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

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

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня
бакалавр
(назва освітнього ступеня)

на тему: *Комп'ютеризована система контролю параметрів мікроклімату приміщення*

Виконав(ла): студент(ка) *IV* курсу, групи *CIc-44*
спеціальності *123 Комп'ютерна інженерія*
(шифр і назва спеціальності)

		Ольховецька Х.А.
	(підпис)	(прізвище та ініціали)
Керівник		*Лещишин Ю.З.*
	(підпис)	(прізвище та ініціали)
Нормоконтроль		*Тиш Є.В.*
	(підпис)	(прізвище та ініціали)
Завідувач кафедри		*Осухівська Г.М.*
	(підпис)	(прізвище та ініціали)
Рецензент		
	(підпис)	(прізвище та ініціали)

Тернопіль 2021

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

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

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

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ *Осухівська Г.М.*
(підпис) (прізвище та ініціали)
« » 20___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

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

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

студенту *Ольховецькій Христині Андріївні*
(прізвище, ім'я, по батькові)

1. Тема роботи *Комп'ютеризована система контролю параметрів мікроклімату приміщення*

Керівник роботи *Лещишин Юрій Зіновійович, к.т.н., доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*10*» *лютого* 20*21* року № *4/7-97*

2. Термін подання студентом завершеної роботи *26.06.2021р.*

3. Вихідні дані до роботи *Технічне завдання*

4. Зміст роботи (перелік питань, які потрібно розробити)
Аналіз ТЗ, Огляд вимог, Аналіз можливих рішень, Розробка узагальненої структури, Вибір апаратної складової, Вибір програмних компонентів, Створення набору правил на базі двох протоколів, Реалізація проектних рішень, Розробка алгоритмів роботи, Збірка та заливка, Тестування.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
КС КРБ 123.176.00.01 Е3
КС КРБ 123.176.00.02
КС КРБ 123.176.00.03
КС КРБ 123.176.00.04 Е1

6. Консультанти розділів роботи

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

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	*Ознайомлення з завданням до кваліфікаційної роботи*	*10.02 – 11.02*	*Виконано*
2.	*Ознайомлення з вимогами до мікроклімату приміщення*	*11.02 – 15.02*	*Виконано*
3.	*ДСН 3.3.6.039-99: N 39 від 01.12.1999*	*15.02 – 17.02*	*Виконано*
4.	*Опрацювання літератури щодо STM32*	*17.02 – 17.03*	*Виконано*
5.	*Підбір апаратних складових*	*17.03 – 25.03*	*Виконано*
6.	*Підбір бази для написання програмної складової*	*25.03 – 30.03*	*Виконано*
7.	*Ознайомлення з OS MBED*	*30.03 – 15.04*	*Виконано*
8.	*Робота над розділом «Аналіз технічного завдання»*	*15.04 – 18.04*	*Виконано*
9.	*Робота над розділом «Проектна частина»*	*18.04 – 05.05*	*Виконано*
10.	*Виконання макетування*	*05.05 – 10.05*	*Виконано*
11.	*Робота над розділом «Практична частина»*	*10.05 – 15.05*	*Виконано*
12.	*Виконання завдань до розділу «Безпека*	*15.05 – 17.05*	*Виконано*
	життєдіяльності, основи охорони праці»		
13.	*Оформлення пояснювальної записки*	*17.05 – 05.06*	*Виконано*
14.	*Нормоконтроль*	*05.06 – 09.06*	*Виконано*
15.	*Перевірка на плагіат*	*10.06 – 16.06*	*Виконано*
16.	*Попередній захист КБ*	*15.06 -18.06*	*Виконано*
16.	*Захист КБ*	*26.06.*	

Студент _____ *Ольховецька Христина Андріївна*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Лещишин Юрій Зіновійович*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Комп’ютеризована система контролю параметрів мікроклімату приміщень // Кваліфікаційна робота бакалавра // Ольховецька Христина Андріївна // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп’ютерно-інформаційних систем і програмної інженерії, кафедра комп’ютерних систем та мереж, група СІс-44 // Тернопіль, 2021 // с. – 64, рис. – 56, табл. – 3, кресл. – , додат. – 2, бібліогр. – 16.

Ключові слова: STM32F767, DMA, UART, TCP SERVER, МІКРОКЛІМАТ, EEPROM, BMP280, АЛГОРИТМ, MBED, VISUAL STUDIO CODE.

В кваліфікаційній роботі бакалавра розроблено комп’ютеризовану систему контролю параметрів мікроклімату приміщень та її програмне забезпечення. На базі мікроконтролера: STM32F767 реалізовано опитування давачів та комунікацію з системою з використанням прямого доступу до пам’яті, що суттєво підвищує її швидкість роботи. Режими роботи системи відображаються на екрані.

Кваліфікаційна робота складається з чотирьох розділів.

У першому розділі проводиться аналіз технічного завдання, а саме вимог до комп’ютерної системи та аналіз можливих рішень.

В другому розділі описується процес проєктування та реалізації проєкту, як вбудованої системи.

В третьому розділі приводиться розробка програмного забезпечення для функціонування пристрою. Розглядаються бібліотеки та реалізація функцій побудованих на основі них, їх алгоритми. Також проводиться тестування на двох платах в реальних умовах експлуатації.

Четвертий розділ описує безпеку життєдіяльності, основи охорони праці.

ABSTRACT

Computer-aided control system of room microclimate parameters // Diploma thesis //Olkhovetska Khrytsyna // Ternopil Ivan Puluj National Technical University, Fac-ulty of Computer Information Systems and Software Engineering, group CIc-44 //Ternopil, 2021 // p. – 64, fig. - 56, tabl.– 3, code snip. – 2, append.1 – , bibliogr. – 16.

Keywords: STM32F767, DMA, UART, TCP SERVER, МІКРОКЛІМАТ, EEPROM, BMP280, АЛГОРИТМ, MBED, VISUAL STUDIO CODE.

In the qualification work of the bachelor the computerized system of control of parameters of a microclimate of the room and its software is developed. Based on the microcontroller: STM32F767, data description and communication with the system using direct access to memory is implemented, which significantly increases its speed. The operating modes of the system are shown on the screen.

Qualification work is created with four sections.

In the first section, we analyze the technical problem, namely, require a computer system and analysis of possible solutions.

Another section describes the process of designing and implementing a project as an embedded system.

The third section describes the development of software for the operation of the device. Libraries and implementation of functions built on their basis, their algorithms are considered. You can also test on two boards in real operating conditions.

The fourth section describes the safety of life, the basis of labor protection.

ЗМІСТ

СПИСОК СКОРОЧЕНЬ

eEPROM – Electrically Erasable Programmable Read-Only Memory

USART – Universal Asynchronous Receiver-Transmitter

NVRAM – Non Volatile Random Access Memory

DMA – Direct Memory Access

SPI – Serial Peripheral Interface

TCP – Transmission Control Protocol

DSP – Digital signal processing

SWD – Serial Wire Debug

ПЗП – Постійний запам’ятовувальний пристій

ВСТУП

Вбудовані системи набули широкого розповсюдження та популярності у використанні, оскільки – це спеціалізована система управління, яка розроблена таким чином, щоб працювати у складі пристрою, яким вона керує, що є перевагою відносно інших. Одним з прикладів так званих вбудованих систем є система моніторингу та контролю мікроклімату приміщення.

Важливість теми моніторингу та автоматизованого контролю мікрокліматом приміщення можна підтвердити численними науковими дослідженнями на тему погіршення самопочуття людей через невідповідність параметрів мікроклімату нормативним, у випадку виробничих приміщень, або ж певним індивідуальним межам у випадку квартири/будинку.

Основною характеристикою нормального функціонування людського організму є процес підтримання відносної динамічної сталості певних параметрів за різноманітних метрологічних умов, яка в більшості випадків забезпечується одним з основних механізмів, а саме терморегуляцією. Терморегуляція – це певне співвідношення теплоутворення (хімічної терморегуляції) і тепловіддачі (фізичної терморегуляції).

Усі характеристики мікроклімату приміщення мають свій вплив на процес терморегуляції. Для прикладу особливо шкідливою є вологість повітря, яка перевищує 70—75 % за температури 30 °С і більше, може статися перегрів тіла, а робота в умовах температури, яка перевищує допустимі межі, викликає прискорення серцебиття та зниження артеріального тиску, послаблює організм, викликає млявість. Проте вплив низької температури також може мати серйозні наслідки – переохолодження організму.

Згідно з результатами досліджень людина є працездатною і нормально себе почуває, якщо температура навколишнього повітря не виходить за межі 18 -20С, відносна вологість — 40-60 %.

Метою даної кваліфікаційної роботи є створення вбудованого програмно-апаратного продукту для збору та автоматизованого управління параметрами мікроклімату приміщення, робота якого буде керуватись за допомогою інтерфейсів UART або ETHERNET на основі розробленої інструкції взаємодії. Система повинна відстежувати та управляти показниками температури, відносної вологості повітря та тиску, за певних умов передбачається автоматизоване охолодження або підігрів приміщення.

Досягнення мети передбачає необхідність розв'язувати такі задачі:

— дослідити основні вимоги до роботи систем керування мікрокліматом приміщення;

— дослідити можливі варіанти апаратної реалізації даної вбудованої системи;

— дослідити можливі варіанти програмної реалізації даної вбудованої системи;

— розробити інструкцію взаємодії користувача та системи для інтерфейсів UART та ETHERNET (використовуючи TCP сервер);

— розробити програмне забезпечення роботи вбудованої системи.

Об’єкт дослідження: процес керування мікрокліматом приміщення з допомогою вбудованої системи.

Предмет дослідження: керування мікрокліматом приміщення з допомогою вбудованої системи.

Пристрій розроблявся на базі NUCLEO F767ZI та в середовищі Visual Studio Code з використанням ОС MBED.

РОЗДІЛ 1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

1.1 Аналіз оптимальних показників мікроклімату

Норми мікроклімату визначенні ДСН 3.3.6.042-99 Міністерства охорони здоров'я України «Санітарні норми мікроклімату виробничих приміщень» в поточній редакції прийнятій 01.12.1999 р. ДСН визначає мікроклімат приміщень як «стан внутрішнього середовища приміщення, що впливає на людину» [1]. В загальному випадку під поняттям внутрішнього середовища розуміють повітря всередині приміщення. Мікроклімат приміщень характеризується комбінацією показників відносної вологості, температури й швидкості руху повітря.

Для забезпечення високого рівня працездатності людини та стану нормального самопочуття показники мікроклімату в приміщенні мають бути оптимальними, тобто такими, щоб не активувати механізм терморегуляції організму, через те що ці показники мають прямий вплив на людину. Якщо параметри відповідають нормі («оптимальні», як вказано в ДСН), то людина відчуває себе комфортно, і організм не витрачає ресурс на пристосування до зовнішніх умов [1].

Мікроклімат житлових та громадських будівель складається з багатьох параметрів, але пріоритетом буде:

— температура повітря;

— вологість повітря;

— тиск.

Оптимальні межі величин показників температури й відносної вологості повітря представлено в табл. 1.

Таблиця 1 – Оптимальні величини температури й відносної вологості повітря

Період року	Категорія робіт	Температура повітря	Відносна вологість
Холодний період року	Легка Іа	22 - 24	60 - 40
	Легка Іб	21 - 23	60 – 40
	Середньої важкості ІІа	19 - 21	60 – 40
	Середньої важкості ІІб	17 – 19	60 – 40
	Важка ІІІ	16 - 18	60 – 40
Теплий період року	Легка Іа	23 - 25	60 - 40
	Легка Іб	22 - 24	60 – 40
	Середньої важкості ІІа	21 - 23	60 – 40
	Середньої важкості ІІб	20 – 22	60 – 40
	Важка ІІІ	18 - 20	60 – 40

При розробці системи в режимі роботи за замовчуванням буде враховано межі оптимальних показників температури й відносної вологості повітря визначених ДСН 3.3.6.042-99 Міністерства охорони здоров'я України «Санітарні норми мікроклімату виробничих приміщень».

1.2 Аналіз вимог до комп'ютерної системи моніторингу та контролю параметрами мікроклімату

Комп’ютерна система контролю параметрів мікроклімату повинна відповідати таким вимогам:

— робота в реальному часі;

— висока частота опитування давача для забезпечення правдивість показників параметрів, які вимірюються;

— давач повинен бути енергоефективним;

— давач повинен мати в наявності вбудовані фільтри, які зменшують вплив зовнішніх шумів на показники параметрів;

— давач повинен підтримувати роботу з інтерфейсами SPI або I2C;

— забезпечення цілісності та збереження даних виміряних показників давача, шляхом використання зовнішнього постійного запам’ятовувача;

— забезпечення доступу до історії всіх даних за період використання системи;

— можливість індивідуального налаштування меж критичних та оптимальних значень показників параметрів мікроклімату приміщення;

— забезпечення альтернативного шляху підключення для керування системою, незалежного від основного;

— використання в роботі прямий Direct Memory Access (DMA) для забезпечення доступу до оперативної пам’яті й зменшення навантаження на мікропроцесор;

— КС повинна бути автономною і повинна працювати незалежно від наявності підключеного користувача;

— КС повинна відповідати на запити відповідно до розробленого внутрішнього набору правил роботи та команд.

1.3 Аналіз можливих рішень поставленого завдання

1.3.1 Огляд плати сімейства STM32F4xx

STM32 F4 – перша серія, яка базується на ядрі ARM Cortex-M4F і має підтримку Digital signal processing (DSP) і чисел з рухомою комою. Розташування портів введення/виводу сумісно з серією F7, проте сам чіп відрізняється тактовою частотою (від 84 до 180 МГц), має 64 КБ вбудованої пам'яті, підтримку протоколу I²S, вбудований годинник реального часу. Пам'ять до 192 КБ SRAM, 64 КБ CCM, 4 КБ NVRAM, 80 байтів NVRAM, що стирається при втручанні. Flash-пам'ять поділяється на блоки 512/1024/2048 для безпосереднього використання, 30 КБ для завантаження, 512 байтів одноразової пам'яті (OTP), 16

байтів для конфігурації. У кожен чіп запрограмований 96-розрядний унікальний номер. периферія USB 2.0 OTG дві CAN-шини один інтерфейс SPI та два SPI / I²S, 3 I²S, 4 USART, 2 UART, SDIO, дванадцять 16-бітних, два 32-бітних і два сторожових таймери, 16/24-канальний АЦП , два ЦАП, від 51 до 140 пінів GPIO, 16 DMA, годинник реального часу, а також апаратний генератор випадкових чисел, У моделях STM32F4x7 присутній Ethernet, MAC і інтерфейс для підключення камери. У моделях STM32F41x / 43x є крипто процесор, що підтримує методи DES, TDES і AES, а також SHA-1 і MD5. У моделях STM32F4x9 присутній LCD-TFT контролер. Робоча напруга знаходиться в діапазоні від 1,8 до 3,6 вольта [13].

1.3.2 Огляд плати сімейства STM32F7xx

Серія MCU STM32H7 на базі Arm ® Cortex ® – M7 використовує технологію незалежної пам'яті (NVM) ST, щоб досягти найвищих в галузі результатів тестів для мікроконтролерів на базі Cortex-M з виконанням до +1327 DMIPS/3224 CoreMark з вбудованої флешпам'яті [13].

Особливостями мікроконтролера є ARM®32-розрядна Cortex®-M7, DPFPU, прискорювач Chrom-ART ™, 216 МГц макс. частота процесора, VDD від 1,7 В до 3,6 В, 2 Мб Flash, 512 КБ SRAM, 114 GPIO портів вводу/виводу із можливістю зовнішнього переривання, три 12-розрядні АЦП з 24 каналами, два 12-розрядних ЦАП-канали, вісім USART/UART, чотири I2C, шість SPI, десять таймерів загального призначення, два таймери розширеного управління, два основних таймери, один таймер малої потужності, два сторожових таймери, три CAN 2.0B, два SAI, SPDIFRX 4 входи, SDMMC, інтерфейс камери, LCD-TFT, USB 2.0 OTG HS / FS, генератор випадкових чисел (TRNG для ентропії HW), Ethernet, два типи ресурсів розширення, підключення Arduino Uno Revision 3, STMicroelectronics Morpho розширювальні роз'єми для повного доступу до всіх входів/виходів STM32, вбудований налагоджувач/програматор ST-LINK/V2-1 із SWD-роз’ємом, перемикач режиму вибору для використання набору як самостійного ST-LINK / V2-1, гнучке живлення плати, USB VBUS або зовнішнє

джерело (3,3 В, 5 В, 7-12 В), точка доступу до управління живленням, дві кнопки: USER і RESET, можливість підключення через USB: три різні інтерфейси, що підтримуються USB, віртуальний порт Com, накопичувач (USB-диск) для програмування drag'n'drop, порт для налагодження, Ethernet 10/100 Мбіт/с [12, 16].

1.3.3 EEPROM W25Q32

EEPROM – програмований ПЗП, який можу електрично очищуватись, один з видів незалежної пам'яті (таких, як PROM і EPROM). Пам'ять такого типу може стиратися і заповнюватися даними до мільйона разів.

Сьогодні класична двох транзисторна технологія EEPROM практично повністю витіснила флешпам'ять типу NOR. Однак назва EEPROM міцно закріпилося за сегментом пам'яті малої місткості незалежно від технології.

Для зберігання даних при втраті живлення буде використано пам’ять W25Q32 яка підключається до контролера за допомогою SPI інтерфейсу [2].

1.3.4 Огляд давача BMP280 для вимірювань

Давач являє собою високоточний цифровий вимірювач атмосферного тиску на базі BMP280 від фірми BOSH. Модуль BMP280 був розроблений фірмою як більш технологічна модель свого попередника BMP180 [17]. Дана модифікація надає користувачеві два послідовних інтерфейсів обміну даними (SPI й I2C), а також 3 режими роботи:

— normal - в даному режимі модуль активується з певною періодичністю, виконує необхідні вимірювання і знову дезактивується. Частота вимірів задається програмним шляхом, а результат зчитується при необхідності.

— sleep - режим максимально низького споживання електроенергії.

— forced - цей режим дозволяє активувати модуль подачею зовнішнього сигналу, що керує після виконання вимірювань, модуль автоматично переходить в режим зниженого енергоспоживання.

Всі вироблені обчислення можуть бути відфільтровані налаштованим програмним фільтром.

До основних технічних характеристик можна віднести наступні [17]:

— напруга живлення: 1.71V - 3.6V;

— інтерфейс обміну даними: I2C або SPI;

— струм, який споживається в робочому режимі: 2.7uA при частоті опитування 1 Гц;

— діапазон вимірювання атмосферного тиску: 300hPa - 1100hPa (± 0.12hPa), що еквівалентно діапазону від -500 до 9000 м над рівнем моря;

— діапазон вимірювання температури: -40 ° C ... + 85 ° C (± 0.01 ° C);

— максимальна частота роботи інтерфейсу I2C: 3.4MHz;

— максимальна частота роботи інтерфейсу SPI: 10 МГц;

— розмір модуля: 21 x 18 мм.

1.3.5 Огляд ОС FreeRTOS для побудови програмної частини КС

FreeRTOS – багатозадачна операційна система реального часу (ОСРВ) для вбудованих систем. Портована на 35 мікропроцесорних архітектур. Поширюється під ліцензією MIT з 2017 року . До 2017 року поширювалася під модифікованою ліцензією GPL з виключенням, що дозволяє розробнику привласнити модифікований код операційної системи [2].

FreeRTOS призначена для роботи на масових мікроконтролерах, які можуть мати низьку швидкодію, малий обсяг оперативної пам'яті, відсутність механізмів управління пам'яті і реалізованих на апаратному рівні механізмів підтримки багатозадачності, в тому числі швидкого перемикання контексту [3].

Диспетчер системи дуже простий і компактний (займає, в залежності від платформи і налаштувань ядра, 4-9 кілобайт), проте підтримує пріоритети процесів, що витісняє і кооперативну багатозадачність, семафори і черги. Починаючи з версії 4, FreeRTOS дозволяє використовувати співпрограми.

Версія 9.1.0 отримала підтримку компілятора ARM Compiler 6. Ядро системи вміщується в кілька файлів.

1.3.6 Огляд OS Mbed для побудови програмної частини КС

Mbed OS – це платформа та операційна система для підключених до Інтернету пристроїв на базі 32-розрядних мікроконтролерів ARM Cortex-M. Такі пристрої також відомі як пристрої Інтернету речей. Проект спільно розробляється ARM та його технологічними партнерами.

Mbed – це ОС без ОС, з можливістю використовувати безліч API без RTOS і просто систему подій, що робить додатки більш економними для центрального процесора з точки зору ресурсів.

Mbed OS надає програмну платформу Mbed C / C ++ та інструменти для створення мікропрограмного забезпечення мікроконтролера, що працює на пристроях IoT. Він складається з основних бібліотек, що забезпечують периферійні драйвери мікроконтролера, мережу, RTOS та середовище виконання, інструменти побудови та сценарії тестування та налагодження. Ці з'єднання можуть бути захищені сумісними бібліотеками SSL / TLS, такими як Mbed TLS або wolfSSL, що підтримує mbed-rtos.

1.3.7 Огляд з’єднання через UART, як основного способу комунікацій користувача та системи

Універсальний асинхронний приймач, UART – вузол обчислювальних пристроїв, призначений для організації зв'язку з іншими цифровими пристроями. Перетворює передані дані в послідовний вид так, щоб було можливо передати їх по одній фізичній цифровий лінії іншому аналогічному влаштуванню. Метод перетворення добре стандартизований і широко застосовується в комп'ютерній техніці (особливо у вбудованих пристроях і системах на кристалі (SoC)).

Являє собою логічну схему, з одного боку підключену до шини обчислювального пристрою, а з іншого має два або більше висновків для зовнішнього з'єднання [10].

1.3.8 Огляд з’єднання через ETHERNET, як альтернативного способу комунікацій користувача та системи

Для забезпечення підключення з допомого ETHERNET пристрій повинен бути сервером для обміну інформації з хостом (користувачем). В даному випадку це можна забезпечии з допомогою TCP сервера.

Transmission Control Protocol (TCP, протокол управління передачею) – один з основних протоколів передачі даних інтернету, призначений для управління передачею даних [2].

У стеці протоколів TCP/IP виконує функції транспортного рівня моделі OSI.

Механізм TCP надає потік даних з попередньою установкою з'єднання, здійснює повторний запит даних в разі втрати даних і усуває дублювання при отриманні двох копій одного пакета, гарантуючи тим самим, на відміну від UDP, цілісність переданих даних і повідомлення відправника про результати передачі.

Реалізації TCP зазвичай вбудовані в ядра ОС. Існують реалізації TCP, що працюють в просторі користувача.

Коли здійснюється передача від комп'ютера до комп'ютера через Інтернет, TCP працює на верхньому рівні між двома кінцевими системами, наприклад, браузером і веб-сервером. TCP здійснює надійну передачу потоку байтів від одного процесу до іншого. TCP реалізує управління потоком, управління перевантаженням, рукостискання, надійну передачу.

РОЗДІЛ 2 ПРОЕКТНА ЧАСТИНА

2.1 Розробка узагальненої структури комп’ютерної системи

Узагальнена схема підключення апаратних компонентів КС контролю параметрами мікроклімату зображено на рис. 2.1. Давач BMP280 та зовнішня пам’ять EEPROM W25Q32 фізично підключаються до мікроконтролера.

Рисунок 2.1 – Узагальнена схема підключення апаратних компонентів КС

Узагальнена структура КС контролю параметрами мікроклімату зображено на рис. 2.2. Взаємодія користувача та системи буде організована за принципом надсилання запитів до КС з допомогою одного з можливих варіантів підключення, мікроконтролер опрацьовує запит, виконуючи дії відповідно до розробленого на базі протоколів UART та TCP набору правил: отримуючи дані з давача, зчитуючи/записуючи показники та надсилаючи відповідь.

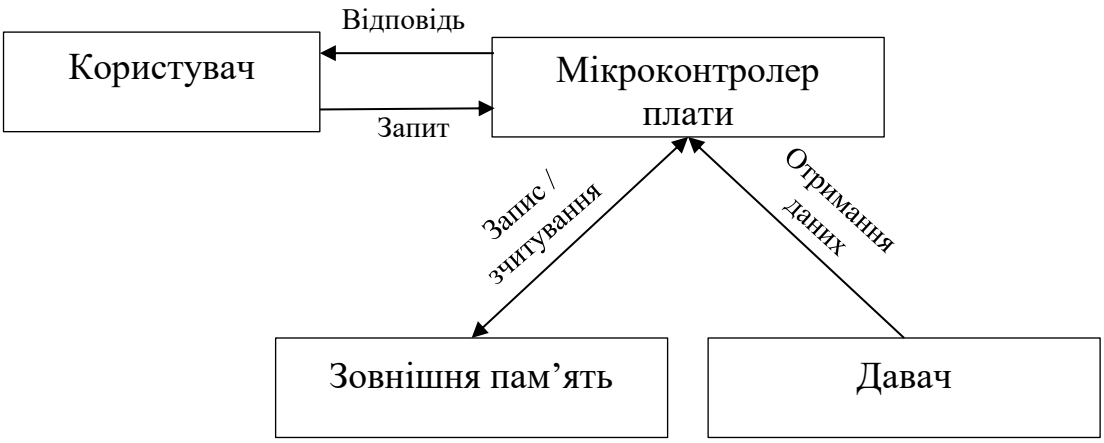

Рисунок 2.2 – Узагальнена структура КС

2.2 Обґрунтування вибору апаратного забезпечення проектованого комп'ютерного засобу

Вбудована система буде побудована на основі STM32F767ZI NUCLEO. Серія MCU STM32F7 на базі Arm ® Cortex ® – M7 використовує технологію незалежної пам'яті (NVM) ST, щоб досягти найвищих в галузі результатів тестів для мікроконтролерів на базі Cortex-M з виконанням до +1327 DMIPS/3224 CoreMark з вбудованої флешпам'яті [16].

Пристрої STM32F7 з вбудованим процесором шифрування/хешування підтримують служби безпеки, такі як безпечна установка прошивки й безпечне завантаження – безпечне оновлення прошивки, що дозволяє встановлювати нові коди додатків в захищеному режимі [15].

Базову обв’язку кристалу з кнопкою скидання, та кварцовим резонатором зображено на рис. 2.3.

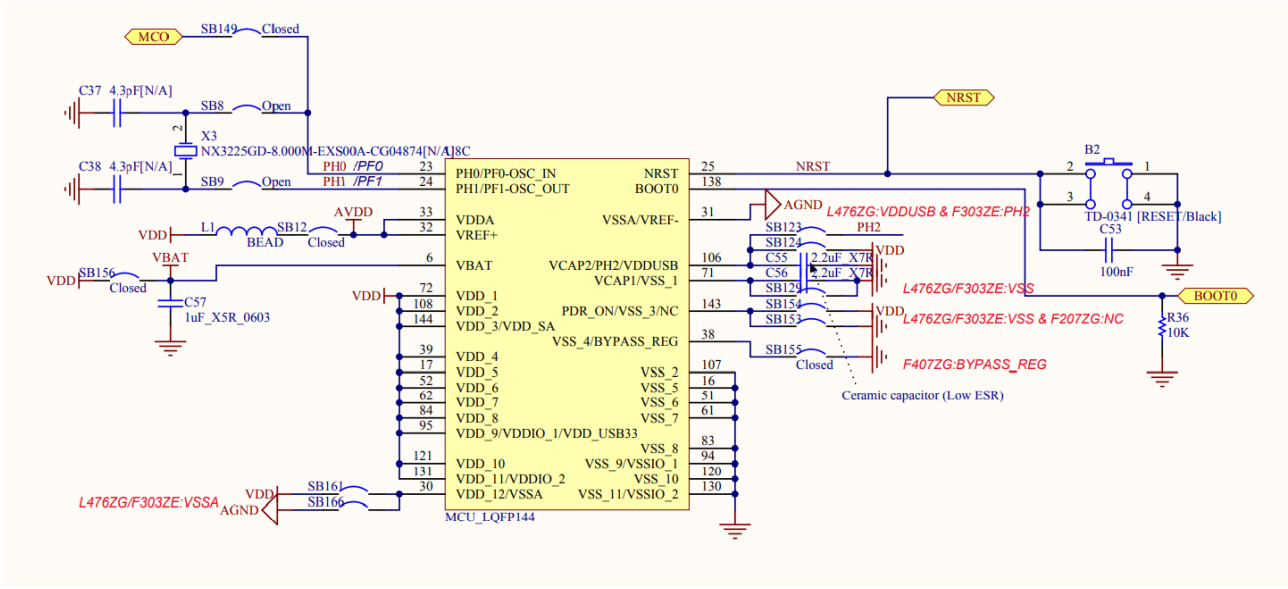

Рисунок 2.3 – Базова обв’язка STM32F767ZI з кнопкою Reset

Базову обв’язку з кнопкою USER зображено на рис. 2.4.

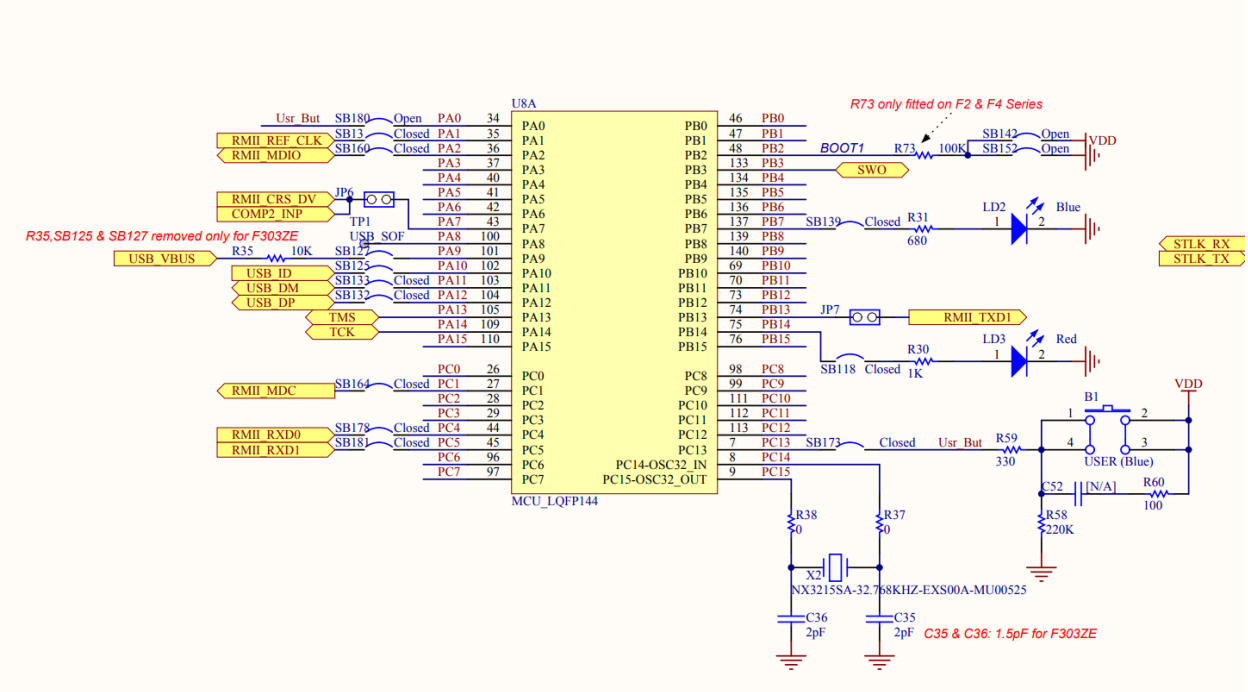

Рисунок 2.4 – Базова обв’язка STM32F767ZI з кнопкою USER

Коли кнопка не натиснута (відкрита), між її ногами немає зв’язку і струм не протікає в результаті на виході IN логічний нуль. Коли кнопка замикається і на виході IN логічна одиниця [15].

Плата має вбудований ST-LINK програматор (SWD – Serial Wire Debug) з виведеним портом для нього, який являє собою USB порт (microUSB). Обв’язку SWD зображено на рис. 2.5 [5].

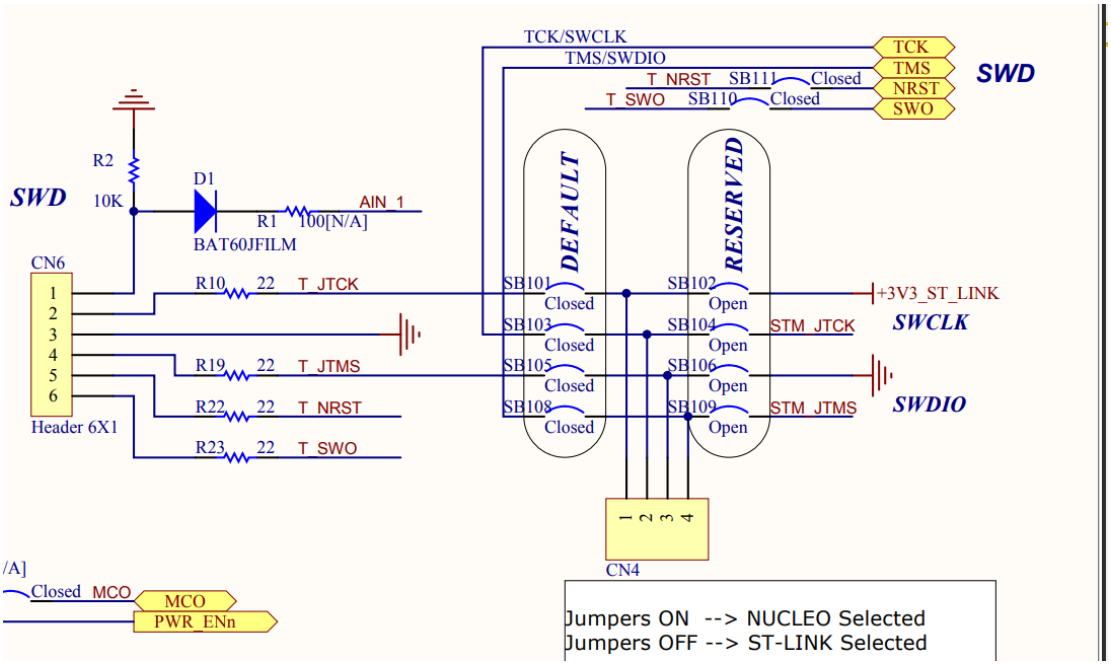

Рисунок 2.5 – Обв’язка SWD

Вхідні дані охоплюють дані з давача BMP280, з допомогою якого і проводяться вимірювання, його базова обв'язка зображена на рис. 2.6 [17].

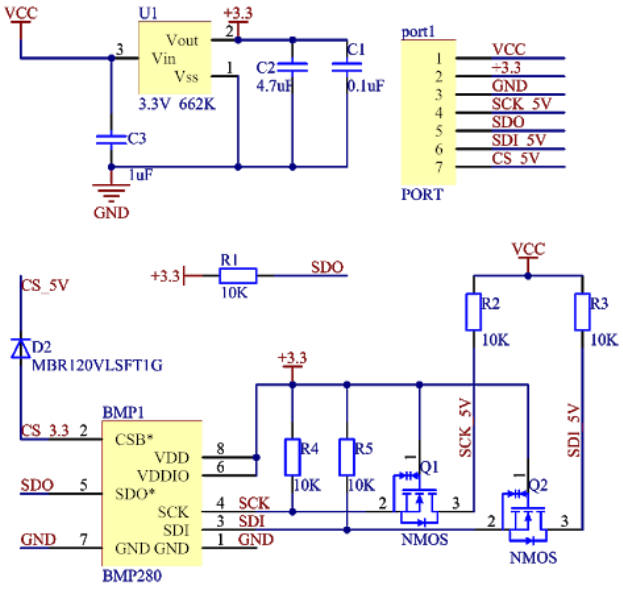

Рисунок 2.6 – Обв’язка давача BMP280

Для запису даних та їх зчитування буде використано W25Q32, яка підключається до системи з допомогою інтерфейсу SPI, схема зображена на рис. 2.7.

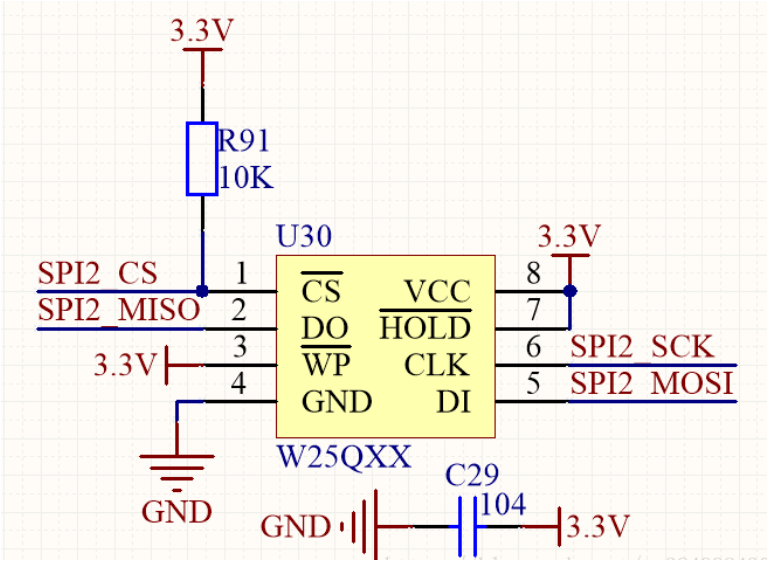

Рисунок 2.7 – Базова обв’язка W25Q32

2.3 Обґрунтування вибору програмного забезпечення проектованого комп'ютерного засобу

2.3.1 Операційна система реального часу

Для написання програми було обрано Mbed OS [14]. Близьким аналогом даної ОС є FreeRTOS, проте Mbed OS має ряд переваг. FreeRTOS – це лише частина ядра для ОС, Mbed охоплює більше компонентів – для прикладу периферійні пристрої, інтерфейси: SPI, UART, I2C, цифрові або аналогові входи або виходи, ETHERNET, USB, зберігання, використовуючи різні файлові системи та запам'ятовувальні пристрої та багато іншого. Більшість з цих компонентів можуть взаємодіяти з ОС у вигляді потоків, семафорів, м’ютексів. FreeRTOS, звичайно, має також безліч функцій RTOS, але їх доведеться поєднувати з інтерфейсами та компонентами вищого рівня самостійно. Зараз існує безліч розширень FreeRTOS+, але не всі вони безплатні, і охоплюють меншу кількість платформ [20].

2.3.2 Методи підключення для комунікації користувача та КС

Для комунікації користувача та системи було обрано протоколи UART та TCP з використанням DMA для швидкодії, оскільки вони повністю задовольнити вимоги до програмної роботи з запитами.

Передача даних в UART здійснюється по одному біту в рівні проміжки часу. Цей часовий проміжок визначається заданою швидкістю UART і для конкретного з'єднання вказується в бодах (що в цьому випадку відповідає бітам в секунду) [10].

Прийнято угоду, що пасивним (за відсутності потоку даних) станом входу і виходу UART є логічна 1. Стартовий біт завжди є логічним 0, тому приймач UART чекає перепаду з 1 в 0 і відраховує від нього часовий проміжок в половину тривалості біта (середина передачі стартового біта). Обмін даними через UART відбувається в дуплексному режимі. Тобто передача і прийом даних можуть відбуватися одночасно. У інтерфейсу UART існують 2 сигналу RX і TX.

При підключенні двох UART пристроїв сигнали RX з'єднуються з сигналами TX. Використовується перехресна схема з'єднання.

Можливості UART в STM32:

— при частоті тактування 72 МГц швидкість передачі даних сягає до 4,5 Мбіт / сек.;

— розмірність переданих даних 8 або 9 біт;

— можуть бути задані формати з 1 або 2 стопових бітів;

— інтерфейс підтримує фізичний рівень протоколу інфрачервоного порту (IRDA);

— також підтримується інтерфейс контактних смарт-карт ISO 7816;

— можна обмінюватися об'єктами можуть відбуватися з використанням контролера прямого доступу до пам'яті (DMA);

— інтерфейс може працювати в синхронному режимі.

Передавальна і приймальна частина працюють абсолютно незалежно один від одного. Зі спільного у них тільки тактовий генератор. Тобто можуть бути залучений тільки приймач, або тільки передавач. Приймач і передавач можуть

працювати з різними пристроями, різними протоколами верхнього рівня. Проте у них повинна бути однакова швидкість і формат даних [4].

2.3.3 Використання DMA при комунікації з системою

Прямий доступ до пам'яті (DMA) – режим обміну даними між пристроями комп'ютера або ж між пристроєм і основною пам'яттю, в якому центральний процесор (ЦП) не бере. Так як дані не пересилаються в ЦП і назад, швидкість передачі збільшується [9].

Процес читання даних з пристрою відбувається за таким принципом: ЦП записує значення в регістри контролера DMA, відправляє пристрою (наприклад, диску) команду на читання даних. Пристрій читає дані (наприклад, з диска) і записує в свою внутрішню пам'ять (буфер). Контролер DMA встановлює на адресну шину адреси пам'яті ПК, відправляє пристрою запит на читання даних з внутрішньої пам'яті (буфера) пристрої. Пристрій отримує запит і при цьому навіть не знає, чи прийшов запит від ЦП або від контролера DMA. Пристрій пересилає чергове слово зі своєї внутрішньої пам'яті (буфера) в оперативну пам'ять ПК за адресою, що знаходиться на адресній шині. Потім пристрій посилає контролеру DMA сигнал, що повідомляє про закінчення запису. Контролер DMA збільшує адреса пам'яті ПК і виставляє його на адресну шину, зменшує значення свого лічильника байтів, знову відправляє запит на читання даних з внутрішньої пам'яті (буфера) пристрої. Цикл повторюється, поки значення лічильника не стане дорівнює нулю. Після закінчення циклу пристрій ініціює переривання процесора, що повідомляє про завершення перенесення даних [13].

Схема роботи DMA в системі зображено на рис. 2.8.

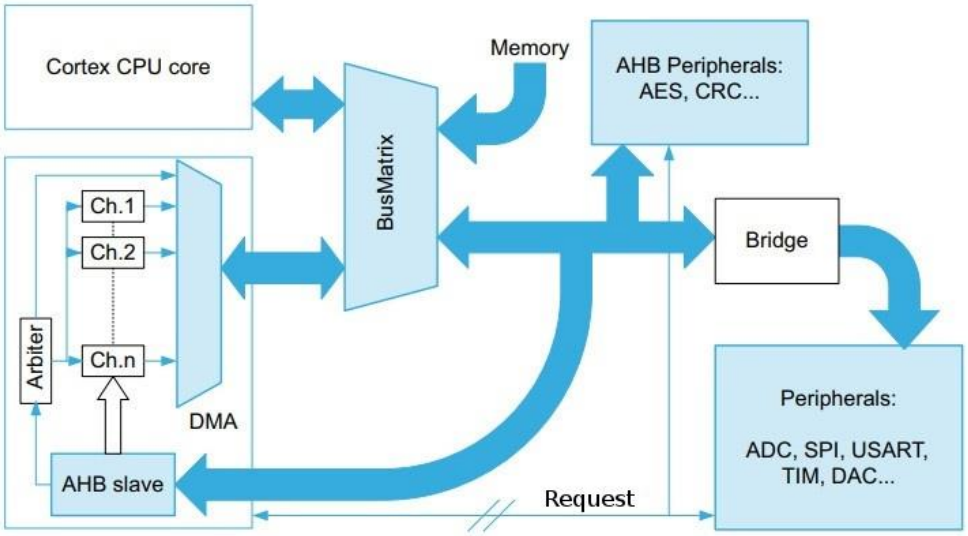

Рисунок 2.8 – Схема взаємодії DMA з іншими складовими системи

2.4 Набір правил та команди для комунікації користувача та системи

Набір правил для комп'ютерної системи буде побудовано на базі комунікаційних протоколів UART і TCP.

Комунікаційний протокол — це обумовлені наперед правила передачі даних між двома пристроями. До основних параметрів, які описує протокол, відносяться [8]:

— тип перевірки помилок, що використовується;

— метод компресії (стискання) інформації (якщо такий є);

— спосіб визначення передаючим пристроєм завершення передачі;

Протокол зв'язку — це набір стандартних правил представлення інформації, передачі сигналів, ідентифікації та виявлення помилок необхідний для обміну інформацією. Простим прикладом пристосованого для голосового зв'язку є радіо диспетчер, що розмовляє з пересувними станціями. Протоколи зв'язку для цифрових комп'ютерних мереж мають багато особливостей, котрі призначені для забезпечення надійного обміну інформацією в умовах неідеального каналу зв'язку.

РОЗДІЛ 3 ПРАКТИЧНА ЧАСТИНА

3.1 Реалізація або моделювання проектних рішень

3.1.1 Створення проекту на базі Mbed OS в Visual Studio Code

Visual Studio Code підтримує фреймворк PlatformIO, з допомогою якого також можна створити проект для даної плати та ОС, проте в ході роботи було виявлено проблеми в підтримці мікроконтролера в версії Mbed для PlatformIO, тому було вирішено перейти на утиліту розробників Mbed – Mbed-Cli, створивши свій target з конфігурацією плати, що і було успішно реалізовано.

Mbed-Cli (mbed command line interface) – набір утиліт на мові Python для автоматизації виконання більшості дій, таких як додавання бібліотек, збірка проекту і т. д [20].

Створення проекту відбувається з допомогою команди mbed new Name_of_project, процес зображено на рис. 3.1.

```
C:\mbed>mbed new COURSE
[mbed] Working path "C:\mbed" (program)
[mbed] Creating new program "COURSE" (git)
[mbed] Adding library "mbed-os" from "https://github.com/ARMmbed/mbed-os" at branch/tag "latest"
[mbed] Updating reference "mbed-os" -> "https://github.com/ARMmbed/mbed-os/#d6784c3ee6ada8e886801d0197904d3ab94c891c"

C:\mbed>
```

Рисунок 3.1 – Створення проекту

Після виконання даної команди в каталозі створення буде згенеровано проект з такими файлами: .mbed, mbed_settings.py, mbed-os.lib та тека mbed-os, рис. 3.2.

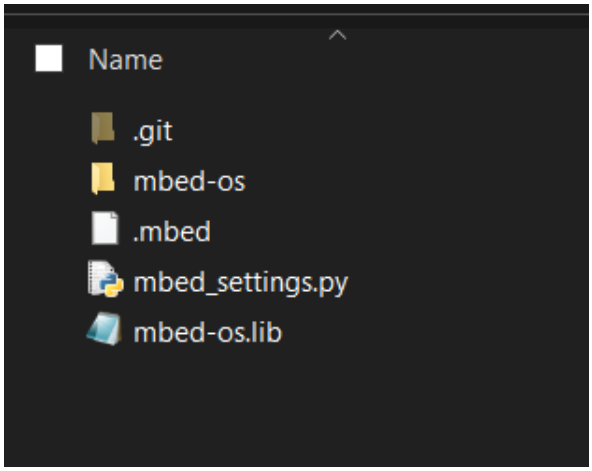

Рисунок 3.2 – Структура проекту

В теці mbed-os знаходиться ОС, структура папок зображена на рис. 3.3.

Name	Date modified	Type	Size
.github	27.12.2020 4:39	File folder	
cmsis	27.12.2020 4:39	File folder	
components	27.12.2020 4:39	File folder	
docs	27.12.2020 4:39	File folder	
drivers	27.12.2020 4:39	File folder	
events	27.12.2020 4:39	File folder	
features	27.12.2020 4:39	File folder	
hal	27.12.2020 4:39	File folder	
platform	27.12.2020 4:39	File folder	
rtos	27.12.2020 4:39	File folder	
targets	27.12.2020 4:39	File folder	
TEST_APPS	27.12.2020 4:39	File folder	
TESTS	27.12.2020 4:39	File folder	
tools	27.12.2020 4:39	File folder	
UNITTESTS	27.12.2020 4:39	File folder	
.astyleignore	27.12.2020 4:39	ASTYLEIGNORE File	2 KB
.astylerc	27.12.2020 4:39	ASTYLERC File	1 KB
.coveragerc	27.12.2020 4:39	COVERAGERC File	1 KB
.gitattributes	27.12.2020 4:39	Text Document	1 KB
.gitignore	27.12.2020 4:39	Text Document	2 KB
.pylintrc	27.12.2020 4:39	PYLINTRC File	1 KB
.travis.yml	27.12.2020 4:39	Yaml Source File	12 KB
CONTRIBUTING.md	27.12.2020 4:39	Markdown Source ...	1 KB
doxyfile_options	27.12.2020 4:39	File	111 KB
DOXYGEN_FRONTPAGE.md	27.12.2020 4:39	Markdown Source ...	1 KB
doxygen_options.json	27.12.2020 4:39	JSON Source File	3 KB
Jenkinsfile	27.12.2020 4:39	File	1 KB

Рисунок 3.3 – Структура файлів в теці mbed-os

З допомогою Visual Studio Code відкриваємо проект з допомогою меню, на рис. 3.4 зображено інтерфейс програми.

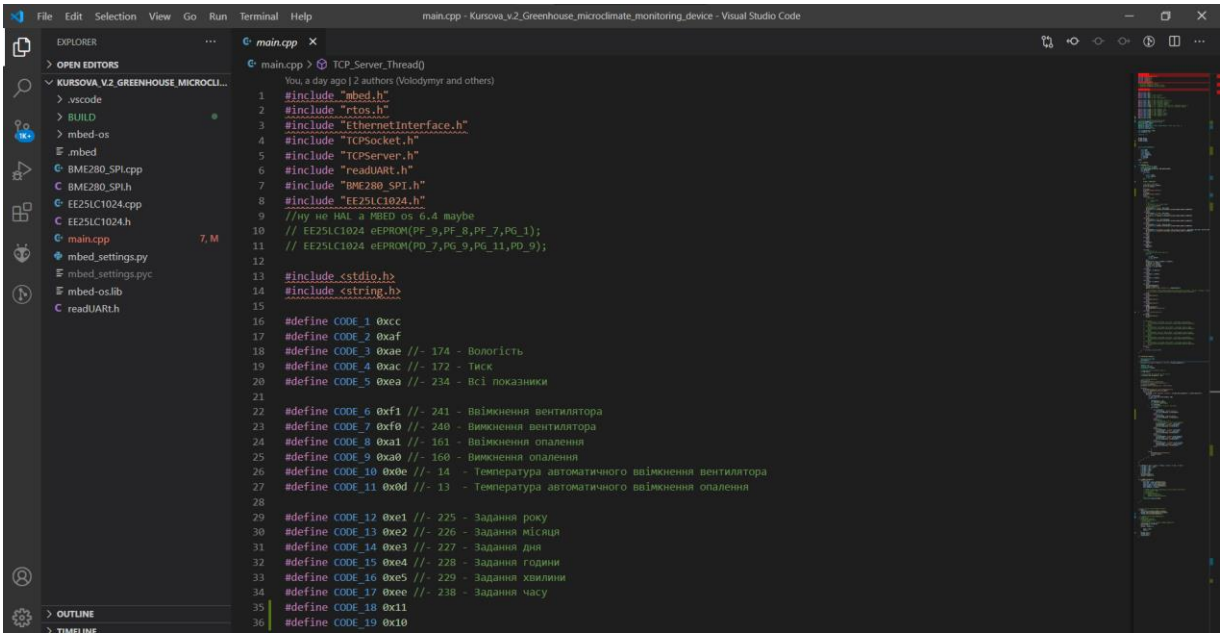

Рисунок 3.4 – Інтерфейс Visual Studio Code

3.1.2 Підключення апаратних компонентів системи

Загальна схема з’єднань в даній комп’ютерній системі керування мікрокліматом приміщення складається з трьох компонентів: BMP280, EEPROM W25Q32 та STM32F767ZI Nucleo [3]. Всі компоненти з’єднуються до плати з допомогою інтерфейсу SPI та відповідно до схеми з’єднань яка подана в табл.3.1.

Таблиця 3.1 – Схема з’єднань компонентів та STM32F767ZI

Компонент	MOSI	MISO	CS	SCK
EE25LC1024	PF_9	PF_8	PF_7	PG_1
BMP280	PC_12	PC_11	PC_10	PC_9

3.2 Розробка алгоритмів роботи програмних модулів пристрою

Загальна блок схема алгоритму роботи вбудованої системи зображено на рис. 3.5. Потоки працюють послідовно з максимальною частотою включень, саме це дозволяє забезпечити можливість відповіді без затримок, як і через з'єднання з допомогою UART так і через з'єднання з допомогою ETHERNET.

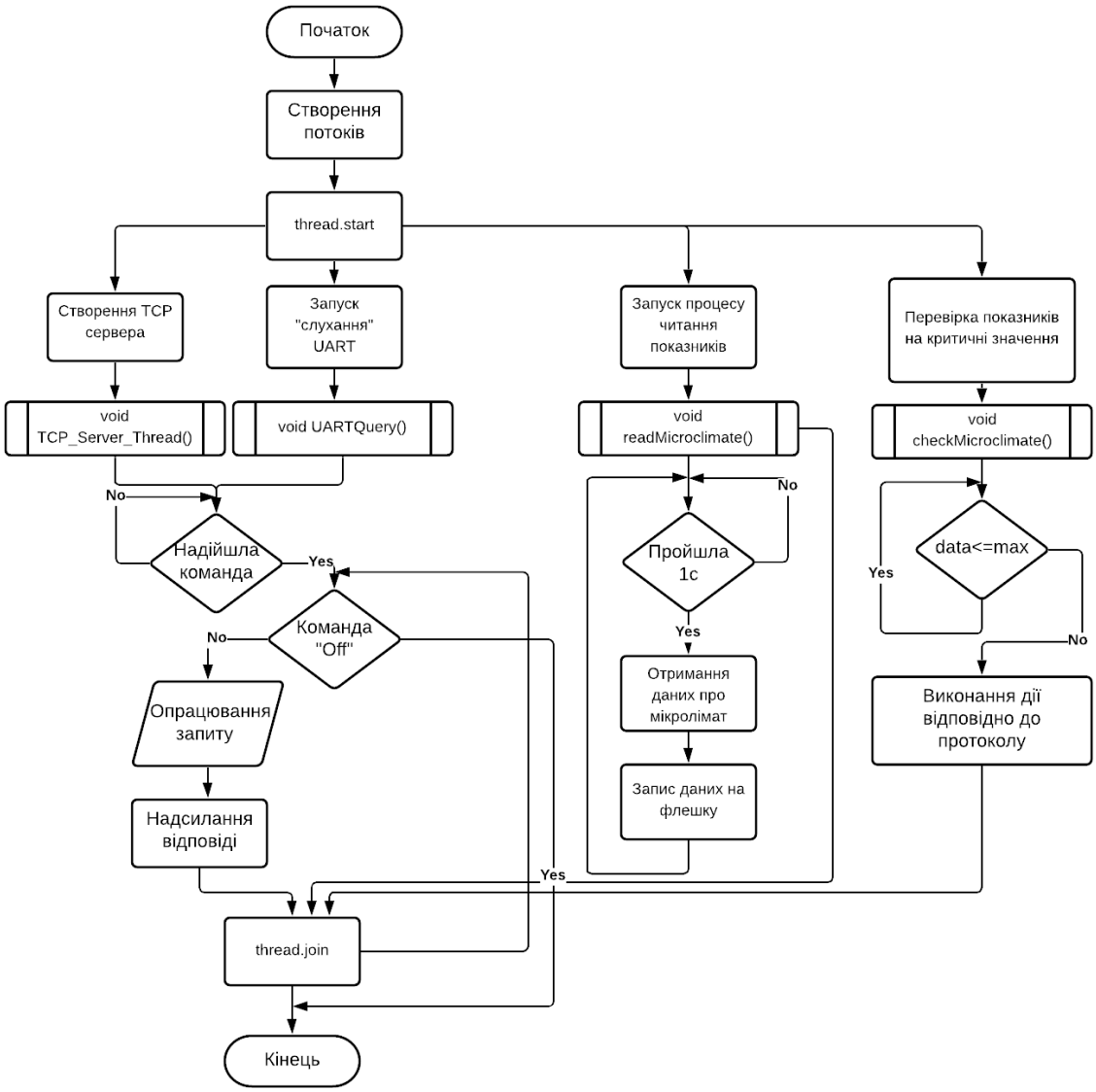

Рисунок 3.5 – Загальна блок схема роботи системи

Оскільки система типу «real-time embedded system» потрібно забезпечити не тільки роботу в реальному часі, але і врахування можливості продовження «часу та дати», встановлених при першому включенні системи, тобто у разі

зникнення живлення система після його відновлення почне працювати зі значенням поточного часу, а не значенням моменту коли система була вимкнена.

Головна програма int main() в вигляді блок-схеми алгоритму зображена на рис. 3.6. Для початку створюються потоки для відповідних підпрограм та виконується їх запуск з використанням callback – функції зворотного виклику, це частина виконуваного коду, що передається як аргумент до іншого коду, який має викликати цей код у відповідь, тобто виконати аргумент у певний момент часу.

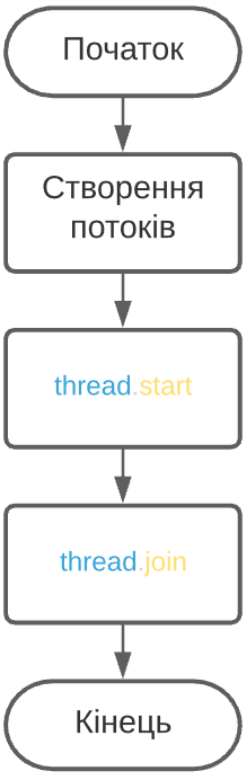

Рисунок 3.6 – Блок схема головної програми int main()

Блок схема алгоритму моніторингу мікроклімату в приміщені зображена на рис. 3.7. Ця підпрограма працює по принципу «вічного циклу», тобто точка завершення відсутня і кожен відлік вагою 1 с супроводжується зчитуванням показників з давача та запис на флешку.

Рисунок 3.7 – Блок схема алгоритму вимірювань мікроклімату в приміщені

Функція UARTQuery() забезпечує «спілкування» користувача з системою з допомогою інтерфейсу UART, рис. 3.8. При запуску функції проводиться перевірка чи це перший запуск – перевірка параметра Date, який містить дані про дату, яка записується разом із даними давача на флешку. У випадку якщо даних про дату немає – запускається відповідна підпрограма, яка після введення даних користувачем – запише їх в змінну. Після даної дії ініціалізується робота з DMA, наступним кроком є перевірка на помилки й, у випадку якщо їх немає, система чекає на команду ззовні від користувача, для подальшої роботи. З допомогою структури switch case було реалізовано роботу по протоколу для даного пристрою.

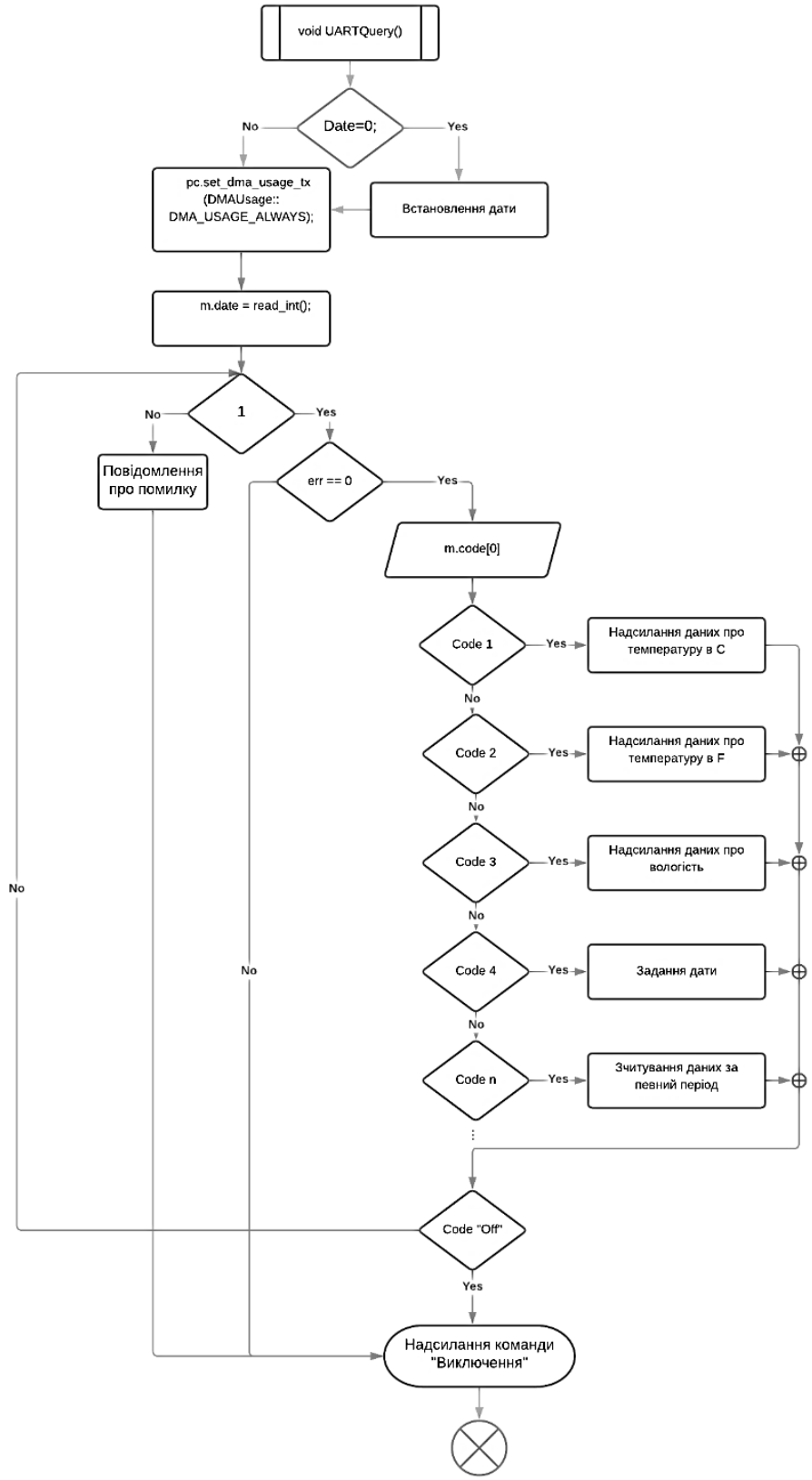

Рисунок 3.8 – Блок схема алгоритму функції UARTQuery()

Функція TCP_Server_Thread()забезпечує «спілкування» користувача з системою з допомогою протоколу ETHERNET, рис. 3.9. Для цього використовується TCP Socket та TCP Server. При запуску функції проводиться

перевірка чи це перший запуск – перевірка параметра Date, який містить дані про дату, яка записується разом із даними давача на флешку. У випадку якщо даних про дату немає – запускається відповідна підпрограма, яка після введення даних користувачем – запише їх в змінну. Після створюється та ініціалізується TCP сервер, який протягом всієї роботи знаходиться в режимі очікувань підключення клієнта, наступним кроком є перевірка на помилки й, у випадку якщо їх немає, система чекає на команду ззовні від користувача, для подальшої роботи. З допомогою структури switch case було реалізовано роботу по протоколу для даного пристрою.

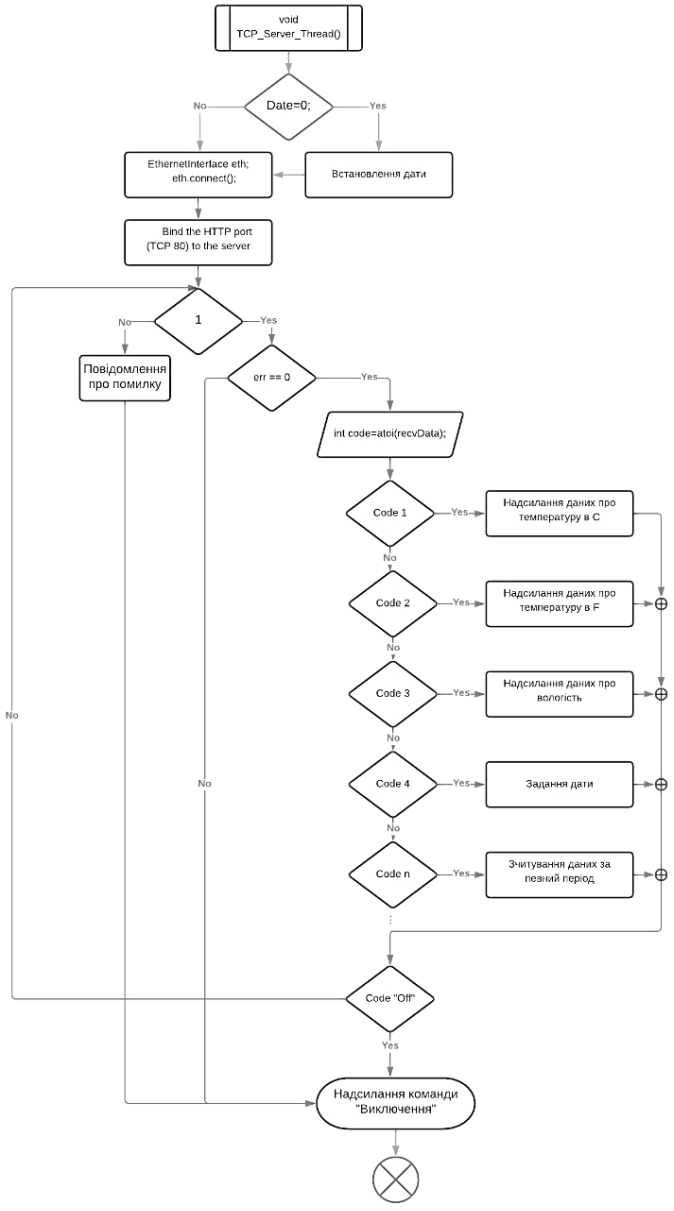

Рисунок 3.9 – Блок схема алгоритму функції TCP_Server_Thread()

3.3 Опис компонентів програмного забезпечення КС

У файлі main.c підключаються хедер файли за допомогою директиви #include. Підключення хедер файлів необхідне для забезпечення доступу до структур, типів даних, прототипів функцій, що знаходяться в окремих модулях, а саме «c» файлах.

В даному проекті було використано такі бібліотеки: mbed.h, rtos.h, EthernetInterface.h, TCPSocket.h, TCPServer.h, readUARt.h, BME280_SPI.h, EE25LC1024.h.

`#include "mbed.h"` – підключається для використання mbed-os і всього функціоналу ОС, в даному файлі знаходяться підключення певних С бібліотек, бібліотек налагодження mbed, периферійних, внутрішніх та неапаратних компонентів. Крім цього відповідно до базових конфігурацій відповідної плати під яку даний проект було створено в бібліотеці спрацьовують перевірки на відповідність умовам і підключаються певні специфічні файли, наприклад:

```
#ifndef MBED_H
#define MBED_H

#include "platform/mbed_version.h"

#if MBED_CONF_RTOS_API_PRESENT
#include "rtos/rtos.h"
#endif

#if MBED_CONF_NSAPI_PRESENT
#include "netsocket/nsapi.h"
#include "netsocket/nsapi_ppp.h"
#endif

#if MBED_CONF_EVENTS_PRESENT
#include "events/mbed_events.h"
#endif

#if MBED_CONF_FILESYSTEM_PRESENT
#include "filesystem/mbed_filesystem.h"
#endif
```

`#include "rtos.h"` – підключається для використання певного функціонала RTOS OS, на якій і побудована mbed, а саме потоки, мютекси, таймери, прапори подій.

`#include "EthernetInterface.h"` – підключається для того, щоб було можливим використання мережевого підключення через ETHERNET, Впровадження NetworkStack для драйвера Ethernet на основі EMAC.

`#include "TCPSocket.h"` – використовується для створення клієнтського TCP сокета. Процеси при такому обміні можуть виконуватися як на одній ЕОМ, так і на різних ЕОМ, пов'язаних між собою мережею. Сокет це абстрактний об'єкт, що являє собою точку з'єднання, яка є кінцевою.

`#include "TCPServer.h"` – підключається для створення TCP сервера, який прослуховує зазначені номери портів, встановлює TCP-сеанси з клієнтами, які ініціюють TCP-з'єднання, а потім обробляє вхідні дані. Сервер TCP може обробляти дані від декількох клієнтів одночасно, створюючи окремі пакети для кожного клієнта і відправляючи підтвердження вихідного клієнтові після аналізу кожного запису або фіксації кожного пакета.

`#include "readUARt.h"` – підключається для того, щоб мати можливість «спілкуватись» з STM32F767ZI з допомогою UART.

`#include "BME280_SPI.h"` – підключається для того, щоб використовувати базовий функціонал роботи давача BMP280: ініціалізація, зчитування температури, вологості, тиску з давача.

`#include "EE25LC1024.h"` – в даній бібліотеці знаходиться базовий функціонал для роботи з флешкою.

Дана вбудована система відповідає за те щоб отримати, опрацювати та видати результат за запитом для цього було реалізовано певні програмні елементи.

Набір правил для взаємодії користувача та КС, побудовані на протоколах UART та TCP, складаються з визначених команд, таких як:

— case 0xcc - 204 - температура в C;

— case 0xaf - 175 - температура в F;

— case 0xae – 174 - вологість;

— case 0xac – 172 – тиск;

— case 0xea – 234 - всі показники;

— case 0xf1 – 241 - ввімкнення вентилятора;

— case 0xf0 – 240 - вимкнення вентилятора;

— case 0xa1 – 161 - ввімкнення опалення;

— case 0xa0 – 160 - вимкнення опалення;

— case 0x0e – 14 - температура автоматичного ввімкнення вентилятора;

— case 0x0d – 13 - температура автоматичного ввімкнення опалення;

— case 0xe1 – 225 - задання року;

— case 0xe2 – 226 - задання місяця;

— case 0xe3 – 227 - задання дня;

— case 0xe4 – 228 - задання години;

— case 0xe5 – 229 - задання хвилини;

— case 0xee – 238 - задання часу;

— case 0xf5 – 245 - історія показників за останні 5 вимірювань;

— case 0xfa – 250 - історія показників за останні 10 вимірювань;

— case 0xaa – 170 - історія всіх показників;

— case 0x60 – 96 - очищення флешки;

— case 0xfe – 254 - температура в F та вологість;

— case 0xfc – 252 - температура в F та тиск;

— case 0xce – 206 - температура в C та вологість;

— case 0xcf – 207 - температура в C та тиск;

— case 0xec – 236 - тиск та вологість.

Як вже згадувалось раніше алгоритм основної програми знаходиться в файлі main.cpp, першим етапом потрібно підключити бібліотеки, які забезпечують функціонування системи, опис основного призначення бібліотеки і базовий функціонал знаходиться в розділі 3.2:

```
#include "mbed.h"
#include "rtos.h"
#include "EthernetInterface.h"
#include "TCPSocket.h"
#include "TCPServer.h"
#include "readUARt.h"
#include "BME280_SPI.h"
#include "EE25LC1024.h"
#include <stdio.h>
#include <string.h>
```

Далі оголошуються змінні відповідних типів для роботи з кожним підключеним елементом та ініціалізуються відповідними пінами підключення:

```
EE25LC1024 eEPROM(PF_9,PF_8,PF_7,PG_1);
Serial pc(USBTX, USBRX);
BME280_SPI sensor(PC_12, PC_11, PC_10, PC_9);
DigitalOut fan(PF_13);
DigitalOut heating(PF_14);
```

EE25LC1024 відповідає за підключену флешку до STM32F767ZI на піни PF_9,PF_8,PF_7,PG_1 та ініціалізацію ними змінної eEPROM.

Serial використовується, щоб створити змінну для роботи з UART та ініціалізувати її пінами для роботи, в цьому випадку USBTX, USBRX.

BME280_SPI використовується для роботи з давачем BMP280, змінна sensor ініціалізується пінами вказаними при оголошенні, mosi – PC_12, miso – PC_11, sclk – PC_10, cs – PC_9.

DigitalOut fan(PF_13) використовується для підключення вентилятора, який спрацьовує при спрацюванні умови на критичні значення показника в випадку коли температура перевищує поріг 25°C.

DigitalOut heating(PF_14) використовується для підключення обігрівача, який спрацьовує при спрацюванні умови на критичні значення показника в випадку коли температура менша за 10°C.

Створення структури tm з ключем t, в яку буде записуватись дата – `struct tm t;`

Створення потоків для забезпечення паралельної роботи системи:

```
Thread thread;
Thread thread1;
Thread thread2;
```

Після оголошується структура в якій будуть зберігатися дані отримані з давача, де в temF – зберігається температура в Фаренгейтах, temC – температура в Цельсіях, prs_hPa – тиск, humidity – відносна вологість повітря, timeWrite – дата для запису на флешку, maxTemp – максимальна температура, minTemp – мінімальна температура.

```
struct microclimateStruct
{
    float temF;
    float temC;
    float prs_hPa;
    float humidity;
    time_t timeWrite;
    int maxTemp;
    int minTemp;
    /* data */
}mcSt;
```

Далі описується в вигляді коду підпрограма UARTQuery(). В даній підпрограмі вказується режим роботи з DMA, а саме DMA_USAGE_ALWAYS, оголошується змінна, в яку будуть записуватись дані для відповіді користувачеві, також використовується при приведення типів об'єднання з ключем m, зчитуємо дані, які надіслав користувач, та з допомогою конструкції switch case виконуємо відповідні обрахунки та формування вихідного повідомлення, рис. 3.10.

```
event_callback_t callback;
pc.set_dma_usage_tx(DMAUsage::DMA_USAGE_ALWAYS);
char dataS[50];
while(true){
    union main
    {
        uint32_t date;
        uint8_t code[4];
    }m;

    m.date = read_int();

    pc.printf("%d\n",m.code[0]);
    switch (m.code[0])
    {
    case 0x11:
    system.start();
    pc.printf("System start\n");
    break;
    case 0x10:
    system.stop();
```

Рисунок 3.10 – Лістинг функції UARTQuery, для роботи з UART

Підпрограма TCP_Server_Thread() створює об’єкт класу EthernetInterface, змінну для сервера та виконує з’єднання з мережею та 80-м портом сервера. Після перевірки на помилки – сервер переходить в режим очікування з’єднання клієнтів, в цьому випадку було додано обмеження в максимальній кількості підключень – це 5 клієнтів. Логіка використання протоколу залишається практично ідентичною відносно зв’язку через UART, рис. 3.11.

```
    eth.connect();

    pc.printf("The    target    IP    address    is    '%s'\r\n",
eth.get_ip_address());

    TCPServer srv;
    TCPSocket clt_sock;
    SocketAddress clt addr;

    /* Open the server on ethernet stack */
    srv.open(&eth);

    /* Bind the HTTP port (TCP 80) to the server */
    srv.bind(eth.get_ip_address(), 80);


    //srv.set_blocking(false);
    while (true) {
     pc.printf("waiting for client\r\n");
     /* Can handle 5 simultaneous connections */
     int err= srv.listen(1);
     pc.printf("server listening error : %d\r\n",err);

     while(1){
          pc.printf("waiting for client connection\r\n");
          err = srv.accept(&clt_sock, &clt_addr);
          if(err == 0){
               pc.printf("client       connected       :%s:%d\r\n",
clt addr.get ip address(), clt addr.get port());
               while(1){
                    char recvData[100];
                    int len = clt_sock.recv(recvData, 100);
```

Рисунок 3.11 – Лістинг функції TCP_Server_Thread(), для роботи з TCP

Частина лістингу створення сервера відбувається з допомогою коду, який ініціалізує сокет сервера та розблоковує з’єднання при виконанні всіх вимог:

```
_lock.lock();
nsapi_error_t ret;
while (true) {
    if (! socket) {
        ret = NSAPI_ERROR_NO_SOCKET;
        break;
    }
```

Першим етапом роботи програми є встановлення часу, за замовчуванням ці дані будуть “2000 р. січень 1 00:00:00” в процесі роботи програми ці дані можна змінювати з допомогою запитів на відповідні адреси. Для встановлення часу необхідно створити об’єкт структури tm та заповнити поля такі як рік, місць, день, година, хвилина, секунда. Після чого виконується команда встановлення оновлення дати.

Після встановлення дати відбувається створення потоку отримання даних з давача струму, оброблення їх, а саме перевірка якщо температура більша ніж 25°C тоді включити систему охолодження якщо температура менша ніж 10°C тоді включити систему обігріву.

В іншому потоці відбувається очікування отримання даних від зовнішніх пристроїв або користувача, після чого пристрій виконує певні дії так як: повернення даних користувачу, встановлення дати, включення або виключення системи охолодження, включення або виключення системи обігріву. Також є можливість передати команду для виключення контролера або очищення даних флешки.

```
    void setTime(int year, int mon, int mday, int hour, int min,
int sec) {
        t.tm_year = year - 1900;
        t.tm_mon =  mon - 1;
        t.tm_mday = mday;
        t.tm_hour = hour;
        t.tm_min = min;
        t.tm_sec = sec;


        set_time(mktime(&t));
         seconds = time(NULL);
    }
```

Підпрограма збору даних давача через певний проміжок часу:

```
void readMicroclimate(){
    while (true) {
        mcSt.temF = sensor.getTemperature();
        mcSt.temC = (mcSt.temF-32)*0.555556;
        mcSt.prs_hPa = sensor.getPressure();
        mcSt.humidity = sensor.getHumidity();
        mcSt.timeWrite = seconds;

}
```

Код int main(), де створюються потоки та викликаються всі функції:

```
int main() {
    thread.start(callback(readMicroclimate));
    thread1.start(callback(TCP_Server_Thread));
    thread2.start(callback(UARTQuery));
    thread2.join();
    thread1.join();
    thread.join();
}
```

3.4 Збірка та заливка програми в STM32F767ZI NUCLEO

Оскільки даний проект було створено з допомогою утиліти Mbed-Cli – після того, як було написано всі потрібні підпрограми та алгоритми процес компілювання та налагодження буде виконуватись також з допомогою даної оболонки.

Для того, щоб скомпілювати проект потрібно зайти в консоль, можна використати термінал Visual Studio Code, але за умови що всі необхідні компоненти заздалегідь встановлені, та використати команду, `mbed compile -m NUCLEO_767ZI - GCC_ARM` , рис. 3.12.

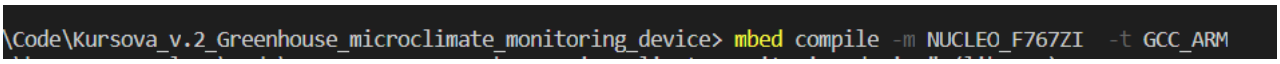

Рисунок 3.12 – Команда для компіляції проекту

Для даного проекту було обрано компілятор GCC_ARM – набір компіляторів спільноти проекту GNU, які нею ж і підтримуються. GCC_ARM – вільне програмне забезпечення і широко використовується для компіляції багатьох програм проекту.

Результат компіляції зображено на рис. 3.13. Було отримано бінарний файл з прошивкою ПЗ даної системи.

```
Building project Kursova_v.2_Greenhouse_microclimate_monitoring_device (NUCLEO_F767ZI, GCC_ARM)
Scan: Kursova_v.2_Greenhouse_microclimate_monitoring_device
Link: Kursova_v.2_Greenhouse_microclimate_monitoring_device
Elf2Bin: Kursova_v.2_Greenhouse_microclimate_monitoring_device
| Module              |     .text |     .data |      .bss |
|---------------------|-----------|-----------|-----------|
| BME280_SPI.o        |  1318(+0) |    0(+0)  |     0(+0) |
| EE25LC1024.o        |   228(+0) |    0(+0)  |     0(+0) |
| [fill]              |   164(+0) |    7(+0)  |    24(+0) |
| [lib]\c.a           | 48792(+0) | 2572(+0)  |   127(+0) |
| [lib]\gcc.a         |   920(+0) |    0(+0)  |     0(+0) |
| [lib]\misc          |   188(+0) |    4(+0)  |    28(+0) |
| [lib]\nosys.a       |    32(+0) |    0(+0)  |     0(+0) |
| [lib]\stdc++.a      |    40(+0) |    0(+0)  |    16(+0) |
| main.o              |  1676(+0) |    0(+0)  |  1228(+0) |
| mbed-os\components  |   114(+0) |    0(+0)  |     0(+0) |
| mbed-os\drivers     |  2662(+0) |    0(+0)  |  1852(+0) |
| mbed-os\events      |  1488(+0) |    0(+0)  |  3108(+0) |
| mbed-os\features    |  2178(+0) |    0(+0)  | 12796(+0) |
| mbed-os\hal         |  1696(+0) |    8(+0)  |   130(+0) |
| mbed-os\platform    |  7074(+0) |  276(+0)  |   385(+0) |
| mbed-os\rtos        | 10588(+0) |  168(+0)  |  6236(+0) |
| mbed-os\targets     | 19938(+0) |    5(+0)  |  1334(+0) |
| Subtotals           | 99096(+0) | 3040(+0)  | 27264(+0) |
Total Static RAM memory (data + bss): 30304(+0) bytes
Total Flash memory (text + data): 102136(+0) bytes
```

Рисунок 3.13 – Результат компіляції проекту

Оскільки плата STM32F767ZI має вбудований програматор схема підключення для вивантаження ПЗ є досить простою як і власне завантаження програми на мікроконтролер. На рис. 3.14 показано спосіб підключення мікроконтролера до пристрою програмування (персонального комп’ютера) даний спосіб може відрізнятись залежно від програматора, в цьому випадку з допомогою підключення через порт micro-USB.

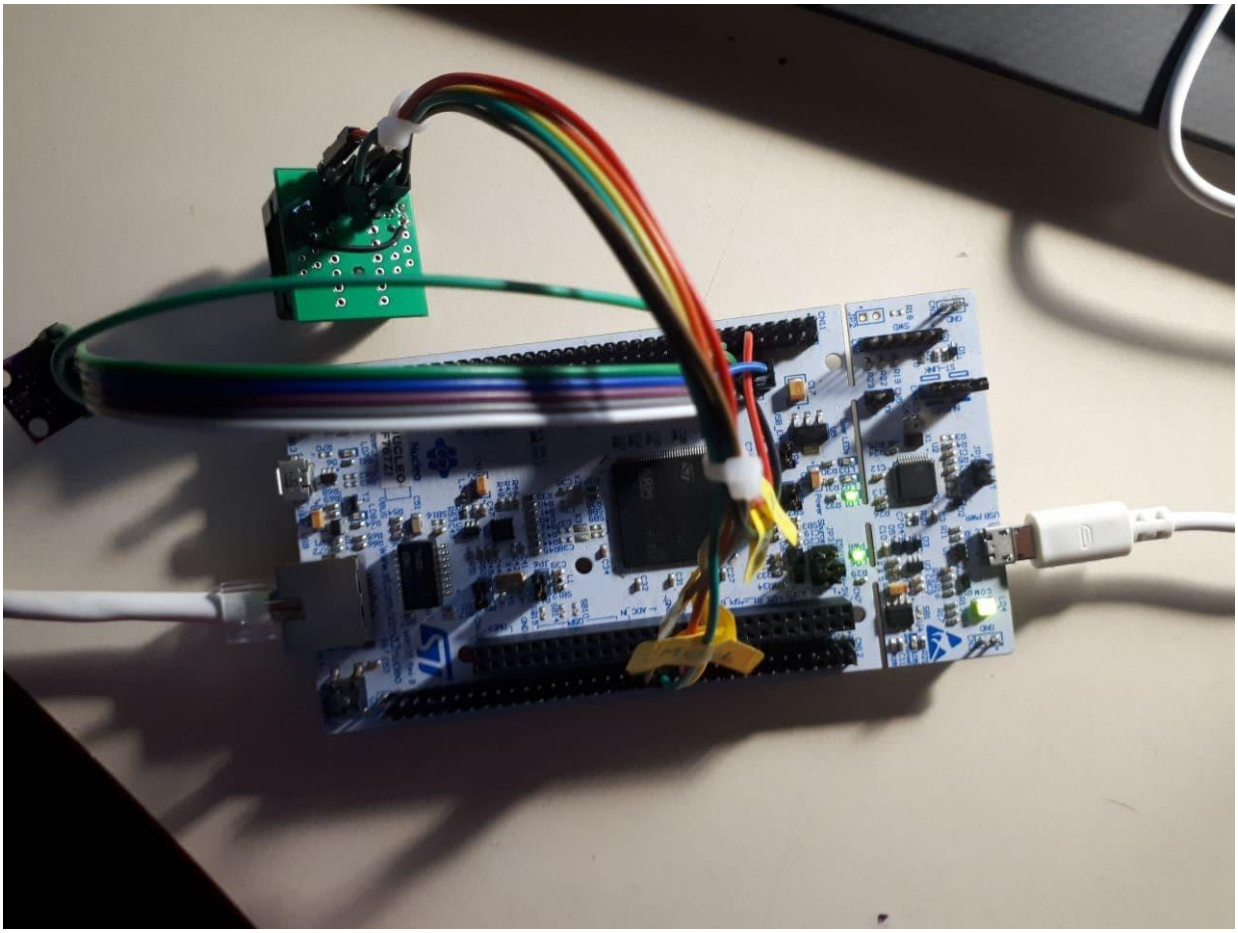

Рисунок 3.14 — Підключення STM32F767ZI

Першим методом вивантаження ПЗ буде використання Mbed-CLI, цей спосіб не підтримується для всіх плат лінійки STM. Для початку потрібно зайти в скачану консоль та використати команду `mbed compile -m NUCLEO_F767ZI - - flash -t GCC_ARM`, результат успішного вивантаження зображено на рис. 3.15.

```
Image: .\BUILD\NUCLEO_F767ZI\GCC_ARM\Kursova_v.2_Greenhouse_microclimate_monitoring_device.bin
        1 file(s) copied.
```

Рисунок 3.15 – Успішне вивантаження bin файла на мікроконтролер

Другим методом є використання ліцензійного програмного забезпечення “ST-LINK Utility” від компанії STMicroelectronics, рис. 3.16.

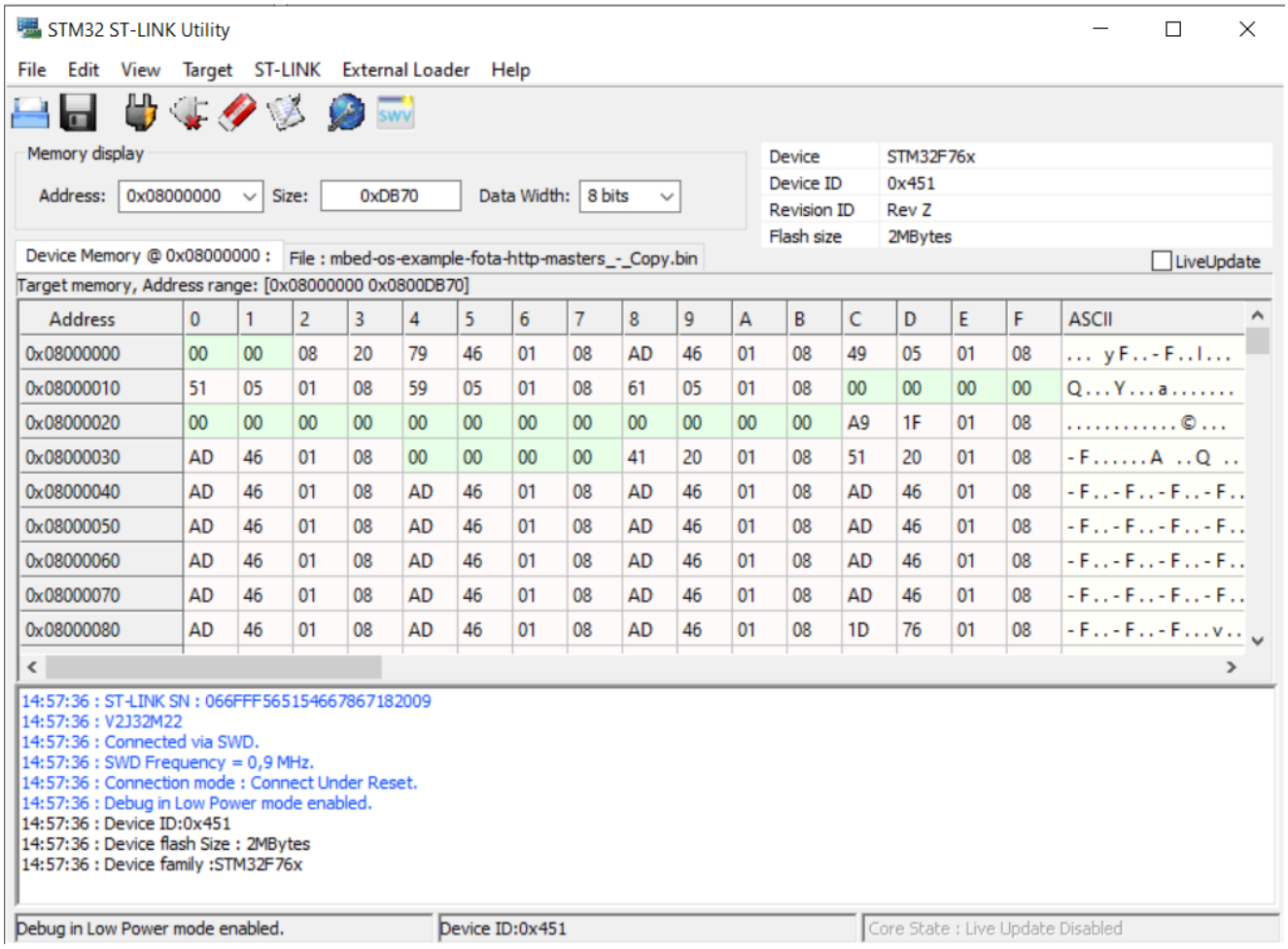

Рисунок 3.16 – Інтерфейс ST-LINK Utility

Після включення програми необхідно налаштувати режим програмування– для цього необхідно відкрити вкладку “Target -> Settings” та встановити налаштування які показано на рис. 3.17.

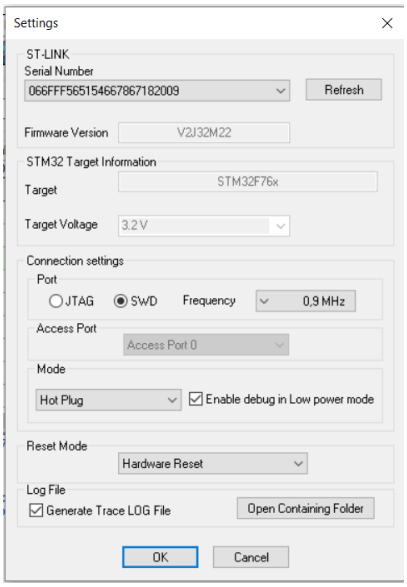

Рисунок 3.17 – Налаштування підключення плати

Після цього, перейшовши на вкладку “Program”, обрати бінарний файл прошивки та вивантажити його на плату, рис. 3.18.

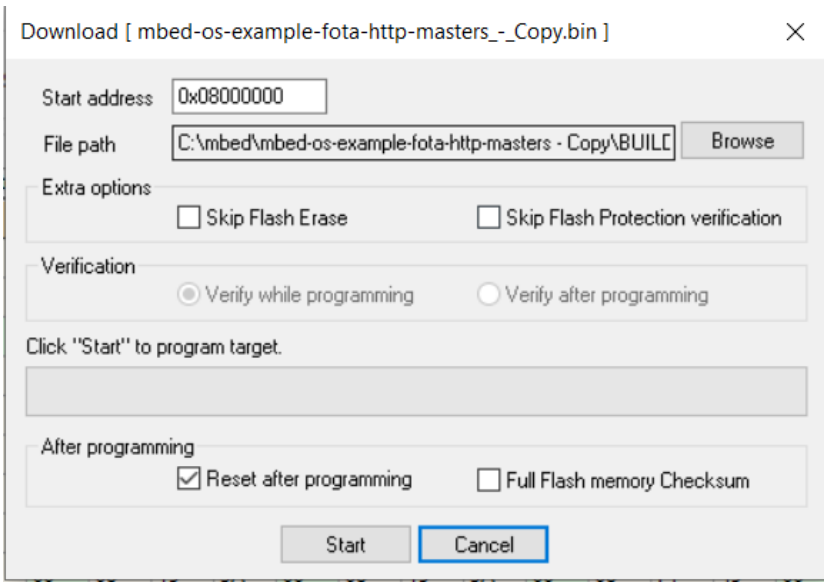

Рисунок 3.18 – Вкладка “Program”

У разі правильного виконання даної послідовності дій – процесор виконає програмний «Reset» та після буде готовим до використання. Для використання тип підключення по UART потрібно підключити плату через порт micro-USB до комп’ютера та використавши програму Docklight, або ж підключити мережевий кабель та програму Hercules.

Структура файлів після компіляції та сформований файл прошивки зображено на рис. 3.19.

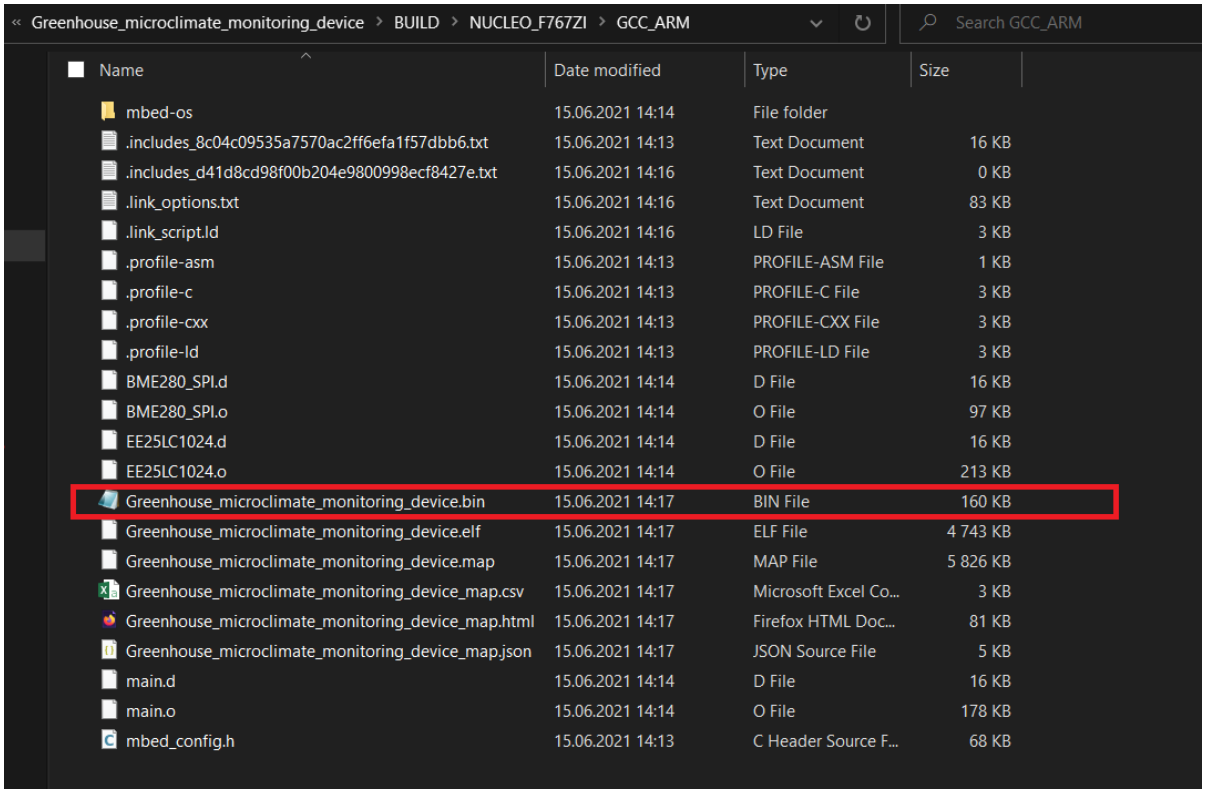

Рисунок 3.19 – Файли після компіляції

3.5 Тестування комп’ютерної системи

Тестування підключення Для тестування підключення до системи через ETHERNET потрібно підключити давач та флешку до плати, рис. 3.20.

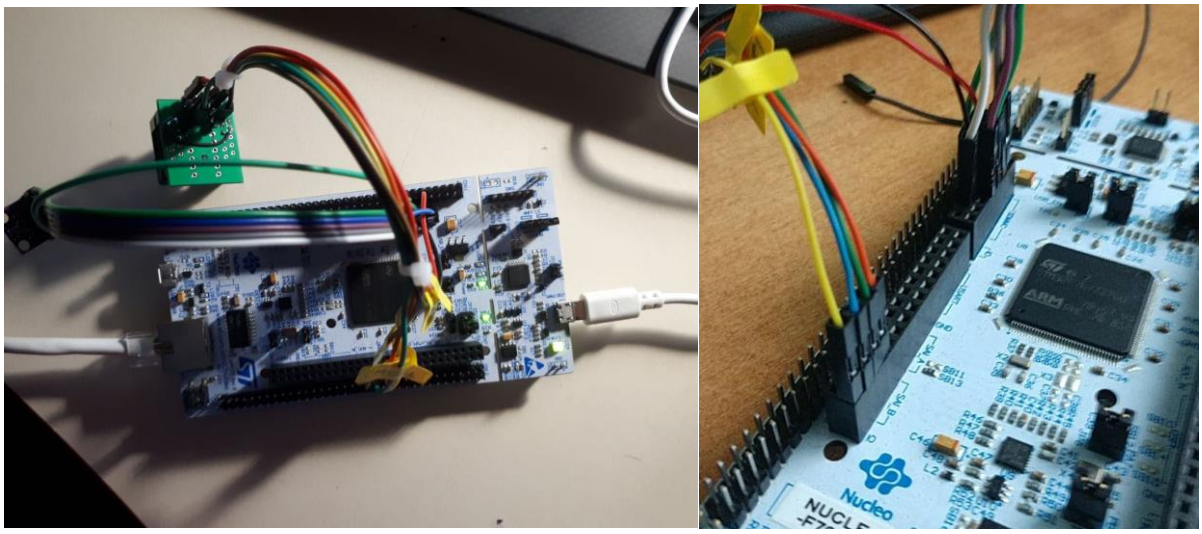

Рисунок 3.20 – Підключені елементи до плати

Для запуску програми можна використовувати надсилання команд на TCP сервер або UART порт. Для того, щоб використовувати команди для ETHERNET протоколу потрібно до плати підключити кабель виду «CAT-5» або так звану «Виту пару» в відповідний порт RJ45 плати STM32F767ZI, з доступом до мережі. Після даних дій потрібно запустити термінал клієнтського сервера TCP/IP. В даному випадку використовується відрите програмне забезпечення Hercules SETUP Utility, рис. 3.21. Утиліта Hercules SETUP – це корисний термінал послідовного порту (термінал RS-485 або RS-232), термінал UDP/IP та термінал TCP/IP клієнтського сервера.

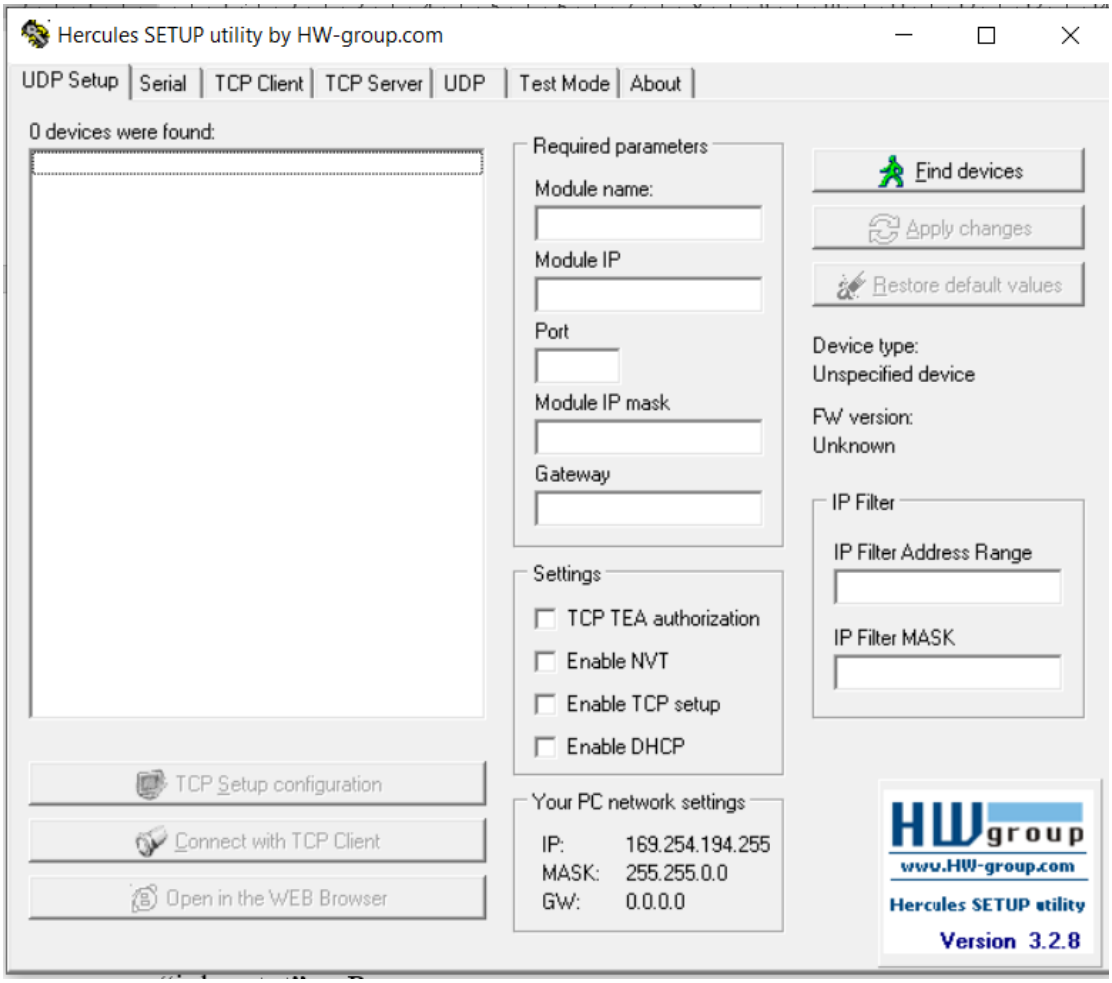

Рисунок 3.21 – Інтерфейс Hercules SETUP Utility

Після цих дій потрібно натиснути кнопку “Reset” на платі і, зайшовши в термінал послідовного порту, можна буде побачити повідомлення з IP адресою для підключення, рис. 3.22.

```
13.12.2020 15:29:49.569 [RX] - The target IP address is '192.168.0.107'<CR><LF>
waiting for client<CR><LF>
```

Рисунок 3.22 – Повідомлення з IP адресою

В програмі Hercules потрібно зайти на вкладку TCP Client та ввести отриману IP адресу та порт для підключення, рис. 3.23.

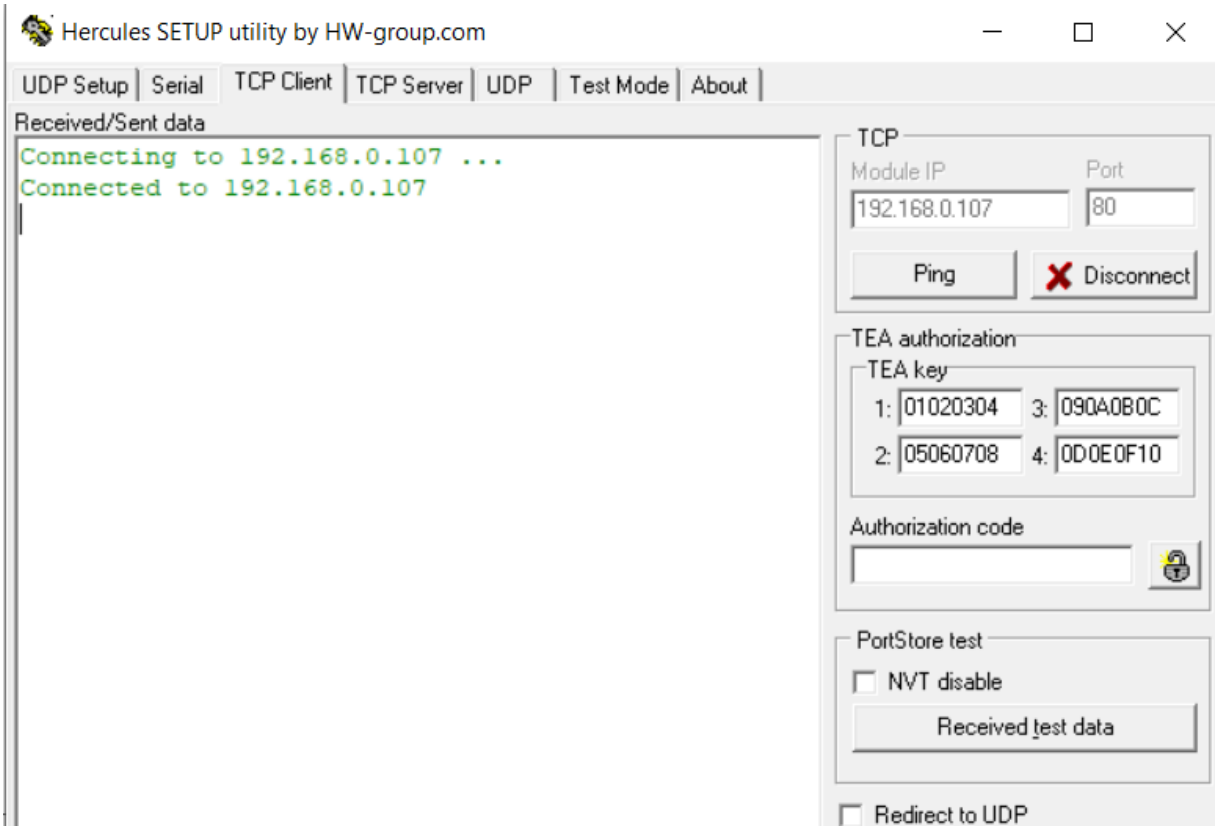

Рисунок 3.23 – Підключення до TCP сервера

Інформація про підключеного клієнта яка виводиться зображена на рис. 3.24.

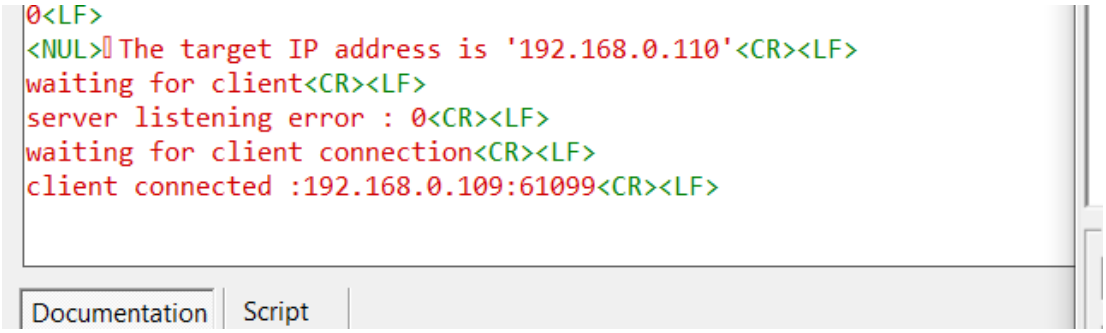

Рисунок 3.24 – Інформація про підключеного користувача

Перевірка системи виведенням температури в Цельсіях та Фаренгейтах, рис. 3.25 та 3.26.

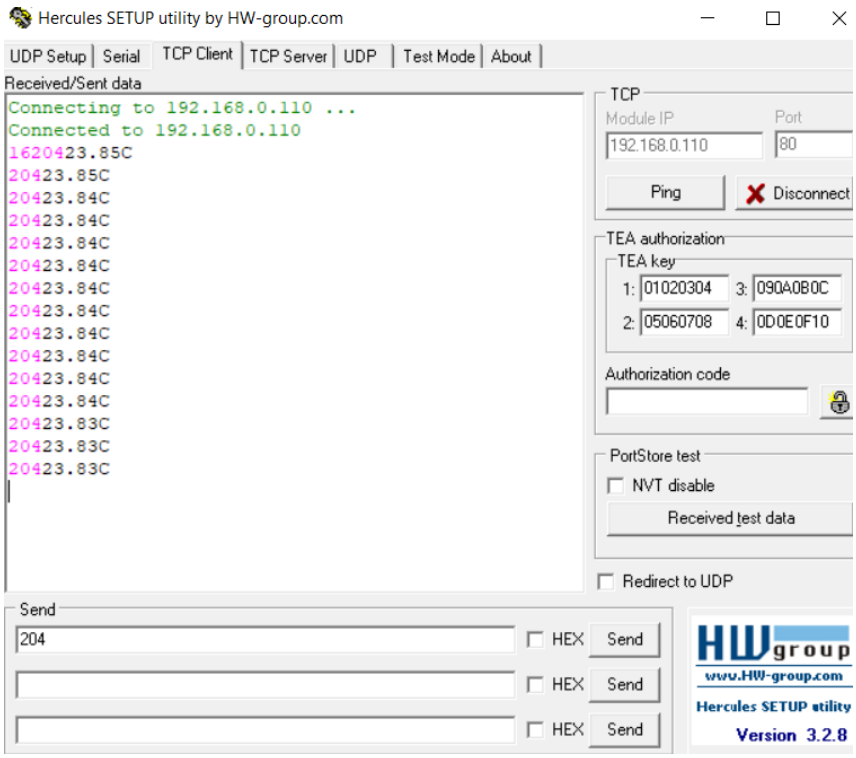

Рисунок 3.25 – Виведення інформації про температуру в Цельсіях

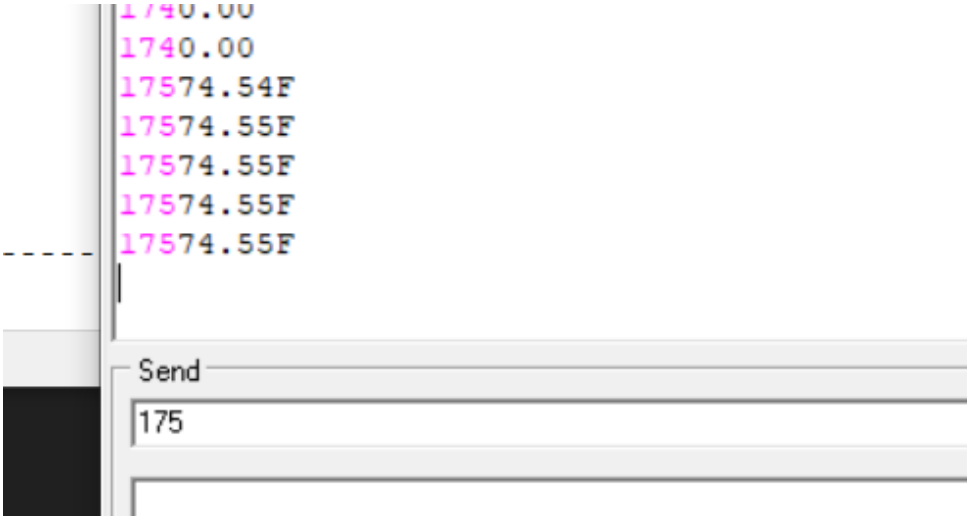

Рисунок 3.26 – Виведення інформації про температуру в Фаренгейтах

Виведення інформації, яка записана на флешці зображено на рис. 3.27.

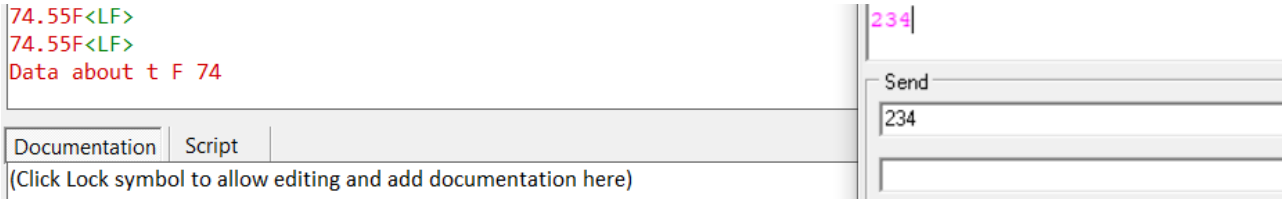

Рисунок 3.27 – Виведення інформації яка записана на флешці

Вимкнути систему можна з допомогою команди 0x10, рис. 3.28.

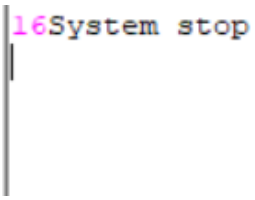

Рисунок 3.28 – Припинення роботи системи

Таким шляхом, надсилаючи відповідні команди протоколу, можна отримувати інформацію з TCP сервера на платі.

Другим методом підключення до даної системи – з допомогою команд на послідовий порт або ж UART. Для даного способу буде використано програмне забезпечення Docklight, рис. 3.29.

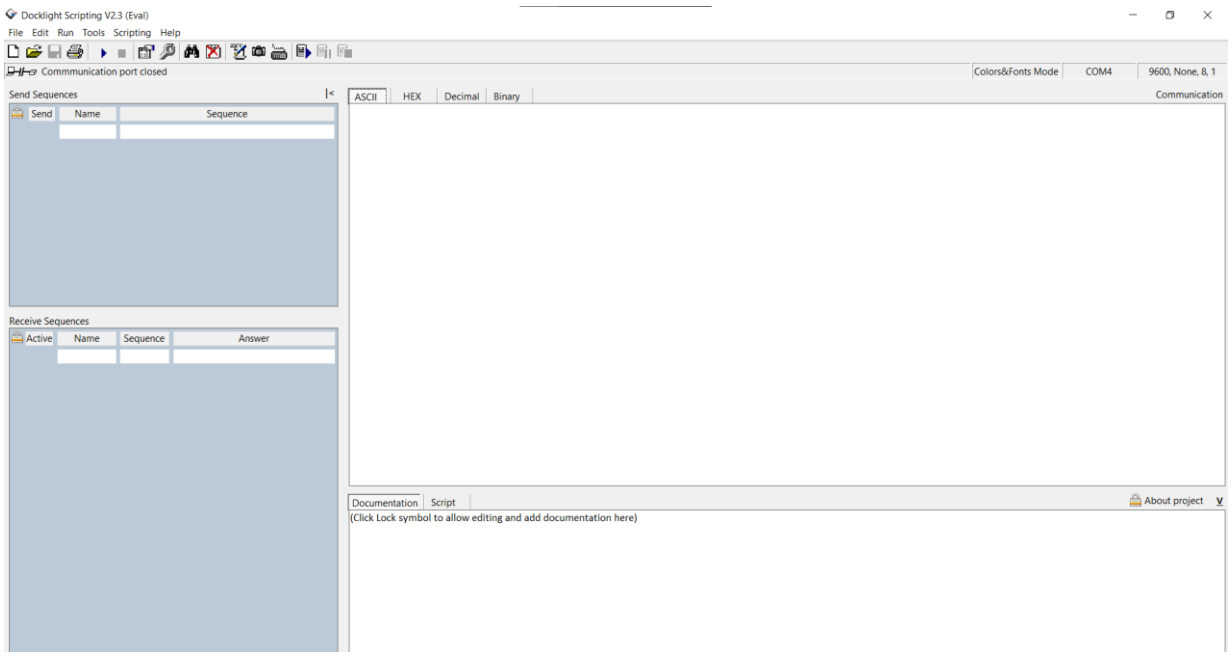

Рисунок 3.29 – Інтерфейс програми Docklight

Для того щоб «слухати» відповідний порт потрібно задати налаштування, які зображено на рис. 3.30.

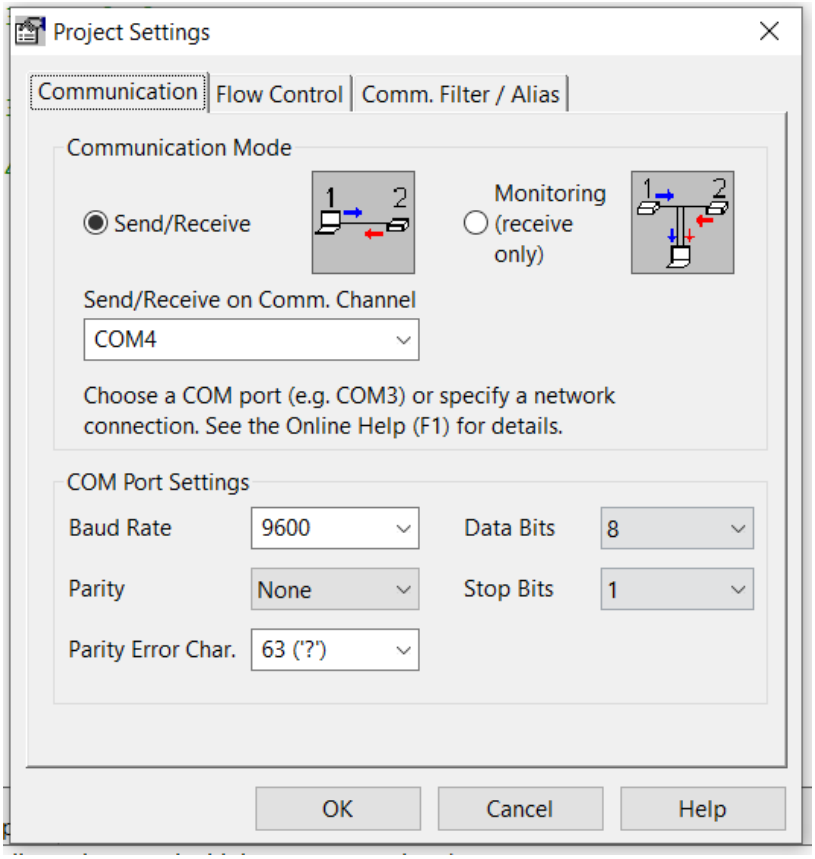

Рисунок 3.30 – Базові налаштування

Після цього на панелі інструментів активувати кнопку, яка зображена на рис. 3.31, для того щоб ми могли щось надсилати в терміналі.

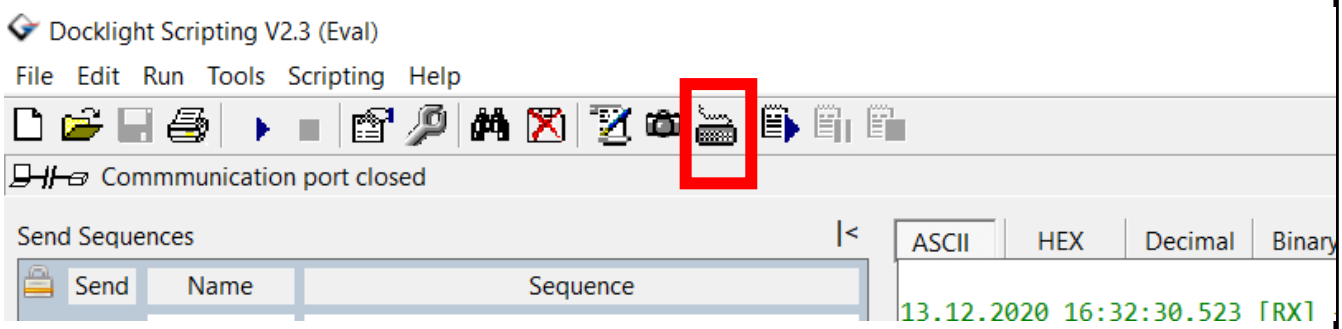

Рисунок 3.31 – Кнопка активації консольної клавіатури

Тепер надсилаючи команди можна «спілкуватись» з платою з допомогою UART, рис. 3.32.

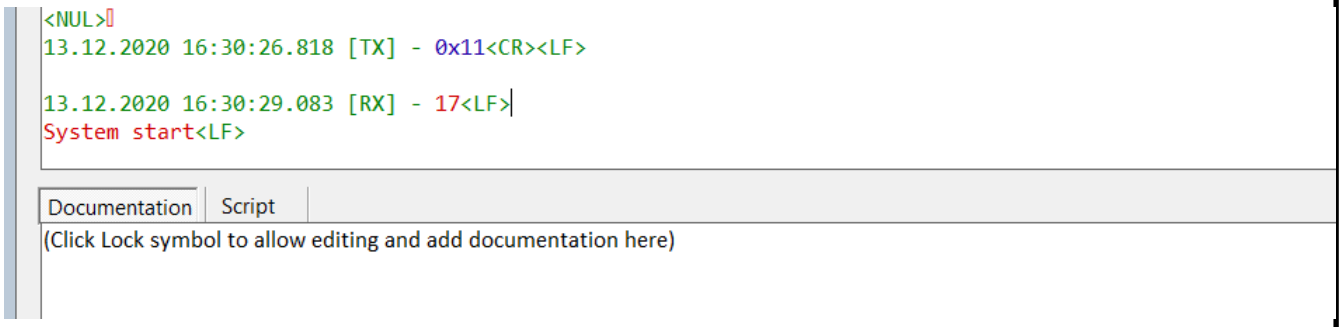

Рисунок 3.32 – Початок роботи з системою

На рис. 3.33 зображено реакцію системи на команди виведення інформації про температуру.

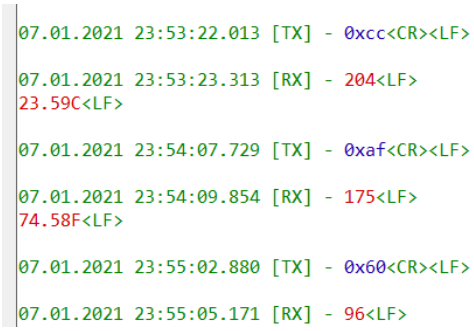

Рисунок 3.33 – Виведення інформації про температуру з допомогою команд

Виведення інформації, яка збережена на флешці зображена на рис. 3.34.

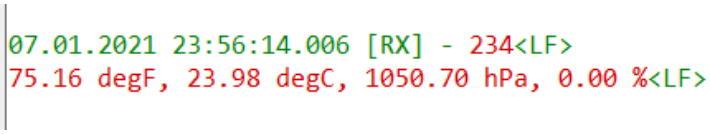

Рисунок 3.34 – Вся інформація яка збережена на флешці

Процес очищення флешки зображено на рис. 3.35.

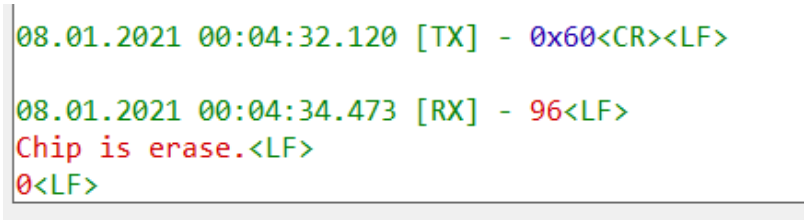

Рисунок 3.35 – Процес очищення флешки

На рис. 3.36 зображено виведення інформації тільки про температуру.

```
08.01.2021 00:03:42.778 [TX] - 0xf5<CR><LF>

08.01.2021 00:03:45.052 [RX] - 245<LF>
Data about t C 23<LF>
```

Рисунок 3.36 – Виведення інформації тільки про температуру

На рис. 3.37 зображено припинення роботи системи з допомогою надсилання команди 0x10, протокол для обох методів є ідентичним.

```
13.12.2020 16:32:39.306 [TX] - 0x10<CR><LF>

13.12.2020 16:32:41.556 [RX] - 16<LF>
System stop<LF>
```

Рисунок 3.37 – Припинення роботи системи

Таким шляхом, надсилаючи відповідні команди протоколу, можна отримувати інформацію з TCP сервера на платі.

Для тестування та налагодження було використано логічний аналізатор Saleae Logic 16 – пристрій для відображення послідовності цифрових сигналів.

Передача пакетів по шині SPI зображена на рис. 3.38.

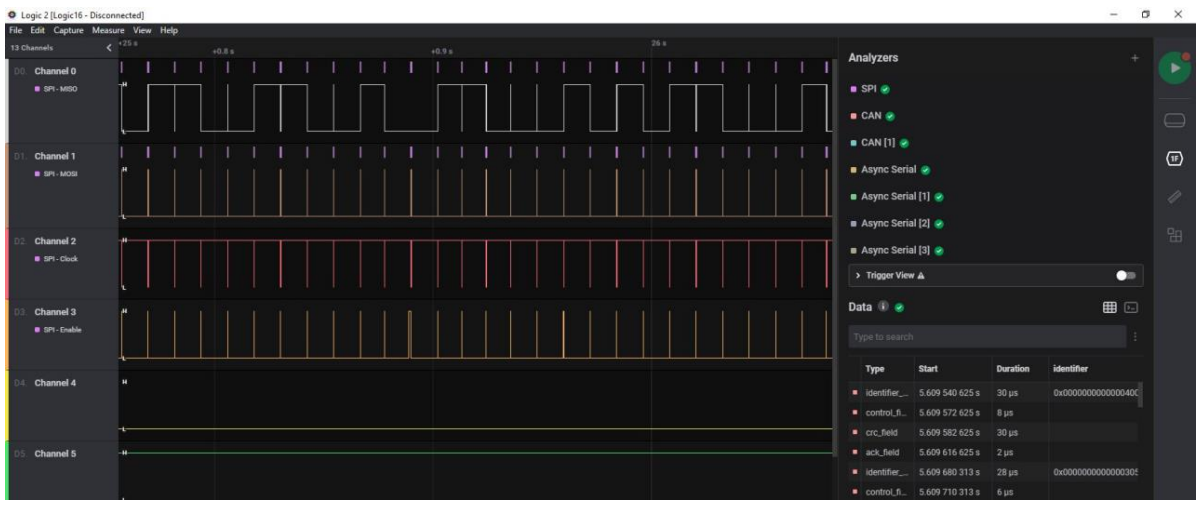

Рисунок 3.38 – Передача даних по шині SPI

Передача та розбір одного пакету даних зображено на рис. 3.39. Оскільки в системі, яка розробляється, давач та модуль зовнішньої пам'яті використовують SPI шину, як основну для забезпечення своєї роботи та комунікації з STM32F767ZI. На рисунку помітно, що перед передачею пін CS стає в одиницю, що свідчить про те що зараз будуть передаватися дані. Після того відбувається надсилання пакету з допомогою MISO, MOSI.

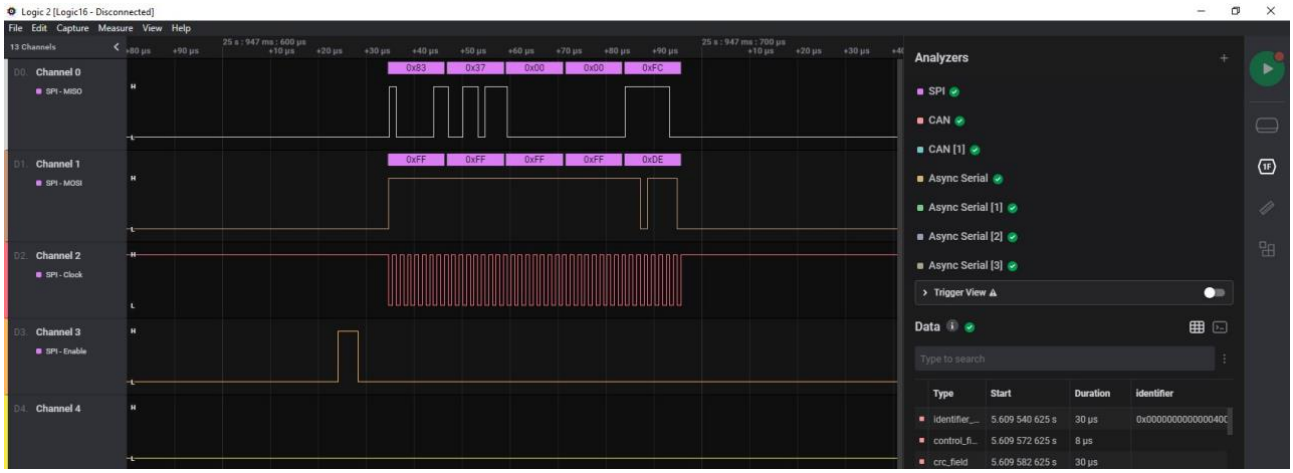

Рисунок 3.39 – Пакет даних який передається по шині SPI

Передача пакетів даних по шині UART зображена на рис. 3.40, а саме надсилання відповіді на запит користувача.

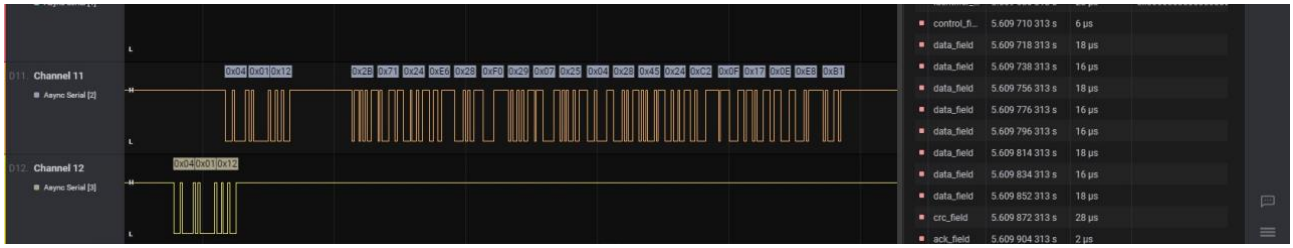

Рисунок 3.40 – Передача даних по шині UART

Передача пакетів даних відбувається відповідно до принципів роботи протоколів SPI та UART, помилки в пакетах відсутні отже вбудована система функціонує згідно ТЗ.

РОЗДІЛ 4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Заходи щодо захисту установки від короткого замикання

Коротке замикання - електричне з'єднання двох точок електричного кола з різними значеннями потенціалу, не передбачене конструкцією пристрою і порушує його нормальну роботу. Коротке замикання може виникати при порушенні ізоляції струмоведучих елементів або внаслідок механічного зіткнення елементів, що працюють без ізоляції. Також коротким замиканням називають стан, коли опір навантаження менше внутрішнього опору джерела живлення [22].

Види коротких замикань. У трифазних електричних мережах розрізняють такі види коротких замикань:

— однофазне (замикання фази на землю);

— двофазне (замикання двох фаз між собою);

— двофазне на землю (2 фази між собою і одночасно на землю);

— трифазне (3 фази між собою).

В електричних машинах можливі короткі замикання:

— міжвиткове - замикання між собою витків обмотки ротора або статора;

— замикання обмотки на металевий корпус.

Методи захисту. Для захисту від короткого замикання приймають спеціальні заходи [23]:

Обмежують струм короткого замикання:

— встановлюють струмообмежуючі електричні реактори;

— застосовують розпаралелення електричних кіл тобто відключення секційних і шиноз’єднувальних вимикачів;

— використовують знижувальні трансформатори з розщепленою обмоткою низької напруги;

— використовують обладнання автоматичного вимкнення – швидкодіючий комутаційний апарат з функцією обмеження струму короткого замикання тобто плавкі запобіжники, автоматичні вимикачі [22];

— застосовують пристрої релейного захисту для відключення пошкоджених ділянок кола;

Реактор являє собою котушку з індуктивним опором. Він підключений в коло по послідовній схемі. При нормальній роботі на реакторі є падіння напруги близько 4%. У разі виникнення КЗ основна частина напруги припадає на реактор. Існує кілька видів реакторів: бетонні, масляні [22].

Основною причиною виникнення коротких замикань є порушення ізоляції електрообладнання.

Порушення ізоляції викликаються:

— перенапруженнями (особливо в мережах з ізольованими нейтралями);

— прямими ударами блискавки;

— старінням ізоляції;

— механічними пошкодженнями ізоляції, проїздом під лініями негабаритних механізмів;

— незадовільним доглядом за обладнанням.

Часто причиною пошкоджень в електричній частині електроустановок є некваліфіковані дії обслуговуючого персоналу.

При використанні спрощених схем з'єднань понижуючих підстанцій використовують спеціальні апарати - короткозамикачі, які створюють навмисні короткі замикання з метою швидких відключень виникли ушкоджень. Таким чином, поряд з короткими замиканнями випадкового характеру в системах електропостачання мають місце також навмисні короткі замикання, що викликаються дією короткозамикачів.

При виникненні коротких замикань в системі електропостачання її загальний опір зменшується, що призводить до збільшення струмів в її гілках в

порівнянні з струмами нормального режиму, а це викликає зниження напруги окремих точок системи електропостачання, яке особливо велике поблизу місця короткого замикання.

Залежно від місця виникнення і тривалості ушкодження його наслідки можуть мати місцевий характер або відбиватися на всій системі електропостачання.

При великій віддаленості короткого замикання величина струму короткого замикання може становити лише незначну частину номінального струму живлення генераторів і виникнення такого короткого замикання сприймається ними як невелике збільшення навантаження. Сильне зниження напруги виходить тільки поблизу місця короткого замикання, в той час як в інших точках системи електропостачання це зниження менш помітно. При розглянутих умовах небезпечні наслідки короткого замикання проявляються лише в найближчих до місця аварії частинах системи електропостачання.

Струм короткого замикання, будучи навіть малим у порівнянні з номінальним струмом генераторів, зазвичай у багато разів перевищує номінальний струм гілки, де сталося коротке замикання. Тому і при короткочасному протіканні струму короткого замикання він може викликати додатковий нагрів струмоведучих елементів і провідників вище допустимого.

Струми короткого замикання викликають між провідниками великі механічні зусилля, які особливо великі на початку процесу короткого замикання, коли струм досягає максимального значення. При недостатній міцності провідників і їх кріплень можуть мати місце руйнування механічного характеру.

Раптове глибоке зниження напруги при короткому замиканні відбивається на роботі споживачів. В першу чергу це стосується двигунів, тому що навіть при короткочасному зниженні напруги на 30-40% вони можуть зупинитися (відбувається перекидання двигунів). [23] Перекидання двигунів важко відбивається на роботі промислового підприємства, так як для відновлення нормального виробничого процесу потрібен тривалий час і несподівана зупинка двигунів може викликати брак продукції підприємства.

При малої віддаленості і достатньої тривалості короткого замикання можливе випадання з синхронізму паралельно працюючих станцій, тобто порушення нормальної роботи всієї електричної системи, що є найнебезпечнішим наслідком короткого замикання.

Виникаючі при замиканнях на землю неврівноважені системи струмів здатні створити магнітні потоки, достатні для наведення в сусідніх колах (лініях зв'язку, трубопроводах) значних ЕРС, небезпечних для обслуговуючого персоналу та апаратури цих кіл [23].

Таким чином, наслідки коротких замикань наступні:

— механічні та термічні пошкодження електрообладнання;

— займання в електроустановках;

— зниження рівня напруги в мережі, що веде до зменшення обертального моменту електродвигунів, їх гальмування, зниження продуктивності або навіть до перекидання їх;

— випадання із синхронізму окремих генераторів, електростанцій і частин електричної системи і виникнення аварій, включаючи системні аварії;

— електромагнітний вплив на лінії зв'язку, комунікації.

4.2 Фізіологічний вплив факторів існування на життєдіяльність людини

Норми мікроклімату визначенні постановою ДСН 3.3.6.042-99 Міністерства охорони здоров'я України «Санітарні норми мікроклімату виробничих приміщень» в поточній редакції прийнятій 01.12.1999 р. Дана постанова визначає мікроклімат приміщень як «стан внутрішнього середовища приміщення, що впливає на людину» [1]. Внутрішнє середовище – це, в більшості випадків, повітря всередині приміщення. Мікроклімат приміщень характеризується головним чином температурою, вологістю та тиском [21]. Цей документ встановлює оптимальні і допустимі значення температури, відносної вологості та швидкості руху повітря, допустиму температуру внутрішніх поверхонь приміщення (стіни, стеля, підлоги) і зовнішніх поверхонь

технологічного обладнання, а також допустиму інтенсивність теплового випромінювання нагрітих поверхонь у приміщенні та відкритих джерел тепла (нагрітий метал, скло, відкритий вогонь тощо) для робочої зони – визначеного простору, в якому знаходяться робочі місця постійного або непостійного (тимчасового) перебування працівників.

Виробниче приміщення – це замкнутий простір в спеціально призначених будинках та спорудах, в яких постійно (по змінах) або періодично (протягом частини робочого дня) здійснюється трудова діяльність людей [1].

Мікроклімат виробничих приміщень – умови внутрішнього середовища цих приміщень, що впливають на тепловий обмін працівників з оточенням шляхом конвекції, кондукції, теплового випромінювання та випаровування вологи. Ці умови визначаються поєднанням температури, відносної вологості та швидкості руху повітря, температури навколишніх поверхонь та інтенсивністю теплового (інфрачервоного) опромінення.

Оптимальні мікрокліматичні умови – поєднання параметрів мікроклімату, які при тривалому та систематичному впливі на людину забезпечують зберігання нормального теплового стану організму без активізації механізмів терморегуляції. Вони забезпечують відчуття теплового комфорту та створюють передумови для високого рівня працездатності [23].

Мікроклімат має прямий вплив на людину. Якщо параметри відповідають нормі («оптимальні», як вказано в ДСН), то людина відчуває себе комфортно, і організм не витрачає ресурс на пристосування до зовнішніх умов [21].

Мікроклімат житлових та громадських будівель складається з багатьох параметрів, але пріоритетом буде:

— температура повітря;

— вологість повітря;

— тиск.

Пристрій керування параметрами мікроклімату повністю враховує межі параметрів мікроклімату визначені в ДСН 3.3.6.042-99 Міністерства охорони здоров'я України «Санітарні норми мікроклімату виробничих приміщень».

ВИСНОВКИ

У кваліфікаційній роботі було розроблено систему керування параметрами мікроклімату, використовуючи плату відлагодження сімейства STM32 Nucleo 144 – STM32F767ZI NUCLEO, високоточний цифровий вимірювач атмосферного тиску, температури та вологості – BMP280 та програмований ПЗП, який може електрично очищуватись, один з видів незалежної пам'яті – eEPROM W25Q32.

Програмне забезпечення системи побудовано на базі ядра MBED OS – платформи та операційної системи для пристроїв Internet of Things, підключених до бази 32-розрядних мікроконтролерів ARM Cortex-M, спільного проекту ARM та його партнерів. Програмно було реалізовано роботу з TCP Server та UART з використання прямого доступу до пам’яті (DMA) для забезпечення взаємодії користувача та системи відповідно до розробленого на основі комунікаційних протоколів UART та TCP набору правил.

Детально дослідивши принцип роботи W25Q32 було розроблено спеціалізовану бібліотеку, яка дає можливість пришвидшити процес запису даних на ПЗП. Також було модернізовано базову бібліотеку для роботи з BPM280.

Розроблена комп’ютерна система була зібрана у вигляді макету на базі плати STM32F767ZI NUCLEO, всі функції працюють справно та система повністю відповідає вимогам ТЗ.

Розроблена комп’ютерна система придатна до використання в реальних умовах експлуатації для моніторингу параметрів мікроклімату приміщення.

БІБЛІОГРАФІЯ

1. ДСН 3.3.6.039-99: N 39 від 01.12.1999 . Київ: Міністерство охорони здоров'я України. 26 с.
2. Architecting High-Performance Embedded Systems: Design and build high-performance real-time digital systems based on FPGAs and custom circuits. Birmingham: Packt Publishing. 376 с.
3. Invidia L. , Palmieri A. , Patrono L. An IoT-oriented Fast Prototyping Platform for BLE-based Star Topology Networks // Наук. стаття / JOURNAL OF COMMUNICATIONS SOFTWARE AND SYSTEMS. Online .№ 2. С. 138-149.
4. Martin T. The Insider's Guide To The STM32 ARM Based Microcontroller // Ebook. URL: https://www.twirpx.com/file/2087462/ (дата звернення: 08.02.2021).
5. Kurniawan A. 2.3 STM32 Driver and ST-LINK/V2 // MicroPython for STM32 Nucleo Technical Workshop / за ред. PE Press. California. С. 125.
6. Назаревич Т. О., Лещишин Ю. З., Міська І. В. Створення вбудованих систем на базі структурно - параметричних моделей цифрових каналів зв'язку // Збірник наукових праць / VIII Науково-технічна конференція «Інформаційні моделі, системи та технології». Тернопіль, 2020. С. 127.
7. Розробка системи зв'язку як інтегрованого елементу роботизованих систем / Лещишин Ю. З., Н.Р. Романишин, В.В. Наконечний, А.О. Паламарчук // Збірник тез доповідей XXI Всеукраїнської науково-практичної конференції / Проблеми створення, розвитку та застосування високотехнологічних систем спеціального призначення з урахуванням досвіду антитерористичної операції. Житомир, 2016. С. 102.
8. R. Rizvi S. Microcontroller Programming . Florida: CRC Press , 2011. 215 с.
9. Ebook Microprocessor Design Principles and Practices With VHDL: Веб-сайт.. URL: http://ebook.pldworld.com/_eBook/FPGA%EF%BC%8FHDL/Micropro

cessor%20Design%20principles%20and%20practices%20with%20Vhdl.pdf (дата звернення: 02.02.2021).

10. Pakdel M. Advanced Programming with STM32 Microcontrollers. London: Elektor, 2020. 212 с.

11. Noviello C. A step-by-step guide to the most complete ARM Cortex-M platform, using a free and powerful development environment based on Eclipse and GCC. Leanpub book: Leanpub, 2016. 831 с.

12. Kurni A. Getting Started With STM32 Nucleo Development. Depok: PE Press; 1st edition (April 17, 2015), 2016. 80 с.

13. Brown G. Discovering the STM32 Microcontroller: eBook / за ред. Indiana University. Indiana, 2016. 240 с.

14. ArmMBED: Веб-сайт.. URL: https://os.mbed.com/docs/mbed-os/v6.11/build-tools/mbed-cli-1.html (дата звернення: 21.01.2021).

15. ST: Веб-сайт.. URL: https://www.st.com/resource/en/datasheet/stm32f765zi.pdf (дата звернення: 21.01.2021).

16. ST: Веб-сайт.. URL: https://www.st.com/en/microcontrollers-microprocessors/stm32f767zi.html (дата звернення: 20.01.2021).

17. Adafruit.com: Веб-сайт.. URL: https://cdn-shop.adafruit.com/datasheets/BST-BMP280-DS001-11.pdf (дата звернення: 20.01.2021).

18. ArmMBED: Веб-сайт.. URL: https://os.mbed.com/docs/mbed-os/v6.11/apis/tcpsocket.html (дата звернення: 11.01.2021).

19. ArmDeveloper: Веб-сайт.. URL: https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain (дата звернення: 17.01.2021).

20. An introduction to Arm Mbed OS 6 : Веб-сайт.. URL: https://os.mbed.com/docs/mbed-os/v6.6/introduction/index.html (дата звернення: 16.01.2021).

21. Гурик О. Я., Король О. І., Сенчишин В. С. Методичні вказівки до лабораторної роботи №2 з дисципліни :”Основи охорони праці” ”Дослідження метеорологічних умов у виробничих приміщеннях” . Тернопіль, 2016. 35 с.

22. Гурик О. Я., Король О. І., Сенчишин В. С. Методичні вказівки до лабораторної роботи №1 з дисципліни ”Основи охорони праці” ” Дослідження характеристик плавких вставок для запобіжників”. Тернопіль, 2015. 25 с.

23. Гурик О. Я., Король О. І., Сенчишин В. С. Методичні вказівки до лабораторної роботи з дисципліни ”Основи охорони праці”. Тернопіль, 2006. 17 с.

ДОДАТОК А

Технічне завдання

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії

Кафедра комп'ютерних систем та мереж

"Затверджую"

Завідувач кафедри КС _____ Осухівська Г.М.

"___" _____2021 р

КОМП'ЮТЕРИЗОВАНА СИСТЕМА КОНТРОЛЮ ПАРАМЕТРІВ МІКРОКЛІМАТУ ПРИМІЩЕНЬ

ТЕХНІЧНЕ ЗАВДАННЯ на __ листках

На здобуття освітньо-кваліфікаційного рівня *бакалавр*

Напрям *123 Комп'ютерна інженерія*

Спеціальність *123 Комп'ютерна інженерія*

«УЗГОДЖЕНО»	«ВИКОНАВЕЦЬ»
Керівник кваліфікаційної роботи	Студентка групи СІс-44
_____к.т.н., доцент	_____ Ольховецька Х.А.
Лещишин Ю.З.	«____» _____ 2021 р
«____» _____ 2021 р	

Тернопіль 2021

1. Назва та підстава для виконання роботи

1.1. Комп’ютеризована система контролю параметрів мікроклімату приміщень.

1.2. Підставою для виконання кваліфікаційної роботи бакалавра (КРБ) є Наказ по Університету (№ 4/7-59 від 01.02.2021 р.).

2. Виконавець

2.1. Студентки групи СІс-44 кафедри КС
Тернопільського національного технічного університету ім. І. Пулюя
Ольховецька Христина Андріївна.

3. Мета роботи

3.1. Метою роботи є розробити структуру та програмне забезпечення комп’ютеризованої системи контролю параметрів мікроклімату приміщень.

4. Склад виробу

4.1. До складу КС повинні входити:

1) давач;
2) зовнішній модуль пам'яті;
3) плата мікроконтролера з можливістю підключення ETHERNET;
4) елемент охолодження (вентилятор);
5) комплект документації.

5. Технічні вимоги

5.1. Вимоги по призначенню.

5.1.1. Комп’ютеризована система повинна мати наступні параметри:

– Діапазон вимірюваної температури, °С +10…+50

– Точність вимірювання температури, °С ±0,1

– Точність вимірювання тиску, hPa ± 0.12

5.1.2. Швидкість обміну даними по повітрю, не менше кбіт/с 1200

5.1.3. Виріб повинен живитись напругою постійного струму, В +12±2

5.2. Вимоги до умов експлуатації:

5.2.1. По умовам експлуатації виріб повинен відповідати вимогам ГОСТ 15150 для УХЛ4.1

5.2.2. Температура експлуатації від 0 до +40°С

5.2.3. Відносна вологість до 100% при t=25°С

5.3. Конструктивні вимоги.

5.3.1. Конструювання корпусу приладу в КРБ не передбачено.

5.3.2. Для побудови системи мають бути використані сучасні компоненти з можливістю поверхневого монтажу друкованого вузла.

5.3.3. При побудові системи необхідно передбачити розміщення роз’ємів живлення і обміну даними.

5.3.4. Габаритні розміри при макетуванні, мм, не більше:

довжина 400

ширина 300

висота 100

5.3.5. Маса макету, кг, не більше 2

5.3.6. Конструкція макету повинна забезпечувати доступ до всіх комплектуючих виробів при тестуванні.

5.4. Вимоги до надійності.

5.4.1. Система повинна відповідати вимогам ДСТУ 2862-94.

5.4.2. Наробка на відмову, не менше 5000 год.

5.5. Вимоги метрології.

5.5.1. Вимірювання параметрів системи при моделюванні повинно виконуватись на універсальних вимірювальних приладах.

6. Економічні показники

6.1. Собівартість системи повинна бути не більше 5000 грн.

7. Вимоги до документації

7.1. Конструкторська документація повинна відповідати вимогам ЄСКД, ДСТУ та ГОСТ.

7.2. До складу документації повинно входити:

1) ПЗ;
2) Електрична схема Е3;
3) Блок схеми алгоритму роботи потоків;
4) Блок схема алгоритму роботи КС;
5) Діаграма потоків даних.

8. Стадії та етапи розробки КРБ

8.1 Стадії та етапи виконання КРБ наведенні в табл. 1.

Таблиця 1 – Стадії та етапи виконання КРБ

	Назва етапу	Строк виконання	
		початок	кінець
1.	Технічне завдання	—	до 26.03.21
2.	Розділ 1	26.03. 21	10.06. 21
3.	Розділ 2	28.03. 21	10.05. 21
4.	Розділ 3	02.04. 21	13.04. 21
5.	Розділ охорона праці	16.04. 21	27.04. 21
6.	Нормоконтроль	21.05. 21	11.06. 21
7.	Попередній захист	11.06. 21	18.06.21
8.	Захист	з 24.06. 21	—

9. В дане ТЗ можуть вноситись зміни по узгодженню сторін.

ДОДАТОК Б.

Перелік елементів

Позначення	Найменування	Кількість	Примітка
	Конденсатори		
C1,C2	104	2	
C3,C4,C10,C11	0.1uF 630V ±10% CL21	4	
C9,C10	16pF 50V ±5% NPO	2	
C7	100pF 5kV ±10% *Y5P*	1	
C2	4.7mF 25V ±20%	1	
C6,C4,C5,C2,C1	100nF ±20% 50V Y5V	5	
	Роз'єми		
J2	Boot-mode jumper	1	
J3	USART1	1	
J4	RJ45+MAG	1	
J5	USART6	1	
K1	Кнопка	1	
LED1, LED2	Діоди	2	
X1	Карцовий резонатор 16МГц - 5032	1	
	Резистори		
R2	MF 47 1W 47Ом 5%	1	
R5	MF 68 2W 68 Ом 5%	1	
R7,R9	MF 220 1 W 220Ом 5%	2	
R10	MF 6.49K 0,125W 6,49кОм ±1%	1	
R6,R8,R11	MF 0,25W 4.7k 1%	3	
R12,R13	MF 27R 2W 27Ом 5 %	2	
R14	MF 4K7 125W 4.7 кОм ± 1%	1	
R15,R22,R23,R24	MF 0,25W 10K 1%	4	
R18,R19	MF 330R 2W 330 Ом 5%	2	
R16,R17,R20,R21	MF 1K 0,5W 1K 5%	4	
R1,R25	MF 0,25W 10K 1%.	2	
R3,R4,R26,R27	MF 0,25W 4.7k 1%.	4	
	Мікросхеми		
U1.1	STM32F76XZXTSTM32F76XZXT	1	
U1.2	STM32F76XZXTSTM32F76XZXT	1	
U2	KSZ8041NL-AM	1	
U3	FT201XS-R	1	
U4	W25QXX	2	
U5	BMP280	1	

ДОДАТОК В

Лістинги програмного забезпечення КС

Лістинг файла Main.cpp

```
#include "mbed.h"
#include "rtos.h"
#include "EthernetInterface.h"
#include "TCPSocket.h"
#include "TCPServer.h"
#include "readUARt.h"
#include "BME280_SPI.h"
#include "EE25LC1024.h"
// EE25LC1024 eEPROM(PF_9,PF_8,PF_7,PG_1);
// EE25LC1024 eEPROM(PD_7,PG_9,PG_11,PD_9);

#include <stdio.h>
#include <string.h>

#define CODE_1 0xcc
#define CODE_2 0xaf
#define CODE_3 0xae //- 174 - Вологість
#define CODE_4 0xac //- 172 - Тиск
#define CODE_5 0xea //- 234 - Всі показники

#define CODE_6 0xf1 //- 241 - Ввімкнення вентилятора
#define CODE_7 0xf0 //- 240 - Вимкнення вентилятора
#define CODE_8 0xa1 //- 161 - Ввімкнення опалення
#define CODE_9 0xa0 //- 160 - Вимкнення опалення
#define CODE_10 0x0e //- 14  - Температура автоматичного ввімкненн
я вентилятора
#define CODE_11 0x0d //- 13  - Температура автоматичного ввімкненн
я опалення

#define CODE_12 0xe1 //- 225 - Задання року
#define CODE_13 0xe2 //- 226 - Задання місяця
#define CODE_14 0xe3 //- 227 - Задання дня
#define CODE_15 0xe4 //- 228 - Задання години
#define CODE_16 0xe5 //- 229 - Задання хвилини
#define CODE_17 0xee //- 238 - Задання часу

EE25LC1024 eEPROM(PB_5,PB_4,PB_3,PB_10);

Serial pc(USBTX, USBRX);
DigitalOut led1(LED1);
BME280_SPI sensor(PC_12, PC_11, PC_10, PC_9); // mosi, miso, sclk,
 cs
DigitalOut fan(PF_13);
DigitalOut heating(PF_14);
```

```
uint writeStartbyte = 1024;
uint coutWrite = 0;

struct tm   t;

Thread thread;
Thread thread1;

struct microclimateStruct
{
    float temF;
    float temC;
    float prs_hPa;
    float humidity;
    time_t timeWrite;
    int maxTemp;
    int minTemp;
    /* data */
}mcSt;

 time_t seconds;

void UARTQuery(){
    Serial pc(USBTX, USBRX);
    event_callback_t callback;
    pc.set_dma_usage_tx(DMAUsage::DMA_USAGE_ALWAYS);
    char dataS[50];
    while(true){
        union main
        {
            uint32_t date;
            uint8_t code[4];
            /* data */
        }m;

        m.date= read_int();

        //int saa1 = read_int();
        pc.printf("%d\n",m.code[0]);
        switch (m.code[0])
        {
        case 0xcc:
            // union main
            // {
            //     uint8_t i[4];
            //     int ch;
            // }m;
            // pc.printf("-> %f \n",mcSt.temF);
            // m.ch = mcSt.temF*100;
            //printf("Hello\n");
            sprintf(dataS, "%.2fC\n", mcSt.temC);
            pc.write((uint8_t*)dataS,strlen(dataS),callback,SERIAL
_EVENT_TX_COMPLETE);
```

```
            break;
         case 0xaf:
            sprintf(dataS, "%.2fF\n", mcSt.temF);
            // pc.printf("temp %.2fv\n", mcSt.temC);
            pc.write((uint8_t*)dataS,strlen(dataS),callback,SERIAL
_EVENT_TX_COMPLETE);
            break;
         case 0x11:
            sprintf(dataS, "System start\n");
            // pc.printf("temp %.2fv\n", mcSt.temC);
            pc.write((uint8_t*)dataS,strlen(dataS),callback,SERIAL
_EVENT_TX_COMPLETE);
            break;
         case 0x10:
            sprintf(dataS, "System stop\n");
            // pc.printf("temp %.2fv\n", mcSt.temC);
            pc.write((uint8_t*)dataS,strlen(dataS),callback,SERIAL
_EVENT_TX_COMPLETE);
            break;
        case 0xae:
            sprintf(dataS, "%.2f", mcSt.humidity);
            pc.write((uint8_t*)dataS,strlen(dataS),callback,SERIAL
_EVENT_TX_COMPLETE);
            break;
        case 0xac:
            sprintf(dataS, "%.2fhPa", mcSt.prs_hPa);
            pc.write((uint8_t*)dataS,strlen(dataS),callback,SERIAL
_EVENT_TX_COMPLETE);
            break;
        case 0xea:
            sprintf(dataS, "%2.2f degF, %2.2f degC, %04.2f hPa, %2
.2f %%\n" , mcSt.temF, mcSt.temC, mcSt.prs_hPa, mcSt.humidity);
            pc.write((uint8_t*)dataS,strlen(dataS),callback,SERIAL
_EVENT_TX_COMPLETE);
            break;
        case 0xf1:
            fan=1;
            break;
        case 0xf0:
            fan=0;
            break;
        case 0xa1:
            heating=1;
            break;
        case 0xa0:
            heating=0;
            break;

        case 0xe1:
            // struct tm   t;
          //  pc.printf("Please enter year:\n");
            union mm
            {
```

```
                int year;
                uint8_t data[2];
                /* data */
            }ss;
            printf("->%d->%d\n", m.code[3], m.code[2]);
            ss.data[0] = m.code[2];
            ss.data[1] = m.code[3];
            printf("->%d\n",ss.year);
            t.tm_year = ss.year-1900;
            break;
        case 0xe2:
            t.tm_mon = m.code[2]-1;
            break;
        case 0xe3:
            t.tm_mday = m.code[2];
            break;
        case 0xe4:
            t.tm_hour = m.code[2];
            break;
        case 0xe5:
            t.tm_min = m.code[2];
            break;
        case 0xee:
            set_time(mktime(&t));
            seconds = time(NULL);
            printf("Time as a basic string = %s", ctime(&seconds))
;

            // sprintf(dataS, "%cYear,%cMonth,%cDay,%cHour,%cMin,%
cSec",t.tm_year,  t.tm_mon, t.tm_mday, t.tm_hour, t.tm_min, t.tm_s
ec);
            // pc.write((uint8_t*)dataS,strlen(dataS),callback,SER
IAL_EVENT_TX_COMPLETE);
            break;
        case 0xf5:
            pc.printf("3231\n");
            break;
        case 0xfa:
            pc.printf("3231\n");
            break;
        case 0xaa:
            pc.printf("3231\n");
            break;
        case 0x60:
            eEPROM.chipErase();
            pc.printf("Chip is erase.\n");
            break;
        case 0x0e:
            mcSt.maxTemp=read_int();
            break;
        case 0x0d:
            pc.printf("3231\n");
            break;
```

```
        // case 0xce:
        //     sprintf(dataS, "%2.2f degC, %2.2f %%\n" , mcSt.temC
, mcSt.humidity);
        //     pc.write((uint8_t*)dataS,strlen(dataS),callback,SER
IAL_EVENT_TX_COMPLETE);
        //     break;
        //  case 0xcf:
        //     sprintf(dataS, "%2.2f degC, %04.2f hPa\n" , mcSt.te
mC, mcSt.prs_hPa);
        //     pc.write((uint8_t*)dataS,strlen(dataS),callback,SER
IAL_EVENT_TX_COMPLETE);
        //     break;
        // case 0xec:
        //     sprintf(dataS, "%2.2f %%, %04.2f hPa\n" , mcSt.humi
dity, mcSt.prs_hPa);
        //     pc.write((uint8_t*)dataS,strlen(dataS),callback,SER
IAL_EVENT_TX_COMPLETE);
        //     break;
        // case 0xfe:
        //     sprintf(dataS, "%2.2f degF, %2.2f %%\n" , mcSt.temF
, mcSt.humidity);
        //     pc.write((uint8_t*)dataS,strlen(dataS),callback,SER
IAL_EVENT_TX_COMPLETE);
        //     break;
        // case 0xfc:
        //     sprintf(dataS, "%2.2f degF, %04.2f hPa\n" , mcSt.te
mF, mcSt.prs_hPa);
        //     pc.write((uint8_t*)dataS,strlen(dataS),callback,SER
IAL_EVENT_TX_COMPLETE);
        //     break;
        default:
            break;
        }
    //    ThisThread::sleep_for(200);
    }
}

void TCP_Server_Thread() {

    EthernetInterface eth;
    eth.connect();

    pc.printf("The target IP address is '%s'\r\n", eth.get_ip_addr
ess());

    TCPServer srv;
    TCPSocket clt_sock;
    SocketAddress clt_addr;

    /* Open the server on ethernet stack */
    srv.open(&eth);
```

```
    /* Bind the HTTP port (TCP 80) to the server */
    srv.bind(eth.get_ip_address(), 80);


    //srv.set_blocking(false);
    while (true) {
    pc.printf("waiting for client\r\n");
    /* Can handle 5 simultaneous connections */
    int err= srv.listen(1);
    pc.printf("server listening error : %d\r\n",err);

    while(1){
        pc.printf("waiting for client connection\r\n");
        err = srv.accept(&clt_sock, &clt_addr);
        if(err == 0){
            pc.printf("client connected :%s:%d\r\n", clt_addr.get_
ip_address(), clt_addr.get_port());
            while(1){
                char recvData[100];
                int len = clt_sock.recv(recvData, 100);
                if(len > 0){

                    recvData[len] = '\0';
                    int code=atoi(recvData);
                   // pc.printf("code%d",code);
                    char dataS[50];
                //     pc.printf(dataS, "%.2fC\n", mcSt.temC);
                    switch(code)
                    {
                        case(CODE_1):
                          //  recvData[len] = '\0';
                           // pc.printf("TA[%d] : %s\r\n",len,recv
Data);
                            pc.printf(dataS, "%.2fC\n", mcSt.temC)
;
                            sprintf(dataS, "%.2fC\n", mcSt.temC);
                            clt_sock.send(dataS, strlen(dataS));
                        break;
                        case(CODE_2):
                            pc.printf(dataS, "%.2fF\n", mcSt.temF)
;
                            sprintf(dataS, "%.2fF\n", mcSt.temF);
                            clt_sock.send(dataS, strlen(dataS));
                        break;
                        case(CODE_3):
                            pc.printf(dataS, "%.2f\n", mcSt.humidi
ty);
                            sprintf(dataS, "%.2f\n", mcSt.humidity
);
                            clt_sock.send(dataS, strlen(dataS));
                        break;
                        case(CODE_4):
```

```
                            pc.printf(dataS, "%.2fPa\n", mcSt.prs_
hPa);
                            sprintf(dataS, "%.2fPa\n", mcSt.prs_hP
a);
                            clt_sock.send(dataS, strlen(dataS));
                        break;

                    }

                }else{
                    pc.printf("disconnected\r\n");
                    clt_sock.close();
                    break;
                }
            }
        }
    }
 }
}

void setTime(int year, int mon, int mday, int hour, int min, int s
ec) {
    t.tm_year = year - 1900;
    t.tm_mon =  mon - 1;
    t.tm_mday = mday;
    t.tm_hour = hour;
    t.tm_min = min;
    t.tm_sec = sec;
    set_time(mktime(&t));
    seconds = time(NULL);
}

void readMicroclimate(){
    while (true) {
        mcSt.temF = sensor.getTemperature();
        mcSt.temC = (mcSt.temF-32)*0.555556;
        mcSt.prs_hPa = sensor.getPressure();
        mcSt.humidity = sensor.getHumidity();
        mcSt.timeWrite = seconds;

        // eEPROM.writeStream(writeStartbyte,(uint8_t*)&mcSt,sizeo
f(mcSt));
        // writeStartbyte +=256;
        // coutWrite+=1;
        // if(coutWrite%10==0){
        //     eEPROM.sectorErase(0);
        //     eEPROM.writeByte(0,coutWrite);
        // }
        ThisThread::sleep_for(1000);

    }
}
```

```
// main() runs in its own thread in the OS
int main() {
    thread.start(callback(readMicroclimate));
    // thread1.start(callback(TCP_Server_Thread));
    thread1.start(callback(UARTQuery));

  //  Thread thread2(TCP_Server_Thread);
 // eEPROM.pageErase32(0);
   // eEPROM.writeByte(0,12);
    //wait_ms(100);
    //pc.printf("Data from 0 addr%d",eEPROM.readByte(0));
    setTime(2000,1,1,00,00,00);
    while (true) {
    seconds = time(NULL);

        led1 = !led1;
        wait(1);
    }
    thread1.join();
    thread.join();
}
```

Лістинг файла BMP280_SPI.cpp

```
#include "mbed.h"
#include "BME280_SPI.h"

BME280_SPI::BME280_SPI(PinName mosi, PinName miso, PinName sclk, P
inName cs)
    :
    _spi(mosi, miso, sclk),
    _cs(cs),
    t_fine(0)
{
    initialize();
}


BME280_SPI::~BME280_SPI()
{
}

void BME280_SPI::initialize()
{
    char cmd[18];

    _cs = 1;
    _spi.format(8, 0); // 8-bit, mode=0
```

```
    _spi.frequency(1000000); // 1MHZ

    _cs = 0;
    _spi.write(0xd0); // chip_id
    cmd[0] = _spi.write(0); // read chip_id
    _cs = 1;

    DEBUG_PRINT("chip_id = 0x%x\n", cmd[0]);

    _cs = 0;
    _spi.write(0xf2 & BME280_SPI_MASK); // ctrl_hum
    _spi.write(0x04); // Humidity oversampling x4
    _cs = 1;

    _cs = 0;
    _spi.write(0xf4 & BME280_SPI_MASK); // ctrl_meas
    _spi.write((4<<5)|(4<<2)|3); // Temparature oversampling x4, P
ressure oversampling x4, Normal mode
    _cs = 1;

    _cs = 0;
    _spi.write(0xf5 & BME280_SPI_MASK); // config
    _spi.write(0xa0); // Standby 1000ms, Filter off, 4-
wire SPI interface
    _cs = 1;

    wait(1);

    _cs = 0;
    _spi.write(0x88); // read dig_T regs
    for(int i = 0; i < 6; i++)
        cmd[i] = _spi.write(0);
    _cs = 1;

    dig_T1 = (cmd[1] << 8) | cmd[0];
    dig_T2 = (cmd[3] << 8) | cmd[2];
    dig_T3 = (cmd[5] << 8) | cmd[4];

    DEBUG_PRINT("dig_T = 0x%x, 0x%x, 0x%x\n", dig_T1, dig_T2, dig_
T3);
    DEBUG_PRINT("dig_T = %d, %d, %d\n", dig_T1, dig_T2, dig_T3);

    _cs = 0;
    _spi.write(0x8e); // read dig_P regs
    for(int i = 0; i < 18; i++)
        cmd[i] = _spi.write(0);
    _cs = 1;

    dig_P1 = (cmd[ 1] << 8) | cmd[ 0];
    dig_P2 = (cmd[ 3] << 8) | cmd[ 2];
    dig_P3 = (cmd[ 5] << 8) | cmd[ 4];
    dig_P4 = (cmd[ 7] << 8) | cmd[ 6];
    dig_P5 = (cmd[ 9] << 8) | cmd[ 8];
```

```
    dig_P6 = (cmd[11] << 8) | cmd[10];
    dig_P7 = (cmd[13] << 8) | cmd[12];
    dig_P8 = (cmd[15] << 8) | cmd[14];
    dig_P9 = (cmd[17] << 8) | cmd[16];

    DEBUG_PRINT("dig_P = 0x%x, 0x%x, 0x%x, 0x%x, 0x%x, 0x%x, 0x%x,
 0x%x, 0x%x\n", dig_P1, dig_P2, dig_P3, dig_P4, dig_P5, dig_P6, di
g_P7, dig_P8, dig_P9);
    DEBUG_PRINT("dig_P = %d, %d, %d, %d, %d, %d, %d, %d, %d\n", di
g_P1, dig_P2, dig_P3, dig_P4, dig_P5, dig_P6, dig_P7, dig_P8, dig_
P9);

    _cs = 0;
    _spi.write(0xA1); // read dig_H1 reg
    cmd[0] = _spi.write(0);
    _cs = 1;

    _cs = 0;
    _spi.write(0xE1); // read dig_H regs
    for(int i = 0; i < 7; i++)
        cmd[1+i] = _spi.write(0);
    _cs = 1;

    dig_H1 = cmd[0];
    dig_H2 = (cmd[2] << 8) | cmd[1];
    dig_H3 = cmd[3];
    dig_H4 = (cmd[4] << 4) | (cmd[5] & 0x0f);
    dig_H5 = (cmd[6] << 4) | ((cmd[5]>>4) & 0x0f);
    dig_H6 = cmd[7];

    DEBUG_PRINT("dig_H = 0x%x, 0x%x, 0x%x, 0x%x, 0x%x, 0x%x\n", di
g_H1, dig_H2, dig_H3, dig_H4, dig_H5, dig_H6);
    DEBUG_PRINT("dig_H = %d, %d, %d, %d, %d, %d\n", dig_H1, dig_H2
, dig_H3, dig_H4, dig_H5, dig_H6);
}

float BME280_SPI::getTemperature()
{
    uint32_t temp_raw;
    float tempf;
    char cmd[3];

    _cs = 0;
    _spi.write(0xfa);
    for(int i = 0; i < 3; i++)
        cmd[i] = _spi.write(0);
    _cs = 1;

    temp_raw = (cmd[0] << 12) | (cmd[1] << 4) | (cmd[2] >> 4);

    int32_t temp;

    temp =
```

```
        (((((temp_raw >> 3) - (dig_T1 << 1))) * dig_T2) >> 11) +
        ((((((temp_raw >> 4) - dig_T1) * ((temp_raw >> 4) - dig_T1
)) >> 12) * dig_T3) >> 14);

    t_fine = temp;
    temp = (temp * 5 + 128) >> 8;
    tempf = (float)temp;

    return (tempf/100.0f);
}

float BME280_SPI::getPressure()
{
    uint32_t press_raw;
    float pressf;
    char cmd[3];

    _cs = 0;
    _spi.write(0xf7); // press_msb
    for(int i = 0; i < 3; i++)
        cmd[i] = _spi.write(0);
    _cs = 1;

    press_raw = (cmd[0] << 12) | (cmd[1] << 4) | (cmd[2] >> 4);

    int32_t var1, var2;
    uint32_t press;

    var1 = (t_fine >> 1) - 64000;
    var2 = (((var1 >> 2) * (var1 >> 2)) >> 11) * dig_P6;
    var2 = var2 + ((var1 * dig_P5) << 1);
    var2 = (var2 >> 2) + (dig_P4 << 16);
    var1 = (((dig_P3 * (((var1 >> 2)*(var1 >> 2)) >> 13)) >> 3) +
((dig_P2 * var1) >> 1)) >> 18;
    var1 = ((32768 + var1) * dig_P1) >> 15;
    if (var1 == 0) {
        return 0;
    }
    press = (((1048576 - press_raw) - (var2 >> 12))) * 3125;
    if(press < 0x80000000) {
        press = (press << 1) / var1;
    } else {
        press = (press / var1) * 2;
    }
    var1 = ((int32_t)dig_P9 * ((int32_t)(((press >> 3) * (press >>
 3)) >> 13))) >> 12;
    var2 = (((int32_t)(press >> 2)) * (int32_t)dig_P8) >> 13;
    press = (press + ((var1 + var2 + dig_P7) >> 4));

    pressf = (float)press;
    return (pressf/100.0f);
}
```

```
float BME280_SPI::getHumidity()
{
    uint32_t hum_raw;
    float humf;
    char cmd[2];

    _cs = 0;
    _spi.write(0xfd); // hum_msb
    for(int i = 0; i < 2; i++)
        cmd[i] = _spi.write(0);
    _cs = 1;

    hum_raw = (cmd[0] << 8) | cmd[1];

    int32_t v_x1;

    v_x1 = t_fine - 76800;
    v_x1 =  (((((hum_raw << 14) -
(((int32_t)dig_H4) << 20) - (((int32_t)dig_H5) * v_x1)) +
                ((int32_t)16384)) >> 15) * (((((((v_x1 * (int32_t)d
ig_H6) >> 10) *
                                                (((v_x1 * ((int32_t)di
g_H3)) >> 11) + 32768)) >> 10) + 2097152) *
                                                (int32_t)dig_H2 + 8192
) >> 14));
    v_x1 = (v_x1 - (((((v_x1 >> 15) * (v_x1 >> 15)) >> 7) * (int32
_t)dig_H1) >> 4));
    v_x1 = (v_x1 < 0 ? 0 : v_x1);
    v_x1 = (v_x1 > 419430400 ? 419430400 : v_x1);

    humf = (float)(v_x1 >> 12);

    return (humf/1024.0f);
}
```

Лістинг файла BMP280_SPI.h

```
#ifndef MBED_BME280_SPI_H
#define MBED_BME280_SPI_H

#include "mbed.h"

#ifdef _DEBUG
extern Serial pc;
#define DEBUG_PRINT(...) pc.printf(__VA_ARGS__)
#else
#define DEBUG_PRINT(...)
#endif

/**  Interface for controlling BME280 Combined humidity and pressu
re sensor
```

```
 *
 * @code
 * #include "mbed.h"
 * #include "BME280_SPI.h"
 *
 * Serial pc(USBTX, USBRX);
 * BME280_SPI sensor(D11, D12, D13, D9); // mosi, miso, sclk, cs
 *
 * int main() {
 *
 *     while(1) {
 *         pc.printf("%2.2f degC, %04.2f hPa, %2.2f %%\n", sensor.
getTemperature(), sensor.getPressure(), sensor.getHumidity());
 *         wait(1);
 *     }
 * }
 *
 * @endcode
 */

/** BME280_SPI class
 *
 *  BME280_SPI: A library to correct environmental data using Bosh
e BME280 environmental sensor device
 *
 */
class BME280_SPI
{
public:

    enum spi_mask {
        BME280_SPI_MASK = 0x7F
    };

    /** Create a BME280 instance
     *  which is connected to specified SPI pins
     *
     * @param mosi SPI MOSI pin
     * @param miso SPI MISO pin
     * @param sclk SPI SCLK pin
     * @param cs device CS pin
     */
    BME280_SPI(PinName mosi, PinName miso, PinName sclk, PinName c
s);

    /** Destructor of BME280_SPI
     */
    virtual ~BME280_SPI();

    /** Initializa BME280 sensor
     *
     *  Configure sensor setting and read parameters for calibrati
on
```

```
     *
     */
    void initialize(void);

    /** Read the current temperature value (degree Celsius) from B
ME280 sensor
     *
     * @return Temperature value (degree Celsius)
     */
    float getTemperature(void);

    /** Read the current pressure value (hectopascal) from BME280
sensor
     *
     * @return Pressure value (hectopascal)
     */
    float getPressure(void);

    /** Read the current humidity value (humidity %) from BME280 s
ensor
     *
     * @return Humidity value (humidity %)
     */
    float getHumidity(void);

private:

    SPI         _spi;
    DigitalOut  _cs;
    uint16_t    dig_T1;
    int16_t     dig_T2, dig_T3;
    uint16_t    dig_P1;
    int16_t     dig_P2, dig_P3, dig_P4, dig_P5, dig_P6, dig_P7, di
g_P8, dig_P9;
    uint16_t    dig_H1, dig_H3;
    int16_t     dig_H2, dig_H4, dig_H5, dig_H6;
    int32_t     t_fine;

};

#endif // MBED_BME280_SPI_H
```

PinNames.h (STM32F767ZI)

```
#ifndef MBED_PINNAMES_H
#define MBED_PINNAMES_H

#include "cmsis.h"
#include "PinNamesTypes.h"

#ifdef __cplusplus
extern "C" {
```

```
#endif

typedef enum {
    ALT0  = 0x100,
    ALT1  = 0x200,
    ALT2  = 0x300,
    ALT3  = 0x400
} ALTx;

typedef enum {
    PA_0  = 0x00,
    PA_0_ALT0 = PA_0 | ALT0,
    PA_0_ALT1 = PA_0 | ALT1,
    PA_1  = 0x01,
    PA_1_ALT0 = PA_1 | ALT0,
    PA_1_ALT1 = PA_1 | ALT1,
    PA_2  = 0x02,
    PA_2_ALT0 = PA_2 | ALT0,
    PA_2_ALT1 = PA_2 | ALT1,
    PA_3  = 0x03,
    PA_3_ALT0 = PA_3 | ALT0,
    PA_3_ALT1 = PA_3 | ALT1,
    PA_4  = 0x04,
    PA_4_ALT0 = PA_4 | ALT0,
    PA_4_ALT1 = PA_4 | ALT1,
    PA_5  = 0x05,
    PA_5_ALT0 = PA_5 | ALT0,
    PA_6  = 0x06,
    PA_6_ALT0 = PA_6 | ALT0,
    PA_7  = 0x07,
    PA_7_ALT0 = PA_7 | ALT0,
    PA_7_ALT1 = PA_7 | ALT1,
    PA_7_ALT2 = PA_7 | ALT2,
    PA_8  = 0x08,
    PA_9  = 0x09,
    PA_10 = 0x0A,
    PA_11 = 0x0B,
    PA_12 = 0x0C,
    PA_13 = 0x0D,
    PA_14 = 0x0E,
    PA_15 = 0x0F,
    PA_15_ALT0 = PA_15 | ALT0,
    PA_15_ALT1 = PA_15 | ALT1,

    PB_0  = 0x10,
    PB_0_ALT0 = PB_0 | ALT0,
    PB_0_ALT1 = PB_0 | ALT1,
    PB_1  = 0x11,
    PB_1_ALT0 = PB_1 | ALT0,
    PB_1_ALT1 = PB_1 | ALT1,
    PB_2  = 0x12,
    PB_3  = 0x13,
    PB_3_ALT0 = PB_3 | ALT0,
    PB_3_ALT1 = PB_3 | ALT1,
```

```
PB_4  = 0x14,
PB_4_ALT0 = PB_4 | ALT0,
PB_4_ALT1 = PB_4 | ALT1,
PB_5  = 0x15,
PB_5_ALT0 = PB_5 | ALT0,
PB_5_ALT1 = PB_5 | ALT1,
PB_6  = 0x16,
PB_6_ALT0 = PB_6 | ALT0,
PB_7  = 0x17,
PB_7_ALT0 = PB_7 | ALT0,
PB_8  = 0x18,
PB_8_ALT0 = PB_8 | ALT0,
PB_8_ALT1 = PB_8 | ALT1,
PB_9  = 0x19,
PB_9_ALT0 = PB_9 | ALT0,
PB_9_ALT1 = PB_9 | ALT1,
PB_10 = 0x1A,
PB_11 = 0x1B,
PB_12 = 0x1C,
PB_13 = 0x1D,
PB_14 = 0x1E,
PB_14_ALT0 = PB_14 | ALT0,
PB_14_ALT1 = PB_14 | ALT1,
PB_15 = 0x1F,
PB_15_ALT0 = PB_15 | ALT0,
PB_15_ALT1 = PB_15 | ALT1,

PC_0  = 0x20,
PC_0_ALT0 = PC_0 | ALT0,
PC_0_ALT1 = PC_0 | ALT1,
PC_1  = 0x21,
PC_1_ALT0 = PC_1 | ALT0,
PC_1_ALT1 = PC_1 | ALT1,
PC_2  = 0x22,
PC_2_ALT0 = PC_2 | ALT0,
PC_2_ALT1 = PC_2 | ALT1,
PC_3  = 0x23,
PC_3_ALT0 = PC_3 | ALT0,
PC_3_ALT1 = PC_3 | ALT1,
PC_4  = 0x24,
PC_4_ALT0 = PC_4 | ALT0,
PC_5  = 0x25,
PC_5_ALT0 = PC_5 | ALT0,
PC_6  = 0x26,
PC_6_ALT0 = PC_6 | ALT0,
PC_7  = 0x27,
PC_7_ALT0 = PC_7 | ALT0,
PC_8  = 0x28,
PC_8_ALT0 = PC_8 | ALT0,
PC_9  = 0x29,
PC_9_ALT0 = PC_9 | ALT0,
PC_10 = 0x2A,
PC_11 = 0x2B,
PC_12 = 0x2C,
```

```
PC_13 = 0x2D,
PC_14 = 0x2E,
PC_15 = 0x2F,

PD_0  = 0x30,
PD_1  = 0x31,
PD_2  = 0x32,
PD_3  = 0x33,
PD_4  = 0x34,
PD_5  = 0x35,
PD_6  = 0x36,
PD_7  = 0x37,
PD_8  = 0x38,
PD_9  = 0x39,
PD_10 = 0x3A,
PD_11 = 0x3B,
PD_12 = 0x3C,
PD_13 = 0x3D,
PD_14 = 0x3E,
PD_15 = 0x3F,

PE_0  = 0x40,
PE_1  = 0x41,
PE_2  = 0x42,
PE_3  = 0x43,
PE_4  = 0x44,
PE_5  = 0x45,
PE_6  = 0x46,
PE_7  = 0x47,
PE_8  = 0x48,
PE_9  = 0x49,
PE_10 = 0x4A,
PE_11 = 0x4B,
PE_12 = 0x4C,
PE_13 = 0x4D,
PE_14 = 0x4E,
PE_15 = 0x4F,

PF_0  = 0x50,
PF_1  = 0x51,
PF_2  = 0x52,
PF_3  = 0x53,
PF_4  = 0x54,
PF_5  = 0x55,
PF_6  = 0x56,
PF_7  = 0x57,
PF_8  = 0x58,
PF_9  = 0x59,
PF_10 = 0x5A,
PF_11 = 0x5B,
PF_12 = 0x5C,
PF_13 = 0x5D,
PF_14 = 0x5E,
```

```
    PF_15 = 0x5F,

    PG_0  = 0x60,
    PG_1  = 0x61,
    PG_2  = 0x62,
    PG_3  = 0x63,
    PG_4  = 0x64,
    PG_5  = 0x65,
    PG_6  = 0x66,
    PG_7  = 0x67,
    PG_8  = 0x68,
    PG_9  = 0x69,
    PG_10 = 0x6A,
    PG_11 = 0x6B,
    PG_12 = 0x6C,
    PG_13 = 0x6D,
    PG_14 = 0x6E,
    PG_15 = 0x6F,

    PH_0  = 0x70,
    PH_1  = 0x71,

    // ADC internal channels
    ADC_TEMP = 0xF0,
    ADC_VREF = 0xF1,
    ADC_VBAT = 0xF2,

    // Arduino connector namings
    A0          = PA_3,
    A1          = PC_0,
    A2          = PC_3,
    A3          = PF_3,
    A4          = PF_5,
    A5          = PF_10,
    D0          = PG_9,
    D1          = PG_14,
    D2          = PF_15,
    D3          = PE_13,
    D4          = PF_14,
    D5          = PE_11,
    D6          = PE_9,
    D7          = PF_13,
    D8          = PF_12,
    D9          = PD_15,
    D10         = PD_14,
    D11         = STM32_D11_SPI_ETHERNET_PIN, /* config in targets
.json file */
    D12         = PA_6,
    D13         = PA_5,
    D14         = PB_9,
    D15         = PB_8,

    // STDIO for console print
```

```
#ifdef MBED_CONF_TARGET_STDIO_UART_TX
    STDIO_UART_TX = MBED_CONF_TARGET_STDIO_UART_TX,
#else
    STDIO_UART_TX = PD_8,
#endif
#ifdef MBED_CONF_TARGET_STDIO_UART_RX
    STDIO_UART_RX = MBED_CONF_TARGET_STDIO_UART_RX,
#else
    STDIO_UART_RX = PD_9,
#endif

    // Generic signals namings
    LED1        = PB_0,  // LD1 = GREEN
    LED2        = PB_7,  // Blue
    LED3        = PB_14, // Red
    LED4        = PB_0,
    USER_BUTTON = PC_13,
    // Standardized button names
    BUTTON1 = USER_BUTTON,
    SERIAL_TX   = STDIO_UART_TX, // Virtual Com Port
    SERIAL_RX   = STDIO_UART_RX, // Virtual Com Port
    USBTX       = STDIO_UART_TX, // Virtual Com Port
    USBRX       = STDIO_UART_RX, // Virtual Com Port
    I2C_SCL     = D15,
    I2C_SDA     = D14,
    SPI_MOSI    = D11,
    SPI_MISO    = D12,
    SPI_SCK     = D13,
    SPI_CS      = D10,
    PWM_OUT     = D9,

    /**** USB FS pins ****/
    USB_OTG_FS_DM = PA_11,
    USB_OTG_FS_DP = PA_12,
    USB_OTG_FS_ID = PA_10,
    USB_OTG_FS_SOF = PA_8,
    USB_OTG_FS_VBUS = PA_9,

    /**** USB HS pins ****/
    USB_OTG_HS_DM = PB_14,
    USB_OTG_HS_DP = PB_15,
    USB_OTG_HS_ID = PB_12,
    USB_OTG_HS_SOF = PA_4,
    USB_OTG_HS_ULPI_CK = PA_5,
    USB_OTG_HS_ULPI_D0 = PA_3,
    USB_OTG_HS_ULPI_D1 = PB_0,
    USB_OTG_HS_ULPI_D2 = PB_1,
    USB_OTG_HS_ULPI_D3 = PB_10,
    USB_OTG_HS_ULPI_D4 = PB_11,
    USB_OTG_HS_ULPI_D5 = PB_12,
    USB_OTG_HS_ULPI_D6 = PB_13,
    USB_OTG_HS_ULPI_D7 = PB_5,
    USB_OTG_HS_ULPI_DIR = PC_2,
```

```
USB_OTG_HS_ULPI_NXT = PC_3,
USB_OTG_HS_ULPI_STP = PC_0,
USB_OTG_HS_VBUS = PB_13,

/**** ETHERNET pins ****/
ETH_COL = PA_3,
ETH_CRS = PA_0,
ETH_CRS_DV = PA_7,
ETH_MDC = PC_1,
ETH_MDIO = PA_2,
ETH_PPS_OUT = PG_8,
ETH_PPS_OUT_ALT0 = PB_5,
ETH_REF_CLK = PA_1,
ETH_RXD0 = PC_4,
ETH_RXD1 = PC_5,
ETH_RXD2 = PB_0,
ETH_RXD3 = PB_1,
ETH_RX_CLK = PA_1,
ETH_RX_DV = PA_7,
ETH_RX_ER = PB_10,
ETH_TXD0 = PB_12,
ETH_TXD0_ALT0 = PG_13,
ETH_TXD1 = PB_13,
ETH_TXD1_ALT0 = PG_14,
ETH_TXD2 = PC_2,
ETH_TXD3 = PE_2,
ETH_TXD3_ALT0 = PB_8,
ETH_TX_CLK = PC_3,
ETH_TX_EN = PB_11,
ETH_TX_EN_ALT0 = PG_11,

/**** OSCILLATOR pins ****/
RCC_OSC32_IN = PC_14,
RCC_OSC32_OUT = PC_15,
RCC_OSC_IN = PH_0,
RCC_OSC_OUT = PH_1,

/**** DEBUG pins ****/
SYS_JTCK_SWCLK = PA_14,
SYS_JTDI = PA_15,
SYS_JTDO_SWO = PB_3,
SYS_JTMS_SWDIO = PA_13,
SYS_JTRST = PB_4,
SYS_TRACECLK = PE_2,
SYS_TRACED0 = PE_3,
SYS_TRACED0_ALT0 = PC_1,
SYS_TRACED0_ALT1 = PG_13,
SYS_TRACED1 = PE_4,
SYS_TRACED1_ALT0 = PC_8,
SYS_TRACED1_ALT1 = PG_14,
SYS_TRACED2 = PE_5,
SYS_TRACED2_ALT0 = PD_2,
SYS_TRACED3 = PE_6,
```

```
    SYS_TRACED3_ALT0 = PC_12,
    SYS_WKUP1 = PA_0,
    SYS_WKUP2 = PA_2,
    SYS_WKUP3 = PC_1,
    SYS_WKUP4 = PC_13,

    // Not connected
    NC = (int)0xFFFFFFFF
} PinName;

#ifdef __cplusplus
}
#endif

#endif
```