю змінних параметрів [1].

Однією з основних переваг використання логгерів даних є можливість автоматично збирати дані цілодобово. Після активації реєстратори даних продовжують свою роботу, залишені без нагляду, для вимірювання і запису інформації протягом визначеного періоду для моніторингу. Це дозволяє забезпечити всебічне, точне уявлення про умови навколишнього середовища і спостережувані параметри технологічного процесу, такі як температура повітря, відносна вологість, тиск, вібрація, навантаження тощо.

Що стосується цільової аудиторії, то реєстратори даних є цінним інструментом для тих, хто хоче постійно контролювати умови, не знаходячись на місці. Екологічні консультанти, керівники будівель, енергетичні аудитори, науковці, проводять дослідження, та багато інших — усі ці професіонали покладаються на реєстратори даних через низку переваг.

#### ❖ *Низька вартість*

В останні роки ціна реєстраторів даних знизилася з прогресом у мікропроцесорах, і один, односенсорний логгер може коштувати менше 50 доларів. Користувачі заощаджують час та гроші, запускаючи та залишаючи реєстратор, який без нагляду проводить вимірювання, замість того, щоб бути на місці особисто [2]. Низька вартість логгерів даних також дозволяє розгортати кілька реєстраторів одночасно, в деяких випадках скорочуючи тривалість проекту та збільшуючи обсяг даних, отриманих для аналізу [2].

Через тривалий час автономної роботи, характерний для багатьох портативних реєстраторів даних, вони можуть бути розгорнуті протягом тривалого періоду часу, тобто менше витраченого часу на дорогу до і від реєстратора для завантаження даних. У випадку дистанційного розміщення пристроїв або в тих випадках, коли використовується багато логгерів, це може бути величезною економією часу та грошей.



### ❖ *Простота використання*

Реєстратори даних, безумовно, використовуються інженерами та науковими співробітниками, але вони також можуть застосовуватись не настільки кваліфікованими людьми, наприклад учнями в шкільних проектах. Щоб залучити більше користувачів пристрої мають бути зручними та зрозумілими у використанні. Для деяких моделей реєстраторів потрібно просто підключити їх до комп'ютера, використати програмне забезпечення даного приладу для налаштування частоти вибірки та часу запуску/зупинки, і роботу можна розпочинати [2]. Реєстратори з технологією Bluetooth (BLE) не потребують навіть комп'ютера, лише мобільного пристрою та додатку [2]. Завантаження даних також простий процес, і їх можна легко проаналізувати, зрозуміти та відформатувати для презентації чи експорту в інші програми.

### ❖ *Надійність*

Після налаштування та розгортання можна розраховувати на реєстратор даних, що працює на акумуляторі, щоб збирати потрібну інформацію стільки, скільки вам потрібно [2]. Немає людських помилок, конфліктів у графіку, поганої погоди, незрозумілого почерку чи будь-якого іншого людського фактору, що може вплинути на збір і точність даних. Незалежно від того чи в приміщенні, чи поза ним, відповідні моделі логгерів даних призначені витримувати широкий спектр екологічних умов, включаючи солону воду, холодильну систему, сильний вітер та постійне сонячне світло [2].

### ❖ *Мітка часу*

Збір даних, безумовно, основне завдання цих пристроїв, і їх мета полягає у наданні об'єктивних, точних, визначених міткою часу показників, які можуть бути використані для тестування теорій, підтвердження операцій, надання записів для наглядових або регуляторних агенцій та встановлення прийнятних рішень. Дані, які збирають логгери, дозволяють краще аналізувати інформацію, а також можуть

заощадити час та гроші в довгостроковій перспективі. Проте без часового показника такий аналіз не принесе багато дивідендів.

### **5.3. Аналіз існуючих програмних засобів логування даних у низькорівневих системах.**

*OM-EL-USB* серії USB-реєстраторів даних включає в себе моделі вимірювання температури, температури та відносної вологості, напруги та струму. Користувач може легко налаштувати USB-реєстратор даних і завантажити збережені дані, підключивши модуль до USB-порту ПК та запустивши просте програмне забезпечення Windows [5]. Параметри налаштування, що вибираються програмним забезпеченням, включають швидкість реєстрації, час запуску, високі/низькі налаштування тривоги та одиниці температури ( $^{\circ}\text{C}$  або  $^{\circ}\text{F}$ ). Кожен реєстратор даних постачається в комплекті з довготривалим літійовим акумулятором [5]. Вбудовані світлодіодні індикатори показують стан реєстратора даних [5].



Рис. 5.1. Логгер даних OM-EL-USB

*Переваги приладу:*

- ✓ багато параметрів для налаштування;

- ✓ довго тримає заряд батареї;
- ✓ низька вартість.

*Недоліки приладу:*

- ✓ відносно габаритний через великий відсік для батареї та USB порт;
- ✓ складність конфігурації — можливість конфігурації лише за допомогою спеціального ПЗ призначеного для персональних комп'ютерів.

**НОВО MX2201** — водонепроникний реєстратор температури, використовує потужність Bluetooth з низьким енергоспоживанням (BLE) для передачі точних вимірів температури прямо на ваш мобільний пристрій. Цей компактний логгер даних є дуже міцним, через це ідеально підходить для вимірювання температури в непростих природних умовах: озерах, океанах, горах та ґрунті. Виділяються такі його функції: зручне налаштування бездротового зв'язку та завантаження через Bluetooth, великий обсяг пам'яті (до 96 000 вимірювань), водонепроникний (до 30 метрів), світлодіодна сигналізація, точність вимірювань  $\pm 0,5$  °C ( $\pm 0,9$  ° F) точність [7].



Рис. 5.2. Логгер даних НОВО MX2201

*Переваги приладу:*

- ✓ дуже компактний;
- ✓ має засоби кріплення;
- ✓ витримує великі навантаження;
- ✓ можливість налаштування через Bluetooth (мобільний телефон).

*Недоліки приладу:*

- ✓ акумулятор;
- ✓ вартість.

*MSR145* — логгер даних високого рівня якості розміром з великий палець, записує понад 2 000 000 виміряних значень — за допомогою додаткової картки microSD, навіть понад мільярд вимірюваних значень — і ідеально підходить для довготривалих вимірювань [6]. Він здатний одночасно вимірювати і записувати різні параметри, такі як температура, вологість, тиск, прискорення і світло. Всі виміряні значення можна швидко перенести на ПК або ноутбук через інтерфейс USB або карту microSD для аналізу даних.



Рис. 5.3. Логгер даних MSR145

*Переваги приладу:*

- ✓ підтримує встановлення додаткової пам'яті;
- ✓ вимірює велику кількість показників;
- ✓ підтримує 2 інтерфейси для зчитування даних;
- ✓ компактні розміри;

- ✓ сповіщення про стан системи через набір діодів.

#### **5.4. Висновки до розділу**

У даному розділі було зроблено детальний розбір задачі логування даних і здійснено загальний опис реєстраторів даних, вказано основну мету роботи аналізованих пристроїв і розписано основні проблеми, що допомагають вирішити ці прилади. Також проведено аналіз пристроїв аналогів, що є популярними на ринку, і було визначено їх переваги та недоліки.

Виконавши розбір предметної області, можна зробити висновок, що розробка логгерів даних є досить поширеною задачею. Викликано це тим, що є безліч типів інформації, збір якої можна автоматизувати, використовуючи логгери даних. Проте немає універсального реєстратора, щоб міг опрацьовувати таку кількість показників. По аналізу пристроїв аналогів також можна з упевненістю сказати, що дана предметна область ще розвивається і відкрита до впровадження нових програмних рішень. З того функціоналу чи характеристик, які можна покращити, можна виділити такі: зменшення габаритів приладів, застосування бездротового інтерфейсу для роботи, збільшення обсягів пам'яті для зберігання даних.

## **6. ПОСТАНОВКА ЗАВДАННЯ ДЛЯ РОЗРОБКИ ПРИСТРОЮ ДЛЯ ЛОГУВАННЯ ДАНИХ СЕНСОРІВ НА ОСНОВІ МІКРОКОНТРОЛЕРА STM32**

### **6.1. Постановка завдання**

Завданням даної бакалаврської кваліфікаційної роботи є створення логгера даних сенсорів на основі мікроконтролера STM32. Така система передбачає виконання двох основних завдань — зчитування показників датчиків і зберігання отриманої інформації в пам'яті пристрою. Оскільки пристрій має підтримувати роботу з кількома сенсорами, потрібно вирішити завдання синхронізації між ними. Тому, щоб реалізувати багатопоточність і використати примітиви синхронізації, буде застосовано операційну систему реального часу FreeRTOS. Це також дозволить зменшити складність програмної реалізації, покращити гнучкість та супровід даної системи.

Особливістю зберігання отриманих даних є правильна взаємодія з флеш пам'яттю пристрою. Таке завдання передбачає аналіз типу пам'яті та особливостей роботи з нею. Тому для збереження інформації буде розроблено алгоритм, за яким дані будуть записуватись у флеш пам'ять. Такий підхід дозволить зменшити можливість втрати даних і зробить процес запису більш зрозумілим.

### **6.2. Специфікація вимог до програмного продукту**

#### **6.2.1. Вступ**

##### **6.2.1.1. Мета створення**

Мета створення даного програмного продукту — забезпечити можливість зчитування показників температури та вологості за певний період часу.

##### **6.2.1.2. Продукти-аналоги**

Як продукти-аналоги було взято: OM-EL-USB, HOBO MX2201, MSR145. Результат їх порівняння зображено в додатку А.

## 6.2.2. Загальний опис

### 6.2.2.1. Характеристики продукту

Основні функції, які повинні бути реалізовані:

- зчитування показника температури
- зчитування показника вологості
- зчитування значення потенціометра (симуляція сенсора)
- зберігання отриманих даних у пам'яті пристрою
- попередження про можливу втрату даних

### 6.2.2.2. Класи користувачів та їх характеристики

Система, що розробляється, не передбачає різних класів користувачів, оскільки вона має один сценарій взаємодії з користувачем. Потрібно лише увімкнути прилад, а далі він продовжує роботу без стороннього втручання — автономно. У додатку Б наведена діаграма прецедентів.

### 6.2.2.3. Середовище функціонування

Пристрій, що розробляється, передбачає використання мікроконтролера STM32F303, через що програмне забезпечення буде залежати від особливостей його архітектури. Логгер даних сенсорів температури і вологості буде працювати правильно на мікроконтролерах даної серії. Апаратні характеристики, необхідні для правильного функціонування системи, зазначені в Таблиці 6.1.

Таблиця 6.1.

Апаратні вимоги

Апаратна характеристика	Вимога
Мікропроцесор	ARM Cortex-M4 32-bit RISC
Тактова частота	72 MHz
Флеш пам'ять	512-Kbyte
Оперативна пам'ять	80-Kbyte

Продовження таблиці 6.1.

Апаратна характеристика	Вимога
Аналогово-цифрові перетворювачі	4
16-бітні таймери	5 загального призначення
16-бітні таймери	1 загального призначення
Апаратна характеристика	Вимога
RTC (real time clock)	1
USARTs	3

### 6.2.3. Характеристики системи

#### 6.2.3.1. Отримання показників температури

##### 6.2.3.1.1. Опис і пріоритет

Пріоритет — високий. Система має отримувати значення температури від сенсора температури і вологості.

##### 6.2.3.1.2. Послідовність дія-відгук

- Система відсилає сигнал датчику про готовність приймати дані.
- Датчик посилає контрольний сигнал.
- Система перевіряє відповідь датчика.
- Датчик посилає виміряне значення температури.
- Система приймає значення.
- Система перевіряє коректність отриманих даних.

##### 6.2.3.1.3. Функціональні вимоги

REQ-1. Підключення та налаштування сенсора температури і вологості.

REQ-2. Перевірка відповіді сенсора, що встановлення з'єднання пройшло вдало.

REQ-3. Перевірка отриманого значення температури по контрольній сумі.



REQ-4. Визначення мітки часу здійснення виміру.

#### **6.2.3.2. Отримання показників вологості**

##### **6.2.3.2.1. Опис і пріоритет**

Пріоритет — високий. Система має отримувати значення вологості від сенсора температури і вологості.

##### **6.2.3.2.2. Послідовність дія-відгук**

- a) Система відсилає сигнал датчику про готовність приймати дані.
- b) Датчик посилає контрольний сигнал.
- c) Система перевіряє відповідь датчика.
- d) Датчик посилає виміряне значення вологості.
- e) Система приймає значення.
- f) Система перевіряє коректність отриманих даних.

##### **6.2.3.2.3. Функціональні вимоги**

REQ-1. Підключення та налаштування сенсора температури і вологості.

REQ-2. Перевірка відповіді сенсора, що встановлення з'єднання пройшло вдало.

REQ-3. Перевірка отриманого значення вологості по контрольній сумі.

REQ-4. Визначення мітки часу здійснення виміру.

#### **6.2.3.3. Отримання показників потенціометра**

##### **6.2.3.3.1. Опис і пріоритет**

Пріоритет — високий. Система має отримувати значення потенціометра для симуляції сенсора.

##### **6.2.3.3.2. Послідовність дія-відгук**

- a) Система виконує налаштування з'єднання з аналогового-цифровим перетворювачем.
- b) Система виконує налаштування аналогового-цифрового перетворювача для можливості отримання даних.
- c) Система зчитує значення потенціометра.

#### **6.2.3.3.3. Функціональні вимоги**

REQ-1. Підключення потенціометра для симуляції сенсора.

REQ-2. Підключення та конфігурація аналогово-цифрового перетворювача.

REQ-3. Визначення мітки часу здійснення виміру.

#### **6.2.3.4. Зберігання показників у конкретному форматі**

##### **6.2.3.4.1. Опис і пріоритет**

Пріоритет — високий. Показники сенсорів мають бути збережені в пам'яті пристрою у визначеному форматі для можливості подальшої роботи з ними.

##### **6.2.3.4.2. Послідовність дія-відгук**

а) Отримане значення сенсора записується в пам'ять пристрою.

##### **6.2.3.4.3. Функціональні вимоги**

REQ-1. Перевірка чи достатньо місця в пам'яті для запису нового значення.

REQ-2. Перевірка чи залишилось достатньо циклів запису в пам'ять (чи можна гарантувати збереження даних).

REQ-3. Дані, що записуються, перевіряються на повторення значення з останнім записом.

REQ-4. Дані мають бути збережені у визначеному форматі.

#### **6.2.3.5. Попередження про можливу втрату даних**

##### **6.2.3.5.1. Опис і пріоритет**

Пріоритет — високий. Система має увімкнути червоний діод при можливій втраті інформації (закінчилась гарантована кількість записів у пам'ять).

##### **6.2.3.5.2. Послідовність дія-відгук**

а) Система виявляє досягнення межі кількості гарантованих записів у пам'ять без втрати.

б) Система має сповістити користувача про можливі втрати інформації, увімкнувши червоний діод.

##### **6.2.3.5.3. Функціональні вимоги**

REQ-1. Підключення червоного діода до системи.

REQ-2. Слідування за кількістю записів у пам'ять.

REQ-3. Можливість увімкнення червоного діода при потребі.

## **6.2.4. Вимоги зовнішніх інтерфейсів**

### **6.2.4.1. Користувацькі інтерфейси**

Дана реалізація логгера даних не передбачає реалізацію користувацького інтерфейсу. Це завдання може стати вдосконаленням цього продукту у наступних версіях. Тоді користувацький інтерфейс може бути розроблений за допомогою LED дисплею та набору кнопок для взаємодії з пристроєм, або з використанням бездротової технології, що дасть можливість працювати з приладом через телефон чи персональний комп'ютер.

### **6.2.4.2. Апаратні інтерфейси**

Програмне забезпечення, що розробляється, буде розгортатися на мікроконтролері STM32F303RE, тому має повністю враховувати особливості даної лінійки мікропроцесорів і їх периферії.

### **6.2.4.3. Програмні інтерфейси**

Програмним інтерфейсом, що дозволить взаємодіяти з мікроконтролером та його периферією, є бібліотека HAL, що надається виробником даних мікроконтролерів – компанією STMicroelectronics. Вона безпосередньо побудована навколо загальної архітектури мікроконтролера і дозволяє значно спростити взаємодію з ним. Використання HAL бібліотеки спрощує повторне використання коду та гарантує просту переносимість на інші пристрої.

Вимогою до програмних інтерфейсів є використання операційної системи реального часу FreeRTOS

### **6.2.4.4. Комунікаційні інтерфейси**

Комунікаційний інтерфейс із сенсором температури і вологості має бути розроблений згідно з вимогами датчика та відповідно до процесу комунікації, на який він розрахований – послідовний інтерфейс.

## **6.2.5. Інші нефункціональні вимоги**

### **6.2.5.1. Вимоги продуктивності**

Оскільки реалізація системи передбачає застосування операційної системи реального часу FreeRTOS, то пристрій буде використовувати добре відлагоджені і оптимізовані засоби багатопоточності. Це зменшить кількість програмних помилок у системі та забезпечить кращу продуктивність.

### **6.2.5.2. Вимоги безпеки**

Оскільки основна система буде комунікувати із сенсором через серійний інтерфейс, це максимально зменшує можливість втрати даних і забезпечує дані від перехоплення. Для реалізації даної версії приладу додаткових заходів безпеки не передбачено.

### **6.2.5.3. Атрибути якості програмного продукту**

- ❖ Відмовостійкість
- ❖ Простота супроводу
- ❖ Надійність
- ❖ Легкорозширюваність

## **6.3. Опис технологій та засобів розробки**

### **6.3.1. Плата NUCLEO-F303RE**

Плати STM32 Nucleo-64 забезпечують доступний і гнучкий спосіб для користувачів випробувати нові концепції та побудувати прототипи, вибираючи з різних комбінацій функціональних можливостей та енергоспоживання, що надаються мікроконтролером STM32 [14].

Основні особливості, що дають плати STM32 Nucleo-64:

- мікроконтролер STM32 в оболонці LQFP64;
- 1 користувацький світлодіод;
- 1 користувацька кнопка та 1 кнопка скидання;
- гнучкі варіанти живлення: ST-LINK, USB VBUS або зовнішні джерела;
- бортовий відлагоджувач, або програматор, ST-LINK з можливістю

підключення через USB: віртуальний COM-порт та порт налагодження.

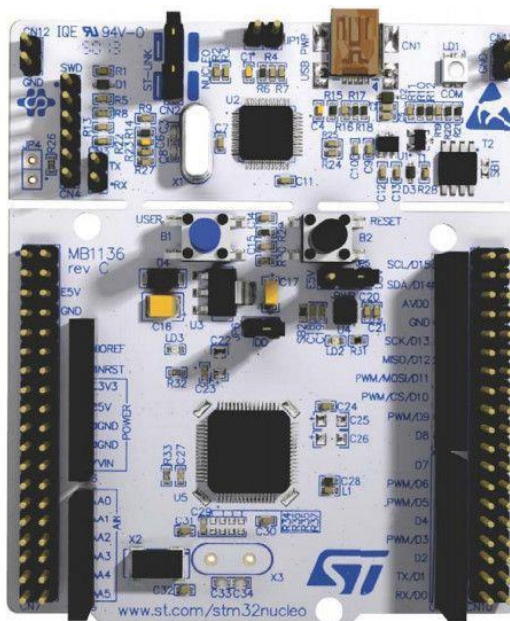


Рис. 6.1. Плата Nucleo-64

### 6.3.2. Датчик температури і вологості DHT11

DHT11 — це базовий, недорогий цифровий датчик температури та вологості. Він використовує ємнісний датчик вологості повітря і термістор для вимірювання температури, на вихід виводить цифровий калібрований сигнал температури і вологості. Він досить простий у використанні, але вимагає ретельного налаштування часу комунікації з мікроконтролером, щоб встановити з'єднання і отримати дані.

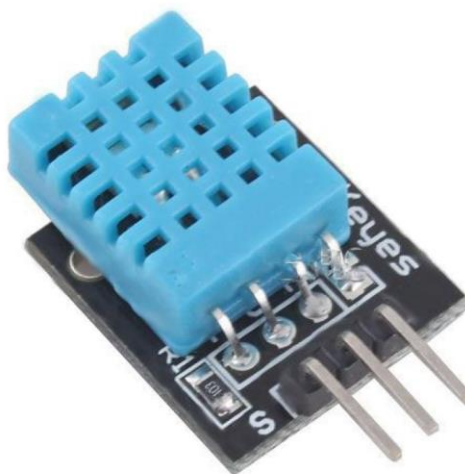


Рис. 6.2. Датчик температури і вологості DHT11

### 6.3.3. Змінний резистор 10кОм (Потенціометр)

Потенціометр являє собою пристрій, здатний розділяти і регулювати напругу за допомогою коректування опору. Діє пристрій як змінний резистор. Два його виведення підключаються до резистивного елемента і з'єднані шляхом постійного опору. Третій же під'єднують до змінним по цій поверхні контакту. Це дозволяє приладу виконувати функцію резистора, роботу якого можна регулювати. Така функція здійснюється за допомогою переміщення ковзаючого контакту з окремим виводом по спеціальному елементу [15]. Положення, в якому вони перебувають один до одного і видаватиме поточне значення напруги.

На сьогоднішній день таке обладнання використовується повсюдно і може виконувати різні функції. Потенціометр може виступати в якості:

- приймача;
- датчика;
- подільника напруги.

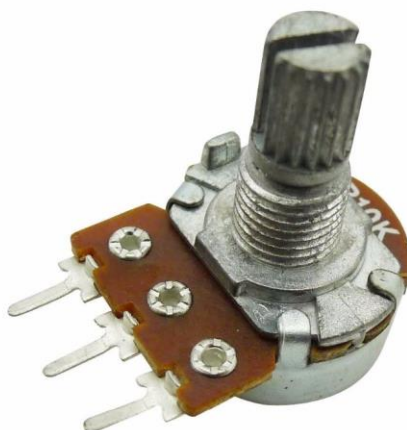


Рис. 6.3. Змінний резистор 10кОм

### 6.3.4. STM32CubeIDE

STM32CubeIDE — це вдосконалена платформа розробки C/C++ з периферійною конфігурацією, генерацією коду, компіляцією коду та

налагодженням мікроконтролерів та мікропроцесорів STM32. Він заснований на основі ECLIPSE/CDT та наборі інструментів GCC для розробки та GDB для налагодження.

#### **6.3.5. STM32 ST-LINK Utility**

STM32 ST-LINK Utility забезпечує просте у використанні та ефективне середовище для читання, запису та перевірки пам'яті пристрою [14].

Інструмент пропонує широкий спектр функцій для програмування внутрішньої пам'яті STM32 (Flash, RAM, OTP та інші), зовнішньої пам'яті, для перевірки запрограмованого вмісту (контрольна сума, перевірка під час та після програмування, порівняння з файлом) та автоматизації програмування STM32.

STM32 Утиліта ST-LINK постачається у вигляді графічного інтерфейсу користувача (GUI) з інтерфейсом командного рядка (CLI) [14].

#### **6.3.6. Real time operating system – FreeRTOS**

FreeRTOS — багатозадачна операційна система реального часу (OSPB) для вбудованих систем. Портована на кілька мікропроцесорних архітектур. FreeRTOS написана на мові Сі з невеликою кількістю асемблерного коду (логіка перемикавання контексту) і її ядро представлено всього 3-ма С файлами [8].

Операційна система реального часу FreeRTOS розрахована на роботу в таких умовах, як низька швидкодія апаратури і малий обсяг оперативної пам'яті, відсутність підтримки на апаратному рівні таких механізмів операційних систем, як блок управління пам'яттю (MMU) і механізми реалізації багатозадачності, такі, як швидке переключення контексту [10].

Планувальник (англ. Scheduler) системи дуже маленький (займає, в залежності від платформи і налаштувань ядра, 4-9 кілобайт) і простий, проте дозволяє задати різні пріоритети процесів, витісняючи і невитісняючи багатозадачність, семафори і черги [10].

#### **6.4. Висновки до розділу**

У цьому розділі детально описано умову завдання, визначено та сформовано набір вимог необхідного функціоналу, обов'язкового для проектування і реалізації. Проведено ґрунтовний аналіз середовища функціонування і апаратних інтерфейсів, що є важливою задачею для вбудованих систем. Також було визначено нефункціональні вимоги та зазначено, які задачі виходять за рамки даної специфікації.

Були проаналізовані програмні інструменти для реалізації поставленої задачі. Середовищем програмування було обрано STM CubeIDE, так як воно найбільш сумісне з мікроконтролером, на якому буде розроблятися система.

Ще був здійснений опис необхідних компонентів для розробки логгера даних: датчик вологості і температури DHT11, потенціометр 10 кОм, плати NUCLEO-64.



## 7. ПРОЕКТУВАННЯ ЛОГГЕРА ДАНИХ СЕНСОРІВ НА ОСНОВІ МІКРОКОНТРОЛЕРА STM32

### 7.1. Налаштування системи мікроконтролера

Логгер даних буде розроблятися на базі мікроконтролера STM32F303RE, що повністю задовільняє визначені вимоги.

Маючи усю необхідну інформацію про будову мікроконтролера та використовуючи документацію виробника, можна провести розподіл периферії та зробити опис взаємодії компонентів пристрою для реалізації логгера даних. Для кращого розуміння процесу конфігурації на рис. 7.1 показано вивід пінів для мікроконтролерів серії STM32F303RE в оболонці LQFP64.

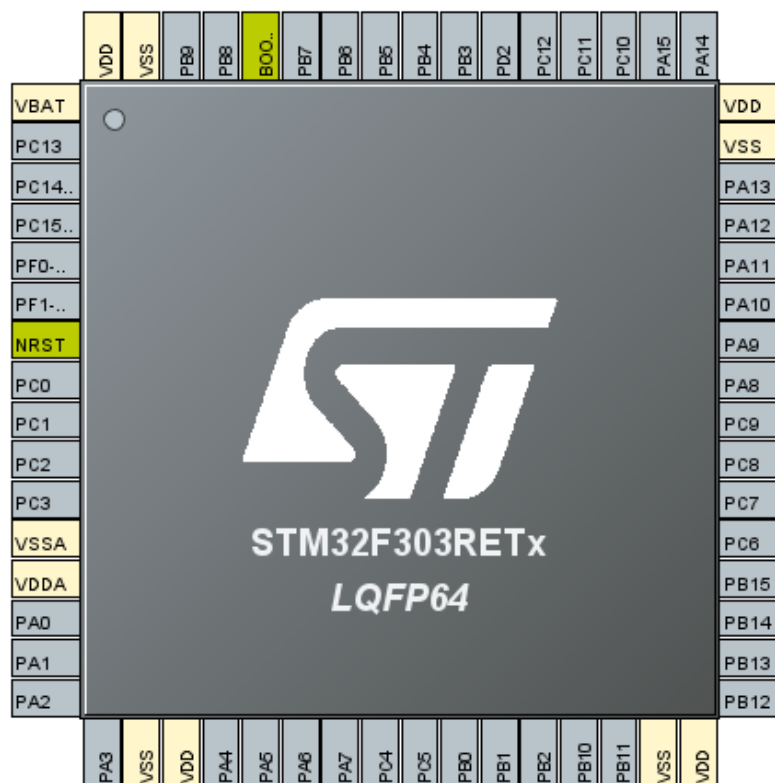


Рис. 7.1. Вивід пінів мікроконтролера STM32F303RE

У даній системі найважливішими з'єднаннями є мікроконтролер – датчик вологості і температури та мікроконтролер – потенціометр (симульований сенсор), адже саме їх комунікація дозволить отримувати дані, без яких ця робота немає сенсу.

### 7.1.1. Організація взаємодії між мікроконтролером і датчиком вологості/температури DHT11

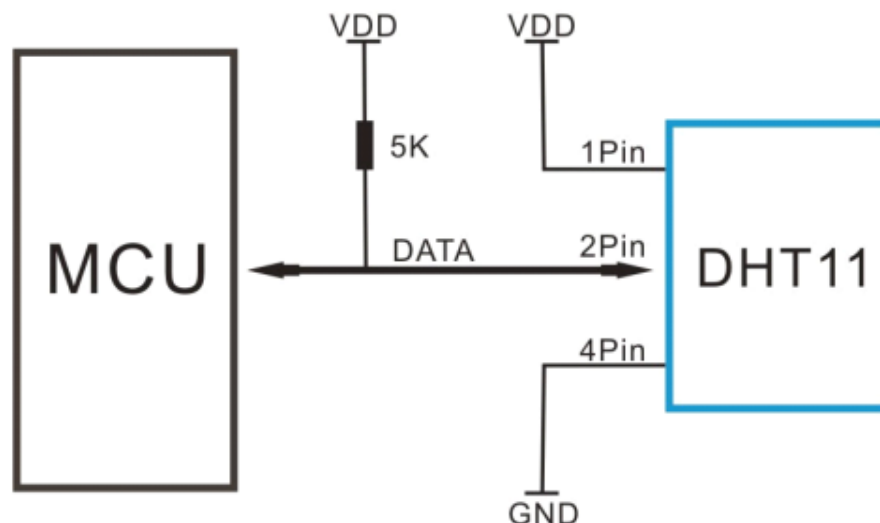


Рис. 7.2. Типове під'єднання сенсора DHT11 і мікропроцесора

Сенсор DHT11 має типову схему підключення до мікроконтролерів чи інших еMBEDED пристроїв. Одна його ножка має бути заземлена, на другу подається живлення і, звичайно, через ще одну відбувається комунікація з іншими пристроями та генерується вихідний сигнал (виміряне значення). Згідно з схемою виконую таке ж підключення у себе земля до землі, 5V до живлення і вихідний сигнал під'єдную до пін PA1.

Формат даних з єдиною шиною використовується для зв'язку та синхронізації між мікроконтролером та DHT11 сенсором. Щоб ініціалізувати датчик, спочатку потрібно поставити пін PA1 в 0 протягом 18 мс і після цього вернути на ньому живлення. Після цього сенсор буде ініціалізований і пошле сигнал у відповідь.

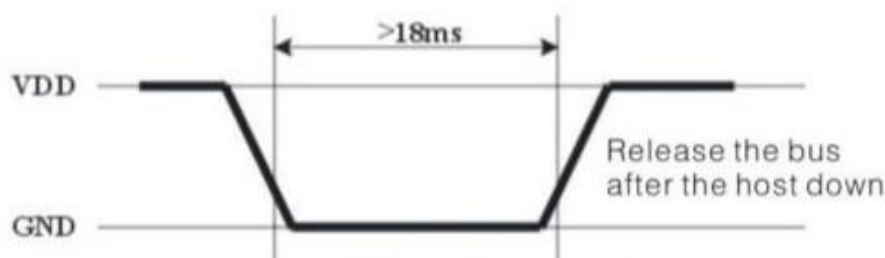


Рис. 7.3. Мікроконтролер починає з'єднання

Щоб визначити присутність мікроконтролера, після отримання сигналу запуску, DHT11 надішле сигнал відповіді. Для цього датчик подасть низький сигнал на 80 мкс, а потім зробить його високим ще на 80 мкс.

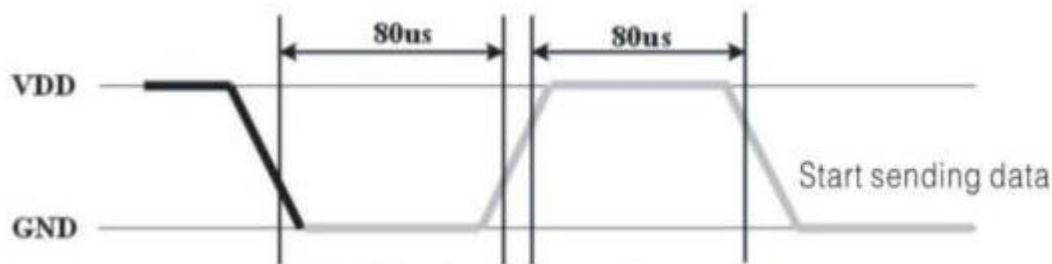


Рис. 7.4. Відповідь сенсора – перевірка з'єднання

Якщо мікроконтролер прийняв відповідь, то тепер DHT11 надішле 40 біт даних. Передача кожного біта починається з низького рівня напруги, який триває 50 мкс, довжина наступного високого сигналу напруги визначає – біт «1» чи «0». Якщо довжина високого рівня напруги становить близько 26-28 мкс, біт дорівнює 0, якщо довжина близько 70 мкс, то біт – 1.

40 біт, що надсилаються DHT11, є такими: 8 біт інтегральних даних (вологість) + 8 бітних десяткових даних (вологість) + 8 біт інтегральних даних (температура) + 8 бітових десяткових даних (температура) + 8 бітна контрольна сума. Якщо передача даних є правильною, контрольна сума повинна бути останнім 8-бітовим значенням – сума 4-х попередніх значень.

### 7.1.2. Організація взаємодії між мікроконтролером і потенціометром

Потенціометр під'єднується до мікроконтролера за тією ж схемою – одна ножка до живлення, інша до заземлення, та що посередині виводить значення.

Оскільки потенціометр є симулятором датчика потрібно, щоб він міг видавати близькі до реальних показники. Тому його буде підключено до 6 каналу аналогово-цифрового перетворювача ADC1, що відповідає піну PA5. Таке рішення було прийнято внаслідок того, що аналогово-цифрові перетворювачі, що під'єднані до інших пінів, можуть давати покази, на які буде впливати інша периферія підключена до того ж піна. 6 канал ADC1 є не зайнятим і не перешкоджає роботі іншої периферії.

### **7.1.3. Підключення діода**

Червоний діод буде сповіщати користувача, що логгер може працювати несправно через умови визначені в специфікації вимог. Його можна під'єднати до піна PA0, так як він не задіяний в інших процесах. Також, щоб діод не згорів потрібно під'єднати до цього простого електричного кола резистор.

### **7.1.4. Налаштування USART**

Піни PA2 і PA3 будуть задіяні для USART інтерфейсу. Використання USART протоколу дозволить виводити дані на COM-порт, а через правильно налаштований термінал дасть можливість відображати дані на екрані ПК. Це спростить відлагодження програми, що є невід'ємною частиною розробки ПЗ.

### **7.1.5. Використання таймерів**

У даній системі передбачається використання 3 таймерів:

- один з 32-бітних таймерів загального призначення буде застосований для синхронізації мікроконтролера з сенсором DHT11, адже їх комунікація базується на зчитуваннях рівня сигналу у точності до мікросекунд;
- RTC (real time clock) буде потрібний для визначення мітки часу, що є необхідною вимогою для роботи логгера;
- ще один 32-бітний таймер загального призначення буде задіяний у планувальнику FreeRTOS.

На рис. 7.5 зображено прототип описаної у цьому пункті системи.

## **7.2. Структура збереження даних в пам'яті**

Реєстратор даних, що розробляється, передбачає вимірювання і збереження 3-х показників: вологість, температура, значення потенціометра. Крім цього, у пам'яті необхідно зберігати мітку часу здійснення виміру. Також постає питання – як розрізняти значення при зчитуванні?

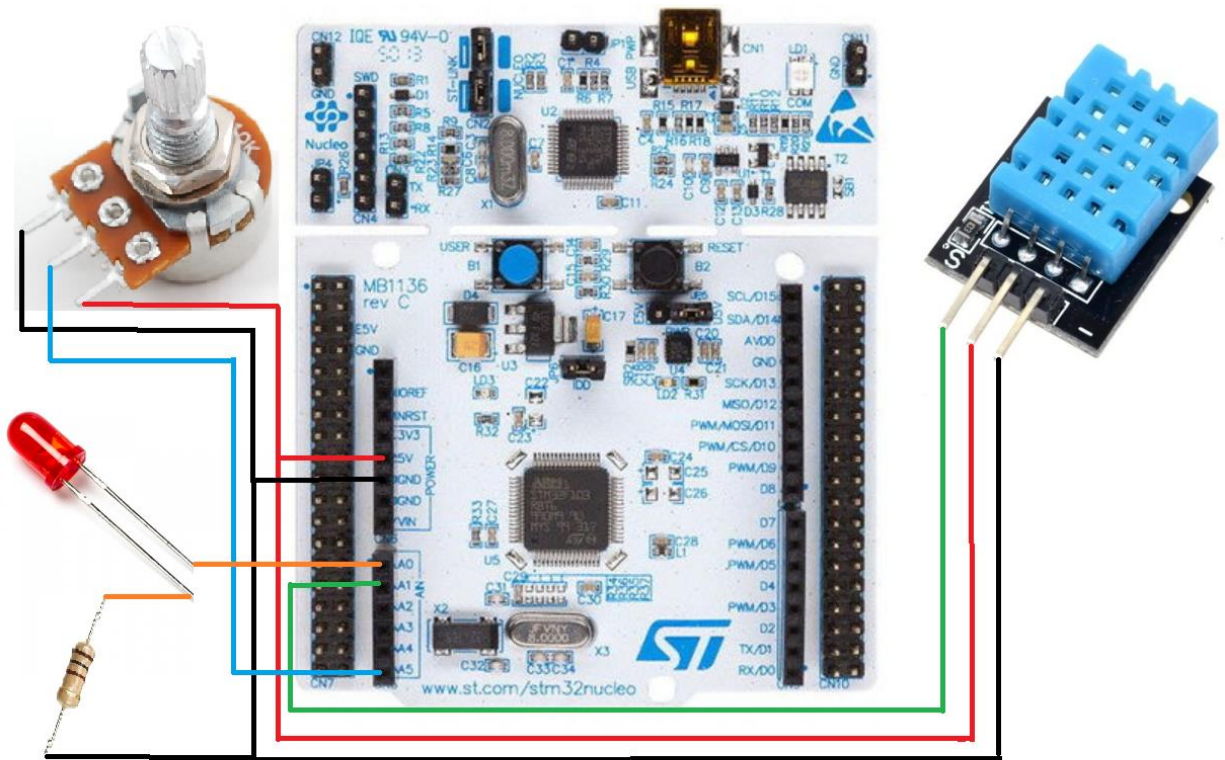


Рис. 7.5. Прототип логгера

Саме тому було розроблено структуру запису даних у пам'ять. Вона однакова для всіх трьох вимірів. Це спростить роботу з даними та забезпечить більшу ефективність. Отже, вигляд цієї структури зображено на рис. 7.6.

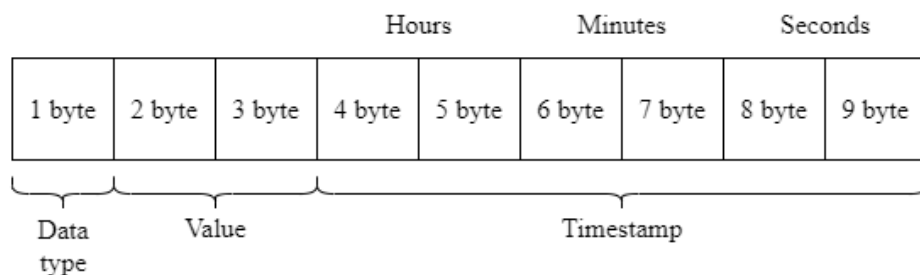


Рис. 7.6. Структура збереження даних у пам'яті

Перший байт буде вказувати на тип даних, що зберігається, тобто температура, вологість чи вимір потенціометра. Далі слідує саме значення, на яке відведено 2 байти, оскільки значення потенціометра лежить в проміжку від 0 до 4095 включно, а температура і вологість складаються з байта десяткових даних і байта інтегрального значення. Наступні 6 байт відведені під мітку часу, що є

стрічкою, яка у свою чергу буде записуватись у формі 2 байти на годину, 2 байти на хвилини, 2 байти на секунди.

### **7.3. Взаємодія з флеш пам'яттю**

Для зберігання інформації, звичайно, потрібно енергонезалежну пам'ять. Оскільки плата Nucleo-64 не містить додаткової пам'яті такого типу, наприклад як вбудований чіп EEPROM як на платі Arduino Uno, дані сенсорів будуть зберігатися в основній (Internal) флеш-пам'яті. Флеш пам'ять володіє властивістю енергонезалежності, тому вона задовільняє вимоги щодо збереження інформації протягом певного періоду часу. Проте, використовуючи такий підхід, обов'язково треба враховувати особливості даного виду пам'яті та її застосування у системі.

#### **7.3.1. Виділення блоку для зберігання даних**

Флеш пам'ять мікроконтролера – це місце, куди записується код при програмуванні приладу, власне він там і зберігається. Саме з цих адрес процесор бере для виконання програмні інструкції. Тому при виборі флеш пам'яті для записування іншого виду інформації, такої як дані сенсорів у цьому випадку, для початку її потрібно налаштувати.

Якщо переглянути додаток В, де зображено структуру флеш пам'яті STM32F303RE, можна замітити, що вона поділена на умовні секції. Адреси з 0x40022000 по 0x40022023 зарезервовані для регістрів, що дозволяють працювати з флеш пам'яттю. Також є невеликий інформаційний блок на 8 Кбайт + 16 байт, що також виділений під інші цілі. Основний блок з 0x08000000 по 0x0807FFFF, позначений як основна пам'ять, складається з 256 сторінок, кожна з яких займає 2 Кбайти. У цих сторінках зберігається прошивка мікроконтролера, зазвичай вона знаходиться у менших адресах, тобто починаючи з 0x08000000. STM CubeIDE при компіляції проекту генерує скрипт лінкувальника, де визначені сектори для розміщення різних типів даних у флеші. Фрагмент такого файлу зображений на рис. 7.7.

```

/* Memories definition */
MEMORY
{
    CCMRAM    (xrw)   : ORIGIN = 0x10000000, LENGTH = 16K
    RAM       (xrw)   : ORIGIN = 0x20000000, LENGTH = 64K
    FLASH    (rx)    : ORIGIN = 0x80000000, LENGTH = 512K - 64K
}

/* Sections */
SECTIONS
{
    /* The startup code into "FLASH" Rom type memory */
    .isr_vector :
    {
        . = ALIGN(4);
        KEEP(*(.isr_vector)) /* Startup code */
        . = ALIGN(4);
    } >FLASH

    /* The program code and other data into "FLASH" Rom type memory */
    .text :
    {
        . = ALIGN(4);
        *(.text)           /* .text sections (code) */
        *(.text*)          /* .text* sections (code) */
        *(.glue_7)          /* glue arm to thumb code */
        *(.glue_7t)         /* glue thumb to arm code */
        *(.eh_frame)

        KEEP (*(.init))
        KEEP (*(.fini))

        . = ALIGN(4);
        _etext = .;         /* define a global symbols at end of code */
    } >FLASH
}

```

Рис. 7.7. Скрипт лінкера згенерований в STM CubeIDE

Тому, щоб зарезервувати певний діапазон адрес, у цьому файлі можна створити власну секцію призначену для збереження показників сенсорів. Таким чином обрана ділянка пам'яті буде зарезервована і це гарантуватиме, що код програми чи інша інформація не буде записана в обрану секцію. Отже, коли логгер буде записувати отримані показники, він не пошкодить іншу інформацію, необхідну для правильної роботи системи.

### 7.3.2. Робота з сторінковою організацією флеш пам'яті

Також потрібно зважати на організацію флеш пам'яті – дані зберігаються посторінково. Через це є обмеження на запис інформації. Правило, яке прописане в документації STM32 по роботі з флеш пам'яттю, каже – щоб перезаписати дані, їх спочатку потрібно стерти. Тобто, коли байт даних одного разу був записаний за певною адресою в флеші, щоб його змінити, треба цей байт скинути, а потім записати потрібне значення. Але на цьому не все. Стирання даних також є операцією, яка описана в документації. Не можна скинути кількість байт, яку хочеш, тому що дані видаляються лише посторінково. Тобто найменша кількість

інформації, яку можна видалити, дорівнює розміру сторінки, у випадку STM32F303RE – це 2 Кбайт, або 2048 байт. Цю особливість також треба враховувати при оновленні інформації, тому що для здійснення цієї операції спочатку потрібно видалити сторінку, а після цього перезаписувати дані.

Запис у флеш пам'ять також не є звичною справою. Перед тим як спробувати доступитись до будь-якої комірки з цього діапазону для її зміни значення, потрібно розблокувати флеш за допомогою присвоєння відповідних значень “ключів” у регістр пам'яті, а саме у FLASH\_KEYR. Якщо цього не зробити і на пряму спробувати доступитись до пам'яті, система аварійно закінчить своє виконання, згенерувавши “HardFault”. Ще одним нюансом роботи з флеш пам'яттю є неможливість запису за раз більше одного програмного слова, що на STM32F303RE рівне 4 байт.

Зважаючи на ці фактори був спроектований наступний алгоритм. Кожного разу, коли потрібно записати дані в пам'ять, перевіряємо чи розмір отриманої інформації може бути збережений цілком. Якщо так, то визначаємо зміщення на сторінці, і записуємо інформацію за обрахованою адресою. У протилежному випадку перевіряємо чи не дійшов алгоритм до кінця відведеної для логів пам'яті. У разі позитивної перевірки потрібно обнулити лічильник сторінок та зміщення відносно початку сторінки, присвоїти поточній сторінці адресу першої, скинути усі дані записані на неї і після цього виконати запис нової інформації. Якщо умова негативна, це означає, що алгоритм записав цілу сторінку. Тому лічильник сторінок збільшується на 1, знаходиться адреса наступної сторінки, виконується її очистка і після цього починається збереження нової інформації. Для візуального відображення цього алгоритму побудована блок-схема, яка знаходиться в додатку Г.

Слід зазначити, що даний алгоритм є одним з найпростіших, але чудово підходить для розв'язання такої задачі. Він нагадує добре відомий метод Round Robin, тільки не для потоків, а для пам'яті пристрою.



#### **7.4. Застосування FreeRTOS**

Система, що розробляється, передбачає використання декількох датчиків і зчитування їх показів паралельно за допомогою реалізації багатозадачності, що надає FreeRTOS. Для виконання задачі достатньо створити 2 потоки – один буде зчитувати та логувати значення потенціометра, інший зчитуватиме дані вологості і температури, так як це робить той самий сенсор. Розподіленим ресурсом у даному випадку буде пам'ять. Щоб синхронізувати 2 потоки достатньо буде використати мютекс, реалізований в FreeRTOS.

На діаграмі послідовності, що в додатку Д, спроектовано, як буде працювати система. Увесь сценарій подій починається з користувача, він має увімкнути реєстратор даних. Спочатку проходить ініціалізація периферії: таймерів, пінів, аналогово-цифрових перетворювачів, комунікаційних протоколів. Далі йде ініціалізація ядра FreeRTOS, після чого логгера даних. Так, як на мікроконтролері STM32F303RE неможливо реалізувати справжню паралельність виконання, розроблена діаграма досить чітко показує, як буде діяти система. Коли потік, що працює з датчиком DHT11, захоплює процесорний час, буде заложене значення вологості-температури. Коли процесорний час переходить до потоку, який зчитує дані потенціометра, буде запис у пам'ять інформації симульованого сенсора.

#### **7.5. Висновки до розділу**

У даному розділі було зроблено проектування системи в цілому. Після аналізу мікроконтролера, на якому здійснюється розробка логгера даних, виконано і розписано стратегію налаштування і конфігурації вбудованої системи. Визначено, які і скільки пінів потрібно для розробки, здійснено опис кожного задіяного і його призначення. Описано використання периферії мікроконтролера. Після цього було розроблено прототип пристрою. Окремо і детально розписано процес роботи з пам'яттю для зберігання даних. До цього розроблено блок-схему, яка описує алгоритм запису в флеш пам'ять. Також проаналізовано застосування FreeRTOS і побудовано діаграму послідовності, яка описує хід роботи системи.

## **8. ВИСНОВКИ ПРО ОТРИМАНІ ПІД ЧАС ПРАКТИКИ РЕЗУЛЬТАТИ**

Проходження практики у ФОП Дячок Р.В. відбулось успішно. Завдання, отримане для виконання, допомогло розібратись у нових технологіях та практиках розробки. Під час ознайомлення з теоретичним матеріалом, необхідним для правильного вирішення поставленої задачі, краще розібрався у напрямку технологій операційних систем реального часу. Саме ж завдання потребувало ґрунтовного дослідження FreeRTOS, що є одною з операційних систем реального часу для вбудованих систем. Тому було розібрано такі аспекти як застосування багатопоточності, примітивів синхронізації, можливості управління пам'яттю і взаємодія між потоками, використовуючи засоби, що реалізовані в FreeRTOS.

Після вивчення потрібного теоретичного матеріалу виконав запуск FreeRTOS на мікроконтролері та створив простий потік для перевірки, що все виконано правильно. Потім було досліджено алгоритми реалізовані у FreeRTOS для управління пам'яттю та використано один з них на самому мікроконтролері, щоб побачити, як все працює на практиці. Далі з настановами керівника практики на підприємстві було розроблено власний алгоритм для динамічного виділення пам'яті у FreeRTOS. Ця розробка дозволяє використовувати декілька вільних ділянок пам'яті розміщених за різними адресами, що допомагає збільшити розмір кучі. Керівник практики підтвердив успішне виконання завдання та схвалив роботу.

По дипломному завданні було досягнуто наступні результати. Під час оглядової частини пересвідчився в актуальності задачі логування даних та необхідності її реалізації у різних типах застосунків. Визначив, що таке логгер даних, як пристрій, які проблеми він вирішує, та, в яких галузях застосовується. Порівняння аналогів дозволило зрозуміти, що існує багато логгерів даних, що розроблені для вирішення якоїсь одної конкретної задачі, та ні один пристрій не є ідеальний для різних типів завдань. Тому розробка логгера для роботи з даними

потрібного типу і у визначеному навколишньому середовищі є реальною і актуальною задачею, яка буде пророблена у ході виконання диплому.

Під час розробки дипломного завдання було складено специфікацію вимог, що визначає основні функціональні вимоги, потрібні для роботи логгера: отримання показників температури і вологості, зчитування значення потенціометра, зберігання даних у пам'яті та сповіщення при можливих проблемах. Також у цій частині було підібрано необхідні засоби розробки: середовище розробки STM CubeIDE, плата Nucleo-64 з мікроконтролером STM32F303RE, датчик DHT11, потенціометр (для симуляції датчика), FreeRTOS.

Проектування логгера даних допомогло краще розібратись у будові мікроконтролера STM32F303RE, на якому має працювати пристрій. Виконано і розписано стратегію налаштування і конфігурації вбудованої системи. Визначено, що наступні піни будуть задіяні для зчитування даних сенсорів: PA1, PA5. Пін PA0 буде відведено для підключення діода. Описано використання периферії мікроконтролера: таймери для синхронізації мікроконтролера з сенсором DHT11, генерації мітки часу, планувальника FreeRTOS, аналогово-цифровий перетворювач ADC1 допоможе у симуляції датчика. На основі цього розроблено прототип пристрою. Окремо і детально розписано процес роботи з пам'яттю для зберігання даних. Також проаналізовано застосування FreeRTOS і побудовано діаграму послідовності, яка описує хід роботи системи. Буде створено 2 потоки для зчитування даних сенсорів, якими буде керувати планувальник FreeRTOS.



<https://uk.wikipedia.org/wiki/Потенціометр>

14. NUCLEO-F303RE [Электронний ресурс]. — Режим доступу: <https://www.st.com/en/evaluation-tools/nucleo-f303re.html>
15. ПОТЕНЦИОМЕТР, КОНСТРУКЦИЯ И ВИДЫ ПРИБОРА [Электронний ресурс]. — Режим доступу: <https://eltaltd.com.ua/blog/potentsiometr-konstruktsiya-i-vidy-pribora>
16. STSW-LINK004 [Электронний ресурс]. — Режим доступу: <https://www.st.com/en/development-tools/stsw-link004.html>

## ДОДАТКИ

## Додаток А. Порівняння логгерів

Таблиця А.1.

## Порівняння логгерів

Характеристики	<i>OM-EL-USB</i>	<i>HOBO MX2201</i>	<i>MSR145</i>
Діапазон виміру температури	-35..+80 °C	-20..+70 °C	-20..+65 °C
Точність вимірів	±1.0 °C	±0.5 °C	±0.5 °C
Розмір	94x20.5x27 mm	3.3x56x18 mm	27x16x53 mm
Вага	43 г	12.75 г	20 г
Обсяг збережених даних	16000 записів	96000 записів	2000000 записів
Тип пам'яті	Вбудована	Вбудована	Вбудована + microSD
Спосіб передачі даних	USB	BLE	USB, microSD
Способи налаштування	Спеціальне ПЗ під Windows	Мобільний додаток під iOS чи Android	Спеціальне ПЗ під Windows

Продовження таблиці А.1.

Характеристики	<i>OM-EL-USB</i>	<i>HOBO MX2201</i>	<i>MSR145</i>
Інтерфейс користувача	2 діоди: увімкненість і статус	2 діоди: увімкненість і сповіщення	4 діоди: : увімкненість, сповіщення, зміна батареї, статус запису

## Додаток Б. Діаграма прецедентів

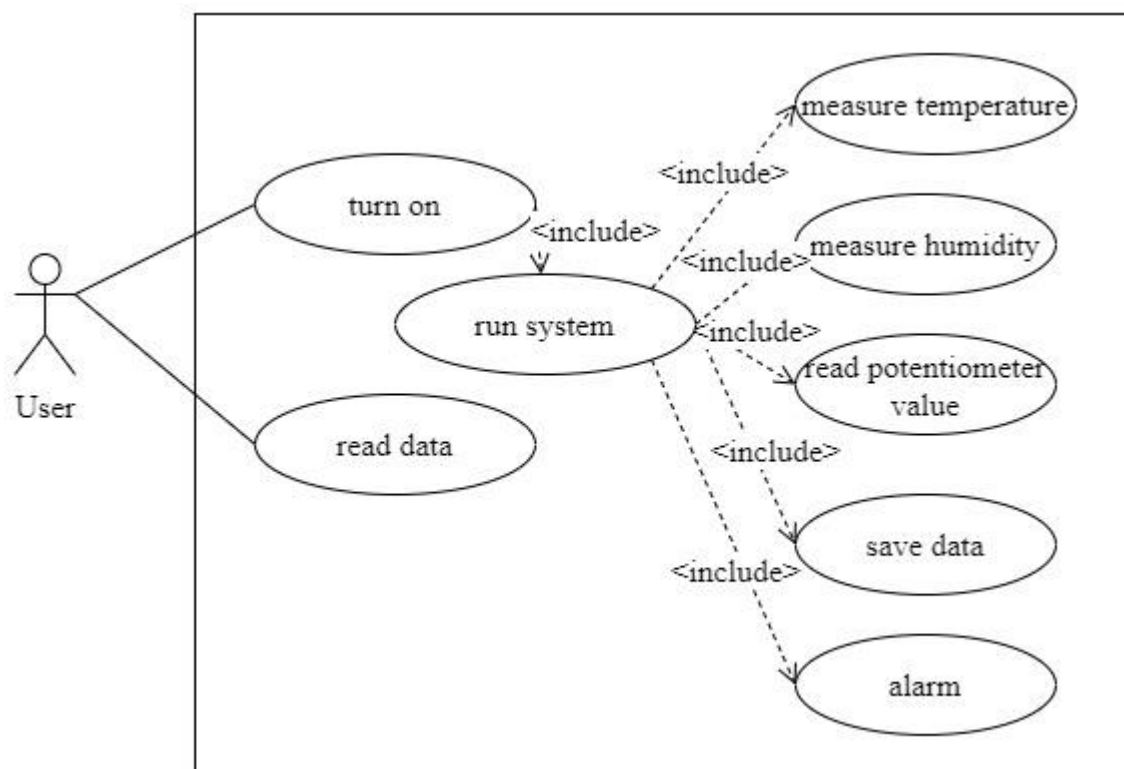


Рис. Б.1. Діаграма прецедентів



## Додаток В. Флеш пам'ять STM32F303RE

Таблиця В.1.

## Флеш пам'ять STM32F303RE

Flash area	Flash memory addresses	Size (bytes)	Name
Main memory	0x0800 0000 - 0x0800 07FF	2 K	Page 0
	0x0800 0800 - 0x0800 0FFF	2 K	Page 1
	0x0800 1000 - 0x0800 17FF	2 K	Page 2
	0x0800 1800 - 0x0800 1FFF	2 K	Page 3
	.	.	.
	.	.	.
	.	.	.
	.	.	.
Information block	0x0807 F800 - 0x0807 FFFF	2 K	Page 255
	0x1FFF D800 - 0x1FFF F7FF	8 K	System memory
Flash memory interface registers	0x1FFF F800 - 0x1FFF F80F	16	Option bytes
	0x4002 2000 - 0x4002 2003	4	FLASH_ACR
	0x4002 2004 - 0x4002 2007	4	FLASH_KEYR
	0x4002 2008 - 0x4002 200B	4	FLASH_OPTKEYR
	0x4002 200C - 0x4002 200F	4	FLASH_SR
	0x4002 2010 - 0x4002 2013	4	FLASH_CR
	0x4002 2014 - 0x4002 2017	4	FLASH_AR
	0x4002 2018 - 0x4002 201B	4	Reserved
	0x4002 201C - 0x4002 201F	4	FLASH_OBR
	0x4002 2020 - 0x4002 2023	4	FLASH_WRPR

## Додаток Г. Блок-схема алгоритму запису у сторінку флеш пам'яті

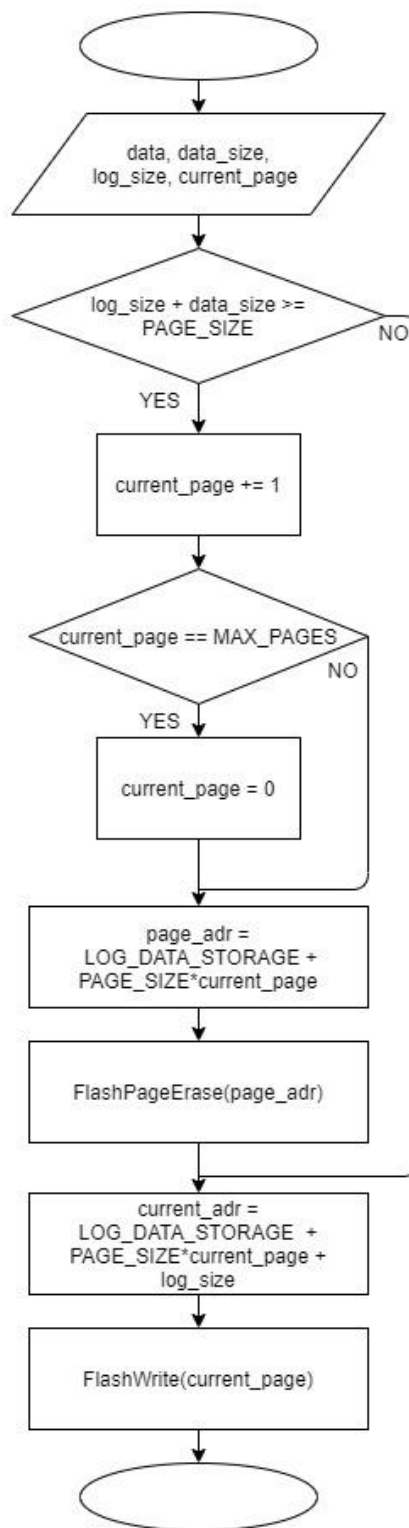


Рис. Г.1. Блок-схема алгоритму запису у сторінку флеш пам'яті

## Додаток Д. Діаграма послідовності

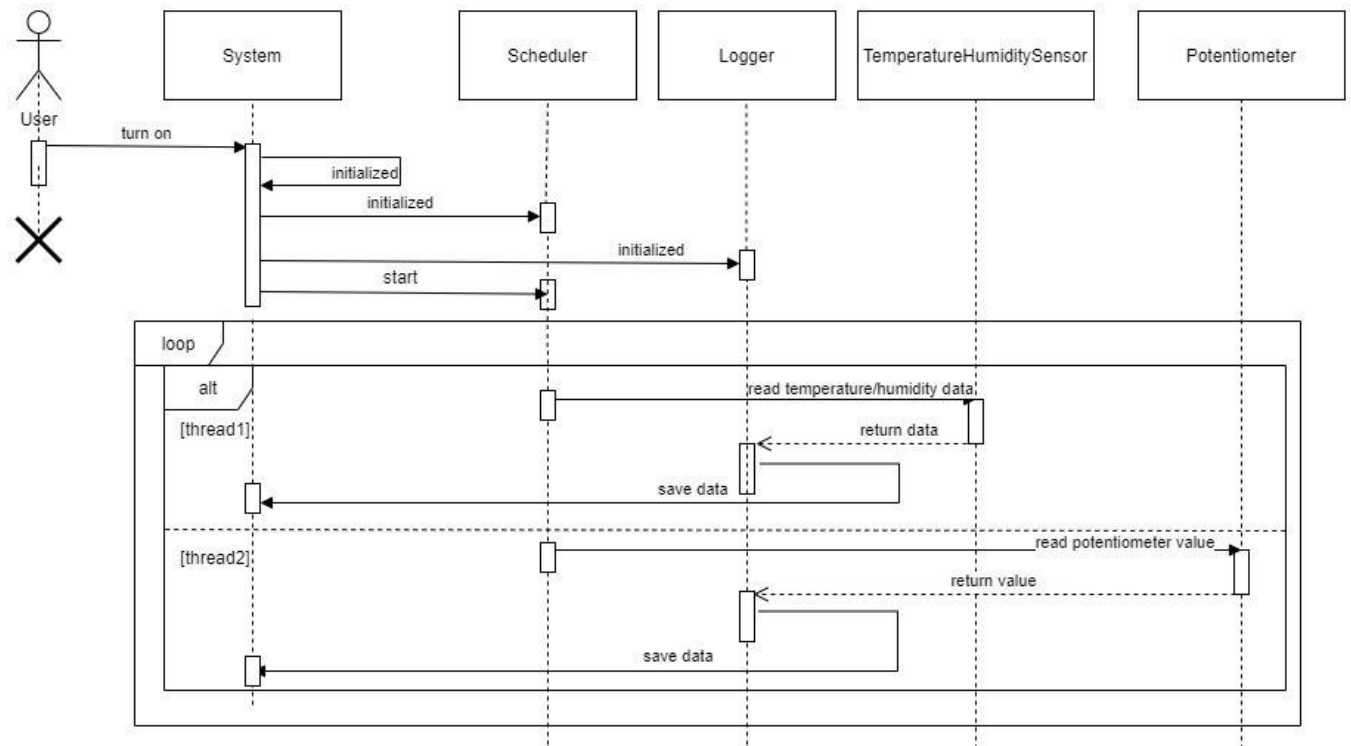


Рис. Д.1. Діаграма послідовності