

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки
Факультет комп'ютерних наук та технологій
(повне найменування інституту, факультету)

Кафедра комп'ютерних систем та мереж
(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)
бакалавра

(ступінь вищої освіти (освітній ступінь))

на тему Розробка дистанційної системи вимірювання атмосферних показників на базі
ATMEGA8

Виконав: студент 4 курсу, групи КНТ-518сп
спеціальності _____

123 «Комп'ютерна інженерія»

(код і найменування спеціальності)

Освітня програма (спеціалізація) _____

«Комп'ютерні системи та мережі»

Назаренко Олександр Іванович

(прізвище та ініціали)

Керівник Тягунова М.Ю

(прізвище та ініціали)

Рецензент Качан О.І.

(прізвище та ініціали)

м. Запоріжжя

2021 рік

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

(повне найменування вищого навчального закладу)

Інститут, факультет інформатики та радіоелектроніки, комп'ютерних наук і технологій
Кафедра «Комп'ютерні системи та мережі»
Ступінь вищої освіти (освітній ступінь) бакалаврський
Спеціальність 123 Комп'ютерна інженерія
(код і найменування)

Освітня програма (спеціалізація) Комп'ютерні системи та мережі
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри Кудерметов Р.К.

“ ” 2020 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТА

Назаренко Олександр Іванович

(прізвище, ім'я, по батькові)

Тема проекту (роботи) Розробка дистанційної системи вимірювання атмосферних показників на базі ATMEGA8

Рівень проекту (роботи) Тягунова Марія Юріївна, к. т. н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом вищого навчального закладу від “ 17 ” березня 2021 року № 81

Строк подання студентом проекту (роботи) 05 травня 2021 року

Вихідні дані до проекту (роботи) аналіз стану досліджень з питань розробки системи вимірювання атмосферних показників

Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

Аналіз предметної області;

Проектування системи;

Реалізація системи.

Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

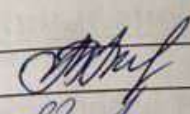
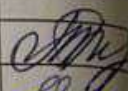
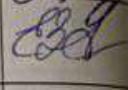
Структура системи

Принципова схема системи

Взаємодія компонентів системи

Структурна схема системи

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконав завдання
1-4	Тягунова М.Ю., к. т. н., доцент		
Нормо-контроль	Зелік О.В., ст викл.		

7. Дата видачі завдання 27.02.2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітки
1	Аналіз предметної області	27.02.2021 р.	
2	Постановка задач		
3	Вибір технологій		
4	Вивчення документації елементів	12.03.2021 р.	
5	Аналіз роботи та взаємодії датчиків тиску	20.03.2021 р.	
6	Аналіз роботи та взаємодії датчиків вологості повітря	01.04.2021 р.	
7	Аналіз роботи та взаємодії датчиків температури	09.04.2021 р.	
8	Робота з мережовим обладнанням	14.04.2021 р.	
9	Оформлення графічного матеріалу	20.04.2021 р.	

Студент


(підпис)

О.І. Назаренко
(ініціали та прізвище)

Керівник проекту (роботи)


(підпис)

М.Ю. Тягунова
(ініціали та прізвище)

РЕФЕРАТ

ПЗ: 105 с., 38 рисунках, 37 таблиць, 23джерел.

AVR, ATMEGA8, МІКРОКОНТРОЛЕР, PROTEUS, ДАТЧИК ВОЛОГОСТІ,
BMP, WI-FI МОДУЛЬ, CODEVISIONAVR, ДАТЧИК

Об'єкт дослідження – системи для вимірювання атмосферних показників на основі Atmega8A.

Мета роботи – організувати вимірювання та контроль атмосферних показників на відстанні.

Пояснювальна записка складається з 3 розділів. Перший розділ присвячено аналізу технічного завдання. Було розглянуто мікроконтролери та платформи на основі яких можна побудувати систему: PIC controller, Arduino, Raspberry pi, Atmega8. Другий розділ описує вибір апаратної частини комп'ютерного модуля. Другий розділ описує вибір апаратної частини комп'ютерного модуля. За для матеріального проектування системи були використані датчики – мікроконтролер Atmega8A, датчик вологості та температури DHT-11, датчик атмосферного тиску та температури BMP180 та Wi-Fi модуль ESP8266 12-f. Третій розділ присвячено програмній розробці. Для створення системи було використано ПЗ AVR CodeVision та проектування у програми Proteus.

В результаті роботи було отримано «домашню метеостанцію» яка передає свої показники до мережі Інтернет, якій в свою чергу передає данні на хмару.

В подальшому розвитку мікросхеми її можна використовувати у тандемі з іншими системами та створити на їх основі «Розумний дім».

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧКИ

ПЗ	- програмне забезпечення
БД	- база даних
ОС	-операційна система
SDK	- набір із засобів розробки, утиліт і документації, який дозволяє програмістам створювати прикладні програми за визначеною технологією або для певної платформи
JDK	- комплект розробника застосунків на мові Java
Java	- об'єктно-орієнтована мова програмування
UML	- Unified Modeling Language
SQLite	- полегшена реляційна система керування базами даних
API	- прикладний програмний інтерфейс
XML	- розширювана мова розмітки
GPS	- система глобального позиціонування
LTE	- мобільного протоколу передавання даних і
JSON	- формат обміну даними
IDE	- інтегроване середовище розробки
SQL-92	- третя версія SQL мови запитів до баз даних
ACID	набір властивостей, що гарантують надійну роботу транзакцій бази даних
ANSI C	- стандарт мови C
TCL	- інтерпретована мова програмування високого рівня
GitHub	- один з найбільших веб-сервісів для спільної розробки програмного забезпечення
IT	- інформаційні технології

ЗМІСТ

Вступ.....	9
1 Аналіз предметної області.....	11
1.1 Вибір керуючої частини систем.....	11
1.1.1 PIC controller.....	12
1.1.2 Пристрій Arduino	13
1.1.3 Raspberry Pi	16
1.1.4 Мікроконтролер Atmega8A	18
1.2 Вибір датчика зчитування атмосферного тиску та температури.	21
1.2.1 Датчики BMP085 та BMP180	22
1.2.2 Датчик атмосферного тиску та температури BMP280.....	25
1.2.3 Датчик атмосферного тиску та температури BME280	26
1.3 Вибір датчика зчитування вологості повітря та температури.	27
1.3.1 Датчик вологості DHT11	28
1.3.2 Датчик вологості DHT22	30
1.3.3 Датчик вологості SHT1х.....	32
1.3.4 Комбінований цифровий датчик температури і вологості AM2320.	34
1.4 Вибір модуля для передачі даних до мережі інтернет.....	35
1.4.1 Інтерфейси для передачі даних у мережі Internet.....	35
1.4.2 Wi-Fi модуль ESP8285	37
1.4.3 Wi-Fi модуль ESP8266	41
1.4.4 Wi-Fi модуль ESP32.....	43
2 Проектування системи.....	47
2.1 Розробка структурної схеми системи	47
2.2 Вибір керуючої частини для системи	47
2.3 Вибір датчика атмосферного тиску для системи	55
2.4 Вибір датчика вологості для системи	61
2.5 Вибір Wi-Fi модуля для системи.....	63
2.6 Блок-схема алгоритму роботи системи.....	79

3 Розробка системи	81
3.1 Розробка принципової схеми системи	81
3.2 Середовище розробки програмного забезпечення для системи	82
3.3 Панель керування системи	83
Висновки.....	85
Перелік джерел посилання	86
Додаток А	88
Додаток Б.....	90
Перелік графічного матеріалу	
Плакат 1 – Структура системи	
Плакат 2 – Принципова схема системи	
Плакат 3 – Взаємодія компонентів системи	
Плакат 4 – Структурна схема системи	

ВСТУП

Системи з дистанційним керуванням - це новий етап розвитку комп'ютерних систем, який значно розширює можливості збору, аналізу і розподілу даних, які людина може перетворити в інформацію і в знання. У цьому сенсі ці системи набувають величезне значення.

Ідеологія дистанційного керування спрямована на підвищення ефективності економіки за рахунок автоматизації процесів в різних сферах діяльності і виключення з них людини.

На початок 2016 року в використанні систем з дистанційним керуванням компанії орієнтуються впершу чергу на масові сегменти систем, де спонуканням кінцевих користувачів до використання рішень і сервісів системи є ринкові стимули, такі як [1]:

- розумний дім - вся інформація про ваш будинок буде знаходитися в вашому смартфоні. Від закриття дверей, до виключення газу в печі. Всі прилади будуть підвладні вам дистанційно через інтернет;

- розумний транспорт – машина яка може керуватися без допомоги водія. Вона має доступ до глобальної мережі інтернет для того щоб визначати маршрут, знаходить дороги на яких немає пробок для того щоб водій доїхав до пункту призначення максимально швидко;

- промисловий сегмент системи – всі виробничі процеси будуть детально виміряні і оптимізовані. Виробництво стане більш надійним - системи моніторингу будуть повідомляти про проблемні ділянки до того, як вони вийдуть з ладу.

Данна схема являється саме представником такої системи та може бути використана у глобальній схемі «Розумний дім». Вона включає в себе датчик температури, тиску, висоти та Wi-Fi модуль, завдяки якому можна передавати данні до глобальної мережі Інтернет.

Ідея розробки схеми була передача даних про температуру з датчиків на сервер, а після цього на телефон користувача. Це дозволяє знати точну погоду в

окремому місці, та мати її у своєму смартфоні, що збережує час. Таким чином, обрана тема бакалаврської роботи є актуальною.

Мета роботи – організувати вимірювання та контроль атмосферних показників на відстанні.

Для досягнення поставленої мети необхідно виконати наступні завдання:

- провести аналіз систем вимірювання з дистанційним керуванням;
- спроектувати власну систему вимірювання з дистанційним керуванням;
- обрати необхідне апаратне та програмне забезпечення системи;
- реалізувати систему.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Вибір керуючої частини системи

Мікроконтролер - це спеціальна мікросхема, призначена для управління різними електронними пристроями. Мікроконтролери вперше з'явилися в тому ж році, що і мікропроцесори загального призначення [2].

Було придумано об'єднати процесор, пам'ять, ПЗУ і периферію всередині одного корпусу, зовні схожого на звичайну мікросхему. З тих пір виробництво мікроконтролерів щорічно у багато разів перевищує виробництво процесорів, а потреба в них не знижується.

Мікроконтролер характеризується великим числом параметрів, оскільки він одночасно є складним програмно-керованим пристроєм і електронним приладом (мікросхемою) [1], [2].

В ході роботи мікроконтролер зчитує команди з пам'яті або порту вводу і виконує їх. Що означає кожна команда, визначається системою команд мікроконтролера. Система команд закладена в архітектурі мікроконтролера і виконання коду команди виражається в проведенні внутрішніми елементами мікросхеми певних мікрооперацій.

На сьогоднішній день існує більше 200 модифікацій мікроконтролерів, сумісних з i8051, що випускаються двома десятками компаній, і велика кількість мікроконтролерів інших типів. Популярністю у розробників користуються 8-бітові мікроконтролери PIC фірми Microchip Technology і AVR фірми Atmel, 16-бітові MSP430 фірми TI, а також 32-бітові мікроконтролери, архітектури ARM, яку розробляє фірма ARM Limited [3].

Завдяки різноманіттю мікроконтролерів, мається вибір між різними видами, та різноманіттям характеристик.

Мікроконтролери та платформи на основі яких можна побудувати систему:

- PIC controller;
- Arduino;

- Raspberry pi;
- Atmega8.

1.1.1 PIC controller

Серія мікроконтролерів, що має гарвардську архітектуру і вироблені американською компанією Microchip Technology Inc [4].

Характеристики мікроконтролера [4]:

- зовнішній та внутрішній режими тактового генератора;
- прецизійний внутрішній генератор 4МГц;
- енергозберігаючий внутрішній генератор 37кГц;
- режим зовнішнього генератора для підключення кварцового або керамічного резонатора;
- режим енергозбереження SLEEP;
- програмовані підтягуючі резистори на входах PORTB;
- сторожовий таймер WDT з окремим генератором;
- режим низьковольтного програмування;
- програмування на платі через послідовний порт (ICSP) ;
- захист коду програми;
- скидання по зниженню напруги живлення BOR;
- скидання по включенню живлення POR;
- таймер включення живлення PWRT і таймер запуску генератора OST;
- широкий діапазон напруги живлення від 2.0В до 5.5В;
- промисловий і розширений температурний діапазон;
- висока витривалість осередків FLASH / EEPROM;
- 100 000 циклів стирання / запису FLASH пам'яті програм;
- 1 000 000 циклів стирання / запису EEPROM пам'яті даних;
- період зберігання даних FLASH / EEPROM пам'яті > 100 років.

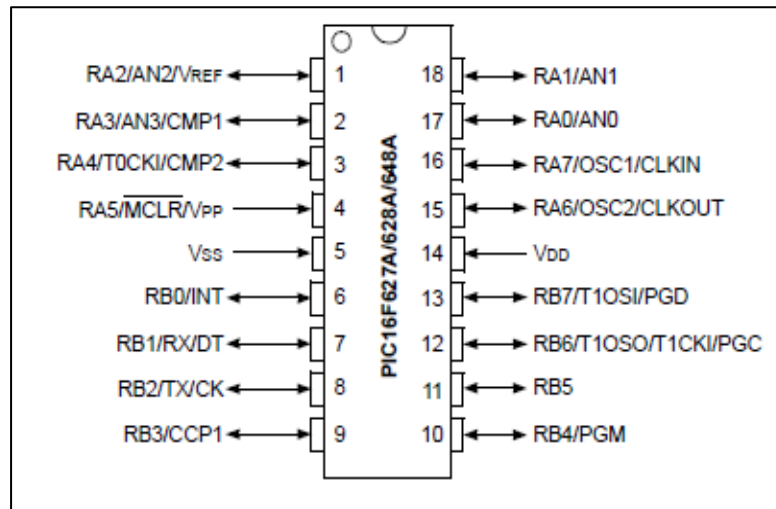


Рисунок 1.1 - Розташування виводів на Pic Controller

1.1.2 Пристрій Arduino

Мікроконтролери для Arduino відрізняються наявністю попередньо прошитого в них завантажувача. За допомогою цього завантажувача користувач завантажує свою програму в мікроконтролер без використання традиційних окремих апаратних програматорів. Завантажувач з'єднується з комп'ютером через інтерфейс USB або за допомогою окремого перехідника UART-USB [5].

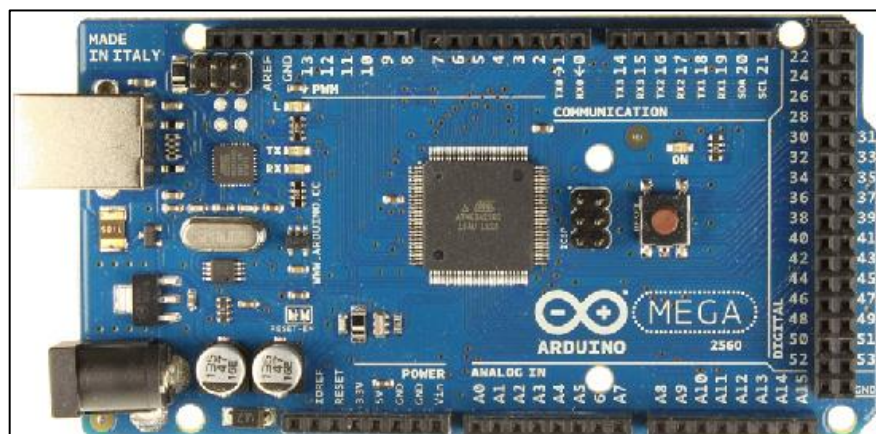


Рисунок 1.2 - Мікроконтролер Arduino

На випадок поломки завантажувача або покупки мікроконтролера без завантажувача розробники надають можливість прошити завантажувач в мікроконтролер самостійно. Для цього в Arduino IDE вбудована підтримка

декількох популярних дешевих програматорів, а більшість плат Arduino має штирьовий роз'єм для внутрисхемного програмування (ICSP для AVR, JTAG для ARM).

В Arduino IDE вбудована можливість створення своїх програмно-апаратних платформ.

Характеристики Arduino [5]:

- 14 цифрових портів входу-виходу;
- 6 аналогових входів;
- частота тактирования 16 МГц;
- USB порт;
- роз'єм внутрисхемного програмування;
- кнопка скидання.

У плати є всі необхідні компоненти для забезпечення роботи мікроконтролера. Досить підключити USB кабель до комп'ютера і подати живлення. Мікроконтролер встановлений на колодці, що дозволяє легко замінити його в разі виходу його з ладу.

Таблиця.1.1 - Характеристики Arduino

Тип мікроконтролера	ATmega328P
Напруга живлення мікроконтролера	5 В
Рекомендуема напруга живлення плати	7 – 12 В
Максимальна допустима напруга плати	6 – 20 В
Цифрові входи-виходи	14 (з них 6 підтримують ШІМ)
Виходи ШІМ модуляції	6
Аналогові входи	6
Допустимий струм цифрових виходів	20 мА
Допустимий струм виходу 3,3 В	50 мА

Продовження таблиці 1.1

Тип мікроконтролера	АТmega328Р
Об'єм флеш пам'яті (FLASH)	32 кБ (з яких 0,5 кБ використовується завантажувачем)
Об'єм оперативної пам'яті (SRAM)	2 кБ
Об'єм енергонезалежної пам'яті (EEPROM)	1 кБ
Частота тактування	16 мГц

Програмування Arduino - контролер програмується з інтегрованого середовища програмного забезпечення Ардуіно (IDE). Програмування відбувається під управлінням резидентного завантажувача по протоколу STK500. Апаратний програматор при цьому не вимагається.

Послідовний інтерфейс: виводи 0 (Rx) і 1 (Tx). Використовуються для прийому (Rx) і передачі (Tx) послідовних даних логічних рівнів TTL. Ці виводи підключені до виводам передачі даних мікросхеми АТmega16U2, яку використовують у якості моста USB-UART [6].

Зовнішні переривання: виводи 2 і 3. Ці виводи можуть бути використані як входи зовнішніх переривань. Програмно можуть бути встановлені на переривання по низькому рівню, позитивному чи негативному фронту, або на зміну рівня сигналу.

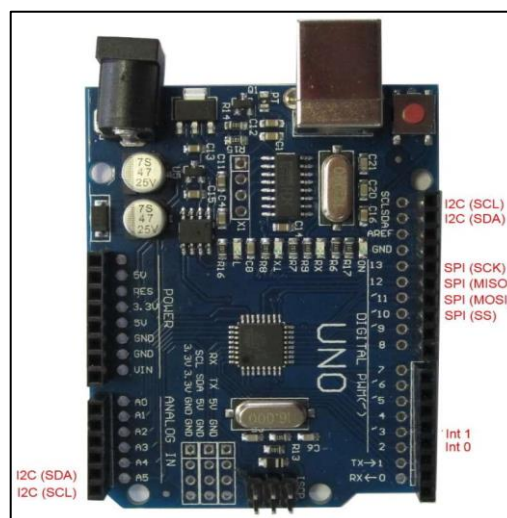


Рисунок 1.3 - Входи та виходи Arduino

ШІМ: виводи 3, 5, 6, 9, 10, 11. Чи можуть працювати в режимі ШІМ модуляції з роздільною здатністю 8 розрядів.

Послідовний інтерфейс SPI: виводи 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).

1.1.3 Raspberry Pi

Raspberry Pi є одноплатним комп'ютером, розміром з кредитну карту, розробленим британським фондом Raspberry Pi Foundation [7].

Наявна в Raspberry Pi система Broadcom BCM2835 на чіпі (SoC) включає процесор ARM1176JZF-S 700 MHz VideoCore IVGPU, і спочатку поставляється з 256 або з 512 мегабайтами оперативної пам'яті.

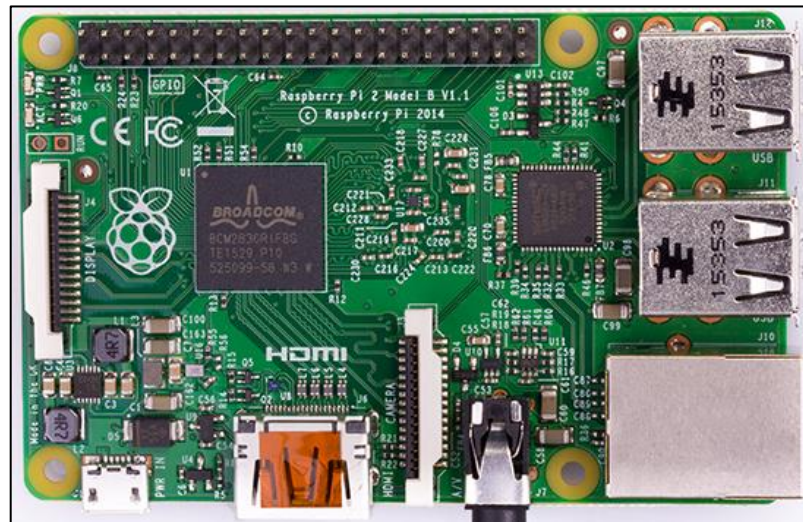


Рисунок 1.4 - Raspberry Pi 2

Модель А (RAM 256MB) і модель В (RAM 512MB) є прототипами до оригінальної британської освітньої моделі BBCMicro Computer, розробленої на базі комп'ютерів Acorn, що спочатку містили процесори типу ARM (сучасна архітектура Raspberry Pi) і операційну систему RISCOS [6], [7].

Raspberry Pi (модель В) також містить 15-контактний інтерфейс камери (CSI) роз'єм. В апаратній DSI роз'ємі інтерфейсу мобільного процесора MIPI доступна підтримка ЖК-панелей. Raspberry Pi використовує систему Linux для основи ядра операційної системи.

Raspberry Pi - ідеальний комп'ютер для постановки різних експериментів. SPC є першим доступним технічним рішенням такого розміру, яке можна використовувати повсюдно для програмування на багатьох мовах (Basic і Python встановлені за замовчуванням) і в якості мікроконтролерів для управління роботизованими пристроями [7].

Важливою характеристикою Raspberry Pi є наявність в ньому інтерфейсу GPIO

GPIO (General Purpose Input / Output) являє собою набір з 26 контактів на платі Raspberry Pi. До них відносяться 3.3V і 5V піни живлення і кілька пінов, які можуть управлятися безпосередньо процесором, і відповідно програмним забезпеченням. Вони можуть комунікувати з зовнішнім обладнанням. Як і в Arduino, настройки пінов можуть бути змінені і вони будуть функціонувати як I2C інтерфейс, UART (Universal Asynchronous Receiver / Transmitter - універсальний асинхронний приймач / передавач), SPI (Serial Peripheral Interface Bus) і PWM (Pulse-Width Modulation). Процедура програмування Raspberry Pi значно складніше, в порівнянні з Arduino, так як Raspberry Pi має багатозадачною операційною системою [8].



Рисунок 1.5 - Групи контактів для підключення різних інтерфейсів

Всі висновки GPIO можуть бути налаштовані як на введення (читання), так і на вивід (запис). Зчитування напруги може бути використано для цифрового введення / виводу. Зокрема, коли напруга менше 1,7 В, то значення дорівнює

логічному 0; більше, ніж 1.7 В є значенням логічної 1 Для контактів виведення, можна встановлювати напругу від 0 В до 3,3 В.

1.1.4 Мікроконтролер Atmega8A

Atmega8A - малопотужний 8-разр. КМОП мікроконтролер, заснований на розширеній AVR RISC - архітектурі. За рахунок виконання більшості інструкцій за один машинний цикл ATmega8 досягає продуктивності 1 млн. операцій в секунду/МГц, що дозволяє проектувальникам систем оптимізувати співвідношення енергоспоживання і швидкодії [9].

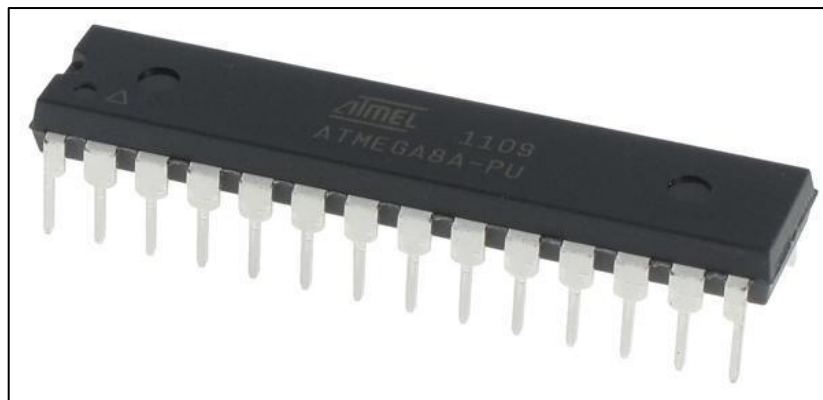


Рисунок 1.6 - Atmega8A

Особливості мікросхеми [10]:

- 8-розрядний високопродуктивний AVR мікроконтролер з малим споживанням;
- прогресивна RISC архітектура;
- 130 високопродуктивних команд, більшість команд виконується за один тактовий цикл;
- 32 8-розрядних робочих регістра загального призначення;
- вбудований 2-цикловий перемножувач;
- незалежна пам'ять програм і даних;
- 8 Кбайт внутрісистемної програмованої Flash пам'яті (In-System Self-Programmable Flash) Забезпечує 1000 циклів стирання / запису;

- додатковий сектор завантажувальних кодів з незалежними бітами блокування;
- режим одночасного читання / запису (Read-While-Write);
- 1 Кбайт вбудованої SRAM;
- програмоване блокування, що забезпечує захист програмних засобів користувача;
- два 8-розрядних таймера / лічильника з окремим попередніми дільником, один з режимом порівняння;
- один 16-розрядний таймер / лічильник з окремим попередніми дільником і режимами захоплення і порівняння;
- лічильник реального часу з окремим генератором;
- три каналу PWM;
- 8-канальний аналого-цифровий перетворювач (в корпусах TQFP і MLF);
- 6 каналів з 10-розрядною точністю;
- 2 канали з 8-розрядною точністю;
- 6-канальний аналого-цифровий перетворювач (в корпусі PDIP);
- 4 канали з 10-розрядною точністю;
- 2 канали з 8-розрядною точністю;
- 2-дротовий послідовний інтерфейс;
- програмований послідовний USART;
- послідовний інтерфейс SPI (провідний / ведений);
- програмований сторожовий таймер з окремим вбудованим генератором;
- ванаалоговий компаратор;
- спеціальні мікроконтролерні функції;
- скидання по подачі живлення і програмований детектор короткочасного зниження напруги живлення.
- вбудований калібрований RC-генератор;
- внутрішні і зовнішні джерела переривань;
- п'ять режимів зниженого споживання: Idle, Power-save, Powerdown, Standby і зниження шумів ADC;

- виводи I / O і корпусу;
- 23 програмовані лінії введення / виводу;
- 28-вивідний корпус PDIP, 32-вивідний корпус TQFP і 32-вивідний корпус MLF;

- робочі напруги: - 2,7 - 5,5 В (ATmega8L) - 4,5 - 5,5 В (ATmega8).

Опис виводів [10]:

- VCC - Напруга живлення цифрових елементів.
- GND – Заземлення.
- Port B (PB7..PB0) Порт В - 8-разр. порт двонаправленого вводу-виводу з внутрішніми XTAL1 / XTAL2 / TOSC1 / TOSC2 які підтягуються до плюса резисторами (вибираються окремо для кожного розряду). Вихідні буфери порту В мають симетричну вихідну характеристику з однаковими втікаючими і витікаючими струмами. При введенні, лінії порту В діятимуть як джерело струму, якщо зовні діє низький рівень і включені підтягує резистори. Виводи порту В знаходяться в третьому стані при виконанні умови скидання, навіть якщо синхронізація не запущена. Залежно від налаштувань фізюз генератора PB6 може бути інверсним входом генератора і введений до внутрішнього годинника через операційний підсилювач.

- Port C (PC5..PC0) Порт С - 7-розр. порт двонаправленого вводу-виводу з внутрішніми підтягуючим до плюса резисторами (вибираються окремо для кожного розряду). Вихідні буфери порту С мають симетричну вихідну характеристику з однаковими втікаючими і витікаючими струмами. При введенні, лінії порту С діятимуть як джерело струму, якщо зовні діє низький рівень і включені підтягує резистори. Виводи порту С знаходяться в третьому стані при виконанні умови скидання, навіть якщо синхронізація не запущена. Якщо фізюз RSTDISBL запрограмований, PC6 використовується в якості контакту введення-виведення.

- Port D (PD7..PD0) Порт D - 8-разр. порт двонаправленого вводу-виводу з внутрішніми підтягуючим до плюса резисторами (вибираються окремо для кожного розряду). Вихідні буфери порту D мають симетричну вихідну

характеристику з однаковими втікаючими і витікаючими струмами. При введенні, лінії порту D будуть діяти як джерело струму, якщо зовні діє низький рівень і включені підтягує резистори.

– RESET - Вхід скидання. Якщо на цей вхід докласти низький рівень тривалістю понад мінімально необхідної буде згенерує скидання незалежно від роботи синхронізації.

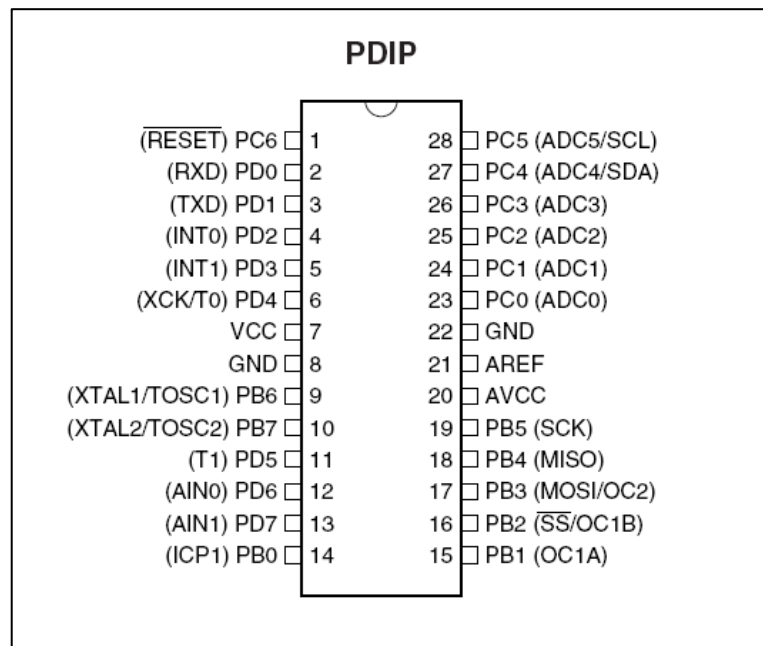


Рисунок 1.7 - Розположення виводів на Atmega8A

1.2 Вибір датчика зчитування атмосферного тиску та температури.

Датчик атмосферного тиску — це пристрій, що дозволяє отримувати і дистанційно передавати сигнал, що відповідає вимірюваному тиску.

Для цієї ролі були обрані та протестовані 4 датчики атмосферного тиску та температури:

- BMP085;
- BMP180;
- BMP280;

– ВМЕ280.

1.2.1 Датчики BMP085 та BMP180

Вимірювач тиску BMP180 не сильно відрізняється від свого попередника, лише тим що має інтерфейс I2C та більш точне вимірювання ніж у BMP085 [11].

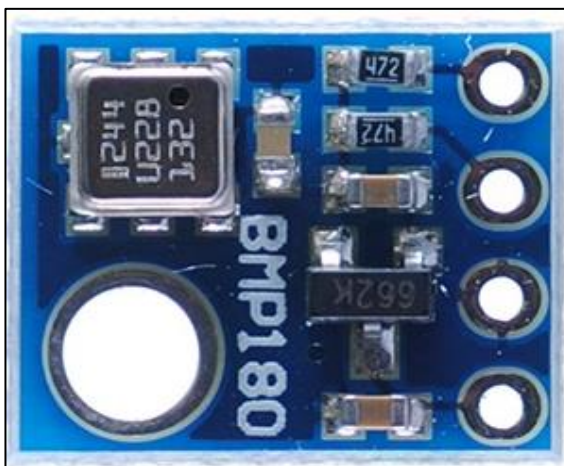


Рисунок 1.8 - BMP180

Функція BMP180 сумісна з наступником BMP085 та починає нове покоління високоточних датчиків тиску та температури для інтернет споживачів.

Інтерфейс I2C дозволяє легко інтегрувати системи з мікроконтролером.

Таблиця. 1.2 - Характеристики BMP180

Живлення датчика	1,8 ... 3,6 В
Живлення шини I2C	1,62 ... 3,6 В
Живлення модуля	3,3 В чи 5 В
Струм	3 ... 32 мкА
Робоча температура	-40 ... 80 °C
Робочий тиск	0 ... 10 МПа
Діапазон тиску	300 ... 1100 гПа
Діапазон температури	0 ... 65 °C
Час перетворень	3 ... 51 мс
Робоча частота шини I2C	3,4 МГц
Резистори на шині I2C	2,2 ... 10 кОм
Адреса датчика на шині I2C	0x77 HEX

Назва реєстрів	Адреса	Дані реєстрів								Значення при скиданні	Доступ
		7 біт	6 біт	5 біт	4 біт	3 біт	2 біт	1 біт	0 біт		
AC1	0xAA	Старший байт								константа	R
	0xAB	Молодший байт								константа	R
AC2	0xAC	Старший байт								константа	R
	0xAD	Молодший байт								константа	R
AC3	0xAE	Старший байт								константа	R
	0xAF	Молодший байт								константа	R
AC4	0xB0	Старший байт								константа	R
	0xB1	Молодший байт								константа	R
AC5	0xB2	Старший байт								константа	R
	0xB3	Молодший байт								константа	R
AC6	0xB4	Старший байт								константа	R
	0xB5	Молодший байт								константа	R
B1	0xB6	Старший байт								константа	R
	0xB7	Молодший байт								константа	R
B2	0xB8	Старший байт								константа	R
	0xB9	Молодший байт								константа	R
MB	0xBA	Старший байт								константа	R
	0xBB	Молодший байт								константа	R
MC	0xBC	Старший байт								константа	R
	0xBD	Молодший байт								константа	R
MD	0xBE	Старший байт								константа	R
	0xBF	Молодший байт								константа	R
Chip Id	0xD0	0	1	0	1	0	1	0	1	константа	R
Soft Reset	0xE0	доступен только для записи								0x00	W
Measurement control	0xF4	OSS		CSO		біти управління вимірюваннями			0x00	R/W	
Out MSB	0xF6	Старший байт результату								0x80	R
Out LSB	0xF7	Молодший байт результату								0x00	R
Out xLSB	0xF8	Доп. біти результату					0	0	0	0x00	R

Рисунок 1.9 - Таблиця реєстрів BMP085 та BMP180

Склад датчика BMP180 [11]:

- BMP180 заснований на п'єзо-резистивної технології для надійності, високої точності і лінійності, а також довгострокової стабільності;
- п'єзо-резистивний датчик (для визначення атмосферного тиску);
- термодатчик (для визначення температури);
- АЦП (аналого-цифровий перетворювач);
- EEPROM (незалежна електрично стирається перепрограмувальна пам'ять);
- RAM (енергозалежна пам'ять, іншими словами ОЗУ);
- мікроконтролер;

– передача даних організована по шинам I2C або SPI (в даному випадку використовується I2C).

Вид виміру	Записуване значення				Час виміру	Струм	Точність виміру
	OSS	CSO	4:0	Результат			
Температура	00	1	01110	00101110 = 0x2E	до 4,5 мс	3 мкА	0,5 °C
Тиск	00	1	10100	00110100 = 0x34	до 4,5 мс	3 мкА	0,6 гПа
Тиск	01	1	10100	01110100 = 0x74	до 7,5 мс	5 мкА	0,5 гПа
Тиск	10	1	10100	10110100 = 0xB4	до 13,5 мс	7 мкА	0,4 гПа
Тиск	11	1	10100	11110100 = 0xF4	до 25,5 мс	12 мкА	0,3 гПа

Рисунок 1.10 - Таблиця вимірів BMP180

Принцип дії датчика BMP180: для передачі даних використовується шина даних I2C - послідовний інтерфейс, призначений для забезпечення зв'язку в інтегральних схемах. Була розроблена в 1980 році фірмою Philips і спочатку призначалася для вбудовування в блоки телевізорів як недорогої шина внутрішнього зв'язку, яка керує електронікою. Пізніше, завдяки дешевизні і простоті реалізації, I2C стала промисловим стандартом, і її застосування поширилося далеко за межі побутових приладів [12].

На даний момент даний інтерфейс використовується для з'єднання низькошвидкісних периферійних компонентів, при регулюванні колірного балансу і контрастності моніторів, забезпечення інформаційного обміну між ЦАП, драйверами, відеопроцесорами, модулями системної пам'яті і багато чого іншого. Тільки фірма Philips виробляє більше сотні найменувань I2C-сумісних пристроїв, призначених для різного застосування. Серед них є вимірювальні прилади і датчики.

Стандартною швидкістю роботи шини I2C є 100 кбіт / с або 10 кбіт / с в режимі зниженої передачі. У 1992 році розробники зробили можливим підключення ще більшої кількості периферійних пристроїв за рахунок впровадження 10-бітної адресації і збільшили швидкість шини до 400 кбіт / с. Представлена в 1998 році версія стандарту 2.0 дозволяла працювати вже зі швидкістю до 3,4 Мбіт / с.

По конструкції шина являє собою всього два дроти, іменовані лініями SCL і SDA. Перша призначена для синхронізації всіх імпульсів які передаються по I2C. Друга відповідає за передачу даних. Лінії зв'язуються по шині з усіма пристроями на платі, які можуть бути двох типів: Master (головний) і Slave (підпорядкований).

Для управління підключеними Slave-компонентами Master формує сигнали start і stop, що визначають початок і закінчення передачі даних відповідно. При цьому сигнал start визначається переходом імпульсів лінії SDA з високого регістру в низький при наявності високого рівня сигналу від SCL, а сигнал stop ініціюється переходом SDA з низького регістра в високий при високому рівні SCL.

В процесі передачі даних від Slave до Master останній повинен підтвердити отримання імпульсів спеціальним сигналом ACK. Якщо він цього не робить, Slave припиняє передачу, розвантажуючи лінію SDA і надаючи можливість Masterу послати сигнал stop [12].

Завдяки таким принципом роботи шина I2C досить просто вирішує виникаючі проблеми із затримками сигналів і одночасної ініціалізацією обміну кількома провідними пристроями без втрати будь-яких даних, що і зумовлює її широку популярність в електронних пристроях.

1.2.2 Датчик атмосферного тиску та температури BMP280

BMP280 на фоні BMP085 та BMP180 – це досить помітний крок вперед, але в цьому є свої мінуси, і головний з них – ціна.

Перше, що слід відзначити в датчику BMP280, – розмір корпусу датчика. Він суттєво зменшився. Його 8-контактний корпус має розміри 2×2.5мм та висоту 0.95мм.

Датчик має інтерфейси I²C та SPI. (BMP085 має тільки I²C, BMP180 були моделі з I²C або SPI) [13].

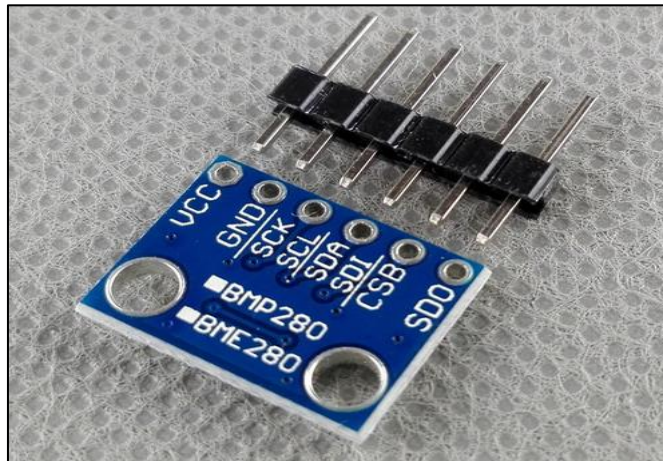


Рисунок 1.11- датчик тискуBMP280

BMP280 має три режими роботи [13]:

- SLEEP – режим сну,
- FORCED – режим аналогічний до роботи BMP085 та BMP180.
- NORMAL – цей режим загоняє датчик у режим циклу, який змушує датчик через деякий час считувати данні, зберігати у пам’яті та засинати.

Таблиця. 1.3 - Порівняння датчиків BMP180 та BMP280

Параметри	BMP180	BMP280
Розмір	3.6*3.8 мм	2.0. * 2.5 мм
Мінімальне VDD	1.80 В	1.71 В
Споживання току	12 мкА	2.7 мкА
Дозвіл тиску	1 Па	0.16 Па
Дозвіл температури	0.1 г.Ц.	0.01 г.Ц.
Інтерфейси	I2C	I2C та SPI
Режими виміру	Температура, тиск, постійний вимір	Температура, тиск, постійний чи періодичний вимір
Швидкість виміру	120 Гц	157 Гц

1.2.3 Датчик атмосферного тиску та температури BME280

Алгоритм роботи з BMP280 і BME280 однаковий. Зчитуються калібрувальні коефіцієнти і дані датчиків температури і тиску. Потім, за вказаними в документації формулами, обчислюються реальні показники температури, тиску, і вологості. Адреси регістрів (коригуючих коефіцієнтів і даних) для роботи з температурою і

тиском у цих датчиків однакові. Формули розрахунку температури і тиску ідентичні. Головне відзначення BME280 у тому що він має гігрометр. За типовим випадком гігрометр в BME280 відключений. Тільки після виконання відповідних налаштувань датчика, можна зчитати актуальні дані гігрометра. Розрахунок вологості виконується за формулою з використанням коригуючих коефіцієнтів. Діапазон вимірювання вологості від 0 до 100% [14].

Як і BMP280, BME280 має три режими роботи - Normal, Forced, Sleep



Рисунок 1.12 – Датчик тиску BME280

У режимі Normal - відповідно до параметрам датчик періодично виконує вимірювання та пропускає значення через цифровий фільтр.

У режимі Forced - датчик виконує вимірювання один раз по команді. Для цього потрібно почекати, поки датчик виконає вимірювання та тільки після цього отримати дані.

У режимі Sleep - датчик переходить у режим мінімального споживання.

1.3 Вибір датчика зчитування вологості повітря та температури

Датчик вологості повітря — це пристрій, що дозволяє отримувати і дистанційно передавати сигнал, що відповідає вимірюванню вологості.

За технічними характеристиками були обрані 4 датчики вологості:

- DHT11;
- DHT22;
- SHT1x;
- Am2320.

1.3.1 Датчик вологості DHT11

Цей датчик DFRobot DHT11 оснащений датчиком температури і вологості з відкаліброваним цифровим вихідним сигналом. Використовуючи ексклюзивну технологію отримання цифрового сигналу і технологію визначення температури і вологості, вона забезпечує високу надійність і відмінну довготривалу стабільність. Цей датчик включає в себе компонент вимірювання вологості резистивного типу і компонент вимірювання температури NTC і підключається до 8-розрядному мікроконтролеру з високою продуктивністю, забезпечуючи відмінну якість, швидку реакцію, захист від перешкод і економічну ефективність [14].

Кожен елемент DHT11 строго відкалібрований і є надзвичайно точним при калібрування вологості. Коефіцієнти калібрування зберігаються у вигляді програм в ОЗП-пам'яті, які використовуються в процесі виявлення внутрішнього сигналу датчика. Однопровідний послідовний інтерфейс робить систему швидше і простіше. Малий розмір, низьке енергоспоживання і передача сигналу до 20 метрів роблять його кращим вибором для різних додатків. Датчик являє собою 4-контактний штирьовий ряд.

Живлення DHT11 становить 3-5,5 В постійного струму. Коли живлення подається на датчик, не потрібно відправляти ніяких даних на нього протягом однієї секунди, щоб передати нестабільний статус мікросхеми. Один конденсатор вартістю 100 нФ може бути доданий між VDD і GND для фільтрації потужності [14], [15].

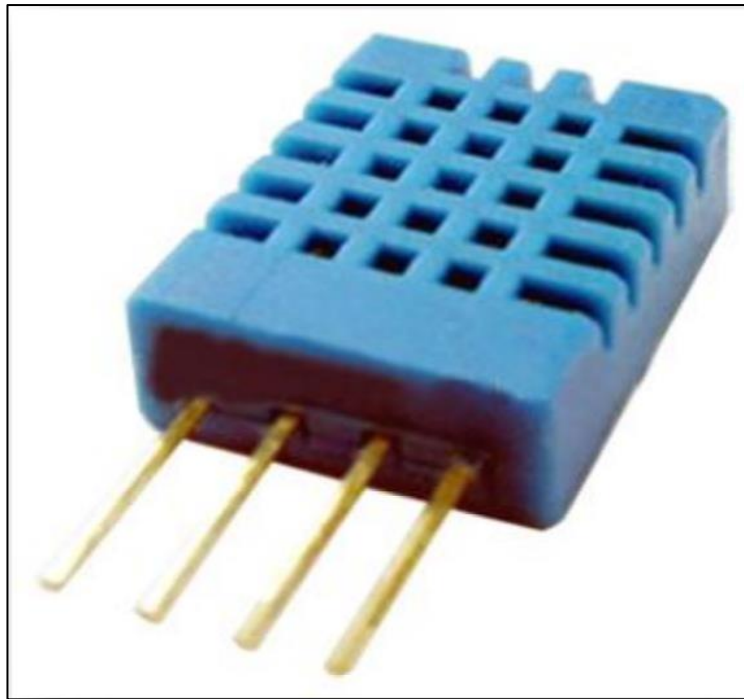


Рисунок 1.13 - Датчик вологості DHT11

Умови експлуатації датчики DHT11 [15]:

- температура зварювання повинна бути нижче 260 градусів Цельсія, а контакт повинен займати менше 10 секунд;
- уникайте використання датчика в умовах роси;
- не використовуйте цей продукт в небезпечних пристроях або аварійної зупинки або в будь-якому іншому випадку, коли відмова DHT11 може привести до травми;
- зберігання: Тримайте датчик при температурі 10-40 °C, вологості <60% відносної вологості.

Застосування датчика DHT11 за межами робочого діапазону, зазначеного в цьому технічному описі, може привести до зсуву сигналу на 3%. Датчик DHT11 може поступово відновлюватися до відкаліброваного стану, коли він повертається в нормальний робочий стан і працює в межах свого діапазону.

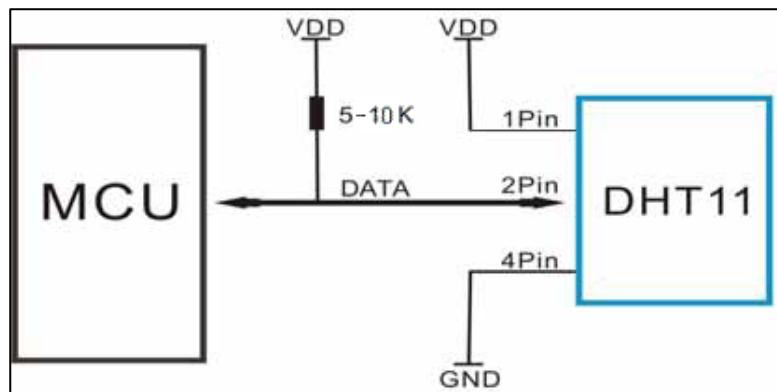


Рисунок 1.14 - Розпіновка DHT11

1.3.2 Датчик вологості DHT22

Характеристика та застосування [16]:

- повноцінний температурний компенсатор;
- відносна вологість і вимір температури;
- калібрований цифровий сигнал;
- відмінна довготривала стабільність;
- довгі відстані передачі;
- низький рівень споживання енергії;
- 4 штиря упаковані і повністю взаємозамінні.

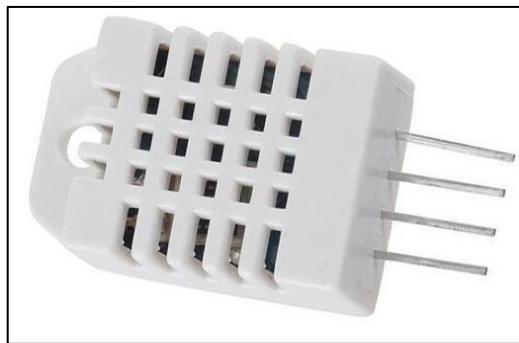


Рисунок 1.15 - Датчик DHT22

DHT22 видає відкалібрований цифровий сигнал. Він використовує ексклюзивну технологію збору цифрового сигналу і технологію визначення вологості, що забезпечує її надійність і стабільність. Його чутливі елементи з'єднані з 8-бітовим одночіповим комп'ютером.

Кожен датчик цієї моделі піддається температурної компенсації і калібрування в точної калібрувальної камері, а коефіцієнт калібрування зберігається в типі програми в ОТР-пам'яті, коли датчик виявляє, він буде вказувати коефіцієнт з пам'яті.

Малий розмір і низьке споживання і довгу відстань передачі (20 м) дозволяють використовувати DHT22 у всіх видах суворих додатків.

Однорядна упаковка з чотирма контактами, що робить з'єднання дуже зручним.

Живлення DHT22 становить 3-3,3 В постійного струму. Коли живлення подається на датчик, не потрібно відправляти ніяких даних на нього протягом однієї секунди, щоб передати нестабільний статус мікросхеми. Один конденсатор вартістю 100 нФ може бути доданий між VDD і GND для фільтрації потужності [16].

Таблиця.1.4 - Характеристики DHT22

Модель	DHT22
Джерело живлення	3.3-6 В постійного струму
Цифровий сигнал	Цифровий сигнал через одну шину
Чутливий елемент	Полімерний конденсатор
Робоча вологість	0-100% Відносної вологості;
Точність вологості	+ -2% Відносної вологості (макс. + 5% відносної вологості); Температура <+ - 0,5 за Цельсієм
Дозвіл або чутливість вологості	0,1% Відносної вологості; Температура 0,1 за Цельсієм
Повторюваність вологості	+ -1% Відносної вологості;
Температура	+ -0,2 за Цельсієм
Гістерезис вологості	+ -0,3% Відносної вологості
Довгострокова стабільність	+ -0,5% Відносної вологості / рік
Період спостереження	Середній: 2 с
Взаємозамінність	Повністю взаємозамінна
Розмір	Малий розмір 14 * 18 * 5,5 мм; Великий розмір 22 * 28 * 5 мм

1.3.3 Датчик вологості SHT1х

SHT1х - це сімейство датчиків Sensorion для вимірювання відносної вологості і температури.

Датчики об'єднують сенсорні елементи і обробку сигналів, забезпечують повністю еталонний цифровий вихід. Для вимірювання відносної вологості використовується унікальний ємнісний сенсорний елемент, а температура вимірюється датчиком зони зазору. Застосовувана технологія CMOSens® гарантує чудову надійність і довготривалу стабільність. Обидва датчика легко підключаються до 14-розрядному аналого-цифровому перетворювача і схемою послідовного інтерфейсу. Це забезпечує чудову якість сигналу, швидкий час відгуку і нечутливість до зовнішніх перешкод (ЕМС). Кожен SHT1х індивідуально відкалібрований в камері з точною вологістю. Коефіцієнти калібрування запрограмовані в ОТР-пам'ять на чіпі. Ці коефіцієнти використовуються для внутрішньої калібрування сигналів від датчиків. Двопровідний послідовний інтерфейс і внутрішнє регулювання напруги дозволяють легко і швидко інтегрувати систему. Крихітний розмір і низьке енергоспоживання роблять SHT1х ідеальним вибором навіть для найвимогливіших додатків [17].

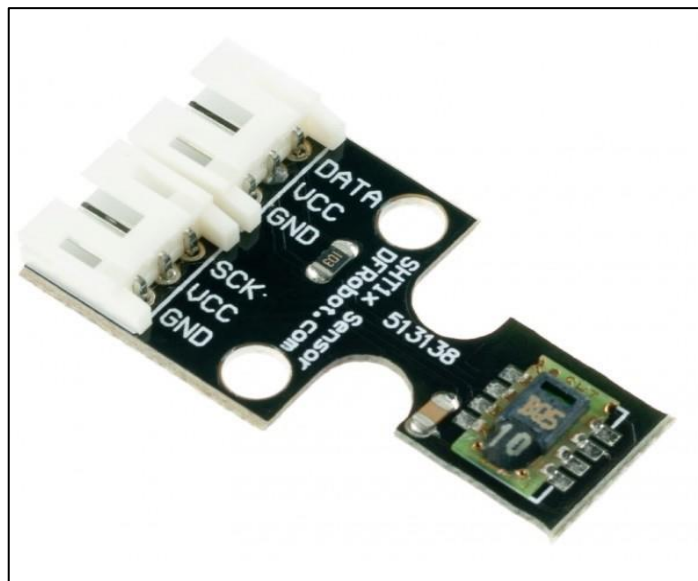


Рисунок 1.16 – Датчик вологості SHT1х

Штирі (VDD, GND) - Напруга живлення SHT1х має бути в діапазоні 2,4 - 5,5 В, рекомендована напруга - 3,3 В. Напруга живлення штирів (VDD) і заземлення (GND) має бути розділене конденсатором на 100 нФ. Послідовний інтерфейс SHT1х оптимізований для зчитування датчиків і ефективного споживання енергії. Сенсор не може бути адресований протоколом I2C, однак датчик може бути підключений до шини I2C без перешкод іншим пристроям, підключеним до шини [17].

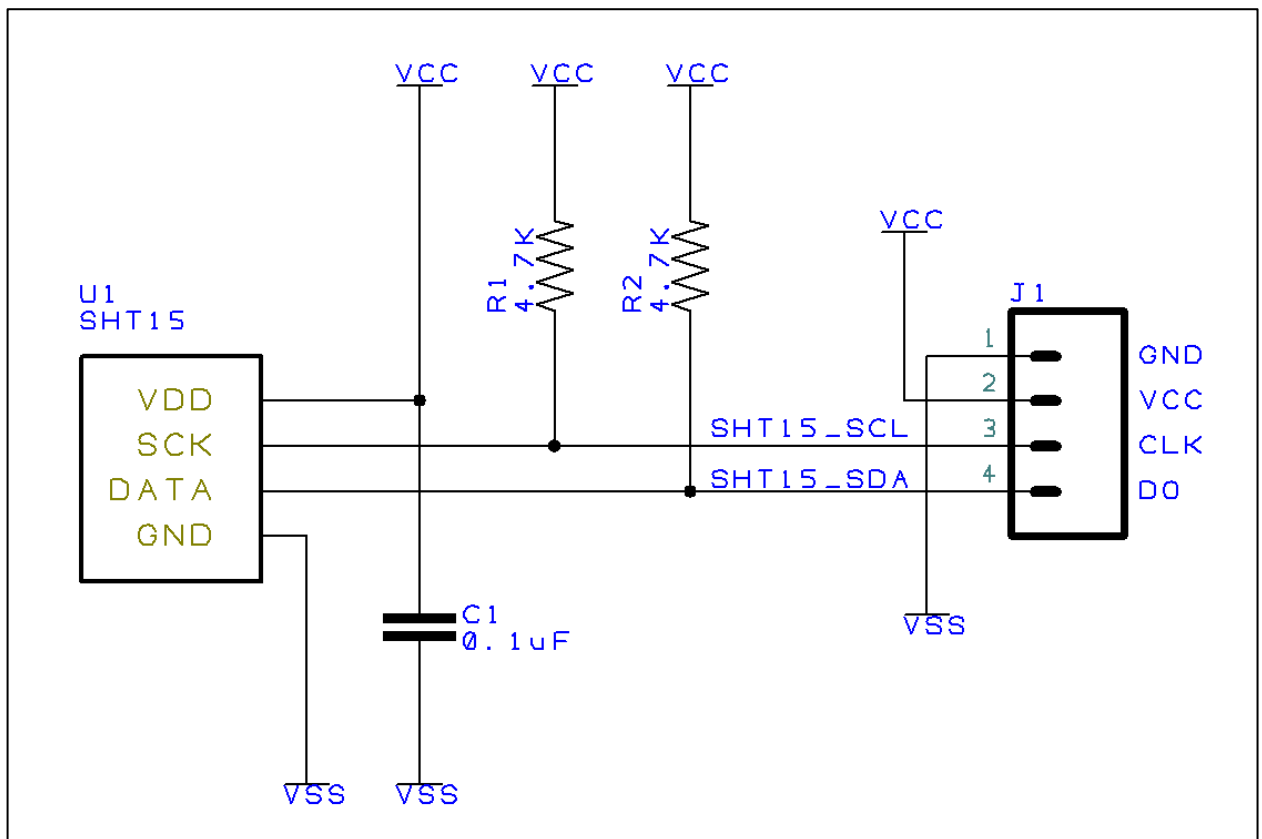


Рисунок 1.17 - Розпіновка SHT1х

Характеристики SHT1х:

- діапазон температур: -40 ... + 128 ° C;
- погрішність температури: $\pm 0,3$ ° C (при 25 ° C);
- діапазон вологості: 0-100%;
- погрішність вологості: $\pm 2\%$ (при вологості 10-90%);
- час реакції: не більше 4 с;

– номінальний струм: 6 мкА.

1.3.4 Комбінований цифровий датчик температури і вологості AM2320

Використовуючи спеціальну технологію визначення температури і вологості, датчик має дуже високу надійність і відмінну довготривалу стабільність. Датчик складається з ємнісного елемента вологості і вбудованих високоточних приладів для вимірювання температури та підключений до високопродуктивної процесору. Продукт володіє відмінною якістю і швидкою відповіддю мікроконтролеру. AM2320 використовує одну шину та два режими обміну по протоколу I2C. Малий розмір, низьке енергоспоживання, відстань передачі сигналу до 20 метрів, роблять датчик кращим вибором для програм вимірювання атмосферного тиску. I2C використовує стандартну послідовність зв'язку, користувач може безпосередньо підключитися до комунікаційної шини I2C без додаткової проводки. Для вологості, температури і іншої цифрової інформації використовуються два режими зв'язку. Контрольна сума CRC з компенсацією температури, користувачам немає необхідності обчислювати вторинний цифровий вихід, і немає необхідності в температурній компенсації вологості [17].

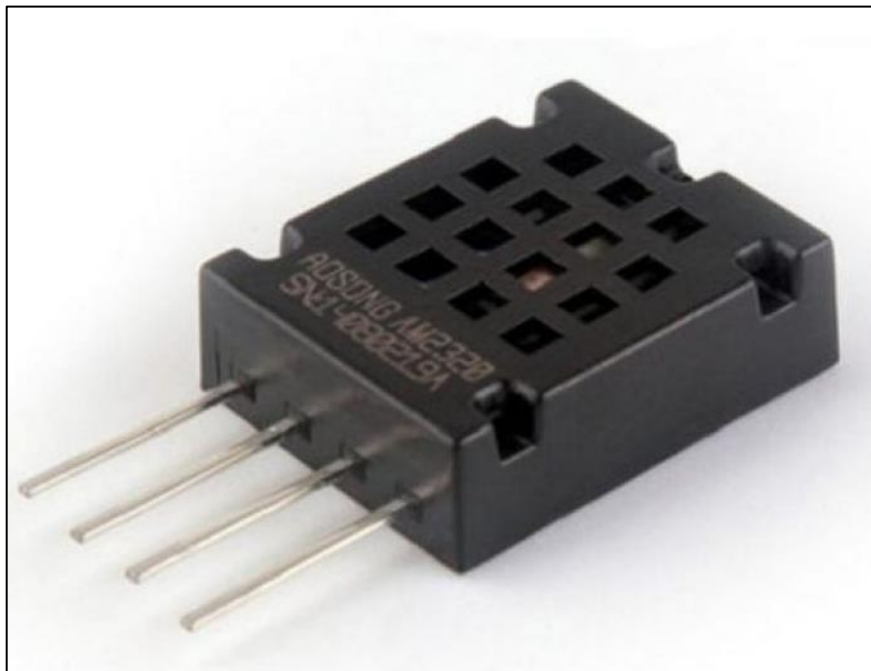


Рисунок 1.18 - Датчик вологості AM2320

Характеристики АМ2320 [17]:

- а) інтерфейс: I2C / 1-провідний;
- б) зовнішні компоненти: підтягуючі резистори 10 кОм (I2C) / 5,1 кОм (1-провід);
- в) відстань між висновками: 2,54 мм;
- г) живлення: 3,1 ... 5,5 (номінально 5В);
- д) Струм:
 - 1) в режимі очікування: 8 ... 10 мкА;
 - 2) в режимі вимірювань: 350 мкА, піковий до 950 мкА;
- е) вимірювання температури:
 - 1) допустимий діапазон: $-40^{\circ}\text{C} \dots 80^{\circ}\text{C}$;
 - 2) максимальна погрішність: $\pm 0,5^{\circ}\text{C}$;
- ж) дозвіл шкали: $0,1^{\circ}\text{C}$;
- з) вимірювання відносної вологості:
 - 1) допустимий діапазон: 0 ... 99,9%;
 - 2) максимальна погрішність: $\pm 3\%$ при $t = 25^{\circ}\text{C}$;
- и) дозвіл шкали: 0.1%;
- к) догляд показань: 0,5% / рік;
- л) мінімальний час між зчитування показань: 2 сек.

1.4 Вибір модуля для передачі даних до мережі інтернет

1.4.1 Інтерфейси для передачі даних у мережі Internet

Існує багато способів передачі даних до мережі інтернет і серед них є різні інтерфейси, такі як: Bluetooth, Wi-Fi, Ethernet та GSM

Bluetooth — технологія бездротового зв'язку, створена у 1998 році групою компаній: Ericsson, IBM, Intel, Nokia, Tshiba.

Нині розробки в області Bluetooth ведуться групою англ. Bluetooth SIG (англ. Special Interest Group), до якої входять також Lucent, Microsoft та інші компанії, чия діяльність пов'язана з мережними технологіями. Основне

призначення Bluetooth — забезпечення економного (з точки зору споживаного струму) і дешевого радіозв'язку між різноманітними типами електронних пристроїв, таких як мобільні телефони та аксесуари до них, портативні та настільні комп'ютери, принтери та інші.

Ethernet - сімейство технологій пакетної передачі даних для комп'ютерних мереж.

Стандарти Ethernet визначають дротяні з'єднання і електричні сигнали на фізичному рівні, формат кадру та протоколи управління доступом до середовища - на канальному рівні моделі OSI. Ethernet в основному описується стандартами IEEE групи 802.3. Ethernet став найпоширенішою технологією ЛВС в середині 1990-х років, витіснивши такі застарілі технології, як ARCNET і Token ring.

GSM - Глобальна система мобільного зв'язку (англ. Global System for Mobile Communications, раніше фр. Groupe Spécial Mobile, GSM) — міжнародний стандарт для мобільного цифрового стільникового зв'язку з розділенням каналу за принципом TDMA та високим рівнем безпеки за рахунок шифрування з відкритим ключем [18].

WiFi — торгова марка Wi-Fi Alliance та загальновживана назва для стандарту IEEE 802.11 передачі цифрових потоків даних по радіоканалах. Обладнання, що відповідає стандарту IEEE 802.11.

Стандарт був розроблений під патронатом Європейського інституту стандартизації електрозв'язку (ETSI) наприкінці 1980-х років.

Ці програми дозволяють розробнику приладу пристосувати її під управління його приладом і передати користувачеві готове рішення. Існує кілька популярних реалізацій концепції системи в плані обміну даними по мережі:

HTML сервер на ESP8266. Контроль і управління пристроєм ведеться через браузер. Великовагове рішення, підходить автономним пристроїв автоматики.

AllJoyn - набирає популярність відкритий протокол великого альянсу виробників цифрової техніки «Allseen».

HTTP запити з використанням протоколів типу REST, XML-RPC (SOAP). Для цього на Wi-Fi модулі запускають спрощений HTTP сервер, без HTML.

Гідність методу - відсутність проблем з налаштуванням файрволлов, HTTP зазвичай відкритий завжди [17], [18].

MQTT. Це простий протокол поверх TCP / IP. Існує велика кількість застосунків верхнього рівня для Android, iOS і інших платформ, що підтримують цей протокол.

SNMP. Розширюваний протокол управління мережевими пристроями. Основний недолік в тому що в більшості мереж файрволли блокують проходження SNMP.

ModBus і інші протоколи промислової автоматизації.

Wi-Fi модулі які підійшли для роботи з системою:

- ESP8285;
- ESP8266;
- ESP32.

1.4.2 Wi-Fi модуль ESP8285

ESP8285 забезпечує високо інтегроване рішення Wi-Fi модулів для задоволення постійно зростаючих вимог користувачів до ефективного використання енергії, компактному дизайну і надійній роботі з системою. Завдяки повній і автономній мережі Wi-Fi, ESP8285 може працювати як автономний модуль, так і підлеглий сервера. Коли ESP8285 запускається на ньому можна побачити світло, що означає готовність роботи датчика. Вбудований високошвидкісний кеш допомагає підвищити продуктивність системи і оптимізувати системну пам'ять. Крім того, ESP8285 можна застосовувати до будь-якої конструкції мікроконтролера в якості адаптера Wi-Fi через інтерфейси SPI / SDIO або I2C / UART [19].

ESP8285 об'єднує антенні перемикачі, підсилювач потужності, низький рівень шуму, підсилювачі, фільтри і модулі управління живленням. Компактна конструкція мінімізує розмір друкованої плати і вимагає мінімальних зовнішніх робіт. Крім функцій Wi-Fi, ESP8285 також інтегрує розширену версію 32-бітного процесора Tensilica L106 Diamond і вбудованого SRAM. Він може бути пов'язаний

із зовнішніми датчиками і іншими пристроями через GPIO. Комплект розробки програмного забезпечення (SDK) надає приклади коду для різних додатків.

Платформа Smart Connectivity Platform (ESCP) Espressif Systems дозволяє використовувати складні функції, включаючи швидкий перехід між режимом сну і повної роботи для енергозберігаючих цілей, попередню обробку сигналів, і механізми для роботи з стільниковим зв'язком, Bluetooth, DDR, LVDS.

IEEE 802.11 — набір стандартів для комунікації в бездротовій локальній мережевій зоні (WLAN) частотних діапазонів 2.4, 3.6 і 5 ГГц. Їх випрацював і підтримує комітет зі стандартів LAN/MAN (IEEE 802) Інституту інженерів з електротехніки та електроніки (IEEE), які визначають взаємодію бездротових комп'ютерних мереж. Базова версія стандарту IEEE 802.11—2007 зазнала наступних доповнень. IEEE 802.11 — початковий стандарт бездротових локальних мереж, заснований на бездротовій передачі даних у діапазоні 2.4 ГГц [19].

Таблиця.1.5 - Головні технічні характеристики ESP8285

Категорія	Пункт	Параметри
Wi-Fi	Стандарти	FCC/CE/TELEC/SRRC
	Протоколи	802.11 b/g/n/e/i
	Діапазон частот	2.4G ~ 2.5G (2400M ~ 2483.5M)
	Тх Потужність	802.11 b: +20 dBm, 802.11 b: +20 dBm
	Антенa	PCB трасування, зовнішній, IPEX роз'єм, керамічний чіп
	Rx Чутливість	802.11 b: -91 dbm (11 Mbps)
Апаратура	Робоча напруга	2.5V ~ 3.6V
	Робочий струм	80 mA
	Діапазон температур	-40°C ~ 125°C
	Діапазон температур зберігання	-40°C ~ 125°C
	Розмір пакета	QFN32-pin (5 mm x 5 mm)
	Зовнішній інтерфейс	-

Продовження таблиці 2.5

Категорія	Пункт	Параметри
Програмне забезпечення	Wi-Fi Режим	Station/SoftAP/SoftAP+Station
	Захист	WPA/WPA2
	Шифрування	WEP/TKIP/AES
	Мережеві протоколи	IPv4, TCP/UDP/HTTP/FTP
	Конфігурацій користувача	Ат команди, сервер, дополнение для Androit/IOS

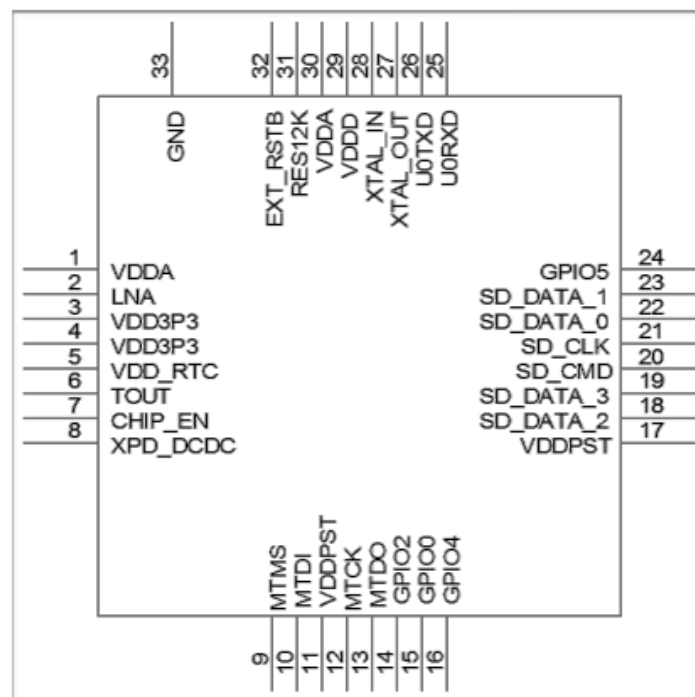


Рисунок 1.19 - Розпіновка ESP8285

Інтерфейс введення / виводу загального призначення (GPIO) ESP8285 має 17 контактів GPIO, які можуть бути призначені різних функцій, програмуючи відповідні регістри.

Ці контакти можуть бути мультиплексованих з іншими функціями, такими як I2C, I2S, UART, PWM, IR

Для операцій з низьким енергоспоживанням GPIO також можуть бути налаштовані на збереження свого стану. Наприклад, коли чіп відключений, все вихідні дозволу можуть бути встановлені на низьке.

Процесор та пам'ять ESP8285

Процесор включає в себе інтерфейси [19]:

- інтерфейс ОЗУ (DBus), який може бути підключений до контролера пам'яті;
- АНВ інтерфейс, який може бути використаний для регістра ESP8285 Wi-Fi

SoC має контролер пам'яті і модулі пам'яті, включаючи SRAM і ПЗУ. MCU може отримати доступ до пам'яті пристрою через інтерфейси IBUS, Dbus і АНВ. Всі блоки пам'яті можуть бути доступні за запитом, в той час як арбітр пам'яті буде вирішувати, що йде послідовність відповідно до часу, коли ці запити будуть отримані за допомогою процесора.

Розмір ОЗУ <50 кБ, коли ESP8285 працює в режимі станції і підключається до маршрутизатора, + секція даних близько 50 кБ.

Wi-Fi ESP8285 реалізує технологію TCP / IP, повний 802.11 b/g/n/e/i WLAN MAC протокол. Він підтримує не тільки базовий набір послуг (BSS) але і P2P за для групової роботи, сумісної з протоколом Wi-Fi P2P. Функції протоколу низького рівня обробляються автоматично датчиком ESP8285.

P2P Wi-Fi Direct має пасивне або активне сканування, а також процедура виявлення P2P виконується в автономному режимі, як тільки була ініційована відповідною командою.

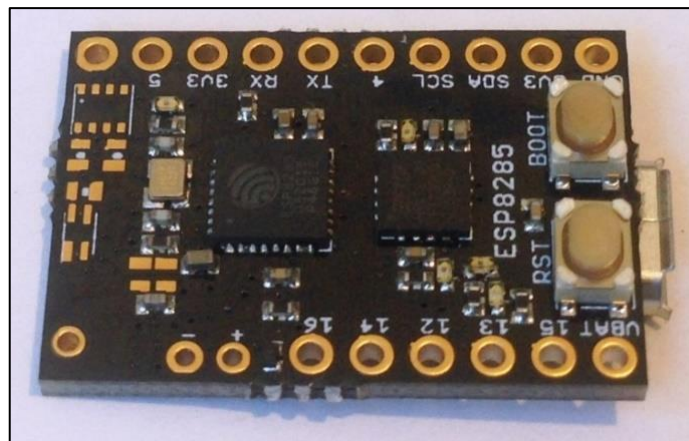


Рисунок 1.20 - Wi-Fi ESP8285

1.4.3 Wi-Fi модуль ESP8266

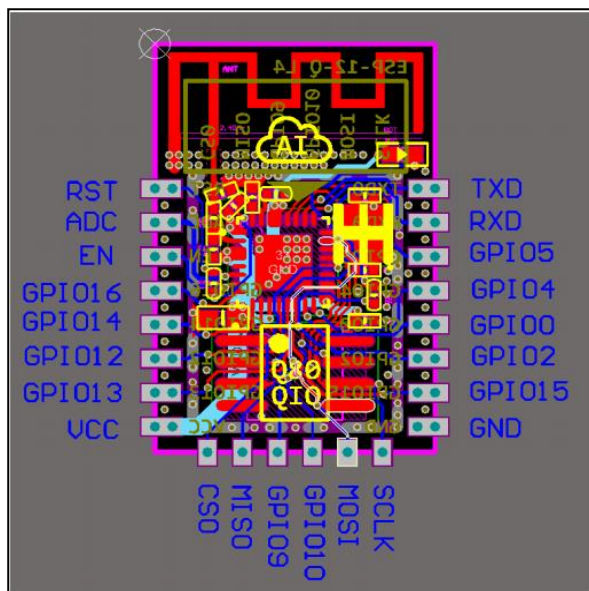
Модуль ESP-12F WiFi розроблений командою Ai-thinker. Основний процесор ESP8266 Tensilica L106, об'єднує в собі ультрасучасні 32-розрядні мікроконтролери з ультранизьким енергоспоживанням з 16-розрядним режимом, підтримкою тактової частоти 80 МГц, 160 МГц, підтримкою RTOS, інтегрованим Wi-Fi MAC / BB / RF / PA / LNA і бортовий антени. Модуль підтримує стандартну угоду IEEE802.11 b / g / n, повний пакет протоколів TCP / IP. Користувачі можуть використовувати модуль для додавання до існуючої мережі або створювати окремий сервер. ESP8266 - це високоінтегрований бездротовий SOC, призначений для розробників мобільних платформ з обмеженим простором і потужністю. Він забезпечує неперевершену здатність вбудовувати можливості Wi-Fi в інші системи або функціонувати як окремий додаток з мінімальними витратами і мінімальними вимогами до пристроїв [20].

Таблиця. 1.6 - Головні технічні характеристики ESP8266

Категорія	Пункт	Параметри
Wi-Fi	Стандарти	FCC/CE/TELEC/SRRC
	Протоколи	802.11 b/g/n
	Діапазон частот	2.4G ~ 2.5G (2400M ~ 2483.5M)
	Тх Потужність	802.11 b: +20 dBm, 802.11 b: +20 dBm
	Антенa	PCB трасування, зовнішній, IPEX роз'єм, керамічний чіп
	Rx Чутливість	802.11 b: -91 dbm (11 Mbps)
Апаратура	Процесор	Tensilica L106 32-bit micro controller
	Переферійні інтерфейси	UART/SDIO/SPI/I2C/I2S/IR Remote Control GPIO/ADC/PWM
	Робоча напруга	2.5V ~ 3.6V
	Робочий струм	80 mA
	Діапазон робочих температур	-40°C ~ 125°C

Продовження таблиці 1.6

Категорія	Пункт	Параметри
Апаратура	Розмір пакета	QFN32-pin (5 mm x 5 mm)
	Зовнішній інтерфейс	-
Програмне забезпечення	Wi-Fi Режим	Station/SoftAP/SoftAP+Station
	Захист	WPA/WPA2
	Шифрування	WEP/TKIP/AES
	Мережеві протоколи	IPv4, TCP/UDP/HTTP/FTP
	Конфігурацій користувача	Ат команди, сервер, дополнение для Androit/IOS



2.21 - Розпіновка ESP8266

Таблиця. 1.7 - Функції контактів ESP8266 12-f

Контакт	Описание функции
RST	Рестарт модуля
ADC	Результат перетворення А / D. Діапазон вхідної напруги 0 ~ 1 В, в Діапазон: від 0 до 1024
EN	Включення чіпу

Продовження таблиці 1.7

Контакт	Описание функции
IO16	GPIO16; Робить режим сну при прийомі на RST.
IO12	GPIO12; HSPI_MISO
Контакт	Описание функции
VCC	Живлення 3.3 V
CS0	Вибір чіпа
MISO	Вхідний сигнал ведучого пристрою
IO9	GPIO9
IO10	GPIO10
MOSI	Вихідні сигнал ведучого пристрою
SCLK	Час
GND	Заземлення
IO15	GPIO15; MTDO; HSPICS; ART0_RTS
IO2	GPIO15; TDO; HSPICS; UART0_RTS
IO0	GPIO0
IO4	GPIO4
IO5	GPIO5
RXD	UART0_RXD; GPIO3
TXD	UART0_RXD; GPIO3

1.4.4 Wi-Fi модуль ESP32

Espressif ESP32 - високо інтегрований, поєднаний (Wi-Fi + Bluetooth) чіп, виконаний для рішень, що вимагають мінімальних показників енергоспоживання.

ESP32 розроблений для переносної електроніки і додатків інтернету речей, виконаний в супер-мініатюрному корпусі 6 x 6 мм, що вимагає для інтеграції близько 10-ти зовнішніх компонентів. Однією з особливостей рішення є наднизьке споживання і гнучкий вибір «сплячих» режимів, що дозволяють отримати цифри до 20мкА (deep sleep mode).

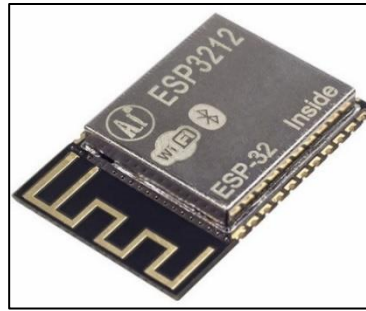


Рисунок 1.22 - Wi-Fi модуль ESP-32

У чіпсеті вже інтегровані:

- приймально-передавальна RF частина;
- підсилювач потужності;
- управління живленням.

ESP32 підтримує весь стек протоколів стандартів Wi-Fi 802.11n і BT4.2, забезпечуючи даний функціонал через інтерфейси SPI / SDIO або I²C / UART.

Чіп Espressif ESP 32 може працювати в якості центрального процесора (підтримка Open CPU) і як підпорядковане пристрій (slave device), кероване мікроконтролером [20].

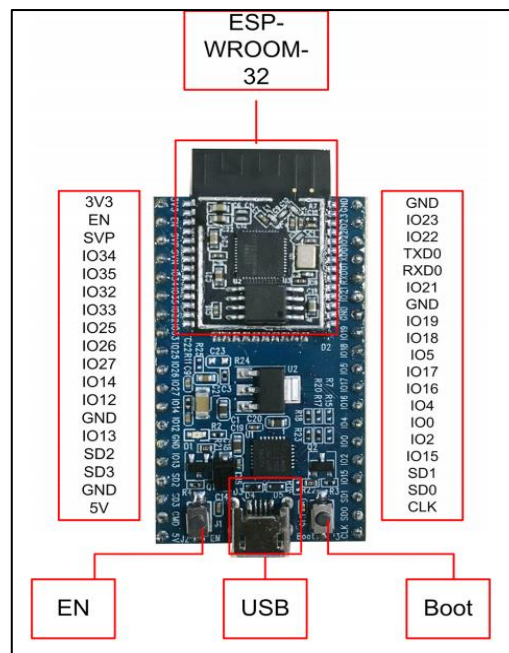


Рисунок 1.23 - Розпіновка Wi-Fi модуля ESP32

Таблиця. 1.8 - Технічні характеристики ESP32

Wi-Fi	802.11 bgn до 150 Мбит/с 4 MAC інтерфейса AMPDU
Bluetooth	- Підтримка BR / EDR / LE - Підтримка передавача і класу 1 і класу 2
CPU	- Двоядерний Xtensa LX6 до 400DMIPS - До 600DMIPS при напрузі ядра 1.2 В - Розмір коду на 15% менше - 6 рівнів переривань
Внутрішня RAM для призначеного для користувача ПО	500kB, 200kB для користувачів
Flash	4 МБ простору що настраюється відображення таблиці для OTA
Внутрішньо LDO для зовнішньої Flash	1.8 В / 3.3 В Підтримка відключення живлення Flash під час «глибокого сну»
SD/eMMC/SDIO Host	Так
SPI	4
GPIO	32
UART	2 з DMA
LED PWM	- підтримується апаратно, 16 каналів, - таймери 20 bit, що працюють на тактовій частоті 80 МГц - 13bit точності при 1 кГц - 15bit точності при 250 Гц
Motor PWM Pluse-counter	Апаратно
Віддалене управління	Підтримується апаратно
Таймер	2 таймера 32 bit і 2 таймера 64 bit 1 RTC-таймер 48 bit
Сторожовий таймер	2 системних Watchdog і 1 RTC watch dog
ADC	16 каналів, 12 bit, підтримка аналогового підсилювача Low Noise Analog Amplifier

Продовження таблиці 1.8

Wi-Fi	802.11 bgn до 150 Мбит/с 4 MAC інтерфейса AMPDU
DAC	2 канала, 10 bit
32k Crystal	Підтримка для режимів «сну»
Датчик сенсора	Підтримується, 10 каналів
Датчик температури	Так
Шифрування Flash	Підтримується апаратно
E-fuse	1 kB, записується користувачем, включаючи MAC-адреси
AES	Підтримка AES-128, AES-192 і AES-256 стандарту FIPS PUB 197
SHA	підтримка SHA-1, SHA-256, SHA-384 і SHA-512 стандарту FIPS PUB 180-4
Відновлення пам'яті в RTC	8Kbyte
ULP-співпроцесор	Підтримка контролю датчиків в режимі «глибокого сну»

Відмінні особливості:

- підтримка dual mode Bluetooth: «classic» і BLE;
- швидкість Wi-Fi: 802.11 b / g / n до 150 Мбіт / с;
- підтримка режимів Wi-Fi: клієнт, точка доступу, Sniffer, Wi-Fi Direct;
- мінімальна чутливість -98 dBm;
- широкий діапазон робочих температур: -40 ° C ... + 125 ° C;

2 ПРОЕКТУВАННЯ СИСТЕМИ

2.1 Розробка структурної схеми системи

Структурна схема — схема, яка визначає основні функціональні частини виробу, їх взаємозв'язки та призначення. Під функціональною частиною розуміють складову частину схеми: елемент, пристрій, функціональну групу, функціональну ланку.

Структурна схема призначена для відображення загальної структури пристрою, тобто його основних блоків, вузлів, частин та головних зв'язків між ними. Із структурної схеми повинно бути зрозуміло, навіщо потрібний даний пристрій і як він працює в основних режимах роботи, як взаємодіють його частини. Позначення елементів структурної схеми можуть обиратись довільно, хоча загальноприйнятих правил виконання схем слід дотримуватись [21].

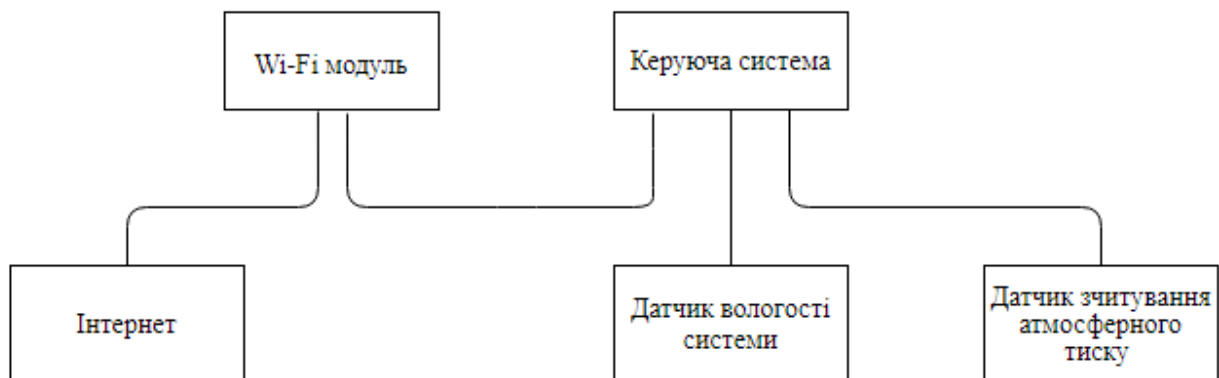


Рисунок 2.1– Структурна схема системи

2.2 Вибір керуючої частини для системи

У якості основи для системи була обрана мікросхема Atmega8A. Ця мікросхема доступна по ціні, дає можливість прошивати її на мові С у компіляторі

AVR CodeVision, та має усі функції щоб виконати роль ядра у системи для обробки атмосферних показників.

Мікроконтролер Atmega8A

Atmega8A - малопотужний 8-разр. КМОП мікроконтролер, заснований на розширеній AVR RISC - архітектурі. За рахунок виконання більшості інструкцій за один машинний цикл ATmega8 досягає продуктивності 1 млн. операцій в секунду/МГц, що дозволяє проектувальникам систем оптимізувати співвідношення енергоспоживання і швидкодії [9], [10].

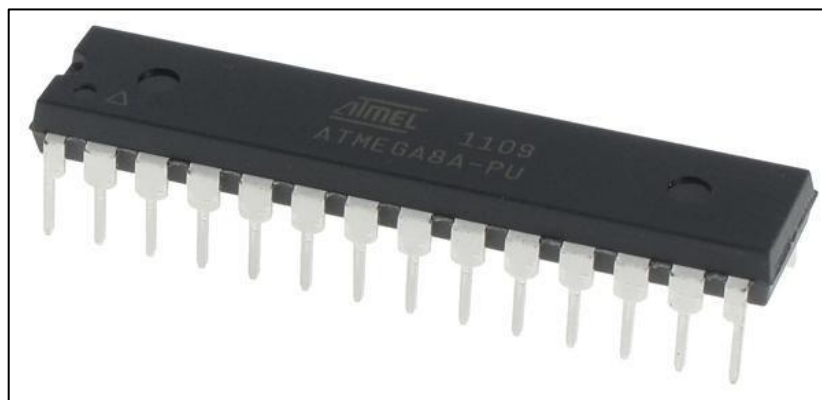


Рисунок 2.2 - Мікроконтролер Atmega8A

Особливості мікросхеми [10]:

- 8-розрядний високопродуктивний AVR мікроконтролер з малим споживанням;
- прогресивна RISC архітектура;
- 130 високопродуктивних команд, більшість команд виконується за один тактовий цикл;
- 32 8-розрядних робочих регістра загального призначення;
- вбудований 2-цикловий перемножувач;
- незалежна пам'ять програм і даних;
- 8 Кбайт внутрісистемної програмованої Flash пам'яті (In-System Self-Programmable Flash) Забезпечує 1000 циклів стирання / запису;

- додатковий сектор завантажувальних кодів з незалежними бітами блокування;
- режим одночасного читання / запису (Read-While-Write);
- 1 Кбайт вбудованої SRAM;
- програмоване блокування, що забезпечує захист програмних засобів користувача;
- два 8-розрядних таймера / лічильника з окремим попередніми дільником, один з режимом порівняння;
- один 16-розрядний таймер / лічильник з окремим попередніми дільником і режимами захоплення і порівняння;
- лічильник реального часу з окремим генератором;
- три каналу PWM;
- 8-канальний аналого-цифровий перетворювач (в корпусах TQFP і MLF);
- 6 каналів з 10-розрядною точністю;
- 2 канали з 8-розрядною точністю;
- 6-канальний аналого-цифровий перетворювач (в корпусі PDIP);
- 4 канали з 10-розрядною точністю;
- 2 канали з 8-розрядною точністю;
- 2-дротовий послідовний інтерфейс;
- програмований послідовний USART;
- послідовний інтерфейс SPI (провідний / ведений);
- програмований сторожовий таймер з окремим вбудованим генератором;
- ванаалоговий компаратор;
- спеціальні мікроконтролерні функції;
- скидання по подачі живлення і програмований детектор короткочасного зниження напруги живлення.
- вбудований калібрований RC-генератор;
- внутрішні і зовнішні джерела переривань;
- п'ять режимів зниженого споживання: Idle, Power-save, Powerdown, Standby і зниження шумів ADC;

- виводи I / O і корпусу;
- 23 програмовані лінії введення / виводу;
- 28-вивідний корпус PDIP, 32-вивідний корпус TQFP і 32-вивідний корпус MLF;

- Робочі напруги: - 2,7 - 5,5 В (ATmega8L) - 4,5 - 5,5 В (ATmega8).

Опис виводів:

1. VCC - Напруга живлення цифрових елементів.

2. GND – Заземлення.

3. Port B (PB7..PB0) Порт В - 8-разр. порт двонаправленого вводу-виводу з внутрішніми XTAL1 / XTAL2 / TOSC1 / TOSC2 які підтягуються до плюса резисторами (вибираються окремо для кожного розряду). Вихідні буфери порту В мають симетричну вихідну характеристику з однаковими втікаючими і витікаючими струмами. При введенні, лінії порту В діятимуть як джерело струму, якщо зовні діє низький рівень і включені підтягує резистори. Виводи порту В знаходяться в третьому стані при виконанні умови скидання, навіть якщо синхронізація не запущена. Залежно від налаштувань фіюз генератора PB6 може бути інверсним входом генератора і введений до внутрішнього годинника через операційний підсилювач.

4. Port C (PC5..PC0) Порт С - 7-розр. порт двонаправленого вводу-виводу з внутрішніми підтягуючим до плюса резисторами (вибираються окремо для кожного розряду). Вихідні буфери порту С мають симетричну вихідну характеристику з однаковими втікаючими і витікаючими струмами. При введенні, лінії порту С діятимуть як джерело струму, якщо зовні діє низький рівень і включені підтягує резистори. Виводи порту С знаходяться в третьому стані при виконанні умови скидання, навіть якщо синхронізація не запущена. Якщо фіюз RSTDISBL запрограмований, PC6 використовується в якості контакту введення-виведення.

5. Port D (PD7..PD0) Порт D - 8-разр. порт двонаправленого вводу-виводу з внутрішніми підтягуючим до плюса резисторами (вибираються окремо для кожного розряду). Вихідні буфери порту D мають симетричну вихідну

характеристику з однаковими втікаючими і витікаючими струмами. При введенні, лінії порту D будуть діяти як джерело струму, якщо зовні діє низький рівень і включені підтягує резистори.

6. RESET - Вхід скидання. Якщо на цей вхід докласти низький рівень тривалістю понад мінімально необхідної буде згенерує скидання незалежно від роботи синхронізації.

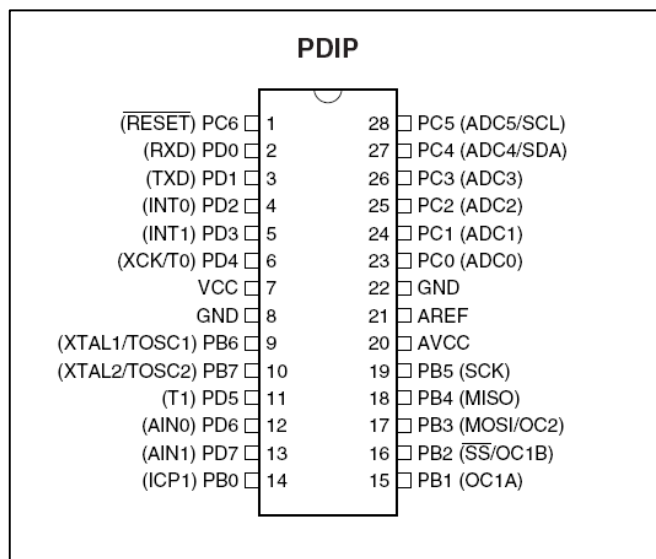


Рисунок 2.2 - Порти Atmega8A

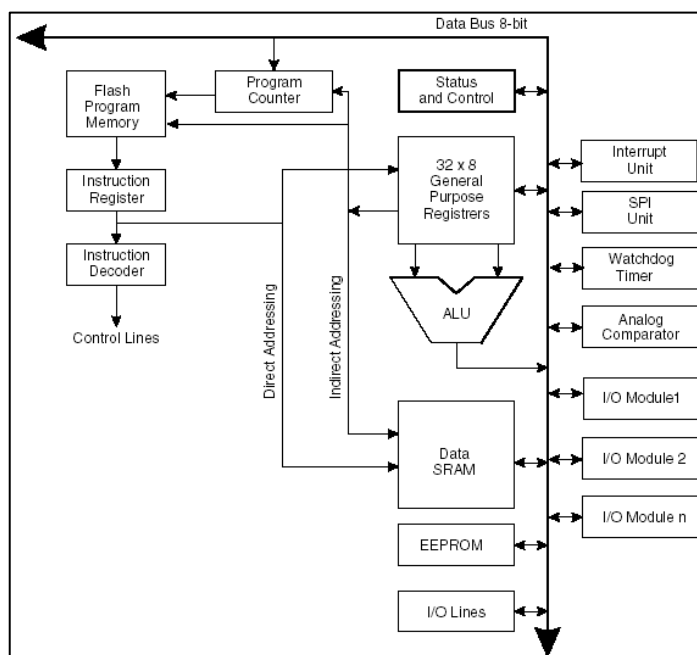


Рисунок 2.3 - Функціональна схема архітектури AVR

З метою досягнення максимальної продуктивності і паралелелізму у AVR мікроконтролерів використовується Гарвардська архітектура з роздільною пам'яттю і шинами програм та даних. Команди в пам'яті програм виконуються з однорівневою конвеєризацією. В процесі виконання однієї інструкції наступна попередньо зчитується з пам'яті програм. Дана концепція дозволяє виконувати одну інструкцію за один машинний цикл. Пам'ять програм являє собою внутрішньосистемну програмовану флеш-пам'ять. Регістровий файл з швидким доступом містить 32 x 8-розр. робочих регістрів загального призначення з однотоковим циклом доступу. Завдяки цьому досягнута ціль однотоковий роботи арифметико-логічного пристрою (АЛП) [12]. При звичайній роботі АЛУ спочатку з реєстрового файлу завантажується два операнда, потім виконується операція, а після результат відправляється назад в регістровий файл і все це відбувається за один машинний цикл. При генерації переривання і виклику підпрограм, адреса повернення з програмного лічильника записується в стек. Стек ефективно розподілений в статичному ОЗУ пам'яті даних і, отже, розмір стека обмежений загальним розміром статичного ОЗУ і використанням його обсягом. У будь-якій програмі відразу після скидання повинна бути виконана ініціалізація покажчика стека (SP). Покажчик стека - SP - доступний на читання і запис в просторі введення-виведення. Доступ до статичного ОЗУ даних може бути легко здійснений через 5 різних режимів адресації архітектури AVR.

6 регістрів з 32 можуть використовуватися як три 16-розр. регістра непрямої адреси для ефективної адресації в межах пам'яті даних. Один з цих покажчиків адреси може також використовуватися як покажчик адреси для доступу до таблиці перетворення у флеш-пам'яті програм. Дані 16-розр. регістри називаються X-регістр, Y-регістр і Z-регістр.

АЛУ підтримує арифметичні і логічні операції між регістрами, а також між константою і регістром. Крім того, АЛУ підтримує дії з одним регістром. Після виконання арифметичної операції регістр статусу оновлюється для відображення результату виконання операцій. Процес виконання програми забезпечується

умовним і безумовним переходом і командами виклику, які в змозі безпосередньо адресувати цілий адресний простір. У більшості інструкцій AVR є єдиний 16-розрядний формат слова. Кожен адреса пам'яті програми містить 16-або 32 розрядну інструкцію.

Флеш-пам'ять програм розділена на дві секції: секція програми початкового завантаження і секція прикладної програми. Обидві секції мають роздільні біти захисту від запису і читання / запису. Інструкція SPM (запис в секцію прикладної програми) повинна використовуватися тільки усередині секції програми початкового завантаження.

Простір пам'яті в архітектурі AVR - лінійне і безперервне.

Гнучкий модуль переривань містить свої керуючі регістри в просторі введення-виведення і має додатковий біт загального дозволу роботи системи переривань в регістрі статусу. У всіх переривань є свій вектор переривання відповідно до таблиці векторів переривань. Переривання мають пріоритет відповідно до позиції їх вектора. Переривання з меншою адресою переривання мають більш високий пріоритет.

Простір пам'яті введення-виведення містить 64 адреси для периферійних функцій ЦП як Регістри Управління, SPI, і інші функції введення-виведення. До Пам'яті введення - виведення можна отримати доступ безпосередньо, або як до пам'яті даних наступна за регістрами за адресами 0x20 - 0x5F.

AVR-мікроконтролер працює в арифметико-логічним безпосереднім зв'язком з усіма 32 універсальними робочими регістрами. АЛУ пристрій дозволяє виконати за один машинний цикл операцію між двома регістрами або між регістром і константою. Операції АЛУ можуть бути класифіковані на три групи: арифметичні, логічні і бітові. Крім того, архітектурою ATmega8 підтримуються операції множення зі знаком і без знака і дробовим форматом.

Регістр статусу містить інформацію про результат тільки що виконаної арифметичної інструкції. Дана інформація може використовуватися для розгалуження програми по умові. Прапори цього регістра в більшості випадків

дозволяють відмовитися від використання інструкцій порівняння, роблячи код програми більш компактним і швидким.

Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Рисунок 2.4 - Регістр статусу SREG AVR-мікроконтролера

Розряд 7 - I: Загальний дозвіл переривань.

Біт загального дозволу переривань використовується для активізації роботи системи переривань. Дозвіл окремих переривань здійснюється у відповідних керуючих регістрах. Якщо біт загального дозволу переривань скинути, то жодне з переривання не буде активним незалежно від їх індивідуальної конфігурації. Біт I скидається в 0 апаратно після генерації запиту на переривання, а після виконання інструкції повернення з переривання RETI знову встановлюється до 1 для виконання наступних переривань.

Розряд 6 - T: Зберігання копійованого біта.

Спеціальні бітові операції BLD (копіювання з T-біта) і BST (копіювання в T-біт) використовують в якості джерела і одержувача даних біт T. Будь біт з реєстрового файлу може бути скопійований в біт T інструкцією BST, а також вміст біта T може бути скопійовано в будь-який біт реєстрового файлу за допомогою інструкції BLD.

Розряд 5 H: Прапор половинного перенесення

Даний прапор встановлюється при виконанні деяких арифметичних інструкцій і відображає про виникнення половинного перенесення. Як правило половинний перенесення широко використовується в двійковій-десятькового арифметиці.

Розряд 4 - S: біт знака, $S = \text{Искл. АБО } (N, ? V)$.

Біт S - результат виконання логічної операції виключає АБО між прапором негативного результату N і прапором переповнення довічного доповнення V. Більш детальна інформація наведена в описі набору інструкцій.

Розряд 3 - V: Прапор переповнення довічного доповнення

Прапор переповнення довічного доповнення V підтримує арифметику з двійковим доповненням.

Розряд 2 - N: Прапор негативного результату.

Прапор негативного результату N відображає, що результатом виконання арифметичної або логічної операції є від'ємне значення.

Розряд 1 - Z: Прапор нульового результату.

Прапор нульового результату Z відображає, що результатом виконання арифметичної або логічної операції є нуль. Більш детальна інформація наведена в описі набору інструкцій.

Розряд 0 - C: Прапор переносу.

Прапор переносу C відображає про виникнення переносу в результаті виконання арифметичної або логічної операції.

2.3 Вибір датчика атмосферного тиску для системи

У якості датчика зчитування атмосферного тиску та температури для системи був обраний вимірювач BMP180. Цей датчик не має нічого зайвого, та підтримує свою малу ціну якістю та своїми функціями.

Вимірювач тиску BMP180 не сильно відрізняється від свого попередника, лише тим що має інтерфейс I2C та більш точне вимірювання, ніж у BMP085 [11].

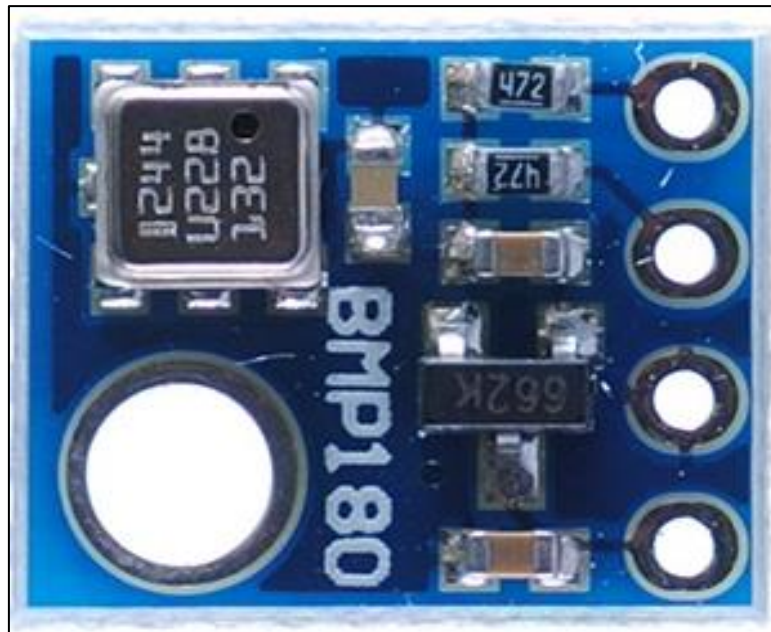


Рисунок 2.5 – Датчик тиску BMP180

Функція BMP180 сумісна з наступником BMP085 та починає нове покоління високоточних датчиків тиску та температури для інтернет споживачів.

Інтерфейс I2C дозволяє легко інтегрувати системи з мікроконтролером.

Склад датчика BMP180:

BMP180 заснований на п'єзо-резистивної технології для надійності, високої точності і лінійності, а також довгострокової стабільності [12].

Таблиця. 2.1 - Характеристики BMP180

Живлення датчика	1,8 ... 3,6 В
Живлення шини I2C	1,62 ... 3,6 В
Живлення модуля	3,3 В чи 5 В
Струм	3 ... 32 мкА
Робоча температура	-40 ... 80 °C
Робочий тиск	0 ... 10 МПа
Діапазон тиску	300 ... 1100 гПа
Діапазон температури	0 ... 65 °C
Час перетворень	3 ... 51 мс
Робоча частота шини I2C	3,4 МГц
Резистори на шині I2C	2,2 ... 10 кОм
Адреса датчика на шині I2C	0x77 HEX

- П'єзо-резистивний датчик (для визначення атмосферного тиску);
- Термодатчик (для визначення температури);
- АЦП (аналого-цифровий перетворювач);
- EEPROM (незалежна електрично стирається перепрограмувальна пам'ять);
- RAM (енергозалежна пам'ять, іншими словами ОЗУ);
- Мікроконтролер;
- Передача даних організована по шинам I2C або SPI (в даному випадку використовується I2C);
- «OSS» - 7 і 6 біти регістра «Measurement Control»: режим точності – може приймати значення від 0 до 3 (00b, 01b, 10b, 11b) на підставі цього значення датчик встановлює співвідношення передискретизації при вимірі тиску (00b: 1, 01b: 2, 10b: 4, 11b: 8);
- «CSO» - 5 біт регістра «Measurement Control»: прапор стану - при установці в «1», датчик почне вимірювання і після всіх перетворень скине його в «0» вказуючи на доступність регістрів «Out ...»;
- 4: 0 біти регістра «Measurement Control»: вид вимірювання - приймають два значення: 01110b або 10100b в першому випадку датчик буде вимірювати температуру, в другому тиск. Результати вимірювань доступні для читання з регістрів «Out MSB», «Out LSB», «Out xLSB», як для тиску, так і для температури.

Вид виміру	Записуване значення				Час виміру	Струм	Точність виміру
	OSS	CSO	4:0	Результат			
Температура	00	1	01110	00101110 = 0x2E	до 4,5 мс	3 мкА	0,5 °C
Тиск	00	1	10100	00110100 = 0x34	до 4,5 мс	3 мкА	0,6 гПа
Тиск	01	1	10100	01110100 = 0x74	до 7,5 мс	5 мкА	0,5 гПа
Тиск	10	1	10100	10110100 = 0xB4	до 13,5 мс	7 мкА	0,4 гПа
Тиск	11	1	10100	11110100 = 0xF4	до 25,5 мс	12 мкА	0,3 гПа

Рисунок 2.6 - Таблиця вимірів BMP180

Назва регістрів	Адреса	Дані регістрів								Значення при скиданні	Доступ
		7 біт	6 біт	5 біт	4 біт	3 біт	2 біт	1 біт	0 біт		
AC1	0xAA	Старший байт								константа	R
	0xAB	Молодший байт								константа	R
AC2	0xAC	Старший байт								константа	R
	0xAD	Молодший байт								константа	R
AC3	0xAE	Старший байт								константа	R
	0xAF	Молодший байт								константа	R
AC4	0xB0	Старший байт								константа	R
	0xB1	Молодший байт								константа	R
AC5	0xB2	Старший байт								константа	R
	0xB3	Молодший байт								константа	R
AC6	0xB4	Старший байт								константа	R
	0xB5	Молодший байт								константа	R
B1	0xB6	Старший байт								константа	R
	0xB7	Молодший байт								константа	R
B2	0xB8	Старший байт								константа	R
	0xB9	Молодший байт								константа	R
MB	0xBA	Старший байт								константа	R
	0xBB	Молодший байт								константа	R
MC	0xBC	Старший байт								константа	R
	0xBD	Молодший байт								константа	R
MD	0xBE	Старший байт								константа	R
	0xBF	Молодший байт								константа	R
Chip Id	0xD0	0	1	0	1	0	1	0	1	константа	R
Soft Reset	0xE0	доступен только для записи								0x00	W
Measurement control	0xF4	OSS		CSO		біты управления измерениями				0x00	R/W
Out MSB	0xF6	Старший байт результата								0x80	R
Out LSB	0xF7	Молодший байт результата								0x00	R
Out xLSB	0xF8	Доп. біти результата					0	0	0	0x00	R

Рисунок 2.7 - Таблиця регістрів BMP085 та BMP180

Алгоритм отримання даних BMP180:

а) Читання калібрувальних коефіцієнтів:

- 1) Short: AC1, AC2, AC3, B1, B2, MB, MC, MD;
- 2) Unsigned short: AC4, AC5, AC6.

б) Зчитування температури:

- 1) записуємо в регістр «Measurement Control» значення: 0x2E;
- 2) очікуємо спад прапора стану «CSO» в «0»;
- 3) читаємо результат з регістрів «Out MSB» і «Out LSB».

в) Зчитування тиску:

- 1) записуємо в регістр «Measurement Control» значення: $0x34 + (OSS \ll 1)$;
- 2) очікуємо спад прапора стану «CSO» в «0»;
- 3) читаємо результат з регістрів «Out MSB», «Out LSB» і «Out xLSB».

Для передачі даних використовується шина даних I2C - послідовний інтерфейс, призначений для забезпечення зв'язку в інтегральних схемах. Була розроблена в 1980 році фірмою Philips і спочатку призначалася для вбудовування в блоки телевізорів як недорога шина внутрішнього зв'язку, яка керує електронікою. Пізніше, завдяки дешевизні і простоті реалізації, I2C стала промисловим стандартом, і її застосування поширилося далеко за межі побутових приладів [21].

На даний момент даний інтерфейс використовується для з'єднання низькошвидкісних периферійних компонентів, при регулюванні колірного балансу і контрастності моніторів, забезпечення інформаційного обміну між ЦАП, драйверами, відеопроцесорами, модулями системної пам'яті і багато чого іншого. Тільки фірма Philips виробляє більше сотні найменувань I2C-сумісних пристроїв, призначених для різного застосування. Серед них є вимірювальні прилади і датчики.

Стандартною швидкістю роботи шини I2C є 100 кбіт / с або 10 кбіт / с в режимі зниженої передачі. У 1992 році розробники зробили можливим підключення ще більшої кількості периферійних пристроїв за рахунок впровадження 10-бітної адресації і збільшили швидкість шини до 400 кбіт / с. Представлена в 1998 році версія стандарту 2.0 дозволяла працювати вже зі швидкістю до 3,4 Мбіт / с.

По конструкції шина являє собою всього два дроти, іменовані лініями SCL і SDA. Перша призначена для синхронізації всіх імпульсів які передаються по I2C. Друга відповідає за передачу даних. Лінії зв'язуються по шині з усіма пристроями на платі, які можуть бути двох типів: Master (головний) і Slave (підпорядкований). Master - це пристрій, який управляє лінією синхронізації імпульсів SCL. Slave - пристрої, які тільки відповідають на запити Mastera.

Для управління підключеними Slave-компонентами Master формує сигнали start і stop, що визначають початок і закінчення передачі даних відповідно. При цьому сигнал start визначається переходом імпульсів лінії SDA з високого регістру в низький при наявності високого рівня сигналу від SCL, а сигнал stop ініціюється переходом SDA з низького регістра в високий при високому рівні SCL.

В процесі передачі даних від Slave до Master останній повинен підтвердити отримання імпульсів спеціальним сигналом ACK. Якщо він цього не робить, Slave припиняє передачу, розвантажуючи лінію SDA і надаючи можливість Master послати сигнал stop. У разі, коли при передачі даних підпорядковане пристрій зайнятий, Master може або зупинити його, або перейти в режим очікування.

Завдяки таким принципом роботи шина I2C досить просто вирішує виникаючі проблеми із затримками сигналів і одночасної ініціалізацією обміну кількома провідними пристроями без втрати будь-яких даних, що і зумовлює її широку популярність в електронних пристроях.

У датчику є герметична камера, одна зі стінок якої є гнучкою мембраною з встановленими на ній тензодатчиками. Мембрана прогинається пропорційно різниці тисків всередині камери і зовні, що впливає на зміну опору тензодатчиків електричного струму. Так само є термодатчик, опір якого змінюється пропорційно температурі. АЦП (аналого-цифровий перетворювач) переводить результати змін датчиків в цифрові дані «некомпенсовані результати», які доступні для читання з регістрів датчика: «Out MSB», «Out LSB» і «Out xLSB».

Для компенсації зазначених результатів (компенсації зсуву, температурної залежності, похибок при виготовленні, неоднорідностей матеріалів і т.д.) кожен датчик калібрується на заводі, і в EEPROM записуються індивідуальні для кожного датчика 11 калібрувальних коефіцієнтів (176 біт), які доступні для читання з регістрів датчика: «AC1», «AC2», «AC3», «AC4», «AC5», «AC6», «B1», «B2», «MB», «MC», «MD».

2.4 Вибір датчика вологості для системи

У якості датчика зчитування вологості та температури для системи був обраний вимірювач DHT11. Цей датчик не має нічого зайвого, та підтримує свою малу ціну якістю та своїми функціями.

Датчик DFRobot DHT11 оснащений датчиком температури і вологості з відкаліброваним цифровим вихідним сигналом. Використовуючи ексклюзивну технологію отримання цифрового сигналу і технологію визначення температури і вологості, вона забезпечує високу надійність і відмінну довготривалу стабільність. Цей датчик включає в себе компонент вимірювання вологості резистивного типу і компонент вимірювання температури NTC і підключається до 8-розрядному мікроконтролеру з високою продуктивністю, забезпечуючи відмінну якість, швидку реакцію, захист від перешкод і економічну ефективність [14].

Кожен елемент DHT11 строго відкалібрований і є надзвичайно точним при калібрування вологості. Коефіцієнти калібрування зберігаються у вигляді програм в ОТР-пам'яті, які використовуються в процесі виявлення внутрішнього сигналу датчика. Однопровідний послідовний інтерфейс робить систему швидше і простіше.

Живлення DHT11 становить 3-5,5 В постійного струму. Коли живлення подається на датчик, не потрібно відправляти ніяких даних на нього протягом однієї секунди, щоб передати нестабільний статус мікросхеми [15].

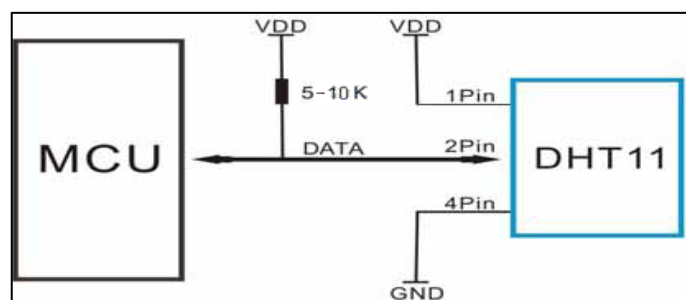


Рисунок 2.8 - Розпіновка DHT11

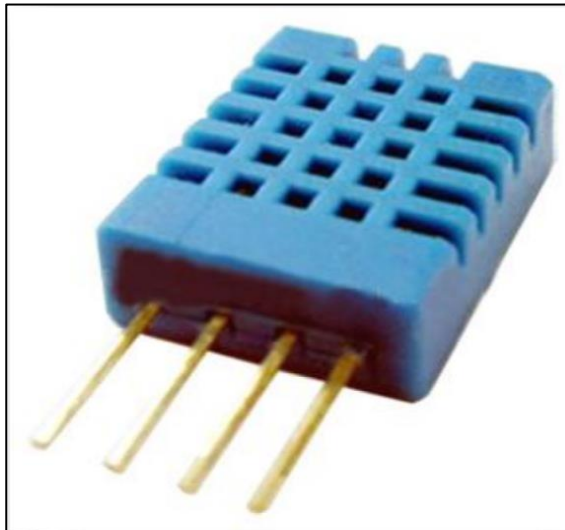


Рисунок 2.9 - Датчик вологості DHT11

Застосування датчика DHT11 за межами робочого діапазону, зазначеного в цьому технічному описі, може привести до зсуву сигналу на 3%. Датчик DHT11 може поступово відновлюватися до відкаліброваного стану, коли він повертається в нормальний робочий стан і працює в межах свого діапазону. Робота датчика DHT11 в ненормальних умовах може прискорити процес старіння датчика.

Увага до хімічних матеріалів. Пари з хімічних матеріалів можуть впливати на чутливі елементи DHT і знижувати його чутливість. Високий ступінь хімічного забруднення може пошкодити датчик [14], [15].

Відносна вологість значною мірою залежить від температури. Незважаючи на те, що технологія температурної компенсації використовується для забезпечення точного вимірювання відносної вологості, як і раніше рекомендується підтримувати датчики в однаковій температурі. DHT11 повинен бути встановлений на місці, наскільки це можливо далеко, від деталей, які можуть виділяти тепло.

Вплив світла: Тривала дія сильного сонячного світла і ультрафіолету може погіршити роботу DHT [16].

Інше:

- температура зварювання повинна бути нижче 260 градусів Цельсія, а контакт повинен займати менше 10 секунд;

- уникайте використання датчика в умовах роси;
- не використовуйте цей продукт в небезпечних пристроях або аварійної зупинки або в будь-якому іншому випадку, коли відмова DHT11 може привести до травми;
- зберігання: Тримайте датчик при температурі 10-40 °C, вологості <60% відносної вологості.

2.5 Вибір Wi-Fi модуля для системи

У якості Wi-Fi модуля для системи була обрана мікросхема ESP8266. Ця мікросхема доступна по ціні, дає можливість передавати данні на хмару при цьому мати легке налаштування і дешеву ціну, та має усі функції щоб виконати роль посередника у системи для обробки атмосферних показників.

UART — тип асинхронного приймача-передавача, компонентів комп'ютерів та периферійних пристроїв, що передає дані між паралельною та послідовною формами. UART звичайно використовується спільно з іншими комунікаційними стандартами, такими як EIA RS-232 [17].

UART це зазвичай окрема мікросхема чи частина мікросхеми, що використовується для з'єднання через комп'ютерний чи периферійний послідовний порт. UART нині загалом включені в мікроконтролери. Здвоєний UART (Dual UART або DUART) об'єднує двоє UART в одній мікросхемі. Багато сучасних мікросхем сьогодні випускаються з можливістю комунікації в синхронному режимі, такі прилади називають USART [18].

Модуль ESP-12F WiFi розроблений командою Ai-thinker. Основний процесор ESP8266 Tensilica L106, об'єднує в собі ультрасучасні 32-розрядні мікроконтролери з ультранизьким енергоспоживанням з 16-розрядним режимом, підтримкою тактової частоти 80 МГц, 160 МГц, підтримкою RTOS, інтегрованим Wi-Fi MAC / BB / RF / PA / LNA і бортовий антени. Модуль підтримує стандартну угоду IEEE802.11 b / g / n, повний пакет протоколів TCP / IP. Користувачі можуть

використовувати модуль для додавання до існуючої мережі або створювати окремий сервер. ESP8266 - це високоінтегрований бездротової SOC, призначений для розробників мобільних платформ з обмеженим простором і потужністю. Він забезпечує неперевершену здатність вбудовувати можливості Wi-Fi в інші системи або функціонувати як окремий додаток з мінімальними витратами і мінімальними вимогами до пристроїв.

ESP8266 створений для використання в розумних розетках, mesh-мережах, IP-камерах, бездротових сенсорах, що носиться електроніці і так далі. Одним словом, ESP8266 з'явився на світ, щоб стати мозком майбутнього «Інтернету речей».

Сценарій використання чіпа в якості керуючого мікроконтролера цікавий тим, що дозволяє створити пристрої, дійсно невеликі і реально довго працюють від батарей. Для роботи з периферією на борту ESP8266 є всі необхідні можливості.

Енергоспоживання - одна з найважливіших характеристик датчика, який претендує стати мозком мільярдів пристроїв Інтернету речей.

Тепер Wi-Fi можна використовувати навіть в автономних датчиках, що працюють на невеликих батареях. Завдяки використанню просунутих механізмів управління енергоспоживанням рішення [19].

Таблиця. 2.2 - Головні технічні характеристики ESP8266

Категорія	Пункт	Параметри
Wi-Fi	Стандарти	FCC/CE/TELEC/SRRC
	Протоколи	802.11 b/g/n
	Діапазон частот	2.4G ~ 2.5G (2400M ~ 2483.5M)
	Тх Потужність	802.11 b: +20 dBm 802.11 b: +20 dBm
	Антенa	PCB трасування, керамічний чіп
	Rx Чутливість	802.11 b: -91 dbm (11 Mbps)
Апаратура	Процесор	Tensilica L106 32-bit micro controller

Продовження таблиці 2.2

Апаратура	Переферійні інтерфейси	UART/SDIO/SPI/I2C/I2S/IR Remote Control GPIO/ADC/PWM
	Робоча напруга	2.5V ~ 3.6V
	Робочий струм	80 mA
	Діапазон робочих температур	-40°C ~ 125°C
	Діапазон температур зберігання	-40°C ~ 125°C
	Розмір пакета	QFN32-pin (5 mm x 5 mm)
	Зовнішній інтерфейс	-
	Wi-Fi Режим	Station/SoftAP/SoftAP+ Station
Програмне забезпечення	Захист	WPA/WPA2
	Шифрування	WEP/TKIP/AES
	Мережеві протоколи	IPv4, TCP/UDP/HTTP/FTP
	Конфігурацій користувача	Ат команди, сервер, дополнение для Android/IOS

Таблиця. 2.3 – Функції контактів ESP8266 12-f

Контакт	Описание функции
RST	Рестарт модуля
ADC	Результат перетворення А / D. Діапазон вхідної напруги 0 ~ 1 В, в Діапазон: від 0 до 1024
EN	Включення чіпу
IO16	GPIO16; Робить режим сну при прийомі на RST.
IO14	GPIO14; HSPI_CLK
IO12	GPIO12; HSPI_MISO
IO13	GPIO13; HSPI_MOSI; UART0_CTS
VCC	Живлення 3.3 V
CS0	Вибір чіпа
MISO	Вхідний сигнал ведучого пристрою

Продовження таблиці 2.3

IO9	GPIO9
IO10	GPIO10
MOSI	Вихідні сигнал ведучого пристрою
SCLK	Час
GND	Заземлення
IO15	GPIO15; MTDO; HSPICS; ART0_RTS
IO2	GPIO15; TDO;HSPICS; UART0_RTS
IO0	GPIO0
IO4	GPIO4
IO5	GPIO5
RXD	UART0_RXD; GPIO3
TXD	UART0_RXD; GPIO3

Передбачено два варіанти використання чіпа: 1) у вигляді моста UART-WIFI, коли модуль на базі ESP8266 підключається до інтернету на базі будь-якого іншого мікроконтролера і управляється АТ-командами, забезпечуючи зв'язок рішення з інфраструктурою Wi-Fi; 2) реалізуючи нове рішення, яке використовує сам чіп ESP8266 в якості керуючого мікроконтролера.

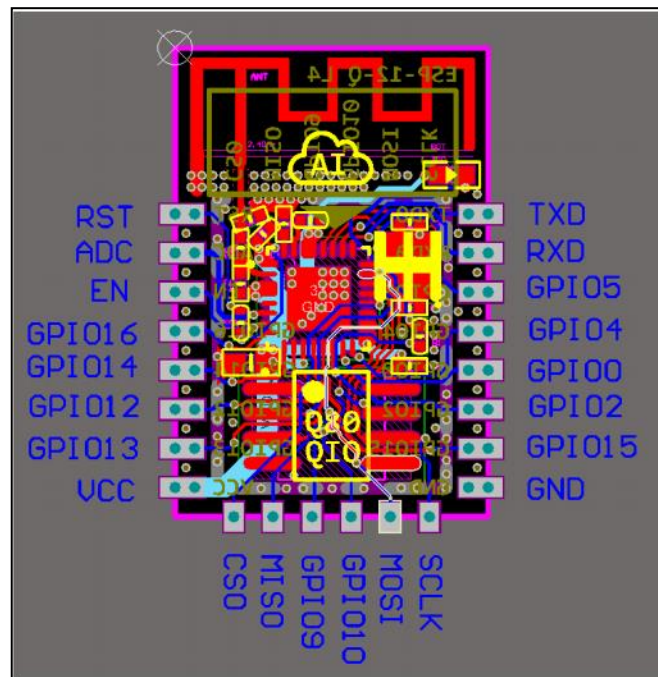


Рисунок 2.10 - Розпіновка ESP8266

Другий варіант сценарію передбачає написання індивідуальної прошивки для управління чипом «зсередини». В даний момент прошивка повинна бути написана для фірмового компілятора. З чим, в основному, і пов'язані вимоги до нерозголошення інформації навколо цього рішення. У доступному для огляду майбутньому виробник планує перейти на використання GCC і ці обмеження будуть зняті [19].

Сценарій використання чіпа в якості керуючого мікроконтролера цікавий тим, що дозволяє створити пристрої, дійсно невеликі і реально довго працюють від батарей. Для роботи з периферією на борту ESP8266 є всі необхідні можливості.

Енергоспоживання - одна з найважливіших характеристик датчика, який претендує стати мозком мільярдів пристроїв Інтернету речей [20].

Тепер Wi-Fi можна використовувати навіть в автономних датчиках, що працюють на невеликих батареях. Завдяки використанню просунутих механізмів управління енергоспоживанням рішення.

215mA в режимі передачі. ESP8266 споживає близько 60uA в режимі глибокого сну (з працюючими годинами реального часу) і менше 1.0mA (DTIM = 3) або менше 0.5mA (DTIM = 10) в режимі підтримки зв'язку з точкою доступу Wi-Fi.

АТ-команди для настройки модуля:

1. АТ – Перевірка модуля

Якщо модуль вдало запустився, то відповідає «ОК»

Таблиця 2.4 - АТ команда Wi-Fi модуля

АТ – Перевірка модуля	
Відповідь	ОК
Параметр	Немає

2. АТ+RST – Перезапуск модуля

Таблиця 2.5 - AT команда Wi-Fi модуля

AT+RST – Перезапуск модуля	
Відповідь	OK
Параметр	Немає

3. AT+GMP - Відобразити версію прошивки.

Версія відображається у вигляді 8 цифр. Перша група 4 цифри - версія AT (наприклад, 0022), друга група 4 - версія SDK (наприклад, 0100).

Таблиця 2.6 - AT команда Wi-Fi модуля

AT+RST – Перезапуск модуля	
Відповідь	<AT version info> <SDK version info> <compile time> OK
Параметр	<AT version info> Інформація про AT версії <SDK version info> Інформація про SDK версії <compile time> Час компіляції

4. AT+GSLP - Перехід в режим зниженого енергоспоживання

Для того, щоб модуль вийшов з режиму сну, необхідно з'єднати висновки XPD_DCDC і EXT_RSTB

Таблиця.2.7 - AT команда Wi-Fi модуля

AT+GSLP - Перехід в режим зниженого енергоспоживання	
Відповідь	«Час» OK
Параметр	Приклад: AT + GSLP = 5000 (5сек)

5. ATE – Відключити, увімкнути ехо AT команд

Ця команда включає, або вимикає відповідь користувачу у виді цієї ж AT команди.

Таблиця. 2.8 - AT команда Wi-Fi модуля

ATE – Відключити, увімкнути ехо AT команд	
Відповідь	OK
Параметр	ATE0: Switch echo off ATE1: Switch echo on

6. AT+RESTORE – Скидання до заводських параметрів

Ця команда використовується для скидання всіх параметрів, збережених у пам'яті та щоб відновити заводські настройки модуля. Чіп буде перезапущений при виконанні цієї команди.

Таблиця. 2.9 - AT команда Wi-Fi модуля

AT+RESTORE – Скидання до заводських параметрів	
Відповідь	OK
Параметр	Restore factory default settings.

7. AT+UART_CUR - Поточна конфігурація UART

Ця команда встановлює поточну конфігурацію UART. Він не записується у флеш-пам'ять. Отже, немає змін у швидкості передачі за замовчуванням.

Таблиця. 2.10 - AT команда Wi-Fi модуля

AT+RESTORE – Скидання до заводських параметрів	
Відповідь	OK
Параметр	швидкість 4400-4608000 біти даних 5: 5 біт 6: 6 біт 7: 7 біт 8: 8 біт стоп біти

Продовження таблиці 2.10

Параметр	1: 1 стоп біт 2: 1.5 стоп біт 3: 2 стоп біт
Параметр	контроль парності 0: немає 1: Odd 2: EVEN flow control 0: flow control відключений 1: включений RTS 2: включений CTS 3: включені обидва RTS і CTS (MTCK - UART0 CTS, MTDO - UART0 RTS) приклад: AT + UART = 115200,8,1,0,0.
Команда	AT+ UART_CUR= baudrate, databits, stopbits, parity, flow control

8. AT+SLEEP – режим сну.

Посилає датчик у режим пониженого енергоживлення, чи очікування.

Таблиця. 2.11 - AT команда Wi-Fi модуля

AT+RESTORE – Скидання до заводських параметрів	
Відповідь	+SLEEP : <режим сна> OK
Параметр	<режим сна> 0: Виключений режим сну 1: Режим очікування 2: Режим глибокого сну
Команда	AT+SLEEP=<sleep mode>

AT команди для налаштування Wi-Fi на модулі:

1. AT+CWMODE_CUR - Поточний режим WiFi

Існує три режими роботи WiFi: режим станції, режим softAP і спільне використання режиму Station і softAP.

Таблиця. 2.12 - АТ команда Wi-Fi модуля

АТ+RESTORE – Скидання до заводських параметрів	
Відповідь	+CWMODE_CUR:<Режим> ОК
Параметр	<Режим> 1: station mode 2: softAP mode 3: softAP + station mode
Команда	АТ+CWMODE_CUR =<Режим>

1. АТ+CWJAP_CUR – підключення до АР точки доступу.

Таблиця. 2.13 - АТ команда Wi-Fi модуля

АТ+CWJAP_CUR – підключення до АР точки доступу.	
Відповідь	ОК or +CWJAP:<error code> FAIL
Параметр	+CWJAP_CUR:<ssid>,<bssid>,<channel>,<rssi> ОК
Команда	АТ+CWJAP_CUR=<ssid>,<pwd>[,<bssid>]
Пояснення	Дані мережі не зберігаються у флеш пам'яті. У разі наявності в SSID або пароль спеціальних символів (', ' ' і '\') їх необхідно екранувати зворотним слешем. АТ + CWJAP_CUR =" ab \\\, с "" 0123456789 \ ""\"

3. АТ+CWLAP - Відобразити список доступних точок доступу

Таблиця. 2.14 - АТ команда Wi-Fi модуля

АТ+CWLAP - Відобразити список доступних точок доступу	
Відповідь	+CWLAP:<ecn>,<ssid>,<rssi>,<mac>,<ch>,<freq offset> ОК ERROR
Параметр	Виводить SSID, метод шифрування, силу сигналу, MAC адреса, номер каналу. Типи шифрування: 0: Open, 1: WEP, 2: WPA_PSK, 3: WPA2_PSK,

Продовження таблиці 2.14

Параметр	4: WPA_WPA2_PSK Приклади: AT + CWLAP = "wifi", "ca: d7: 19: d8: a6: 44", 6 або пошук WiFi мережі з ім'ям "home" AT + CWLAP = "home", ""4 AT+CWQAP – Відключення від точки доступу.
Команда	AT+CWLAP=<ssid>,<mac>,<ch>
Пояснення	AT + CWLAP показує всі доступні точки доступу

4. AT+CWQAP – Відключення від точки доступу.

Таблиця.3.15 - AT команда Wi-Fi модуля

AT+CWQAP – Відключення від точки доступу.	
Відповідь	OK

5. AT+CWSAP_CUR - Створити SoftAP (точку доступу) для поточного сеансу

Таблиця. 2.16 - AT команда Wi-Fi модуля

AT+CWSAP_CUR - Створити SoftAP (точку доступу) для поточного сеансу	
Відповідь	OK ERROR
Параметр	Команда доступна тільки коли модуль знаходиться в режимі SoftAP (точка доступу). Потрібно AT + RST. SSID і пароль вказуються в подвійних лапках. Пароль не більше 64 символів. Типи шифрування: 0: Open, 2: WPA_PSK, 3: WPA2_PSK, 4: WPA_WPA2_PSK (Шифрування WEP недоступно в цій версії) Приклад: AT + CWSAP_CUR = "ESP8266", "1234567890", 5,3
Команда	AT+CWSAP_CUR=<ssid>,<pwd>,<chl>,<ecn>,<max conn>

Продовження таблиці 2.16

Пояснення	AT + CWSAP_CUR = <ідентифікатор мережі>, «пароль», «канал», «тип шифрування»
-----------	--

6. AT+CWLIF - Відобразити IP адреси станцій, підключених до SoftAP точки доступу

Таблиця 2.17 - AT команда Wi-Fi модуля

AT+CWLIF - Відобразити IP адреси станцій, підключених до SoftAP точки доступу	
Відповідь	<IP addr>,<mac> OK
Параметр	1: IP адреса 2: Mac адреса
Команда	(Тільки для режимів 2-SoftAP і 3-Station + SoftAP)

7. AT+CWDHCP – Увімкнути або вимкнути DHCP сервер.

Таблиця 2.18 - AT команда Wi-Fi модуля

AT+CWDHCP – Увімкнути або вимкнути DHCP сервер.	
Відповідь	OK
Параметр	<Режим> 0: ESP8266 SoftAP 1: ESP8266 station 2: SoftAP і station <Вкл> 0: Вимкнути DHCP 1: Включити DHCP Приклад: AT + CWDHCP_CUR = 0,1
Команда	AT+CWDHCP=<mode>,<en>

8. AT+CWAUTOCONN – Автопідключення до AP (точки доступу) чи ні.

Таблиця 2.19 - AT команда Wi-Fi модуля

AT+CWAUTOCONN – Автопідключення до AP (точки доступу) чи ні.	
Відповідь	OK
Параметр	<Вкл> 0: чи не підключатися автоматично до точки доступу після старту модуля 1: підключатися автоматично до точки доступу після старту модуля
Команда	AT+CWAUTOCONN=<enable>

9. AT+CIPSTAMAC – Встановити Mac адресу на станцію.

Таблиця 2.20 - AT команда Wi-Fi модуля

AT+CIPSTAMAC – Встановити Mac адресу на станцію.	
Відповідь	OK
Параметр	Приклад: AT+CIPAPMAC_CUR="1a:fe:36:97:d5:7b"
Команда	AT+CIPSTAMAC=<mac>

10. AT+CIPSTA – Встановити IP адресу на станцію

Таблиця. 2.21 - AT команда Wi-Fi модуля

AT+CIPSTAMAC – Встановити Mac адресу на станцію.	
Відповідь	OK
Параметр	ip - ip адреса у вигляді рядка, <шлюз> - шлюз, <маска> - маска підмережі. Приклад: AT + CIPSTA_CUR = "192.168.6.100", "192.168.6.1", "255.255.255.0"
Команда	AT+CIPSTAMAC=<mac>

11. AT+CIPAP – Встановити IP адресу на softAP.

Таблиця. 2.22 - AT команда Wi-Fi модуля

AT+CIPAP – Встановити IP адресу на softAP	
Відповідь	+CIPAP:<IP> OK
Параметр	Пример: AT+CIPAP_CUR="192.168.5.1"
Команда	AT+CIPAP=<IP>[,<gateway>,<netmask>]

AT команди для налаштування TCP/IP:

1. AT+CIPSTATUS - Відобразити статус підключення

Таблиця 2.23 - AT команда Wi-Fi модуля

AT+CIPSTATUS - Відобразити статус підключення	
Відповідь	STATUS:<stat>+CIPSTATUS: <link ID>, <type>, <remote_IP>, <remote_port>, <local_port>, <tetype>
Параметр	Повертає = ID з'єднання 0-4, <Тип> = тип з'єднання (2 - отримано IP, 3 - підключений, 4 - відключений)"TCP" або "UDP",= Віддалений IP адреса,= Віддалений порт, = тип зв'язку: 0: приєднаний як клієнт, 1: як сервер
Команда	AT+CIPSTATUS

2. AT+CIPSTART – початок підключення

Таблиця 2.24 - AT команда Wi-Fi модуля

AT+CIPSTART – початок підключення	
Відповідь	OK or ERROR If connection already exists, returns ALREAY CONNECT
Параметр	ID з'єднання 0-4, <Тип> = TCP або UDP, <адреса> = IP адреса віддаленого хоста, <порт> = порт віддаленого хоста. [<Локальний порт>] тільки для UDP [<Режим>] тільки для UDP

Продовження таблиці 2.24

Параметр	Повертає "OK", "ERROR" або "ALREADY CONNECT" 0: destination peer entity of UDP will not change. 1: destination peer entity of UDP can change once. 2: destination peer entity of UDP is allowed to change. використовується тільки спільно Приклад: AT + CIPSTART = "TCP", "192.168.101.110", 1000
Команда	Одиночне підключення: AT+CIPSTART=<type>,<remote IP>,<remote port>[,<UDP local port>,<UDP mode>][,<TCP keep alive>] Множинне підключення AT+CIPSTART=<link ID>, <type>, <remote IP> ,<remote port>[,<UDP local port>,<UDP mode>][,<TCP keep alive>]

3. AT+CIPSEND – Відправити данні

Таблиця 2.25 - AT команда Wi-Fi модуля

AT+CIPSEND – Відправити данні	
Відповідь	Повертає ">" після відправки команди. Якщо передача даних підтверджена починає передачу даних. Якщо з'єднання не може бути встановлено або обривається під час відправки, повертає ERROR якщо все пройшло добре відправляє SEND OK
Параметр	Довжина даних в пакеті до 2048 байт. Після отримання даної команди модуль виводить запрошення ">" і переходить в режим прийому даних через UART, після прийому даних необхідної довжини передає їх в радіоканал. При успішній передачі повертає "SEND OK". При невдачі "ERROR". У режимі "unvarnished transmission mode " перервати режим прийому даних і перейти в командний режим можна послідовністю "+++" в окремому пакеті. Між пакетами інтервал 20мс. Приклади можна знайти в документі "Espressif AT Command Examples"
Команда	AT+CIPSEND

4. AT+CIPCLOSE – Закриває TCP чи UDP підключення.

Таблиця. 2.26 - AT команда Wi-Fi модуля

AT+CIPCLOSE – Закриває TCP чи UDP підключення.	
Відповідь	OK ERROR
Параметр	Повертає "LINK IS NOT" або "UNLINK" якщо ID з'єднання вже розірвано, ERROR якщо з'єднання немає. Якщо в режимі клієнта = 5, то закриваються всі з'єднання "
Команда	AT+CIPCLOSE=<link ID>

5. AT+CIFSR - Відобразити локальні IP адреси

Таблиця. 2.27 - AT команда Wi-Fi модуля

AT+CIFSR - Відобразити локальні IP адреси	
Відповідь	+ CIFSR:<IP address> OK чи ERROR
Параметр	Повертає IP адреса ESP8266 SoftAP і IP address ESP8266 Station
Команда	AT+ CIFSR

6. AT+CIPMUX - Вибрати режим одиночного або множинних підключень

Таблиця. 2.28 - AT команда Wi-Fi модуля

AT+CIPMUX - Вибрати режим одиночного або множинних підключень	
Відповідь	OK If already connected, returns Link is builded
Параметр	0 = поодинокі підключення 1 = множинні підключення. Змінити режим можна тільки після закриття всіх підключень. Якщо запущено сервер, то потрібне перезавантаження модуля. "AT + CIPMUX = 1" команда доступна тільки при "AT + CIPMODE = 0"
Команда	AT+CIPMUX=<mode>

7. AT+CIPSERVER - Запустити (перезапустити) TCP сервер

Таблиця. 2.29 – AT команда Wi-Fi модуля

AT+CIPSERVER – Запустити (перезапустити) TCP сервер	
Відповідь	OK If already connected, returns Link is builded
Параметр	0 = сервер відключений, 1 = сервер запущений. Порт вказувати необов'язково, за замовчуванням: 333. Для запуску сервера модуль повинен бути в режимі множинних підключень AT + CIPMUX = 1. Приклади: AT + CIPMUX = 1 AT + CIPSERVER = 1,1001
Команда	AT+CIPSERVER=<mode>[,<port>]

2.6 Блок-схема алгоритму роботи системи

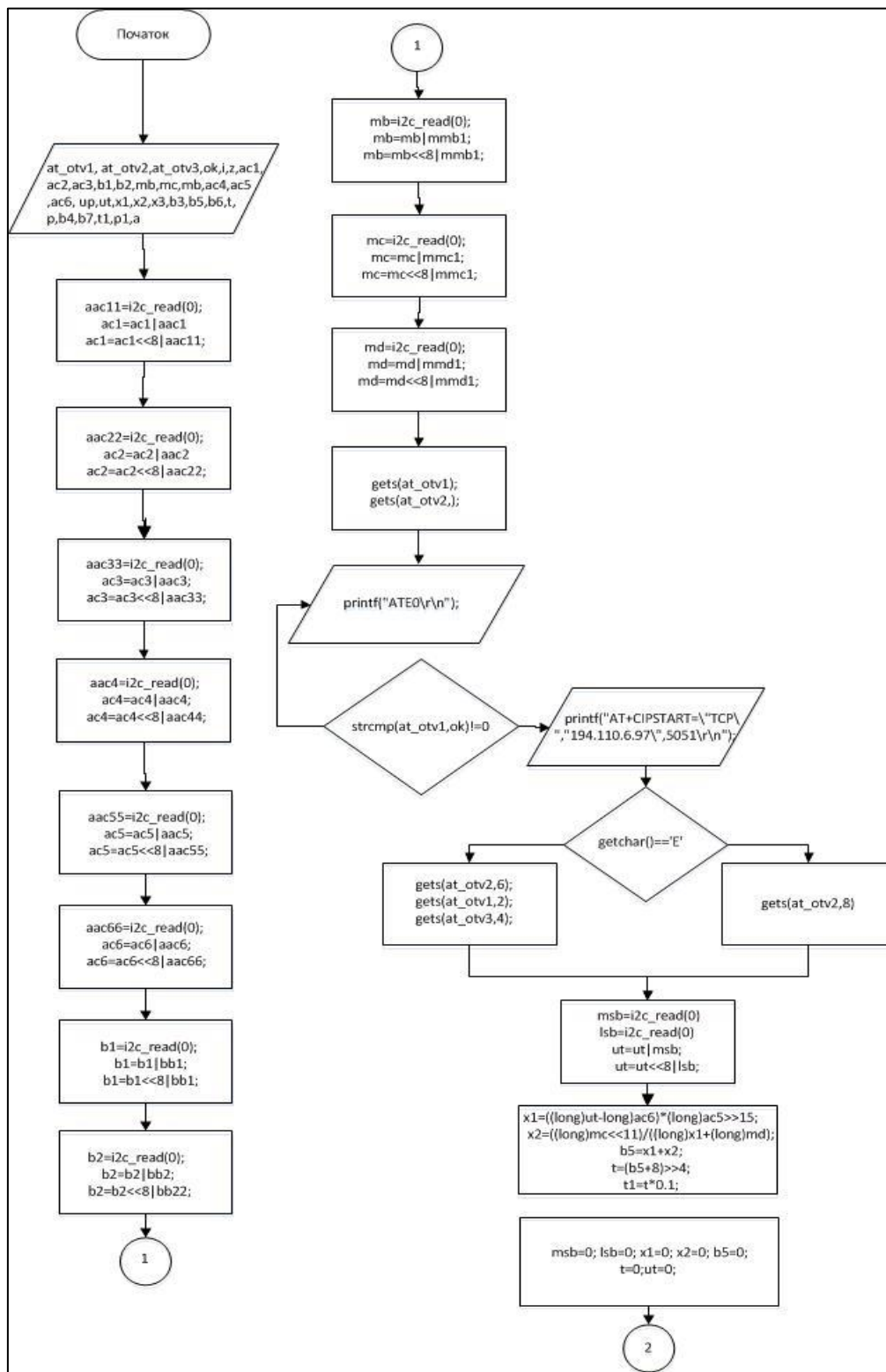


Рисунок 2.11 – Блок схема 1 частина.

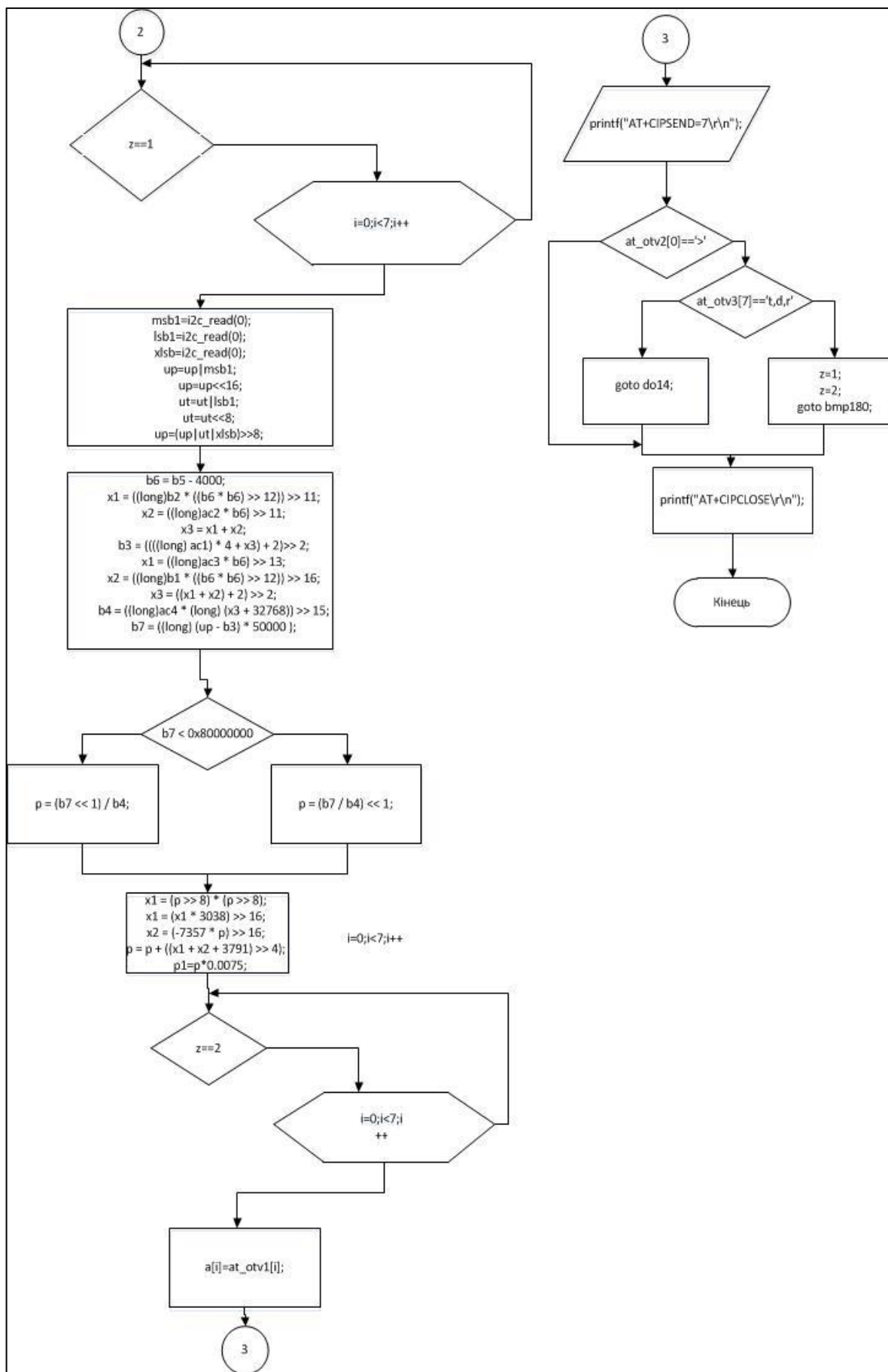


Рисунок 2.12 – Блок схема 2 частина

3 РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Розробка принципової схеми системи

Електрична принципова схема є видом електричної схеми виробу, що дає найповніше уявлення про склад і принцип його роботи. При виконанні схем цифрової обчислювальної техніки керуються ГОСТ 2.708-81. Цей вид кресленника не враховує габаритних розмірів і реального розташування деталей об'єкта.

На принциповій електричній схемі (ГОСТ 2.709-89) зображуються всі складові частини виробу і зв'язки між ними, та елементи, якими закінчуються вхідні та вихідні ланки електричних кіл (роз'єми, затискачі тощо) [21].

Принципові електричні схеми призначенні для повного відображення взаємозв'язків пристроїв з урахуванням принципів їх дії і послідовності роботи. На принципових електричних схемах електричні елементи зображують за допомогою умовних позначень, а також вказують лінії зв'язків між ними, блоками та модулями. На схемі, також, розміщується наступна інформація: умовне зображення принципу роботи функціональних вузлів, пояснювальні написи, частини окремих елементів, діаграми переключення контактів, а також перелік використовуваних в даній схемі пристроїв [22].

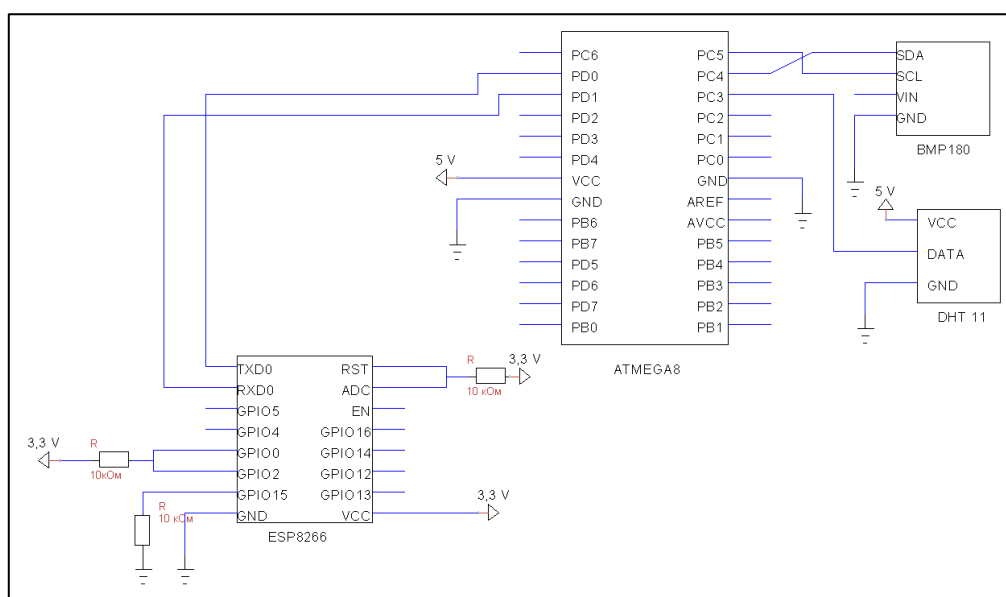


Рисунок 3.1 – Принципова схема системи

3.2 Середовище розробки програмного забезпечення для системи

Для розробки програми «домашньої метеостанції» було використане середовище AVR CodeVision.

Програма являє собою інтегроване середовище розробки ПО для AVR мікроконтролерів.

З основних достоїнств CodeVisionAVR можна відзначити те, що він не дуже складний для самостійного освоєння, підтримує всі численне сімейство мікроконтролерів AVR (включаючи чіпи з ядром ATxmega), формує ємний і результативний програмний код. Крім компілювання середовище розробки здатна записати створену програму в пам'ять мікроконтролера.

Модуль прошивки може взаємодіяти з усіма популярними программаторами (AVR910, STK200 / 300 і багатьма іншими). Редактор дозволяє працювати з двома проектами одночасно, розміщувати закладки, налаштовувати час автоматичного збереження результатів.

Основними модулями, що входять до складу CodeVisionAVR є [23]:

- транслятори програм, складених на Сі-подібних мовах або асемблері, в машинний код для AVR;
- елементи для ініціалізації периферійних пристроїв;
- модуль для роботи з платою налагодження STK-500;
- компоненти взаємодії з зовнішніми программаторами;
- редактор первинного коду;
- термінальний модуль.

Результат роботи в програмі CodeVisionAVR може бути представлений у вигляді HEX, ROM або BIN-файлу для прямої прошивки мікроконтролера шляхом використання стороннього програматора.

Крім цього, програма може бути передана в форматі COFF (файл відладчика) або OBJ. Число бібліотек CodeVisionAVR зростає з кожною новою версією і

включає в себе підтримку продукції від таких відомих виробників як Philips, National Semiconductor, Maxim-Dallas Semiconductor і багатьох інших.

3.3 Панель керування системи

Система передає свої данні атмосферних показників до серверу, звідки і можуть бути зчитані данні.

Сервер — у комп'ютерній термінології термін може стосуватися окремого комп'ютера чи програми. Головною ознакою в обох випадках є здатність машини чи програми переважну кількість часу працювати автономно, без втручання людини, реагуючи на зовнішні події відповідно до встановленого програмного забезпечення. Втручання людини відбувається під час встановлення серверу і під час його сервісного обслуговування. Часто це роблять окремі адміністратори серверів з вищою кваліфікацією.

Сервер приймає 4 типа даних:

- тиск;
- температуру;
- вологість;
- скидання.

Данные:

d

d

Отправить

712,35

713,63

710.10

Рисунок 3.2 – Відправлення значення тиску сервером

Данные:

t

Отправить

Рисунок 3.3 – Відправлення значення температури сервером

ВИСНОВКИ

В процесі розробки данного пристрою було проаналізовано декілька варіантів комплектуючих елементів:

- PIC controller;
- Arduino;
- Raspberry pi;
- Atmega8.

Мною було вибрано саме ATMEGA8, тому що в цьому пристрої можливо швидко й динамічно змінювати налаштування конфігурації. Він володіє найліпшою сумуїстністю з іншими датчиками та модулями. Також одним з головних чинників у виборі стала ціна та доступність данного пристрою на ринку.

Вагомою проблемою у ході цієї роботи було складання програми, для повного співпрацювання датчиків та Wi-Fi модуля у рамках однієї системи. Завдяки аналізу та вибору саме таких елементів у результаті вийшов повноцінний аналог системи для вимірювання даних погоди, та який у майбутньому житті можна використовувати у системі розмуний дім.

Підводячи підсумок зі зробленої роботи, можна вивести, що найголовніша проблема системи це її висока ціна та тяжке впровадження у повсякденне життя. Тому що для людей це великий крок – довіритися технологіям, які будуть знати про них все. На прикладі розумного будинку, можна зрозуміти, що він може зчитувате усе наше повсякденне життя, для того щоб впрощувати його.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Антонов В. Н. Интеллектуально-экспертная система обработки документов, подсказки принятия решений: метод построения и реализации
2. УСиМ. – 1995. –№3.–С82-85. 6. Антонов В. Н. Проектирование объектноориентированных интеллектуальных АРМ УСиМ.
3. Електроніка і мікросхемотехніка: Підручник для студентів вищих навчальних закладів, що навчаються за напрямками “Електромеханіка” та “Електротехніка”: У 4-х т. /Сенько В.І., Панасенко М.В., Сенько Є.В. та ін. Т2. Аналогові та імпульсні пристрої. –Харків: Фоліо, 2002.
4. Електроніка та мікросхемотехніка [Електронний ресурс]: навчальний посібник для студентів напряму підготовки 6.050702 «Електромеханіка» / А. А. Щерба, К. К. Победаш, В. А. Святненко: – Київ: НТУУ «КПІ», 2013. – 360 с. Режим доступу: <http://ela.kpi.ua/handle/123456789/3569>
5. Електротехніка та електроніка. Теоретичні відомості, розрахунки та дослідження за підтримкою комп'ютерних технологій: Навч. посіб. /Щерба А.А., Рябенський В.М., Кучеренко М.Є., Победаш .К.К. та ін. – К.: "Корнійчук", 2007, - 488 с. з іл.
6. Колонтаєвський Ю.П., Сосков А.Г. Промислова електроніка та мікросхемотехніка: теорія і практикум: Навч. посіб. 2-е вид. – К.: Каравела, 2004, - 432 с.
7. Руденко В.С., Сенько В.І., Трифонюк В.В. Основы промышленной электроники. – К.: Выща шк., 1985. – 400 с.
8. Горбачев Г.Н., Чаплыгин Е.Е. Промышленная электроника. –М.: Энергоатомиздат, 1988.
9. Забродин Ю.С. Промышленная электроника. –М.: ВШ, 1982.
10. Скаржепа В.А., Луценко А.Н. Электроника и микросхемотехника. Ч.1. Электронные устройства информационной автоматики. –К.: Выща шк., 1989.

11. Опадчий Ю.Ф., Глудкин О.П., Гуров А.И. Аналоговая и цифровая электроника. – М.: Горячая Линия – Телеком, 2000.
12. Прищепа М.М., Погребняк В.П. Мікроелектроніка. - Ч.1. Елементи мікроелектроніки / М.М. Прищепа, В.П. Погребняк. - Київ: Вища школа, 2004. - 432 с.
13. Сенько В.І. Електроніка і мікросхемотехніка. Т.1. Елементна база електронних пристроїв / В.І. Сенько, М.В. Панасенко, Є.В. Сенько. - Київ: Обереги, 2000. - 300 с.
14. Лавриненко В.Ю. Справочник по полупроводниковым приборам. - Киев: Техніка, 1984. - 410 с.
15. <https://habrahabr.ru/company/exante/blog/321548/>
16. <https://rb.ru/opinion/blockchain/>
17. <http://blockgeeks.com/guides/smart-contracts/>
18. <https://cryptor.net/kriptovalyuty/ethereum-vs-emercoin-o-raznice-podhodovv-ispolzovanii-blokcheyna>
19. <https://habrahabr.ru/post/319868/>
20. <https://www.ethereum.org/dao>
21. <https://habrahabr.ru/post/153865/>
22. <http://coin-lab.com/ethereum-glava-6-kontrakty-i-tranzaktsii/>
23. <https://habrahabr.ru/company/microsoft/blog/311690/>

ДОДАТОК А

АЛГОРИТМ РОЗРОБКИ СИСТЕМИ

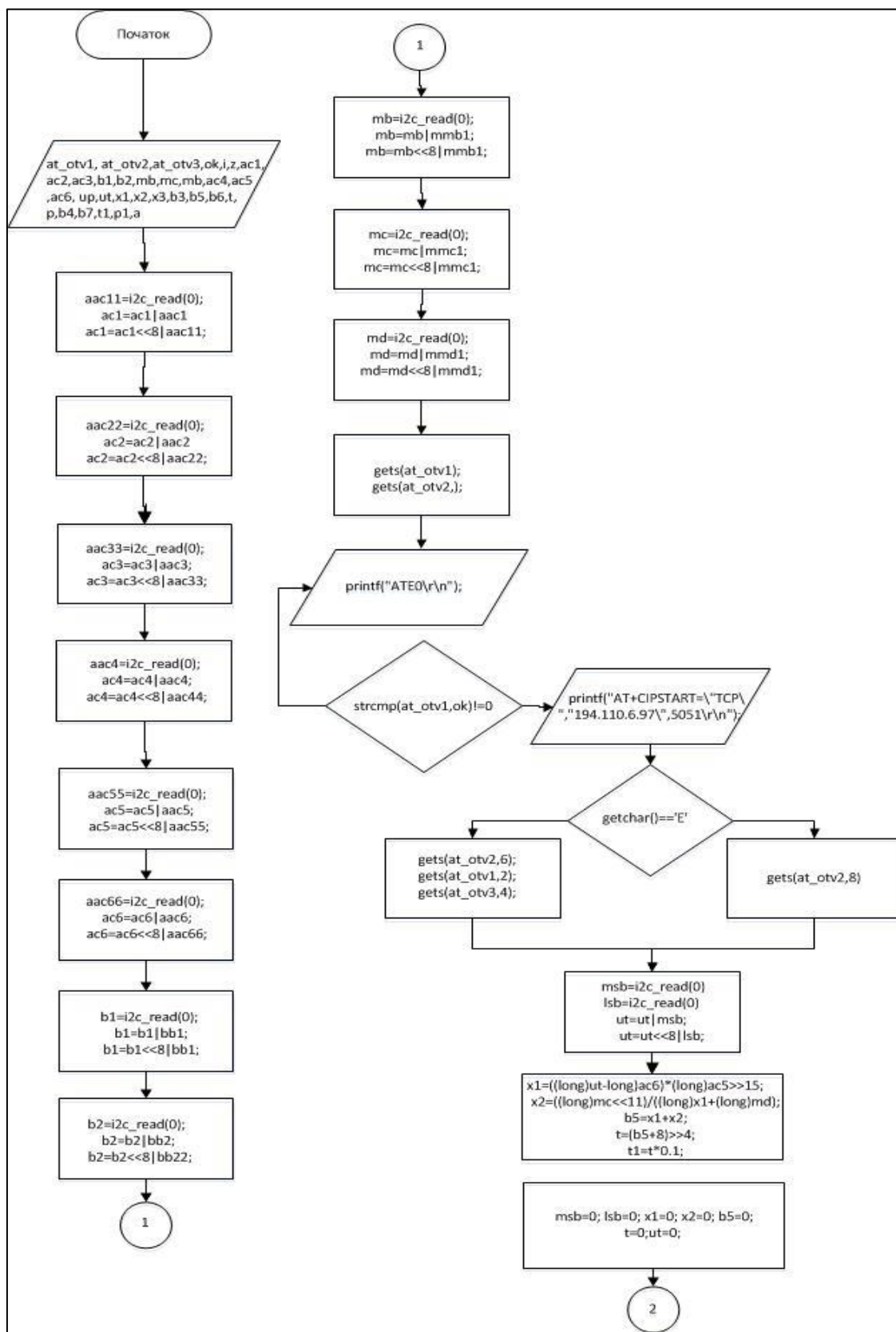


Рисунок А.1 – Початок алгоритму

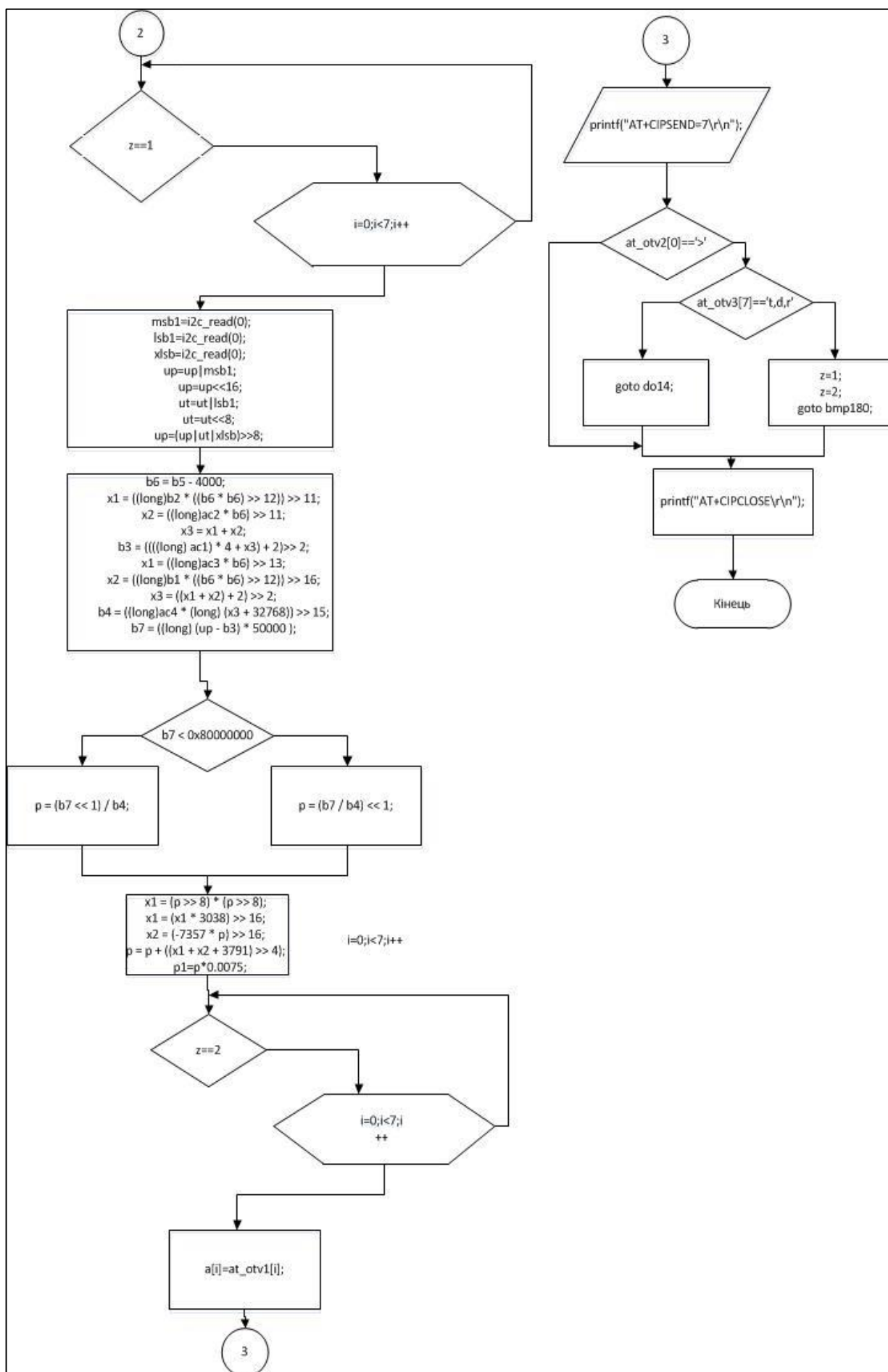


Рисунок А.2 – кінець алгоритму

ДОДАТОК Б

КОД ПРОГРАМИ

Лістинг Б.1 – Введення змінних

```
#include <mega8.h>
#include <delay.h>
#include <string.h>
#include <stdlib.h>
// I2C Bus functions
#include <i2c.h>
// Declare your global variables here
char at_otv1[60];
char at_otv2[60];
char at_otv3[60];
char ok[40]="\rOK\r";
char i=0,z=0;
short ac1=0,ac2=0,ac3=0,b1=0,b2=0,mb=0,mc=0,md=0;
unsigned short ac4=0,ac5=0,ac6=0;
char
aac1=0,aac11=0,aac2=0,aac22=0,aac3=0,aac33=0,bb1=0,bb11=0,bb2=0,bb22
=0,mmb=0,mmb1=0,mmc=0,mmc1=0,mmd=0,mmd1=0,aac4=0,aac44=0,aac5=0,aac5
5=0,aac6=0,aac66=0,msb=0,msb1=0,lsb=0,lsb1=0,xlsb=0;
long up=0,ut=0,x1=0,x2=0,x3=0,b3=0,b5=0,b6=0,t=0,p=0;
unsigned long b4=0,b7=0;
float t1=0,p1=0;
char a[7]="nachal";

#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)
#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<DOR)
```

Лістинг Б.2 – Буфер приймача USART

```
// USART Receiver buffer
#define RX_BUFFER_SIZE 160
char rx_buffer[RX_BUFFER_SIZE];

#if RX_BUFFER_SIZE <= 256
unsigned char rx_wr_index=0,rx_rd_index=0;
#else
unsigned int rx_wr_index=0,rx_rd_index=0;
#endif

#if RX_BUFFER_SIZE < 256
unsigned char rx_counter=0;
#else
unsigned int rx_counter=0;
#endif
```

Лістинг Б.3 – Прапор для переповнення буфера приймача USART

```
// This flag is set on USART Receiver buffer overflow
bit rx_buffer_overflow;
```

Лістинг Б.4 – Підпрограма обслуговування переривання приймача USART

```
// USART Receiver interrupt service routine
interrupt [USART_RXC] void usart_rx_isr(void)
{
    char status,data;
    status=UCSRA;
    data=UDR;
    if ((status & (FRAMING_ERROR | PARITY_ERROR |
DATA_OVERRUN))==0)
    {
        rx_buffer[rx_wr_index++]=data;
#ifdef RX_BUFFER_SIZE == 256
        // special case for receiver buffer size=256
        if (++rx_counter == 0) rx_buffer_overflow=1;
#else
        if (rx_wr_index == RX_BUFFER_SIZE) rx_wr_index=0;
        if (++rx_counter == RX_BUFFER_SIZE)
        {
            rx_counter=0;
            rx_buffer_overflow=1;
        }
#endif
    }
}

#ifdef _DEBUG_TERMINAL_IO_
// Get a character from the USART Receiver buffer
#define _ALTERNATE_GETCHAR_
#pragma used+
char getchar(void)
{
    char data;
    while (rx_counter==0);
    data=rx_buffer[rx_rd_index++];
#ifdef RX_BUFFER_SIZE != 256
    if (rx_rd_index == RX_BUFFER_SIZE) rx_rd_index=0;
#endif
    #asm("cli")
    --rx_counter;
    #asm("sei")
    return data;
}
#pragma used-
#endif

// USART Transmitter buffer
#define TX_BUFFER_SIZE 160
char tx_buffer[TX_BUFFER_SIZE];

#ifdef TX_BUFFER_SIZE <= 256
```

```

unsigned char tx_wr_index=0,tx_rd_index=0;
#else
unsigned int tx_wr_index=0,tx_rd_index=0;
#endif

#if TX_BUFFER_SIZE < 256
unsigned char tx_counter=0;
#else
unsigned int tx_counter=0;
#endif

// USART Transmitter interrupt service routine
interrupt [USART_TXC] void usart_tx_isr(void)
{
    if (tx_counter)
    {
        --tx_counter;
        UDR=tx_buffer[tx_rd_index++];
#if TX_BUFFER_SIZE != 256
        if (tx_rd_index == TX_BUFFER_SIZE) tx_rd_index=0;
#endif
    }
}

#ifndef _DEBUG_TERMINAL_IO_
// Write a character to the USART Transmitter buffer
#define _ALTERNATE_PUTCHAR_
#pragma used+
void putchar(char c)
{
    while (tx_counter == TX_BUFFER_SIZE);
    #asm("cli")
    if (tx_counter || ((UCSRA & DATA_REGISTER_EMPTY)==0))
    {
        tx_buffer[tx_wr_index++]=c;
#if TX_BUFFER_SIZE != 256
        if (tx_wr_index == TX_BUFFER_SIZE) tx_wr_index=0;
#endif
        ++tx_counter;
    }
    else
        UDR=c;
    #asm("sei")
}
#pragma used-
#endif

```

Лістинг Б.5 – Стандартні функції вводу / виводу

```

// Standard Input/Output functions
#include <stdio.h>

void main(void)

```

```

{
// Declare your local variables here

// Input/Output Ports initialization
// Port B initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In
Bit1=In Bit0=In
    DDRB=(0<<DDB7) | (0<<DDB6) | (0<<DDB5) | (0<<DDB4) | (0<<DDB3)
| (0<<DDB2) | (0<<DDB1) | (0<<DDB0);
    // State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T
Bit0=T
    PORTB=(0<<PORTB7) | (0<<PORTB6) | (0<<PORTB5) | (0<<PORTB4) |
(0<<PORTB3) | (0<<PORTB2) | (0<<PORTB1) | (0<<PORTB0);

    // Port C initialization
// Function: Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In
Bit0=In
    DDRC=(0<<DDC6) | (0<<DDC5) | (0<<DDC4) | (0<<DDC3) | (0<<DDC2)
| (0<<DDC1) | (0<<DDC0);
    // State: Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
    PORTC=(0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) | (0<<PORTC3) |
(0<<PORTC2) | (0<<PORTC1) | (0<<PORTC0);

    // Port D initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In
Bit1=In Bit0=In
    DDRD=(0<<DDD7) | (0<<DDD6) | (0<<DDD5) | (0<<DDD4) | (0<<DDD3)
| (0<<DDD2) | (0<<DDD1) | (0<<DDD0);
    // State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T
Bit0=T
    PORTD=(0<<PORTD7) | (0<<PORTD6) | (0<<PORTD5) | (0<<PORTD4) |
(0<<PORTD3) | (0<<PORTD2) | (0<<PORTD1) | (0<<PORTD0);

    // USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud Rate: 4800
    UCSRA=(0<<RXC) | (0<<TXC) | (0<<UDRE) | (0<<FE) | (0<<DOR) |
(0<<UPE) | (0<<U2X) | (0<<MPCM);
    UCSRB=(1<<RXCIE) | (1<<TXCIE) | (0<<UDRIE) | (1<<RXEN) |
(1<<TXEN) | (0<<UCSZ2) | (0<<RXB8) | (0<<TXB8);
    UCSRC=(1<<URSEL) | (0<<UMSEL) | (0<<UPM1) | (0<<UPM0) |
(0<<USBS) | (1<<UCSZ1) | (1<<UCSZ0) | (0<<UCPOL);
    UBRRH=0x00;
    UBRRL=0x33;

    // Bit-Banged I2C Bus initialization
// I2C Port: PORTC
// I2C SDA bit: 4
// I2C SCL bit: 5

```

```

// Bit Rate: 100 kHz
// Note: I2C settings are specified in the
// Project|Configure|C Compiler|Libraries|I2C menu.
i2c_init();

//AC1
i2c_start(); //Кидаем команду "Старт" на шину I2C
i2c_write(0xEE); //Записываем адрес акса в режиме записи.
i2c_write(0xAA); //Указываем адрес регистра ACCEL_ZOUT_H
i2c_stop();

i2c_start(); //Кидаем команду "Старт" на шину I2C
i2c_write(0xEF); //Записываем адрес гироскопа в режиме
чтения.
aac1=i2c_read(0); //Читаем число из устройства с нулевым
адресом и присваиваем переменной "a".
i2c_stop();

i2c_start(); //Кидаем команду "Старт" на шину I2C
i2c_write(0xEE); //Записываем адрес акса в режиме записи.
i2c_write(0xAA); //Указываем адрес регистра ACCEL_ZOUT_H
i2c_stop();

i2c_start(); //Кидаем команду "Старт" на шину I2C
i2c_write(0xEF); //Записываем адрес гироскопа в режиме
чтения.
aac11=i2c_read(0); //Читаем число из устройства с нулевым
адресом и присваиваем переменной "a".
i2c_stop();
ac1=ac1|aac1;
ac1=ac1<<8|aac11;

// AC2

i2c_start(); //Кидаем команду "Старт" на шину I2C
i2c_write(0xEE); //Записываем адрес акса в режиме записи.
i2c_write(0xAC); //Указываем адрес регистра ACCEL_ZOUT_H
i2c_stop();

i2c_start(); //Кидаем команду "Старт" на шину I2C
i2c_write(0xEF); //Записываем адрес гироскопа в режиме
чтения.
aac2=i2c_read(0); //Читаем число из устройства с нулевым
адресом и присваиваем переменной "a".
i2c_stop();

i2c_start(); //Кидаем команду "Старт" на шину I2C
i2c_write(0xEE); //Записываем адрес акса в режиме записи.
i2c_write(0xAD); //Указываем адрес регистра ACCEL_ZOUT_H
i2c_stop();

i2c_start(); //Кидаем команду "Старт" на шину I2C

```

```

        i2c_write(0xEF); //Записываем адрес гироскопа в режиме
чтения.
        aac22=i2c_read(0); //Читаем число из устройства с нулевым
адресом и присваиваем переменной "a".
        i2c_stop();
        ac2=ac2|aac2;
        ac2=ac2<<8|aac22;

//AC3

        i2c_start(); //Кидаем команду "Старт" на шину I2C
        i2c_write(0xEE); //Записываем адрес акса в режиме записи.
        i2c_write(0xAE); //Указываем адрес регистра ACCEL_ZOUT_H
        i2c_stop();

        i2c_start(); //Кидаем команду "Старт" на шину I2C
        i2c_write(0xEF); //Записываем адрес гироскопа в режиме
чтения.
        aac3=i2c_read(0); //Читаем число из устройства с нулевым
адресом и присваиваем переменной "a".
        i2c_stop();

        i2c_start(); //Кидаем команду "Старт" на шину I2C
        i2c_write(0xEE); //Записываем адрес акса в режиме записи.
        i2c_write(0xAF); //Указываем адрес регистра ACCEL_ZOUT_H
        i2c_stop();

        i2c_start(); //Кидаем команду "Старт" на шину I2C
        i2c_write(0xEF); //Записываем адрес гироскопа в режиме
чтения.
        aac33=i2c_read(0); //Читаем число из устройства с нулевым
адресом и присваиваем переменной "a".
        i2c_stop();
        ac3=ac3|aac3;
        ac3=ac3<<8|aac33;

//AC4

        i2c_start(); //Кидаем команду "Старт" на шину I2C
        i2c_write(0xEE); //Записываем адрес акса в режиме записи.
        i2c_write(0xB0); //Указываем адрес регистра ACCEL_ZOUT_H
        i2c_stop();

        i2c_start(); //Кидаем команду "Старт" на шину I2C
        i2c_write(0xEF); //Записываем адрес гироскопа в режиме
чтения.
        aac4=i2c_read(0); //Читаем число из устройства с нулевым
адресом и присваиваем переменной "a".
        i2c_stop();

        i2c_start(); //Кидаем команду "Старт" на шину I2C
        i2c_write(0xEE); //Записываем адрес акса в режиме записи.
        i2c_write(0xB1); //Указываем адрес регистра ACCEL_ZOUT_H

```

```

i2c_stop();

i2c_start(); //Кидаем команду "Старт" на шину I2C
i2c_write(0xEF); //Записываем адрес гироскопа в режиме
чтения.
aac4=i2c_read(0); //Читаем число из устройства с нулевым
адресом и присваиваем переменной "a".
i2c_stop();
ac4=ac4|aac4;
ac4=ac4<<8|aac4;

//AC5

i2c_start(); //Кидаем команду "Старт" на шину I2C
i2c_write(0xEE); //Записываем адрес акса в режиме записи.
i2c_write(0xB2); //Указываем адрес регистра ACCEL_ZOUT_H
i2c_stop();

i2c_start(); //Кидаем команду "Старт" на шину I2C
i2c_write(0xEF); //Записываем адрес гироскопа в режиме
чтения.
aac5=i2c_read(0); //Читаем число из устройства с нулевым
адресом и присваиваем переменной "a".
i2c_stop();

i2c_start(); //Кидаем команду "Старт" на шину I2C
i2c_write(0xEE); //Записываем адрес акса в режиме записи.
i2c_write(0xB3); //Указываем адрес регистра ACCEL_ZOUT_H
i2c_stop();

i2c_start(); //Кидаем команду "Старт" на шину I2C
i2c_write(0xEF); //Записываем адрес гироскопа в режиме
чтения.
aac55=i2c_read(0); //Читаем число из устройства с нулевым
адресом и присваиваем переменной "a".
i2c_stop();
ac5=ac5|aac5;
ac5=ac5<<8|aac55;

//AC6

i2c_start(); //Кидаем команду "Старт" на шину I2C
i2c_write(0xEE); //Записываем адрес акса в режиме записи.
i2c_write(0xB4); //Указываем адрес регистра ACCEL_ZOUT_H
i2c_stop();

i2c_start(); //Кидаем команду "Старт" на шину I2C
i2c_write(0xEF); //Записываем адрес гироскопа в режиме
чтения.
aac6=i2c_read(0); //Читаем число из устройства с нулевым
адресом и присваиваем переменной "a".
i2c_stop();

```

```

i2c_start(); //Кидаем команду "Старт" на шину I2C
i2c_write(0xEE); //Записываем адрес акса в режиме записи.
i2c_write(0xB5); //Указываем адрес регистра ACCEL_ZOUT_H
i2c_stop();

i2c_start(); //Кидаем команду "Старт" на шину I2C
i2c_write(0xEF); //Записываем адрес гироскопа в режиме
чтения.
aac66=i2c_read(0); //Читаем число из устройства с нулевым
адресом и присваиваем переменной "a".
i2c_stop();
ac6=ac6|aac6;
ac6=ac6<<8|aac66;

//B1

i2c_start(); //Кидаем команду "Старт" на шину I2C
i2c_write(0xEE); //Записываем адрес акса в режиме записи.
i2c_write(0xB6); //Указываем адрес регистра ACCEL_ZOUT_H
i2c_stop();

i2c_start(); //Кидаем команду "Старт" на шину I2C
i2c_write(0xEF); //Записываем адрес гироскопа в режиме
чтения.
bb1=i2c_read(0); //Читаем число из устройства с нулевым
адресом и присваиваем переменной "a".
i2c_stop();

i2c_start(); //Кидаем команду "Старт" на шину I2C
i2c_write(0xEE); //Записываем адрес акса в режиме записи.
i2c_write(0xB7); //Указываем адрес регистра ACCEL_ZOUT_H
i2c_stop();

i2c_start(); //Кидаем команду "Старт" на шину I2C
i2c_write(0xEF); //Записываем адрес гироскопа в режиме
чтения.
bb11=i2c_read(0); //Читаем число из устройства с нулевым
адресом и присваиваем переменной "a".
i2c_stop();
b1=b1|bb1;
b1=b1<<8|bb11;

//B2

i2c_start(); //Кидаем команду "Старт" на шину I2C
i2c_write(0xEE); //Записываем адрес акса в режиме записи.
i2c_write(0xB8); //Указываем адрес регистра ACCEL_ZOUT_H
i2c_stop();

i2c_start(); //Кидаем команду "Старт" на шину I2C
i2c_write(0xEF); //Записываем адрес гироскопа в режиме
чтения.

```

```

        bb2=i2c_read(0); //Читаем число из устройства с нулевым
адресом и присваиваем переменной "a".
        i2c_stop();

        i2c_start(); //Кидаем команду "Старт" на шину I2C
        i2c_write(0xEE); //Записываем адрес акса в режиме записи.
        i2c_write(0xB9); //Указываем адрес регистра ACCEL_ZOUT_H
        i2c_stop();

        i2c_start(); //Кидаем команду "Старт" на шину I2C
        i2c_write(0xEF); //Записываем адрес гироскопа в режиме
чтения.
        bb22=i2c_read(0); //Читаем число из устройства с нулевым
адресом и присваиваем переменной "a".
        i2c_stop();
        b1=b1|bb1;
        b2=b2<<8|bb22;

        //MB

        i2c_start(); //Кидаем команду "Старт" на шину I2C
        i2c_write(0xEE); //Записываем адрес акса в режиме записи.
        i2c_write(0xBA); //Указываем адрес регистра ACCEL_ZOUT_H
        i2c_stop();

        i2c_start(); //Кидаем команду "Старт" на шину I2C
        i2c_write(0xEF); //Записываем адрес гироскопа в режиме
чтения.
        mmb=i2c_read(0); //Читаем число из устройства с нулевым
адресом и присваиваем переменной "a".
        i2c_stop();

        i2c_start(); //Кидаем команду "Старт" на шину I2C
        i2c_write(0xEE); //Записываем адрес акса в режиме записи.
        i2c_write(0xBB); //Указываем адрес регистра ACCEL_ZOUT_H
        i2c_stop();

        i2c_start(); //Кидаем команду "Старт" на шину I2C
        i2c_write(0xEF); //Записываем адрес гироскопа в режиме
чтения.
        mmb1=i2c_read(0); //Читаем число из устройства с нулевым
адресом и присваиваем переменной "a".
        i2c_stop();
        mb=mb|mmb;
        mb=mb<<8|mmb1;

        //MC

        i2c_start(); //Кидаем команду "Старт" на шину I2C
        i2c_write(0xEE); //Записываем адрес акса в режиме записи.
        i2c_write(0xBC); //Указываем адрес регистра ACCEL_ZOUT_H
        i2c_stop();

```

```

        i2c_start(); //Кидаем команду "Старт" на шину I2C
        i2c_write(0xEF); //Записываем адрес гироскопа в режиме
чтения.
        mmc=i2c_read(0); //Читаем число из устройства с нулевым
адресом и присваиваем переменной "a".
        i2c_stop();

        i2c_start(); //Кидаем команду "Старт" на шину I2C
        i2c_write(0xEE); //Записываем адрес акса в режиме записи.
        i2c_write(0xBD); //Указываем адрес регистра ACCEL_ZOUT_H
        i2c_stop();

        i2c_start(); //Кидаем команду "Старт" на шину I2C
        i2c_write(0xEF); //Записываем адрес гироскопа в режиме
чтения.
        mmc1=i2c_read(0); //Читаем число из устройства с нулевым
адресом и присваиваем переменной "a".
        i2c_stop();
        mc=mc|mmc;
        mc=mc<<8|mmc1;

        //MD

        i2c_start(); //Кидаем команду "Старт" на шину I2C
        i2c_write(0xEE); //Записываем адрес акса в режиме записи.
        i2c_write(0xBE); //Указываем адрес регистра ACCEL_ZOUT_H
        i2c_stop();

        i2c_start(); //Кидаем команду "Старт" на шину I2C
        i2c_write(0xEF); //Записываем адрес гироскопа в режиме
чтения.
        mmd=i2c_read(0); //Читаем число из устройства с нулевым
адресом и присваиваем переменной "a".
        i2c_stop();

        i2c_start(); //Кидаем команду "Старт" на шину I2C
        i2c_write(0xEE); //Записываем адрес акса в режиме записи.
        i2c_write(0xBF); //Указываем адрес регистра ACCEL_ZOUT_H
        i2c_stop();

        i2c_start(); //Кидаем команду "Старт" на шину I2C
        i2c_write(0xEF); //Записываем адрес гироскопа в режиме
чтения.
        mmd1=i2c_read(0); //Читаем число из устройства с нулевым
адресом и присваиваем переменной "a".
        i2c_stop();
        md=md|mmd;
        md=md<<8|mmd1;

```

Лістинг Б.6 – Глобальні переривання включення

```
// Global enable interrupts
```

```

#asm("sei")

while (1)
{
    delay_ms(2000);
    gets(at_otv1,40);
    gets(at_otv2,32);
    gets(at_otv1,2);
    gets(at_otv2,7);
    gets(at_otv1,16);
    //gets(at_otv2,16);
    gets(at_otv1,13);
    do1:
    memset(at_otv1,0,60);
    memset(at_otv2,0,60);
    printf("ATE0\r\n");
    delay_ms(3000);
    gets(at_otv2,7);
    gets(at_otv1,2);
    gets(at_otv3,4);
    strcat(at_otv1,at_otv3);
    if(strcmp(at_otv1,ok)!=0) {
        gets(at_otv2,6);
        delay_ms(3000);
        goto do1;}
    do3:
    memset(at_otv1,0,40);
    memset(at_otv2,0,40);
    memset(at_otv3,0,40);

    printf("AT+CIPSTART=\"TCP\", \"194.110.6.97\", 5051\r\n");
    delay_ms(3000);
    gets(at_otv1,2);
    if(getchar()=='E')
    {gets(at_otv2,8);
    delay_ms(3000);
    goto do3;}
    else{
    gets(at_otv2,6);
    gets(at_otv1,2);
    gets(at_otv3,4);
    strcat(at_otv1,at_otv3);};
    if(strcmp(at_otv1,ok)!=0){delay_ms(3000);
    goto do3;}

    bmp180:
    //kod bmp;

    i2c_start();
    i2c_write(0xEE);//записать
    i2c_write(0xF4);//адрес куда регистра,адрес частота
    i2c_write(0x2E);//данные, что
    записать,8000/(1+0x07)=1000

```

```

        i2c_stop();
        delay_ms(5);

        i2c_start(); //Кидаем команду "Старт" на шину I2C
        i2c_write(0xEE); //Записываем адрес акса в режиме записи.
        i2c_write(0xF6); //Указываем адрес регистра ACCEL_ZOUT_H
        i2c_stop();

        i2c_start(); //Кидаем команду "Старт" на шину I2C
        i2c_write(0xEF); //Записываем адрес гироскопа в режиме
чтения.
        msb=i2c_read(0); //Читаем число из устройства с нулевым
адресом и присваиваем переменной "a".
        i2c_stop(); //Останавливаем шину.

        i2c_start(); //Кидаем команду "Старт" на шину I2C
        i2c_write(0xEE); //Записываем адрес акса в режиме записи.
        i2c_write(0xF7); //Указываем адрес регистра ACCEL_ZOUT_H

        i2c_start(); //Кидаем команду "Старт" на шину I2C
        i2c_write(0xEF); //Записываем адрес гироскопа в режиме
чтения.
        lsb=i2c_read(0); //Читаем число из устройства с нулевым
адресом и присваиваем переменной "a".
        i2c_stop(); //Останавливаем шину.

        ut=ut|msb;
        ut=ut<<8|lsb;

        x1=((long)ut-(long)ac6)*(long)ac5>>15;
        x2=((long)mc<<11)/((long)x1+(long)md);
        b5=x1+x2;
        t=(b5+8)>>4;
        t1=t*0.1;

        memset(at_otv1,0,7);
        ftoa(t1,1,at_otv1);

        msb=0;
        lsb=0;
        x1=0;
        x2=0;
        b5=0;
        t=0;
        ut=0;
        if (z==1)
        {
            for (i=0;i<7;i++)

            {
                a[i]=at_otv1[i];
            }
        }
};

```

```

goto do14;
};

i2c_start();
i2c_write(0xEE); //записать
i2c_write(0xF4); //адрес куда регистра, адрес частота
i2c_write(0x34); //данные, что
записать, 8000/(1+0x07)=1000
i2c_stop();
delay_ms(5);

i2c_start(); //Кидаем команду "Старт" на шину I2C
i2c_write(0xEE); //Записываем адрес акса в режиме записи.
i2c_write(0xF6); //Указываем адрес регистра ACCEL_ZOUT_H
i2c_stop();

i2c_start(); //Кидаем команду "Старт" на шину I2C
i2c_write(0xEF); //Записываем адрес гироскопа в режиме
чтения.
msb1=i2c_read(0); //Читаем число из устройства с нулевым
адресом и присваиваем переменной "a".
i2c_stop(); //Останавливаем шину.

i2c_start(); //Кидаем команду "Старт" на шину I2C
i2c_write(0xEE); //Записываем адрес акса в режиме записи.
i2c_write(0xF7); //Указываем адрес регистра ACCEL_ZOUT_H

i2c_start(); //Кидаем команду "Старт" на шину I2C
i2c_write(0xEF); //Записываем адрес гироскопа в режиме
чтения.
lsb1=i2c_read(0); //Читаем число из устройства с нулевым
адресом и присваиваем переменной "a".
i2c_stop(); //Останавливаем шину.

i2c_start(); //Кидаем команду "Старт" на шину I2C
i2c_write(0xEE); //Записываем адрес акса в режиме записи.
i2c_write(0xF8); //Указываем адрес регистра ACCEL_ZOUT_H

i2c_start(); //Кидаем команду "Старт" на шину I2C
i2c_write(0xEF); //Записываем адрес гироскопа в режиме
чтения.
xlsb=i2c_read(0); //Читаем число из устройства с нулевым
адресом и присваиваем переменной "a".
i2c_stop(); //Останавливаем шину.

up=up|msb1;
up=up<<16;
ut=ut|lsb1;
ut=ut<<8;
up=(up|ut|xlsb)>>8;

```

```

    b6 = b5 - 4000;
x1 = ((long)b2 * ((b6 * b6) >> 12)) >> 11;
x2 = ((long)ac2 * b6) >> 11;
x3 = x1 + x2;
b3 = (((long) ac1) * 4 + x3) + 2)>> 2;
x1 = ((long)ac3 * b6) >> 13;
x2 = ((long)b1 * ((b6 * b6) >> 12)) >> 16;
x3 = ((x1 + x2) + 2) >> 2;
b4 = ((long)ac4 * (long) (x3 + 32768)) >> 15;
b7 = ((long) (up - b3) * 50000 );

if (b7 < 0x80000000)
{
    p = (b7 << 1) / b4;
}
else
{
    p = (b7 / b4) << 1;
};

x1 = (p >> 8) * (p >> 8);
x1 = (x1 * 3038) >> 16;
x2 = (-7357 * p) >> 16;
p = p + ((x1 + x2 + 3791) >> 4);
p1=p*0.0075;

    memset(at_otv1,0,7);
    ftoa(p1,1,at_otv1);
    msb1=0;
    lsb1=0;
    x1=0;
    x2=0;
    x3=0;
    b6=0;
    b4=0;
    b7=0;
    p=0;
    up=0;
    ut=0;
    if (z==2)
    {
        for(i=0;i<7;i++)
        {
            a[i]=at_otv1[i];
        };
        goto do14;
    };

    do14:
        memset(at_otv1,0,40);
        memset(at_otv2,0,40);
        memset(at_otv3,0,40);
        printf("AT+CIPSEND=7\r\n");
        //delay_ms(3000);

```

```

        gets(at_otv1,2);
gets(at_otv3,4);
gets(at_otv2,2);
if(at_otv2[0]=='>')
{
puts(a);
putchar('\r');
putchar('\n');
//memset(a,0,7);
//delay_ms(500);
        gets(at_otv1,2);
        gets(at_otv1,11);
        gets(at_otv1,2);
        gets(at_otv1,14);
        gets(at_otv1,2);
        gets(at_otv1,9);
gets(at_otv1,2);
        gets(at_otv3,8);
        if(at_otv3[7]=='t')
        { z=1;
goto bmp180;  };

        if(at_otv3[7]=='d')
        {z=2;
        goto bmp180;};

        if(at_otv3[7]=='r')
        { goto do15;  };

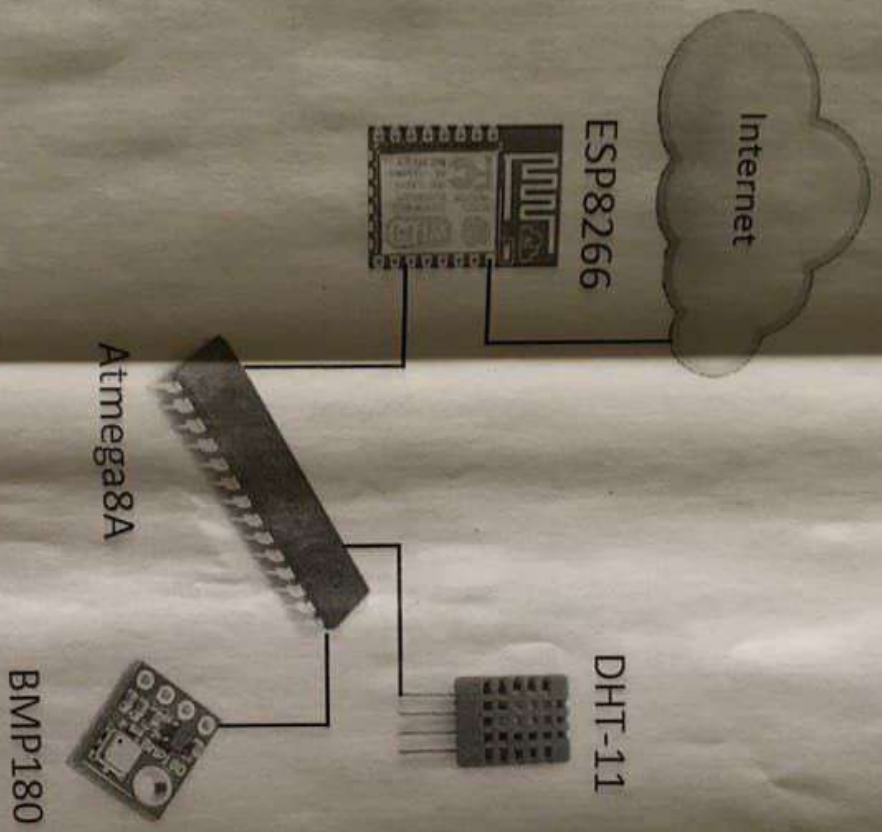
goto do14;
}
else{goto do14;};

do15:
memset(at_otv1,0,40);
printf("AT+CIPCLOSE\r\n");
gets(at_otv1,8);
gets(at_otv1,2);
gets(at_otv2,4);
goto do3;

}
}

```

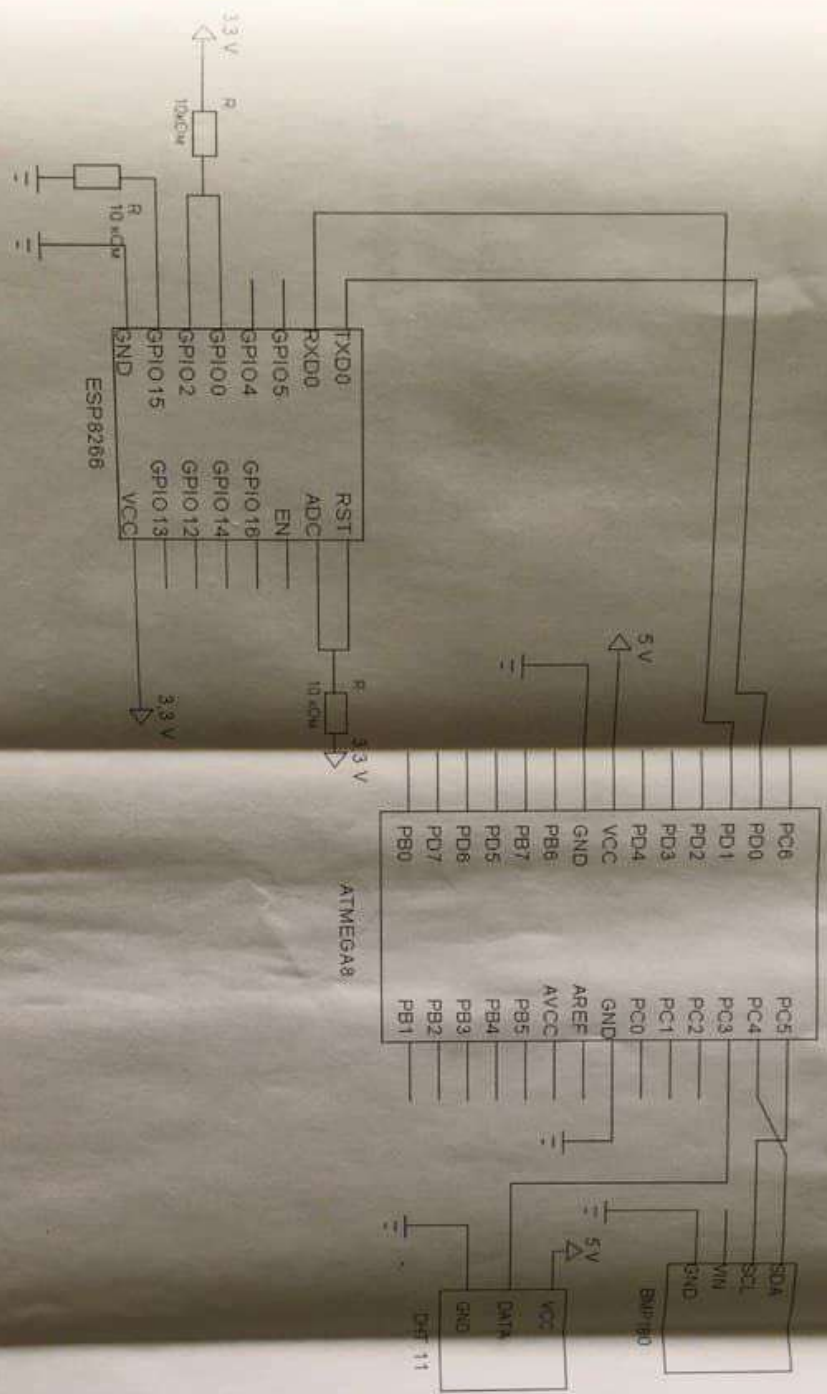
Структура системи



13.02070849.00062 ПП1																															
Розробка системи автоматизованого вимірювання показників в атмосферних умовах в концепції Інтернет речей																															
Структура системи																															
<table><tr><td colspan="2">Дат.</td><td colspan="2">Місяц</td><td colspan="2">Рік</td></tr><tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr><tr><td colspan="2">Діагн. з</td><td colspan="4">Діагн. 4</td></tr></table>														Дат.		Місяц		Рік		1	2	3	4	5	6	Діагн. з		Діагн. 4			
Дат.		Місяц		Рік																											
1	2	3	4	5	6																										
Діагн. з		Діагн. 4																													
НЗ «Запорізька політехнічна ВНУ» КНТ-537ср																															
Зав.		Львів		№ докум.		Шифр		Дата		Зав.		Львів		№ докум.		Шифр		Дата													
Розроб.		Назаренко О.І.		Технічна М.О.						Розроб.		Назаренко О.І.		Технічна М.О.																	
Перев.										Перев.																					
Т. експерт										Т. експерт																					
Накази										Накази																					
Затв.										Затв.																					

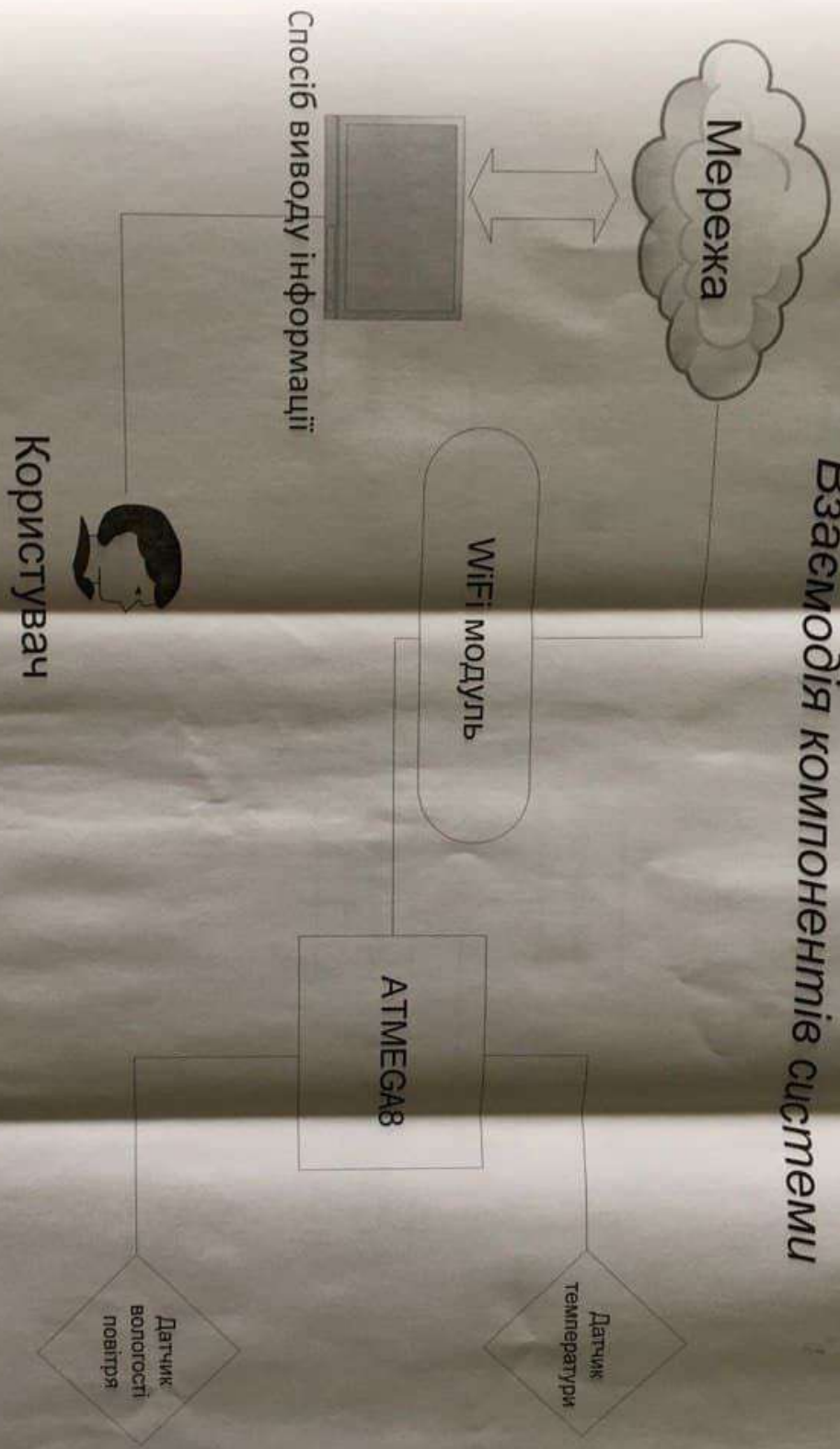
НЗ «Зарплати»
розробника
КНТ-537СН

Принципова схема системи



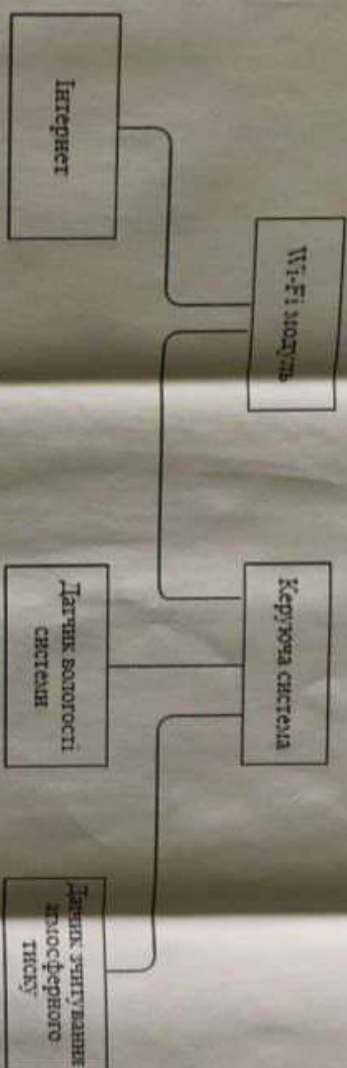
13.02070849.00062 ПП2				Розробка системи відслідковування атмосферних показників в концепції Інтернет речей			
Стор.	Варіант	М. розроб.	Ім'я	Ім'я	Дата	Лист	
Розроб.	Назаренко О.І.	Ім'я	Ім'я	Ім'я	Дата	Лист	
Перев.	Ткаченко М.Ю.	Ім'я	Ім'я	Ім'я	Дата	Лист	
Т. контроль.		Ім'я	Ім'я	Ім'я	Дата	Лист	
Н. контроль.	Зоря О.В.	Ім'я	Ім'я	Ім'я	Дата	Лист	
Замов.	Міністерство Д.С.	Ім'я	Ім'я	Ім'я	Дата	Лист	
(Принципова схема системи)				ІНУ в Закарпатській області Міністерство: КНТ-53/ІТ			

Взаємодія компонентів системи



13.02070849.00062 ПЛЗ									
Ім'я	Піде	М. Ім'я	Піде	Ім'я	Розробка системи впровадженні атмосферних показників в контексті Інтернет речей				
Підпис	Підпис	Підпис	Підпис	Підпис					
Підпис	Підпис	Підпис	Підпис	Підпис					
Підпис	Підпис	Підпис	Підпис	Підпис					
Назва	Назва	Назва	Назва	Назва	Взаємодія компонентів системи				
Назва	Назва	Назва	Назва	Назва					
ІН «комп'ютерна технологія» КНТ-53109									

Структурна схема системи



13.02070849.00062 ПП4			
Назва	Модель	Питання	Відповідь
Розробка	Інженер О.І.	Розробка системи вимірювання атмосферних показників в контейнері Імпортів Речей	
Технік	Технік М.В.		
Технік	Технік О.В.		
Інженер	Інженер Р.В.		
Інженер	Інженер Р.В.		
Структурна схема системи		ІНУ «Запорізька політехніка» КНТ-537сн	