МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) Автоматизації і комп'ютерних систем
Кафедра Інформаційних технологій, штучного інтелекту і кібербезпеки

«До захисту в ЕК»	**«До захисту допущено»**
Директор інституту(декан факультету)	Завідувач кафедри
_____ Андрій ФОРСЮК	_____ Сергій ГРИБКОВ
(підпис) (ім'я та прізвище)	(підпис) (ім'я та прізвище)
«10» червня 2024р.	«10» червня 2024р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

зі спеціальності 122 «Комп'ютерні науки»
(код та назва спеціальності)
освітньо-професійної програми Комп'ютерні науки

на тему: Розроблення інформаційної системи обробки метеоданих портативної метеостанції на базі ESP8266

Виконав: здобувач 4 курсу, групи КН-4-2

Коломієць Іван Вікторович _____
(прізвище, ім'я, по батькові повністю) (підпис)

Керівник Андріюк Олена Петрівна _____
(прізвище , ім'я та по батькові повністю) (підпис)

Консультанти _____ _____
(ім'я та прізвище) (підпис)

_____ _____
(ім'я та прізвище) (підпис)

_____ _____
(ім'я та прізвище) (підпис)

Рецензент _____ _____
(ім'я та прізвище) (підпис)

Я як здобувач(ка) Національного університету харчових технологій розумію і підтримую політику університету з академічної доброчесності. Я не надавав(-ла) і не одержував(-ла) недозволеної допомоги під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Здобувач _____
(підпис)

Київ - 2024р.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) _Автоматизації і комп'ютерних систем_
Кафедра _Інформаційних технологій, штучного інтелекту і кібербезпеки_
Освітній ступінь _бакалавр_
Спеціальність _122 «Комп'ютерні науки»_
Освітньо-професійна програма _ Комп'ютерні науки_

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інформаційних технологій, штучного інтелекту і кібербезпеки
_____ Сергій ГРИБКОВ

" 15 " квітня 2024 року

З А В Д А Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА

Коломійця Івана Вікторовича
(прізвище, ім'я, по батькові)

1. Тема роботи «Розроблення інформаційної системи обробки метеоданих портативної метеостанції на базі ESP8266»
керівник роботи Андріюк Олена Петрівна доцент, к.ф – м.н.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від 15 квітня 2024 року № 279-кс
2. Строк подання здобувачем роботи 27.05.2024 р.
3. Вхідні дані до роботи
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
1) Системний аналіз об'єкту дослідження
2) Проектування інформаційної системи
3) Охорона праці та техніка безпеки
5. Перелік графічного матеріалу
1) Функціональна схема
2) Інтерфейс системи

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1.	Андріюк О.П. , доцент	15.04.24	15.04.24
2.	Андріюк О.П., доцент	15.04.24	15.04.24
3.	Андріюк О.П.. , доцент	15.04.24	15.04.24

7. Дата видачі завдання 15 квітня 2024 року

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів виконання кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Дослідження предметної області та постановка завдання на проектування	15.04.2024 – 20.04.2024	Виконано
2	Створення інформаційної системи	20.04.2024 – 15.05.2024	Виконано
3	Оформлення пояснювальної записки	15.05.2024 – 26.05.2024	Виконано
4	Оформлення презентації	26.04.2024 – 27.05.2024	Виконано

Здобувач _____ Іван КОЛОМІЄЦЬ

(підпис) (прізвище та ініціали)

Керівник роботи _____ Олена АНДРІЮК

(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Дипломний проєкт: 67 сторінок, 52 рисунків, 15 джерел, 1 таблиця.

Дана робота на тему «Розроблення інформаційної системи обробки метеоданих портативної метеостанції на базі ESP8266» має в собі мету опис процесу створення повноцінної системи обробки даних портативної метеостанції. Розглянемо усі аспекти розробки починаючи від аналізу впливу метрологічних даних на людину, закінчуючи поширенням даних в загальний доступ та надання можливості отримувати дані іншим користувачам.

КЛЮЧОВІ СЛОВА: ESP8266, ARDUINO IDE, TELEGRAM, BOT, API, HTML, WEB, CSS, BMP280, BH1750, I2C, C/C++, СИСТЕМА, МЕТЕОСТАНЦІЯ, МЕТЕОДАНІ.

ANNOTATION.

Diploma project: 67 pages, 52 figures, 15 sources, 1 table.

This work on the topic “Development of an information system for processing meteorological data of a portable weather station based on ESP8266” aims to describe the process of creating a full-fledged data processing system for a portable weather station. We will consider all aspects of the development, from analyzing the impact of metrological data on humans to distributing data to the public and providing the ability to receive data to other users.

KEYWORDS: ESP8266, ARDUINO IDE, TELEGRAM, BOT, API, HTML, WEB, CSS, BMP280, BH1750, I2C, C/C++, SYSTEM, WEATHER STATION, METEOROLOGICAL DATA.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

IoT – Інтернет речей.

ЕРС – Електромагнітний індуктивний ефект.

ТКО – Температурний коефіцієнт опору.

I2C – протокол передачі даних.

USB – інтерфейс для з’єднання пристроїв.

Wi-Fi – технологія бездротового зв’язку.

IDE – інтегроване середовище розробки.

SDA – лінія серійних даних.

SCL – лінія серійного годинника.

API – набір правил та механізмів для взаємодії програм між собою.

SSL/TLS – криптографічні протоколи.

ПК – Персональний комп’ютер.

ДСТУ – Державний стандарт України.

ТЕС – Теплова електростанція.

ГЕС – Гідроелектростанція.

СЕС – Сонячна електростанція.

ВСТУП

З поширенням Інтернету речей (IoT) та розвитком бездротових технологій портативні метеостанції мають вирішальне значення для моніторингу та збору даних про погоду в режимі реального часу, їх зручність та мобільність роблять їх незамінним інструментом для наукових досліджень, сільськогосподарського сектору. метеорологічні діагностичні станції, туризм та багато інших галузей.

У той же час розробка мікроконтролерних технологій і вбудованих систем відкриває нові можливості для створення і вдосконалення портативних метеостанцій. 1. Один з цих прогресивних мікроконтролерів, ESP8266, є ідеальною платформою для впровадження портативних пристроїв для збору та обробки даних про погоду.

Метою створення даної кваліфікаційної роботи є розробка інформаційної системи обробки метеоданих портативної метеостанції на базі ESP8266.

Відповідно до мети були визначені такі завдання:

- Провести аналіз впливу метеоданих на життя людини
- Проаналізувати використання систем контролю метеоданих в інших сферах
- Обґрунтувати доцільність проєктування і розробки
- Скласти технічне завдання
- Описати комплекс задач розробки системи

РОЗДІЛ 1. СИСТЕМНИЙ АНАЛІЗ ОБ'ЄКТА ДОСЛІДЖЕННЯ ТА ПОСТАНОВКА ЗАДАЧІ НА ПРОЄКТУВАННЯ

1.1. Вплив метеоданих на життя людини

В сучасному світі наявність актуальних даних про навколишнє середовище є необхідністю для комфортного життя. Починаючи від вибору одягу для майбутньої прогулянки і закінчуючи вибором оптимальної температури та вологості для комфортної роботи. Люди які часто мандрують між різними місцями України, відмічають що в залежності від міста в якому вони перебувають змінюється їхнє самопочуття та продуктивність. Причиною цього є зміна температури, вологості, тиску та інше.

Вплив метрологічних даних на людей може бути значним, особливо з точки зору здоров'я та безпеки. Метеодані стосуються вимірювання та оцінки різних фізичних величин, таких як температура, вологість, тиск та освітлення. Деякі аспекти впливу на людину включають:

Комфорт і здоров'я: правильно структуровані параметри навколишнього середовища, визначені з використанням метрологічних даних, можуть мати позитивний вплив на комфорт і здоров'я людини. Наприклад, оптимальна температура і вологість сприяють підтримці фізичного і психічного комфорту.

Безпека праці: Вимірюючи такі параметри, як рівень шуму, освітлення та рівень радіації, ви можете визначити потенційну небезпеку для здоров'я ваших співробітників. Наприклад, високий рівень шуму може призвести до погіршення слуху, а недостатнє освітлення може призвести до втоми та погіршення зору.

Якість життя: підтримка оптимальних умов життя та праці, що забезпечуються кількісними даними, сприяє поліпшенню якості життя. Наприклад, правильне регулювання рівня вологості може запобігти появі грибків та алергічних реакцій.

Професійна продуктивність: оптимальні умови праці, створені на основі метрологічних даних, сприяють підвищенню продуктивності праці працівників і зниження кількості виробничих дефектів.

Управління стресом: дефекти робочого середовища, такі як надмірний шум і недостатнє освітлення, можуть викликати стрес і погіршити психічне здоров'я. Належний облік даних зважування може допомогти зменшити ці ризики.

1.2. Використання систем контролю метеоданих в інших сферах

Використання систем контролю метеоданих в сферах промисловості, сільського господарства, транспорту, енергетики та інших галузях є ключовим для забезпечення ефективності та безпеки різних процесів.

У сільському господарстві системи контролю метеоданих використовуються для оптимізації сільськогосподарських процесів, таких як полив, внесення добрив, врожаїв та планування сівозмін. Це допомагає фермерам підвищувати врожайність та ефективність виробництва.

У транспорті системи контролю метеоданих використовуються для безпечної експлуатації транспортних засобів, планування маршрутів та уникнення небезпечних погодних умов. Це дозволяє зменшити ризики дорожніх аварій та затримок у транспортному русі.

У сфері енергетики системи контролю метеоданих використовуються для оптимізації виробництва електроенергії з відновлювальних джерел, таких як сонячна та вітрова енергія. Вони допомагають передбачати погодні умови та планувати роботу електростанцій з урахуванням прогнозованих змін у погоді.

1.2.1. Використання систем контролю метеоданих в сільському господарстві

Використання систем контролю метеоданих в сільському господарстві має велике значення для всіх аспектів сільськогосподарської діяльності. Особливо в сучасних умовах, коли зміни клімату стають більш непередбачуваними, точна інформація про погоду є важливою для фермерів.

Системи контролю метеоданих допомагають у плануванні робіт на полях. Знання погодних умов дозволяє фермерам оптимізувати час висівання насіння, поливу, захист від шкідників та хвороб, а також збирання врожаю. Вони можуть реагувати на прогнозовані погодні явища, такі як дощі, засухи, або морози, змінюючи плани та заходи.

Збирання та аналіз метеоданих допомагає визначити оптимальні умови для росту рослин. Враховуючи температуру, вологість, інтенсивність сонячного світла та інші показники, фермери можуть підтримувати найкращі умови для здоров'я та росту своїх культур.

Контроль метеоданих допомагає в управлінні ресурсами, такими як вода та добрива. Знання про погоду дозволяє ефективно використовувати ці ресурси, уникати їхнього пере- або недовикористання.

Використання метеоданих допомагає впоратися зі змінами клімату та мінімізувати ризики для сільськогосподарських підприємств. Вони дозволяють адаптуватися до нових умов та збільшувати стійкість сільського господарства до зовнішніх впливів.

1.2.2. Використання систем контролю метеоданих в транспортній сфері

Використання систем контролю метеоданих у транспортній сфері є ключовим для забезпечення безпеки та оптимальності переміщень транспортних засобів.

Ці системи надають можливість передбачати погодні умови, що дозволяє транспортним компаніям адаптувати маршрути та графіки перевезень згідно з очікуваними умовами. Наприклад, прогноз погоди може вплинути на рішення про зміну маршруту або затримки руху, що забезпечує безпеку пасажирів та вантажів.

Також ці системи допомагають в оптимізації використання транспортних засобів. Вони забезпечують інформацію про дорожні умови, такі як стан доріг та обмеження видимості, що допомагає водіям та операторам приймати обґрунтовані рішення про безпеку та швидкість руху.

Системи контролю метеоданих сприяють плануванню обслуговування транспортних засобів, надаючи інформацію про умови для технічного обслуговування та ремонту, а також прогнозуючи можливі затримки через погодні умови.

Використання систем контролю метеоданих у транспортній сфері сприяє підвищенню безпеки та ефективності руху транспортних засобів, зменшенню ризиків та витрат, а також покращенню задоволення користувачів послуг транспорту.

1.2.3. Використання систем контролю метеоданих в енергетичній сфері

В енергетичній сфері використання систем контролю метеоданих є ключовим аспектом для ефективного управління енергетичними ресурсами.

Такі системи дозволяють енергетичним компаніям передбачати споживання електроенергії, враховуючи погодні умови. Наприклад, вони можуть передбачити пікове споживання під час теплих або холодних періодів, допомагаючи підготувати виробництво енергії для забезпечення пікового попиту.

Зокрема, такі системи сприяють оптимізації використання альтернативних джерел енергії, таких як сонячні батареї та вітрові турбіни. Шляхом аналізу погодних умов вони дозволяють передбачити виробництво електроенергії з цих джерел, що сприяє ефективному використанню відновлювальних джерел енергії.

Системи контролю метеоданих відіграють важливу роль у запобіганні аварій та надзвичайних ситуацій, пов'язаних зі зміною погоди. Інформація, яку вони надають про погодні умови, дозволяє приймати вчасні заходи для зменшення ризиків та підвищення надійності енергопостачання.

Яскравим прикладом є Україна, постійні ракетні атаки, щоразу приносять проблеми енергетичній сфері України. Через це активно збільшує роль сонячних електростанцій. Завдяки простоті встановлення та обслуговування вони є прекрасним аналогом діючим ТЕС та ГЕС, які підчас ракетних атак є однією з головних цілей ворога. Завдяки аналізу метрологічних даних можна легко з’ясувати в яких регіонах краще встановлювати СЕС для максимальної ефективності їх роботи та забезпечити максимальну генерацію енергії.

Використання систем контролю метеоданих у енергетичній сфері допомагає забезпечити стабільне постачання електроенергії, підвищити ефективність використання енергії та зменшити негативний вплив на довкілля.

1.3. Методи вимірювання метеоданих

Вимірювання метеорологічних даних-це процес збору та аналізу різних атмосферних параметрів, таких як температура, вологість, тиск, швидкість

вітру та кількість опадів. Ці дані використовуються при плануванні та управлінні різними галузями промисловості, такими як прогнозування погоди, наукові дослідження, Сільське господарство, транспорт та будівництво.

Існує багато різних способів вимірювання даних про погоду, але це може змінюватися залежно від типу даних, які ви збираєте, а також умов та цілей конкретного вимірювання. Деякі з найбільш поширених методів включають::

Метеорологічні станції: це прилади, які розміщені в різних місцях для вимірювання погодних умов. Вони можуть включати датчики температури, вологості, тиску, опадів, вітру та інших параметрів, які надають дані про погоду.

Радіозон: цей метод полягає у запуску метеорологічного радіозонду, який передає дані про температуру, вологість та інші параметри у верхні шари атмосфери. Інформація радіозондів використовується для складання вертикальних профілів атмосферних умов.

Супутникове спостереження: супутникові системи надають інформацію про погодні умови на великій території. Вони вимірюють температуру поверхні Землі, розподіл хмар, концентрацію атмосферних газів та інші параметри.

Вимірювання літаками та безпілотниками: цей метод використовується для отримання точних даних у важкодоступних місцях.

Наземні метеорологічні мережі: це мережі станцій, розташованих на Землі, які використовуються для збору даних про погоду для певного регіону. Ці дані можуть включати не тільки автоматичні вимірювальні прилади, але і спостереження людини.

1.3.1. Методи вимірювання температури

Вимірювання температури за допомогою термопара

Термопара — чутливий елемент термоелектричного перетворювача у вигляді двох ізольованих провідників із різнорідних матеріалів, з'єднаних на

одному кінці, принцип дії якого ґрунтується на використанні термоелектричного ефекту для вимірювання температури[12]. Звичайна термопара це контакт двох різних спаяних металів(термоелектродів). У якості термоелектродів використовують як і чисті метали – платина, залізо, мідь так і сплави – алюмель, константан, копель, платинорідій, хромель. В основу дії термопари покладено ефект Зеєбека. Тобто при нагріванні контакту утворюється ЕРС який пропорційний різниці температур. Діапазон вимірювань становить -50-120

Вимірювання температури за допомогою терморезисторів

Терморезистор – це нелінійний резистор, виготовлений із напівпровідникового матеріалу, опір якого залежить від температури[13].Однією з характеристик терморезисторів є те, що їхній температурний коефіцієнт опору (ТСО) дуже великий, набагато вищий, ніж у металів. Перевагами терморезисторів є простота використання, здатність працювати в суворих кліматичних умовах і характерна стабільність у часі. До основних параметрів термістора відносяться:

- номінальний опір;
- температурний коефіцієнт опору;
- висока чутливість;
- діапазон робочих температур;
- максимально допустима розсіяна потужність;
- низька вартість;
- маленький розмір.

Всі терморезистори можна розділити на дві групи: терморезистори з негативним ТКО – термістори, з позитивним ТКО – позистори.

Значення ТКО для різних термісторів при кімнатній температурі знаходиться в межах 0,8-6% на градус.

Для позисторів це значення може досягати і десятків відсотків на один градус.

Термістори можна з легко використовувати для вимірювання температури навколишнього середовища.

1.3.2. Методи вимірювання атмосферного тиску

Вимірювання атмосферного тиску за допомогою ємнісного датчика тиску

Ємнісні давачі тиску використовують кремнієві діафрагми. Під дією тиску діафрагма зміщується відносно опорної пластини, тим самим змінюючи пропускну здатність між ними. Ці датчики тиску найкраще працюють при низькому тиску. Монолітний ємнісний давач тиску виготовлений з кристалу кремнію, має досить високу стабільність робочих характеристик. Рух діафрагми може забезпечити 25% зміни ємності, що дозволяє безпосередньо оцифрувати отриману величину тиску.

Вимірювання атмосферного тиску за допомогою тензорезистивного датчика тиску. У цих датчиках використовується два елемента: пластина заданої площі(діафрагма), а також детектор(тензорезистори), який генерує сигнал пропорційний тиску.

Датчик тиску з кремнієвою діафрагмою складається з діафрагми та тензорезистором вбудованого в неї шляхом дифузії. Оскільки монокристалічний кремній має досить хороші властивості пружності, то такий датчик не має інертності та гістерезису при високих тисках.

Максимальна вихідна напруга цих даних датчиків дуже мала і становить кілька сотень мілівольт, тому на його виходах розміщують підсилювач сигналу. Головним недоліком цього датчика є висока температурна чутливість, тому при проєктуванні пристроїв на їх основі необхідно створювати коло температурної залежності.

1.3.3. Методи вимірювання яскравості освітлення

Вимірювання яскравості освітлення є важливим кроком у процесі проектування та оцінки ефективності освітлення в будь-якій кімнаті або на відкритому повітрі. Це процес, який дозволяє вимірювати рівень освітленості, який визначається певною площею або площею поверхні. Вимірювання яскравості є ключем до створення комфортних умов для роботи, відпочинку, досліджень та інших видів діяльності, що вимагають освітлення.

Існує кілька методів вимірювання яскравості освітлення, вони використовуються відповідно до конкретних вимог та умов вимірювання:

Вимірювання за допомогою люксметра (люксметра): цей метод вимірює освітлення, створюване джерелом світла на певній поверхні. Люксметр вимірює кількість світла, що падає на один квадратний метр поверхні, і виражає результат як Lux. Блок Lux відповідає яскравості 1 свічки, розташованої на відстані 1 метра від поверхні.

Фотометричне вимірювання (фотометричне): фотометри вимірюють інтенсивність світла і збуджуються в певному напрямку. Використовуючи цей метод, ви можете оцінити, наскільки яскраво освітлена певна частина кімнати або приміщення.

Спектрофотометрія: цей метод дозволяє аналізувати спектральний склад світла, тобто його розподіл по довжинах хвиль. Це важливо, оскільки різні кольори можуть по-різному впливати на ефективність освітлення та візуальний комфорт. Спектрофотометр дозволяє визначити конкретні кольори, присутні в джерелі світла, і те, як вони впливають на зір.

Інтегральні методи вимірювання: ці методи поєднують різні аспекти вимірювання, такі як освітлення, розподіл світла в просторі та спектральний склад. Це дозволяє більш ретельно оцінити якість освітлення.

Незалежно від обраного методу вимірювання, при оцінці яскравості освітлення важливо враховувати конкретні умови і деталі потреб користувача.

Вимірювання яскравості є важливим інструментом для забезпечення комфортних і безпечних умов в різних умовах.

1.4. Розробка функціональної схеми

Розробка функціональних схем є важливим етапом у процесі створення інформаційної системи. Схема визначає всі необхідні функції, необхідні для забезпечення роботи системи. Ось чому це важливо:

Особливості: функціональні схеми допомагають точно визначити, які системні функції потрібно реалізувати. Це важливо для розуміння того, як працює система та які функції вона пропонує користувачам.

Оптимізація: розробка функціональних схем дозволяє виявити і виправити можливі недоліки у функціонуванні системи на ранній стадії розробки. Це забезпечує оптимальну роботу системи і може задовольнити потреби користувачів.

Вимоги: за допомогою розробки функціональних схем можна уточнити вимоги до системи, з'ясувати, які саме функції повинна виконувати система, і визначити, які параметри метеорологічних даних необхідно виміряти.

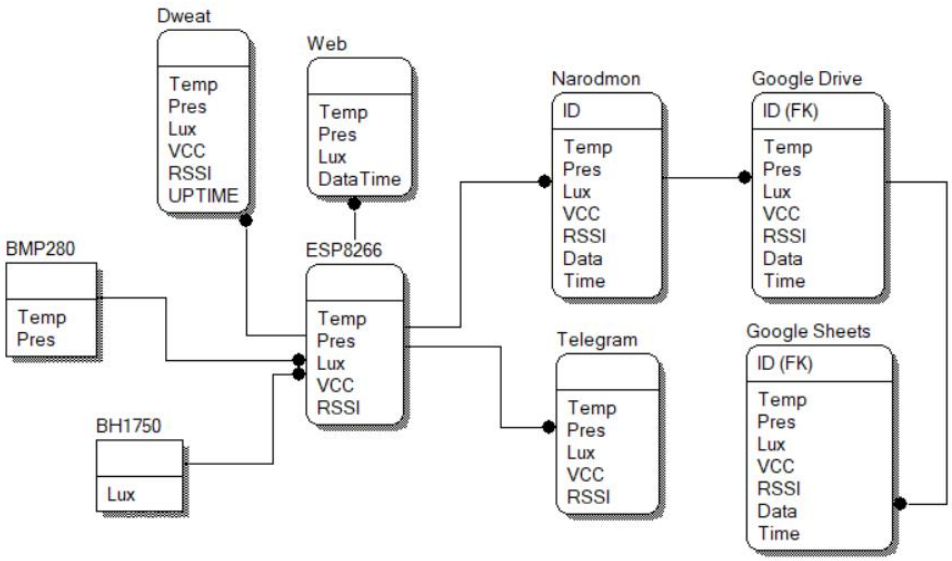

Рисунок 1.1 – Функціональна схема інформаційної системи

Функціональна схема розроблюваної інформаційної системи повинна містити наступні елементи:

- Вхідні дані: параметри що вимірюються датчиками та мікроконтролером.
- Обробка інформації: поширення даних на загальнодоступні сервіси; обробка запитів з Telegram; поширення даних на веб – сервер.
- Вихідні дані: візуальне відображення даних на веб – сервері та загальнодоступних сервісах; відповіді на запити з Telegram; збереження даних в Google Drive для подальшої роботи в Google Sheets.

1.5. Обґрунтування доцільності проектування та створення власної системи

Проєктування та розробка системи обробки метеоданих портативної метеостанції може бути доцільним рішенням з кількох причин:

Зручність: Наявність власної системи обробки даних дає можливість постійно відслідковувати метеодані рівень яких можна корегувати в залежності від того що потрібно користувачеві.

Мобільність: портативна метеостанція дозволяє збирати дані про погоду в будь-якому місці та у будь-який час. Це може стати корисним для наукових досліджень, дослідження клімату певної зони та мікроклімату приміщення.

Варіативність: Розробка власної системи дає можливість змінювати та вдосконалювати в залежності від задач які потрібні користувачу. Це може бути встановлення додаткових датчиків або зміна швидкості передачі даних.

Доступність: Систему можна так налаштувати щоб користувач мав змогу отримати дані в будь-якій точці світу, головною умовою має бути лише наявність Інтернету в системи та користувача.

1.6. Висновок до розділу 1

В даному розділі був проведений аналіз впливу метеоданих на різні сфери життєдіяльності людини.Було визначено що контроль метеоданих є чи не одним з головним фактором в ефективній роботі сфери сільськогосподарства, енергетики та транспрорту.

В результаті аналізу методу вимірювання метеорологічних даних важливість та надійність цих методів стали важливим фактором для забезпечення якості та надійності отриманих метеорологічних даних. Вимірювання параметрів погоди, таких як температура та тиск, є основою для розуміння зміни клімату та прогнозування погодних явищ.

За допомогою функціональної схеми було здійснено ретельний аналіз процесу збору, обробки та відображення метеоданих. Вхідні дані,отримані від датчиків і мікроконтролера,є основою для подальшої обробки інформації.

Обробка інформації включає кілька важливих етапів, таких як розповсюдження даних на загальнодоступні сервіси, обробка запитів від Telegram та відображення даних на веб-серверах.

Вихідні дані відображаються у вигляді візуального відображення на веб-сервері і в загальнодоступних сервісах, що полегшує доступ до них. Також, система надає можливість відповідати на запити від Telegram, що є додатковим зручним каналом зв'язку. Іншим важливим фактором є зберігання даних на Google Диску для подальшого використання в Google Таблицях, що дозволяє проводити подальший аналіз та обробку даних.

РОЗДІЛ 2. ТЕХНІЧНЕ ЗАВДАННЯ

2.1.Призначення і склад системи

Портативна метеостанція призначення для вимірювання температури та тиску навколишнього середовища та яскравості освітлення.

Система повинна складатися з 3 вимірювальних каналів. Система може працювати в двох режимах:

- портативний режим роботи;
- стендовий режим роботи;

В портативному режимі роботи система повинна передавати вимірювальну інформацію через I2C інтерфейс на мікроконтролер ESP8266, який в свою чергу буде передавати данні через WIFI на різноманітні ресурси які будуть підключенні до системи.

В стендовому режимі роботи результати вимірювань повинні передаватися через USB на ПК для подальшої обробки та аналізу.

2.2. Галузі застосування

Портативна метеостанція призначення для вимірювання температури та тиску навколишнього середовища та яскравості освітлення. Застосовується для власного використання, також можливе комерційне застосування

2.3. Цілі створення системи

- Зручність: Система дозволяє відслідковувати метеодані в зручній для користувача формі.
- Ефективність: Система дає змогу отримувати дані в режимі реального часу
- Доступність: Система надає можливість будь-якому користувачеві переглянути метеодані в місці встановлення станції за допомогою підключених до неї сервісів.
- Варіативність: Система має можливість підключення інших засобів збору інформації

2.4.Вимоги до системи

- Режим перегляду даних в режимі реального часу
- Можливість віддаленого перегляду даних
- Підключення інших користувачів
- Портативність

2.4.1 Технічні та метрологічні вимоги

Розроблювана система обробки метеоданих портативної метеостанції на базі ESP8266 має складатися з 3 каналів вимірювання метеоданих , блоку обробки вимірювальної інформації і забезпечувати відповідні метрологічні характеристики

Таблиця 2.1 Метрологічні характеристики

№	Найменування параметру	Позначення	Початкове – кінцеве значення	Абсолютна похибка вимірювань
1	Температура	T	-40 - +50	±1
2	Тиск	P	500 - 1000	±1
3	Яскравість освітлення	L	0 – 10000	±1

Система повинна виконувати автоматичне вимірювання метеоданих та передавати вимірянні данні через USB або WIFI інтерфейс

2.4.2 Вимоги до електроживлення

Живлення повинно здійснюватися від джерела постійної напруги +5В або 3.3В

2.4.3 Вимоги стійкості до зовнішніх впливів

Система повинна зберігати працездатність за умов зовнішнього середовища, які відповідають санітарним нормам мікроклімату виробничих приміщень ДСН 3.3.6.042-99

2.5. Вимоги до документації

На систему розробляється комплекс документації у складі: технічне завдання та технічний проєкт

Документація на систему розробляється у відповідності з вимогами Державних стандартів серії 19«Єдина система програмної документації».

6. Джерела розробки

При розроблені технічного завдання на системо використано наступні документи:

- ДСТУ 3008-2015. Документація. Звіти у сфері науки і техніки. Структура та правила оформлювання[14].
- ДСТУ Б В.2.5-82:2016 Електробезпека в будівлях і спорудах. Вимоги до захисних заходів від ураження електричним струмом[15].
- ДСТУ EN 60204-1:2015 Безпечність машин. Електрообладнання машин. Частина 1. Загальні вимоги (EN 60204-1:2006; A1:2009; AC:2010, IDT). Зі зміною [9].
- ДСТУ ISO 6309:2007. Пожежна безпека. Загальні вимоги [10].

РОЗДІЛ 3. ОПИС КОМПЛЕКСУ ЗАДАЧ РОЗРОБКИ СИСТЕМИ

3.1. Алгоритмізація та реалізація комплексу задач системи

3.1.1. Підключення бібліотек необхідних для реалізації проєкту

Середовищем розробки було вибрано Arduino IDE[1]. Arduino IDE – це середовище програмування, спеціально призначене для розробки програмного забезпечення для мікроконтролерів. Вона володіє простим та зрозумілим інтерфейсом, який включає все необхідне для написання коду, завантаження коду та взаємодію з мікроконтролером. Arduino IDE має вбудовані засоби для редагування та компіляції коду та відладки. Вона має велику спільноту користувачів, що дає можливість обміну знань та розвитку проєктів. Це середовище чудово підходить як для початківців, так і досвідчених розробників.

Першим етапом є підключення всіх необхідних бібліотек для реалізації системи. Частина бібліотек є влаштованими в середовище розробки Arduino IDE, іншу частину потрібно було встановити самостійно за допомогою влаштованих функцій

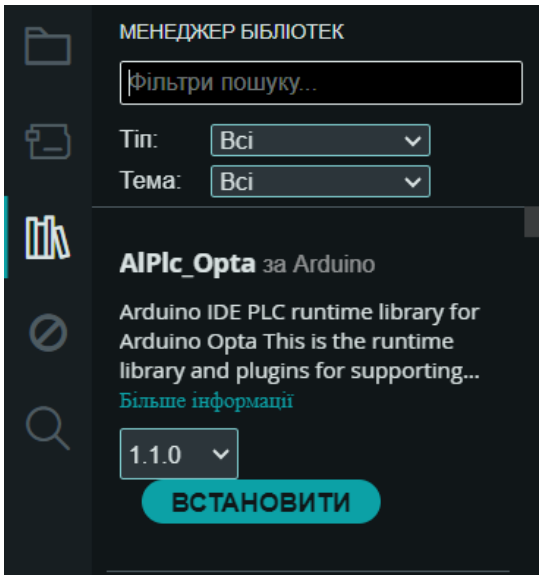

Рисунок 3.1 – Влаштований менеджер бібліотек в середовище Arduino IDE

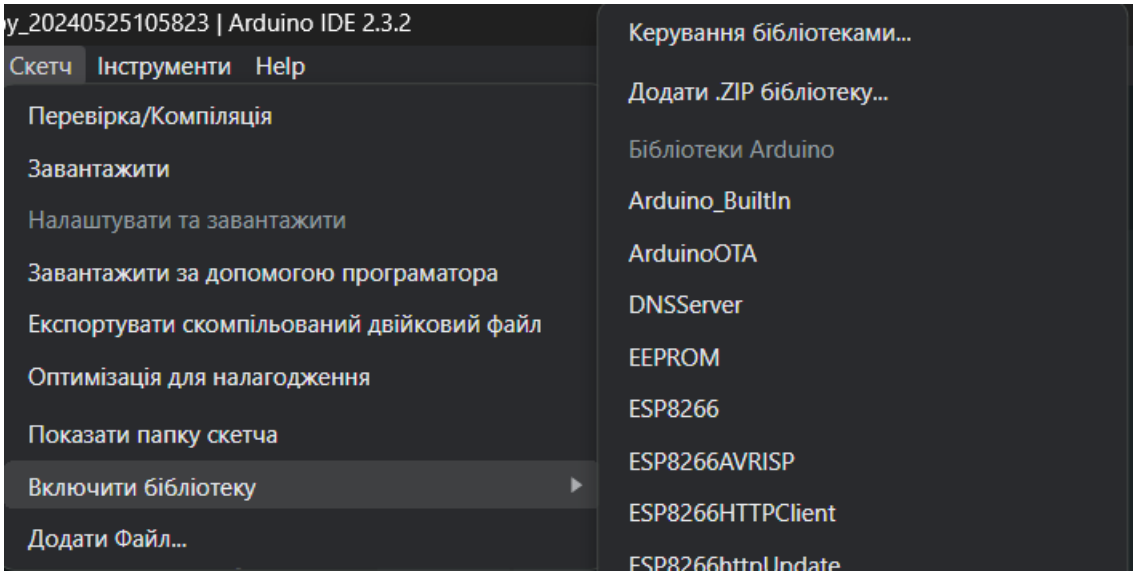

Рисунок 3.2 – Додавання сторонніх бібліотек за допомогою влаштованої функції “ Додати .ZIP бібліотеку… ”

```
#include <ESP8266WiFi.h>
#include <Wire.h>
#include "WiFiClient.h"
#include "ESP8266WebServer.h"
#include "ESP8266HTTPClient.h"
#include "ESP8266mDNS.h"
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP280.h>
#include <BH1750.h>
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>
#include <ArduinoJson.h>
```

Рисунок 3.3 – Зображення під’єднаних бібліотек для реалізації системи

3.1.2. Оголошення констант та змінних для реалізації системи

Оголошення констант та змінних є одним із головних етапів реалізації, саме завдяки правильним оголошенням система легко читається, правильно функціонує та зменшується ймовірність проблем в подальшому.

```
ADC_MODE(ADC_VCC);
Adafruit_BMP280 bmp; // I2C
BH1750 lightMeter;
WiFiClient client;
ESP8266WebServer server(80);

#define host "eu.narodmon.com"
#define httpPort 8283
#define dweet "dweet.io"
#define dweetPort 80
#define interval 5000
#define BOT_TOKEN "6561845762:AAFul7SqNUqrJwoxJeY8SibZ9rGgM3PZ4is"

X509List cert(TELEGRAM_CERTIFICATE_ROOT);
WiFiClientSecure secured_client;
UniversalTelegramBot bot(BOT_TOKEN, secured_client);

float vcc, T, P, L;
int rssi;
unsigned long lasttime;
unsigned long lastNarodmonSendTime = 0;
const unsigned long narodmonSendInterval = 5 * 60 * 1000; // 5 хвилин у мілісекундах
```

Рисунок 3.4 – Зображення оголошених констант та змінних для реалізації системи

ADC_MODE(ADC_VVC) – це фунція в мікроконтролері ESP8266 яка встановлює режим рвимірювання напруги за допомогою аналого – цифрового конвектора (ADC).

Adafruit_BMP280 bmp – це ініціалізація датчика BMP280 з назвою bmp для з’єднання через шину I2C. I2C – двонаправлена послідовна шина зв’язку, яка використовується для сполучення мікроконтролерів, сенсорів та переферії.Вона використовує дві лінії дл комунікації: лінію даних(SDA) та лінію годинника(SCL).

BH1750 lightMeter – це ініціалізація датчика BH1750 з назвою lightMeter для з’єднання через шину I2C.

WiFiClient client – це ініцалізація об’єкта WiFi для взаємодії з WiFi мережею.

ESP8266WebServer server(80) – це ініціалізація об’єкта веб-сервера на порту 80.

#define host “eu.narodmon.com” – це деректива визначає символьну константу host зі значення“eu.narodmon.com”.

#define httpPort 8283 – це деректива визначає символьну константу httpPort зі значення 8283.

#define dweet “dweet.io” – це деректива визначає символьну константу dweet зі значення “dweet.io”.

#define dweetPort 80 – це деректива визначає символьну константу dweetPort зі значення 80.

#define interval 5000 – це деректива визначає символьну константу interval зі значення 5000.

#define BOT_TOKEN 6561845762:AAFul7SqNUqrJwoxJeY8SibZ9rGgM3PZ4is – це деректива визначає символьну константу BOT_TOKEN зі значення 6561845762:AAFul7SqNUqrJwoxJeY8SibZ9rGgM3PZ4is

X509List cert(TELEGRAM_CERTIFICATE_ROOT) – це ініціалізація класу данних для представлення списків сертифікатів. Сертифікати використовуяться для аунтифікації сторін у мережевому зв’язку.

WiFiClientSecure secured_client – це ініцалізація класу данних , що надає можливісіть встановлення захищеного з’єднання через протоколи SSL/TLS.

UniversalTelegramBot bot(BOT_TOKEN, secured_client) – це ініціалізація об’єкта UniversalTelegramBot з назвою bot для взаємодії з Telegram Bot API за допомогою захищеного каналу зв’язку який був оголошений раніше.\

Float vcc, T, P, L – це оголошення змінних типу Float з іменами vcc, T, P, L.

int rssi – це оголошення змінної типу int з іменем rssi.

Unsigned long lasttime – це оголошення змінної Unsigned long з іменем lasttime для зберігання значення часу в мікросекундах або мілісекундах.

Unsigned long lastNarodmonSendTime = 0 – це оголошення змінної Unsigned long з іменем long lastNarodmonSendTime зі значенням 0 для зберігання значення часу в мікросекундах або мілісекундах

Const Unsigned long lastNarodmonSendTime = 5 * 60 * 1000 – це оголошення константи для встановлення інтервалу часу між відправленнями даних на сервер Narodmon.

3.1.3. Створення веб-сторінки та створення запитів для коректного відображення інформації на ній

Для коректного відображення даних на веб-сторінці потрібно написати правильні запити та згенерувати приємну стилізацію самої сторінки.

```
String generateWebpage() {
  String webpage = "<!DOCTYPE html>";
  webpage += "<html>";
  webpage += "<head>";
  webpage += "<link rel=\"stylesheet\" href=\"https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.4/css/all.min.css\">";
  webpage += "<style>";
  webpage += "body {";
  webpage += "  font-family: Arial, Helvetica, sans-serif;";
  webpage += "  background-color: #f2f2f2;";
  webpage += "}";
  webpage += ".card {";
  webpage += "  background-color: white;";
  webpage += "  border-radius: 10px;";
  webpage += "  box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);";
  webpage += "  transition: 0.3s;";
  webpage += "  width: 300px;";
  webpage += "  margin: auto;";
  webpage += "  padding: 20px;";
  webpage += "}";
  webpage += ".container {";
  webpage += "  text-align: left;";
  webpage += "}";
  webpage += ".icon {";
  webpage += "  float: left;";
  webpage += "  margin-right: 10px;";
  webpage += "}";
  webpage += "</style>";
  webpage += "<script>";
  webpage += "function updateTime() {";
  webpage += "  var now = new Date();";
  webpage += "  var dateElement = document.getElementById('date-time');";
  webpage += "  dateElement.innerHTML = now.toLocaleString();";
```

Рисунок 3.5 – Частина коду генерації веб – сторінки

```
  webpage += "}";
  webpage += "setInterval(updateTime, 1000);"; // Оновлюємо кожну секунду
  webpage += "function fetchData() {";
  webpage += "  fetch('/data')";
  webpage += "    .then(response => response.json())";
  webpage += "    .then(data => {";
  webpage += "      document.getElementById('temp').innerText = data.temperature.toFixed(2) + ' °C';";
  webpage += "      document.getElementById('pres').innerText = data.pressure.toFixed(2) + ' mmHg';";
  webpage += "      document.getElementById('lux').innerText = data.light.toFixed(2) + ' lx';";
  webpage += "    });";
  webpage += "}";
  webpage += "setInterval(fetchData, 1000);"; // Оновлюємо кожну секунду
  webpage += "</script>";
  webpage += "</head>";
  webpage += "<body onload=\"updateTime(); fetchData();\">";
  webpage += "<div class=\"card\">";
  webpage += "  <div class=\"container\">";
  webpage += "    <h4><b>Temperature</b></h4>";
  webpage += "    <div class=\"icon\"><i class=\"fas fa-thermometer-half\"></i></div>";
  webpage += "    <span id=\"temp\">Loading...</span><br>";
  webpage += "    <h4><b>Pressure</b></h4>";
  webpage += "    <div class=\"icon\"><i class=\"fas fa-tachometer-alt\"></i></div>";
  webpage += "    <span id=\"pres\">Loading...</span><br>";
  webpage += "    <h4><b>Light Intensity</b></h4>";
  webpage += "    <div class=\"icon\"><i class=\"fas fa-lightbulb\"></i></div>";
  webpage += "    <span id=\"lux\">Loading...</span><br>";
  webpage += "    <h4><b>Date and Time</b></h4>";
  webpage += "    <div class=\"icon\"><i class=\"far fa-calendar-alt\"></i></div>";
  webpage += "    <span id=\"date-time\"></span>";
  webpage += "  </div>";
  webpage += "</div>";
  webpage += "</body>";
  webpage += "</html>";
```

Рисунок 3.6 – Друга частина коду генерації веб – сторінки

```
  webpage += "</body>";
  webpage += "</html>";
  return webpage;
}

void handleData() {
  StaticJsonDocument<200> data;

  data["temperature"] = T;
  data["pressure"] = P;
  data["light"] = L;

  String jsonData;
  serializeJson(data, jsonData);

  server.send_P(200, "application/json", jsonData.c_str());
}

void handleWebpage() {
  server.send(200, "text/html", generateWebpage());
}
```

Рисунок 3.7 – Зображення функцій для обробки запитів на дані та генерацію HTML – сторінки.

Функція generateWebpage() генерує HTML – сторінку із стилізацією CSS та скриптами JavaScript[11]. На сторінці містяться контейнери для відображення метрологічних даних та поточного дата/часу.

Функція handleData() обробляє запити на дані. Вона створює об’єкти з мтерологічними даними , а потім відправляє ці дані у відповідь на запит

Функція handleWebpage() відповідає на запити на головну сторінку веб – сервера, надсилаючи згенеровану HTML – сторінку.

3.1.4. Підключення до WiFi мережі

Підключення до WiFi мережі є головним аспектом в поширенні даних які збирає метеостанція[8]. Для забезпечення коректної роботи підключення до мережі було розділено на 3 функції:

- Підключення до попередньо заданої мережі
- Підключення до мережі які відсканувала ESP8266
- Перевірка підключення до мережі

3.1.4.1. Підключення до попередньо заданої мережі

Була написана функція яка передбачає підключенння до попердньо заданої мережі. Ця функція була реалізована для того щоб у випадку коли користува немає змоги напряму під’єднатися до метеостанції для введення данних своєї мережі, станція мала змогу під’єднатися до Інтернету для коректної роботи

```
121
122 void connectToPreferredWiFi() {
123   // Введіть дані вашої перевіреної мережі
124   const char* preferredSSID = "ESP8266";
125   const char* preferredPassword = "12345678";
126
127   Serial.println("Спроба підключення до перевіреної мережі Wi-Fi...");
128
129   WiFi.begin(preferredSSID, preferredPassword);
130
131   unsigned long startAttemptTime = millis(); // Початок спроби підключення
132   while (WiFi.status() != WL_CONNECTED) {
133     delay(500);
134     Serial.print(".");
135     if (millis() - startAttemptTime > 10000) { // Якщо пройшло більше 10 секунд, вважаємо це невдалою спробою
136       Serial.println("Не вдалося підключитися до перевіреної мережі Wi-Fi.");
137       return; // Виходимо з функції, оскільки не вдалося підключитися до переданої мережі
138     }
139   }
140
141   // Якщо підключено успішно, виводимо IP-адресу та продовжуємо виконання програми
142   Serial.println("");
143   Serial.println("Успішно підключено до перевіреної мережі Wi-Fi!");
144   Serial.println("Ваш IP-адреса " + WiFi.localIP().toString());
145 }
146
```

Рисунок 3.8 – Зображення реалізації функції підключення до заданої мережі

В функції задані дані про мережу до якої підключитися. Якщо підключення відбулося функція виводить в консоль відповідні рядки коду. У випадку якщо підключення не відбулося протягом 10 секунд, система виходить з функції щоб продовжити пілючення в наступній функції

3.1.4.2. Підключення до мережі які відсканувала ESP8266

Якщо не відбулося підключення до попердньо заданої мережі, система переходить до наступної функції підключення до мережі.

```
146
147  void connectWiFi() {
148    // Спочатку спробуємо підключитися до заданої мережі
149    connectToPreferredWiFi();
150
151    // Якщо не вдалося підключитися до заданої мережі, виконаємо процес сканування та підключення до іншої мережі
152    if (WiFi.status() != WL_CONNECTED) {
153      Serial.println("Не вдалося підключитися до перевіреної мережі Wi-Fi. Запускаємо процес сканування мереж...");
154      while (true) {
155        boolean wifiFound = false;
156        int i, n;
157
158        WiFi.mode(WIFI_STA);
159        WiFi.disconnect();
160        delay(100);
161        Serial.println("Проведення сканування мереж Wi-Fi...");
162        int nbVisibleNetworks = WiFi.scanNetworks();
163        Serial.println(F("Сканування завершено!"));
164        if (nbVisibleNetworks == 0) {
165          Serial.println("Мережі Wi-Fi не знайдено!");
166          continue;
167        }
168        Serial.print(nbVisibleNetworks);
169        Serial.println(" мереж знайдено");
170
171        // Виводимо список доступних мереж
172        for (i = 0; i < nbVisibleNetworks; ++i) {
173          Serial.println(WiFi.SSID(i));
174        }
175
```

Рисунок 3.9 – Зображення першої частини функції підключення до мережі

```
175
176        // Введення імені мережі та пароля через Serial Monitor
177        Serial.println("Введіть SSID мережі, до якої ви хочете підключитися:");
178        while (!Serial.available()) {} // Очікування вводу
179        String ssidInput = Serial.readStringUntil('\n');
180        Serial.println("Введіть пароль:");
181        while (!Serial.available()) {} // Очікування вводу
182        String passwordInput = Serial.readStringUntil('\n');
183
184        // Перевірка, чи знайдено введену мережу в списку доступних
185        for (i = 0; i < nbVisibleNetworks; ++i) {
186          if (WiFi.SSID(i) == ssidInput) {
187            wifiFound = true;
188            break;
189          }
190        }
191        if (!wifiFound) {
192          Serial.println("Введена мережа не знайдена!");
193          continue;
194        }
195
196        // Підключення до введеної мережі
197        Serial.print("Підключення до ");
198        Serial.println(ssidInput);
199        WiFi.begin(ssidInput.c_str(), passwordInput.c_str());
200        unsigned long startAttemptTime = millis(); // Початок спроби підключення
201        while (WiFi.status() != WL_CONNECTED) {
202          delay(500);
203          Serial.print(".");
204          if (millis() - startAttemptTime > 10000) { // Якщо пройшло більше 10 секунд, вважаємо це невдалою спробою
205            Serial.println("Час очікування підключення вичерпано!");
206            break; // Виходимо з циклу
207          }
```

Рисунок 3.10 – Зображення другої частини функції підключення до мережі

```
195
196       // Підключення до введеної мережі
197       Serial.print("Підключення до ");
198       Serial.println(ssidInput);
199       WiFi.begin(ssidInput.c_str(), passwordInput.c_str());
200       unsigned long startAttemptTime = millis(); // Початок спроби підключення
201       while (WiFi.status() != WL_CONNECTED) {
202         delay(500);
203         Serial.print(".");
204         if (millis() - startAttemptTime > 10000) { // Якщо пройшло більше 10 секунд, вважаємо це невдалою спробою
205           Serial.println("Час очікування підключення вичерпано!");
206           break; // Виходимо з циклу
207         }
208       }
209       if (WiFi.status() == WL_CONNECTED) { // Якщо підключено успішно, виходимо з циклу
210         Serial.println("");
211         Serial.println("Успішно підключено до мережі Wi-Fi!");
212         Serial.println("Ваш IP-адреса " + WiFi.localIP().toString());
213         break;
214       } else { // Якщо підключення не вдалося
215         Serial.println("Помилка підключення, повторна спроба...");
216       }
217     }
218   }
219 }
220
```

Рисунок 3.11 – Зображення третьої частини функції підключення до мережі

В функції реалізовано сканування мережі, після чого в консоль видається список наявних мереж. Потрібно ввести назву необхідної мережі, після чого ввести пароль до цієї мережі. У випадку якщо відбудеться помилка під час підключення, система напише про це в консоль та цикл підключення почнеться знову.

3.1.4.3. Перевірка підключення до мережі

Може виникнути ситуація коли мереже відімкнеться і система почне працювати не коректно, дані перестануть передаватися. Саме для цього була реалізована функція перевірки підключення

```
void checkWiFiConnection() {
  if (WiFi.status() != WL_CONNECTED) {
    Serial.println("Втрачено з'єднання з Wi-Fi. Повторна спроба підключення...");
    connectWiFi(); // Повторне підключення до Wi-Fi
  }
}
```

Рисунок 3.12 – Зображення реалізації функції перевірки підключення

У випадку коли підключення зникло, система це бачить і починає цикл підключення до мережі задля забезпечення коректної роботи

3.1.5. Підключення до Telegram Bot

Підключення до Telegram Bot є одним із методі поширення метеоданих. В боті який був створений попердньо були налаштовані команди для отримання данних з системи. Тому можна умовно розділити на два етапи:

- підключення до Telegram API;
- обробка отриманої інформації з бота.

3.1.5.1. Підключення до Telegram API

Telegram – це один з найпопулярніших месенджерів, який дозволяє користувачам обмінюватися повідомленнями, аудіо та відеофайлами. Головною перевагою є механізм шифрування який забезпечує конфіденційність даних користувачів. Завдяки відкритому API, розробники різноманітних додатків та сервісів, мають можливість інтеграції з месенджером[2].

```
227
228  void setupTelegram() {
229    Serial.println("Підключення Telegram bot...");
230    secured_client.setTrustAnchors(&cert); //  Додати кореневий сертифікат для api.telegram.org
231
232    configTime(0, 0, "pool.ntp.org"); // Отримати час UTC через NTP
233    time_t now = time(nullptr);
234    while (now < 24 * 3600) {
235      delay(100);
236      now = time(nullptr);
237    }
238
239    Serial.println("Telegram bot підключено!");
240
241  }
242
```

Рисунок 3.13 – Зображення реалізації функції підключення до Telegram API

Завдяки попередньо заданим константам та ініційованих об’єктів функція не займає багато місця і має читабельний вигляд. В консоль виводять повідомлення про початок і кінець підключення. Підключення відбувається по захищеному каналу, тому вразливість з’єднання мінімальна.

3.1.5.2. Обробка інформації отриманої з бота

Попередньо було створено Telegram Bot який включає в собі основні команди для отримання метеоданих[7]. Після створення Telegram Bot, був отриманий BOT_TOKEN для зв’язку з ботом. Цей токен необхідний для коректної роботи і був оголошений на початку

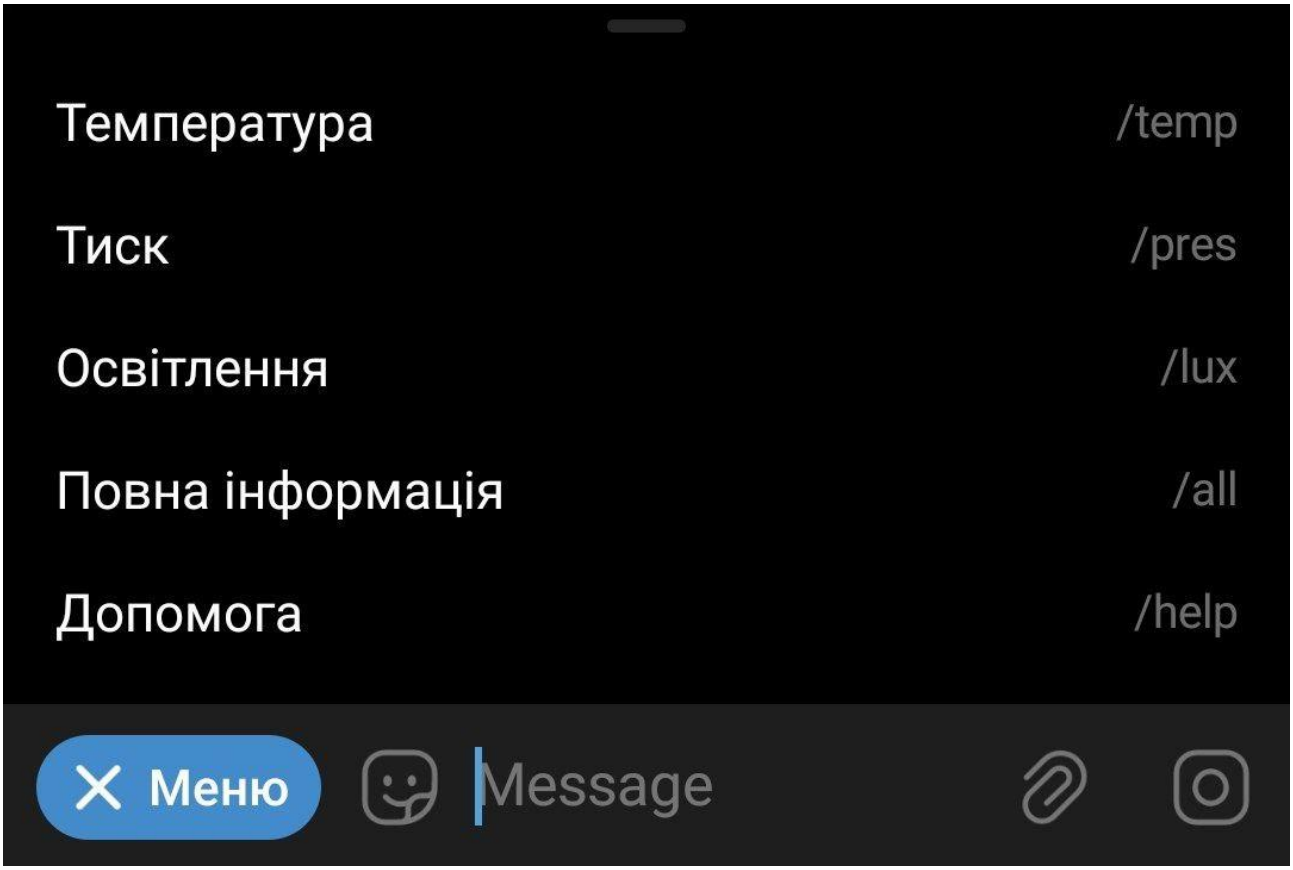

Рисунок 3.14 – Зображення меню в Telegram Bot

В меню відображаються всі команди для отримання метеоданих. Після натискання однієї з команд система фіксує звернення і посилає повідомлення яке відповідає зверненню

```
243  void handleNewMessages(int numNewMessages) {
244    Serial.print("handleNewMessages ");
245    Serial.println(numNewMessages);
246
247    for (int i = 0; i < numNewMessages; i++) {
248      String text = bot.messages[i].text;
249
250      if (text == "/temp") {
251        String message = "Температура: " + String(T) + "°C\n";
252        bot.sendMessage(bot.messages[i].chat_id, message, "");
253      } else if (text == "/pres") {
254        String message = "Тиск: " + String(P) + " мм. рт. ст.\n";
255        bot.sendMessage(bot.messages[i].chat_id, message, "");
256      } else if (text == "/lux") {
257        String message = "Яскравість освітлення: " + String(L) + " люкс\n";
258        bot.sendMessage(bot.messages[i].chat_id, message, "");
259      } else if (text == "/all") {
260        String message = "Температура: " + String(T) + "°C\n";
261        message += "Тиск: " + String(P) + " мм. рт. ст.\n";
262        message += "Яскравість освітлення: " + String(L) + " люкс\n";
263        message += "Рівень напруги: " + String(vcc) + " V\n";
264        message += "Рівень сигналу: " + String(WiFi.RSSI()) + "dBm\n";
265        bot.sendMessage(bot.messages[i].chat_id, message, "Markdown");
266      } else if (text == "/help") {
267        String welcome = "ESP8266 WiFi Telegram !\n";
268        welcome += "Для отримання даних оберіть потрібну команду:\n\n";
269        welcome += "/temp : Температура\n";
270        welcome += "/pres : Тиск\n";
271        welcome += "/lux : Яскравість освітлення\n";
272        welcome += "/all : Загальні значення\n";
273        bot.sendMessage(bot.messages[i].chat_id, welcome, "Markdown");
274      }
275    }
```

Рисунок 3.15 – Зображення реалізації функції відповіді на команди з Telegram Bot

Функція обробляє нові повідомлення , що надходять від користувачів з Telegram Bot. Після перевірки повідомлення на відповідність до заданих команд, функція відправляє відповідне повідомлення яке належить команді. У випадку якщо повідомлення не відповідає жодні з команд. Система просто ігнорує це повідомлення.

Кожне повідомлення має унікальний ідентифікатор чату “chat_id”. Завдяки цьому ідентифікатору система має можливість відправляти відповідь

саме тому користовачу який надіслав повідомлення і невиникає помилок з дублювання відповідей іншим користувачам.

3.1.6. Вимірювання метрологічних параметрів

Метрологічні дані головний аспект в роботі даної системи. Коректна робота датчиків має фундаментальне значення. В цій системі два датчика вимірювання метеданних:

- BMP280
- BH1750

BMP280 – це високоточний датчик для вимірювання температури та

атмосмосферного тиску в оточуючому середовищі[5]. Датчик здатний вимірювати температуру в діапазоні від -40°C до +85°C, а тиск в діапазоні від 300гПа до 1100гПа. Підключення може відбуватися за допомогою I2C шини або SPI шини.

BH1750 – це цифровий датчик який використовується для вимірювання інтенсивності світла в оточуюючому середовищі[6]. Датчик завдяки точності вимірювань здатний працювати в широкому діапазоні освітлення. Для підключення датчика використовується I2C шина або SPI шина

В нашому випададку підлкючення за допомогою I2C шини, так як є більш зручним та комфортним у використанні.

3.1.6.1. Вимірювання температури та тиску

Вимірювання температури та тиску відбувається завдяки датчику BMP280. Тому необхідно правильно його ініціалізувати для коректної роботии системи.

```
277
278 void measure() {
279   Wire.begin();
280   vcc = ((float)ESP.getVcc()) / 1000.0;
281   rssi = WiFi.RSSI();
282   T = bmp.readTemperature();
283   P = bmp.readPressure();
284   P = P / 133.322;
285   Serial.print("T= ");
286   Serial.print(T, 2);
287   Serial.print(" C; P= ");
288   Serial.print(P , 2);
289   Serial.print(" mmHg; VCC= ");
290   Serial.print(vcc, 3);
291   Serial.print(" V, RSSI= ");
292   Serial.print(WiFi.RSSI());
293   Serial.println(" dBm ");
294 }
```

Рисунок 3.16 – Зображення реалізації функції вимірювання даних з датчика BMP280

За допомогою команди “Wire.begin()” ініціалізується з’єднання датчика з мікроконтролером по шині I2C.

Отримання данних температури та тиску відбувається за допомогою команд “bmp.readTemperature” та “bmp.readPressure” які зберігають їх у змінні T та P відповідно. Для комфортного відображення тиску в звичній для нас формі, відбувається конвертація значення з одиниць виміру тиску Паскаль в мм ртутного стопчика за допомогою ділення на 133.332

“ESP.getVcc()” та “WiFi.RSSI()” команди які допомагають дізнатися напругу живлення мікроконтролера та отриматі рівень сигналу WiFi.

Вимірянні данні виводяться в консоль як один із методів відображення інформації

3.1.6.2. Вимірювання яскравості освітлення

Яскравість освітлення є третім метеопараметром які відслідковує система для відображення в подальшому

```
295
296  void luxt() {
297    Wire.begin();
298    lightMeter.begin();
299    L = lightMeter.readLightLevel();
300    Serial.print("L= ");
301    Serial.print(L, 2);
302    Serial.print(" Lx ");
303  }
304
```

Рисунок 3.17 – Зображення реалізації функції виміру яскравості освітлення

У випадку як із датчиком BMP280 ініціалізація з'єднання з шиною I2C вібдувається за допомогою команди “Wire.begin()”.

Команда “lightMeter.begin()” ініціалізує датчик освітленості

За допомогою “L = lightMeter.readLightLeverl()” отримується значення яскравості освітлення та зберігається в змінній “L”

Отримане значення виводить у консоль як один із методів відображення.

3.1.7. Поширення отриманих даних в загальний доступ

Поширення отриманих метеодних в загальний доступ є одним із методів відображення інформації та розвитку проєкту.

На даний час існує чимало платформ які дають можливість розробникам показувати свої розробки. Такі сервіси об’єднують спільноту по всьом світі даючи можливість вдосконалювати свої проєкти та допомагати іншим. Одними із таких сервісів є Narodmon та Dweet.

3.1.7.1. Платформа Narodmon та реалізація функції поширення даних на нього

Narodmon – це сервіс для збору та обміну даними про навколишнє середовище[3]. Він дозволяє користувачам збирати дані за допомогою датчиків, моніторити їх та використовувати спільно з іншими користувачами

На платформі є можливість створювати власні проєкти та додавати дані з власних датчиків, щоб інші користувачі мали можливість ними користуватися. Однією із особливістю платформи є відображення даних на мапі. Це дає можливість для моніторингу та аналізу метрологічних даних в різних куточках світу. Дані можна використовувати і в наукових дослідженнях, так як за часту датчики які використовують користувачі є доволі високоточними і мають невелику похибку

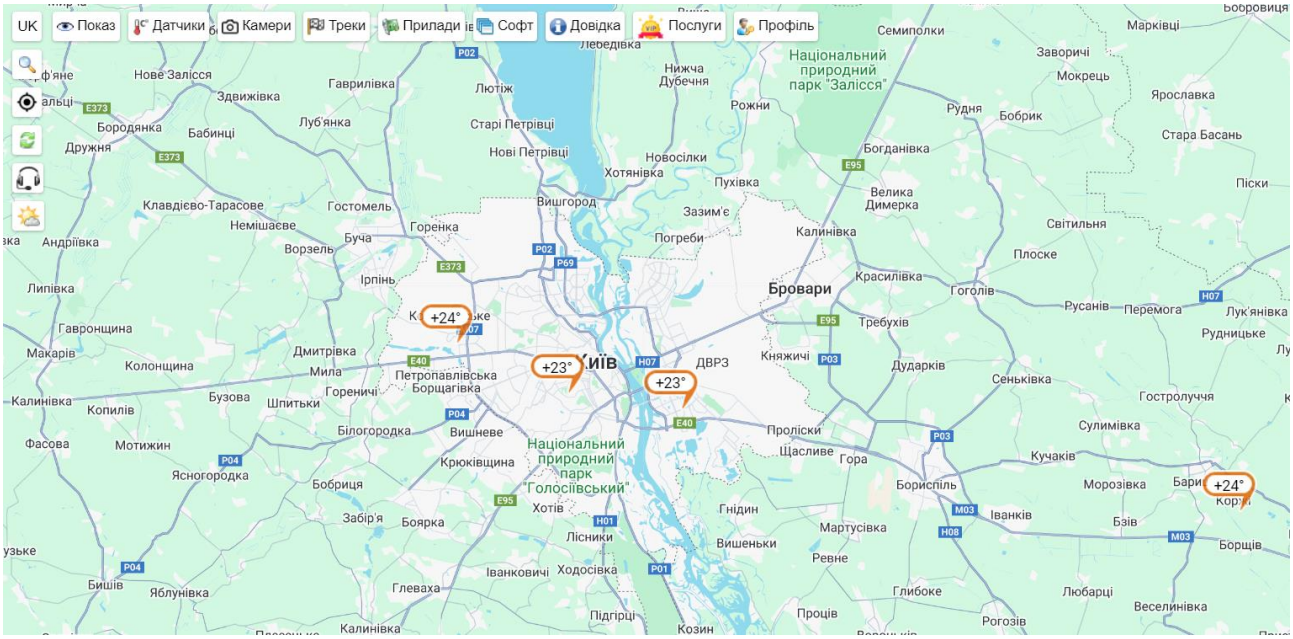

Рисунок 3.18 – Зображення сервісу Narodmon

Тому однією із цілей системи було поширення даних до сервісу Narodmon

```
305  void narodmonSend() {
306    unsigned long currentTime = millis();
307
308    // Перевіряємо, чи минуло вже narodmonSendInterval мілісекунд
309    if (currentTime - lastNarodmonSendTime >= narodmonSendInterval) {
310      if (!client.connect(host, httpPort)) {
311        Serial.println("Connection to narodmon failed");
312        return;
313      }
314      Serial.println("Sending data on narodmon ...");
315      client.print("#");
316      client.print(WiFi.macAddress());
317      client.print("#");
318      client.print("ESP8266+BMP280");
319      client.println();
320      client.print("#temp#");
321      client.println(T);
322      client.print("#pres#");
323      client.println(P);
324      client.print("#lx#");
325      client.println(L);
326      client.print("#rssi#");
327      client.println(rssi);
328      client.print("#vcc#");
329      client.println(vcc);
330      client.println("##");
331      delay(10);
332      Serial.print("Requesting: ");
333      while (client.available()) {
```

Рисунок 3.19 – Зображення першої частини реалізації функції передачі даних на сервіс Narodmon

```
333      while (client.available()) {
334        String line = client.readStringUntil('\r');
335        Serial.println(line);
336      }
337      client.stop();
338      Serial.println("narodmon ok!");
339
340      // Оновлюємо час останнього надсилання
341      lastNarodmonSendTime = currentTime;
342    }
343  }
344
```

Рисунок 3.20 – Зображення другої части реалізації функції передачі даних на сервіс Narodmon

На платформі існують інтервальні обмеження передачі данних, мінімум 5 хвилин, тому в функції було реалізовано таймер який перевіряє коли відбувалась остання передача данних і коли можна посилати дані в наступний раз. Якщо відбувається помилка під час підключення до сервісу в консоль виводиться відповідне повідомлення; якщо відправка відбулася успішно, про це також виводиться відповідне повідомлення

3.1.7.2. Платформа Dweet.io та поширення даних на неї

Dweet.io – це платформа для обміну даними між пристроями через інтернет[4]. Вона дозволяє відправляти дані з різноматніх датчиків або пристроїв до хмарного сервісу, де вони можуть бути збереженні та доступні для використання

На платформі можна побачити різноманіття датчиків та пристроїв які під’єднують користувачі з усього світу. Вона надає можливість зберігання даних на власному хмарному середовищі для подальшого використання користувачем. Це дає можливість для подальшого аналізу та використання данних в різноматних цілях.

Рисунок 3.21 – Головна сторінка платформи Dweet.io

Тому з ціллю зберігання данних і подальшого їхнього використання я вирішив поширити дані на цей сервіс

```
void dweetSend() {
  if (!client.connect(dweet, dweetPort)) {
    Serial.println("connection dweet failed");
    return;
  }
  client.print(String("GET /dweet/for/ESP8266_BMP280?Temperature=")  + String(T, 2)
               + "&Pressure=" + String(P, 2)
               + "&Lx=" + String(L, 2)
               + "&RSSI=" + String(rssi)
               + "&VCC=" + String(vcc, 3)
               + "&Uptime=" + String(millis() / 1000)
               + " HTTP/1.1\r\n" +
               "Host: " + dweet + "\r\n" +
               "Connection: close\r\n\r\n");
  delay(10);
  while (client.available()) {
    String line = client.readStringUntil('\r');
    Serial.print(line);
  }
  client.stop();
  Serial.println("dweet ok!");
}
```

Рисунок 3.22 – Зображення реалізації функції поширення даних на сервіс Dweet.io

По аналогічному принципу як і при підключені до сервісу Narodmon спочатку відбувається перевірка підключення до сервісу, виводяться в консоль відповідні повідомлення про успішність операції

Після успішного підключення створюється відповідний запит із заними сенсорів який надсилається на сервер. Коли система отримала відповідь від сервера, виводиться в консоль повідомлення про виконання задачі.

3.1.8. Ініціалізація функцій

Правильна ініціалізація функцій є найголовнішим аспектом в коректній роботі системі. Навіть якщо всі функції написано правильно, а їхня ініціалізація

не коректно, вся система буде працювати не вірно. Тому потрібно все правильно реалізувати для стабільної та правильної роботи системи.

```
367
368  void setup() {
369    Serial.begin(115200);
370    bmp.begin(0x76);
371    connectWiFi();
372    setupTelegram();
373    Serial.print("MAC адреса: ");
374    Serial.println(WiFi.macAddress());
375    server.on("/", HTTP_GET, handleWebpage);
376    server.on("/data", HTTP_GET, handleData);
377    server.begin();
378  }
```

Рисунок 3.23 – Зображення реалізації функції “setup()”

Функція “setup()” є однією з двох основних функцій для коректної роботи системи на мікроконтролері ESP8266. В ній реалізовано запуск функцій які будуть виконуватися під час запуску системи

“Serial.begin(115200)” – ініціалізація серійного порта на швидкості 115200 біт/с.

“bmp.begin(0x76)” – ініціалізація сенсора BMP280 на адресі I2C шини 0x76.

“connectWiFi()” – ініціалізація функції підключення до Wi-Fi.

“setupTelegram()” – ініціалізація функції підключення до Telegram.

“Serial.print(“”)” та “Serial.println(WiFi.macAddress())” – виводять MAC – адресу мікроконтролера на монітор серійного порта.

“server.on(“/”, HTTP_GET, handleWebpage)” та “server.on(“/data”, HTTP_GET, handleData)” – ініціалізація налаштувань сервера для обробки запитів.

“server.begin()” – ініціалізація сервера.

```
379
380 void loop() {
381   if (millis() > lasttime + interval) {
382     lasttime = millis();
383     measure();
384     luxt();
385     checkWiFiConnection();
386     narodmonSend();
387     dweetSend();
388     int numNewMessages = bot.getUpdates(bot.last_message_received + 1);
389     while (numNewMessages) {
390       Serial.println("got response");
391       handleNewMessages(numNewMessages);
392       numNewMessages = bot.getUpdates(bot.last_message_received + 1);
393     }
394     lasttime = millis();
395   }
396   server.handleClient();
397 }
```

Рисунок 3.24 – Зображення реалізації функції “loop()”

Функція “loop()” є другою з двох основних функцій для коректної роботи системи на мікроконтролері ESP8266. В ній реалізовано запуск функцій які будуть виконуватися під час роботи системи. Мета цієй фукції – це виконувати дії, які повинні виконуватися циклічно або з деякою переодичністю

“if (millis() > lasttime + interval)” – це умова для перевірки часу з моменту виконання останніх дій.

“measure(), luxt(), checkWiFiConnection(), narodmonSend(), dweetSend()” – це ініціалізація функцій які повинні виконуватися з певною переодичністю.

“int numNewMessages = bot.getUpdates(bot.last_message_received +1)” – це рядок який отримує всі нові повідомлення з Telegram.

“while(numNewMessages)” – це цикл який спрямований для обробкм повідомлень з Telegram.

“lasttime = millis()” – це рядок для оновлення значення lasttime, щоб вірно вибрати момент виконання циклу.

“server.handleClient()” – це обробка запитів до сервера, розміщюється поза основної умови, щоб виконувався постійно.

3.1.9. Збереження даних, аналіз та прогнозування

Збереження даних є чи не одною з найголовніших функцій будь-якої системи. Це дає можливість аналізувати та прогнозувати майбутні події. Але для зберігання даних необхідна велика кількість вільної пам’яті. Мікроконтролер має на борту 4 мегабайти пам’яті, цього достатньо для комфортної роботи системи, але замало для збереження даних на певний період часу. Саме для цього сервіс Narodmon дає можливість своїм користувачам зберагати дані на власному хмарному середовищі. Це найкраще рішення для систем які немають великого обсягу пам’яті. Також платформа надає можливість збереження даних на хмарне середовище користувача, цим і скористаємося.

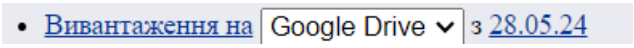

Рисунок 3.25 – Вивантаження даних на Google Drive

На Google Drive дані зберігаються в окремі папці. Зберігання відбувається за кожен день роботи системи в форматі csv. Цей формат доступний для Google Sheets та Microsoft Excel, де дані відображаються у вигляді таблиці і які в подальшому можна використовувати для побудови графіків та аналізу.

1717016400	30.05.2024	12:00:00 AM	18.73	740.79	0	-42	3
1717016700	30.05.2024	12:05:00 AM	18.63	740.73	0	-42	3
1717017000	30.05.2024	12:10:00 AM	18.84	740.62	0	-44	3
1717017300	30.05.2024	12:15:00 AM	18.94	740.6	0	-43	3
1717017660	30.05.2024	12:21:00 AM	19.17	740.56	0	-43	3
1717017960	30.05.2024	12:26:00 AM	19.17	740.51	0	-43	3
1717018260	30.05.2024	12:31:00 AM	19.22	740.52	0	-47	3
1717018560	30.05.2024	12:36:00 AM	19.24	740.49	0	-43	3
1717018860	30.05.2024	12:41:00 AM	19.32	740.42	0	-45	3
1717019160	30.05.2024	12:46:00 AM	19.31	740.38	0	-42	3
1717019460	30.05.2024	12:51:00 AM	19.38	740.41	0	-43	3
1717019760	30.05.2024	12:56:00 AM	19.35	740.37	0	-47	3
1717020060	30.05.2024	1:01:00 AM	19.26	740.36	0	-43	3
1717020360	30.05.2024	1:06:00 AM	19.34	740.36	0	-43	3
1717020780	30.05.2024	1:13:00 AM	19.57	740.33	0	-43	3.03
1717021080	30.05.2024	1:18:00 AM	19.54	740.39	0	-44	3.03
1717021440	30.05.2024	1:24:00 AM	19.55	740.36	0	-45	3.03
1717022040	30.05.2024	1:34:00 AM	19.67	740.34	0	-46	3.03
1717022340	30.05.2024	1:39:00 AM	19.64	740.34	0	-44	3.03
1717022700	30.05.2024	1:45:00 AM	19.63	740.31	0	-45	3.03
1717023180	30.05.2024	1:53:00 AM	19.62	740.26	0	-45	3.01
1717023540	30.05.2024	1:59:00 AM	19.55	740.21	0	-43	3.01
1717023840	30.05.2024	2:04:00 AM	19.51	740.16	0	-48	3.01
1717024200	30.05.2024	2:10:00 AM	19.5	740.01	0	-46	3.03
1717024860	30.05.2024	2:21:00 AM	19.48	739.93	0	-48	3.01
1717025220	30.05.2024	2:27:00 AM	19.39	740.02	0	-47	3.03
1717025520	30.05.2024	2:32:00 AM	19.42	740.05	0	-48	3.03
1717025820	30.05.2024	2:37:00 AM	19.54	740.13	0	-48	3.03

Рисунок 3.26 – Відображення даних у вигляді таблиці у Google Sheets

Ці дані можна використовувати для створення графіків, аналізу та прогнозування. Також платформа надає можливість будувати графіки

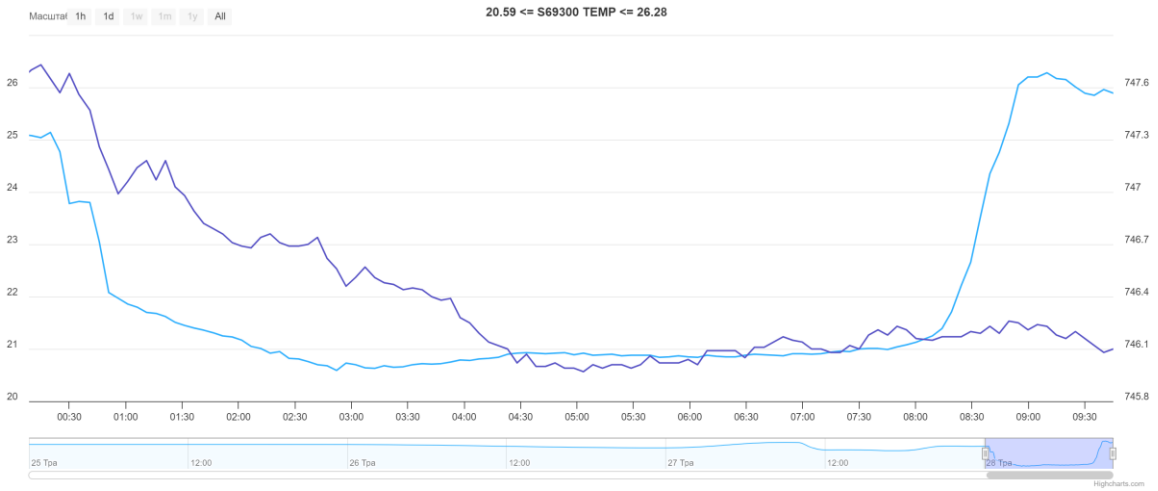

Рисунок 3.27 – Побудований графік температури та тиску на платформі Narodmon

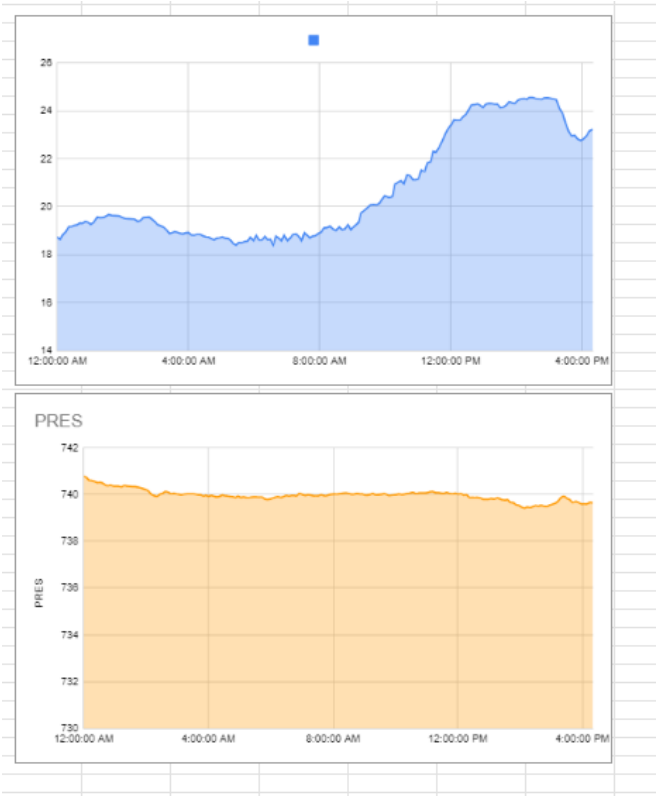

Рисунок 3.28 – Побудований графік температури та тиску в Google Sheets

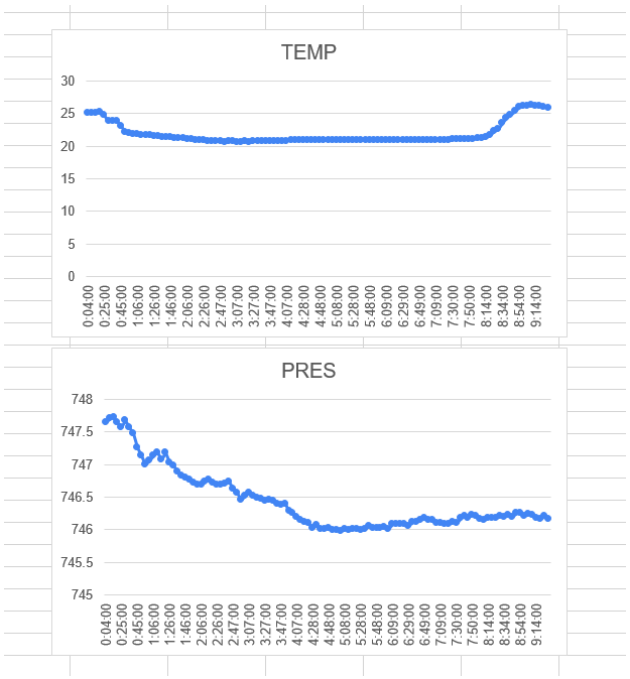

Рисунок 3.29 – Побудований графік температури та тиску в Microsoft Excel

Рисунок 3.30 – Прогнозування температури в Microsoft Excel

Рисунок 3.31 – Прогнозування тиску в Microsoft Excel

Завдяки можливості зберігати виміряні системою дані ми отримуємо можливість для детального аналізу та прогнозування. Це збільшує спектр наших можливостей та дозволяє робити багато речей. Ми можемо використовувати ці дані для розуміння тенденції, виявлення взаємозв’язків, передбачення можливих подій та обґрунтування наших рішень.

3.2. Інструкція користувача

3.2.1. Портативний режим роботи

В портативному режимі роботи метеостанція працює незалежно від ПК, тому необхідно створити точку доступу Wi-Fi або перейменувати існуючу відповідно до заданих параметрів в системі:

SSID: ESP8266

Password: 12345678

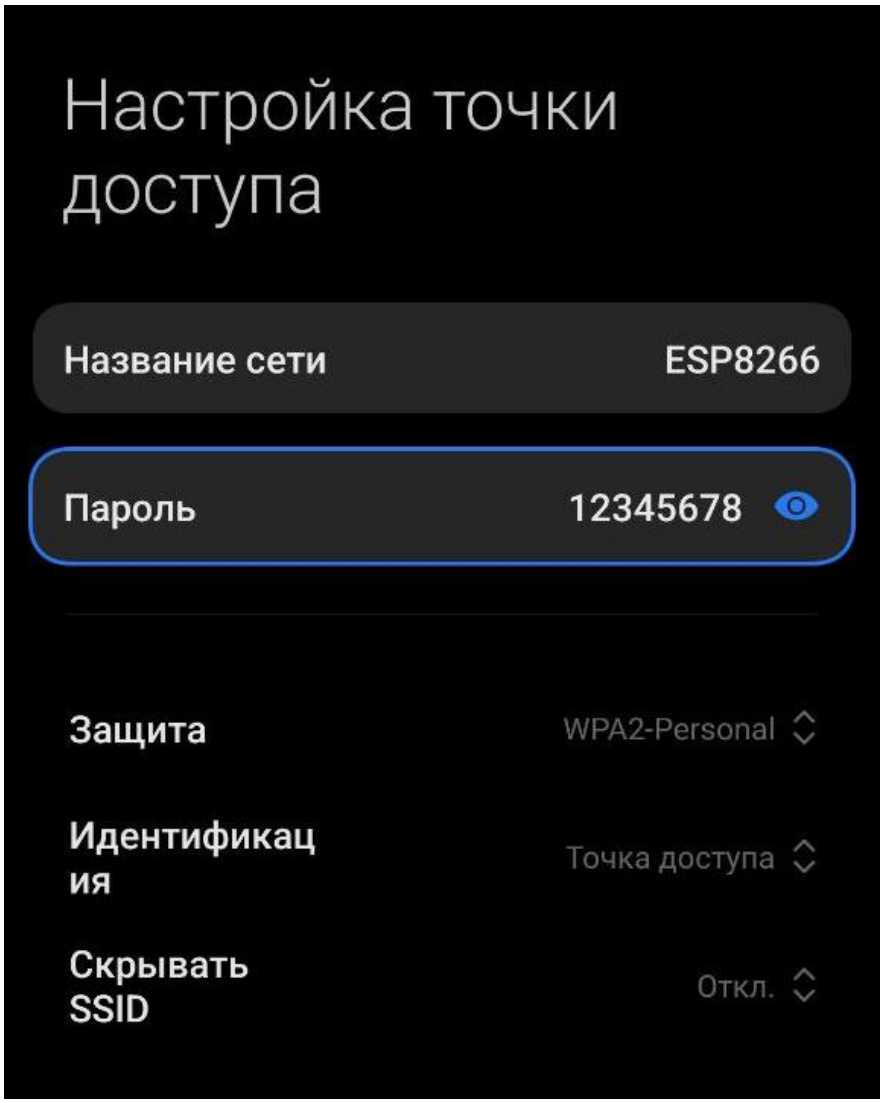

Рисунок 3.32 – Налаштування точки доступу Wi-Fi

Після створення точки доступу згідно заданих параметрів необхідно під’єднати живлення до системи. Це можна зробити або через акумулятор який описувався вище або через кабель USB-A – microUSB та елемент живлення

Рисунок 3.33 – Підключення мікроконтролера до елемента живлення за допомогою кабелю USB-A – microUSB

Після чого мікроконтролер подасть сигнал світлодіодом про подачу живлення

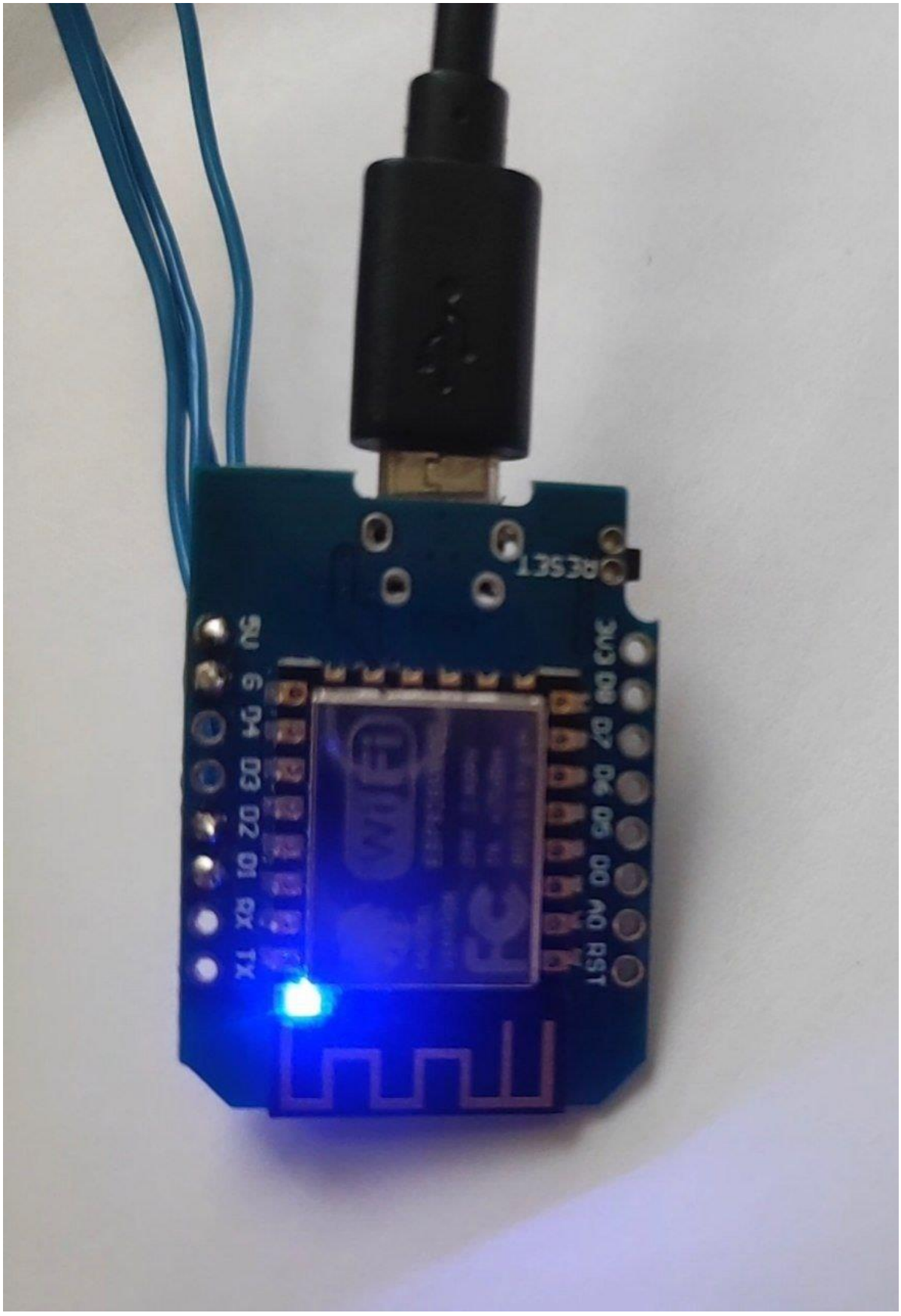

Рисунок 3.34 – Індикація світлодіода про початок роботи

З моменту підключення живлення необхідно почекати приблизно 10-15 секунд. За цей час система під ‘єднається до заданої мережі і розпочне свою роботу

Щоб почати перегляд даних які відслідковує метеостанція необхідно під ‘єднатися до бота в мессенджері Telegram, увівши ім’я бота у пошук @espbmp_bot.

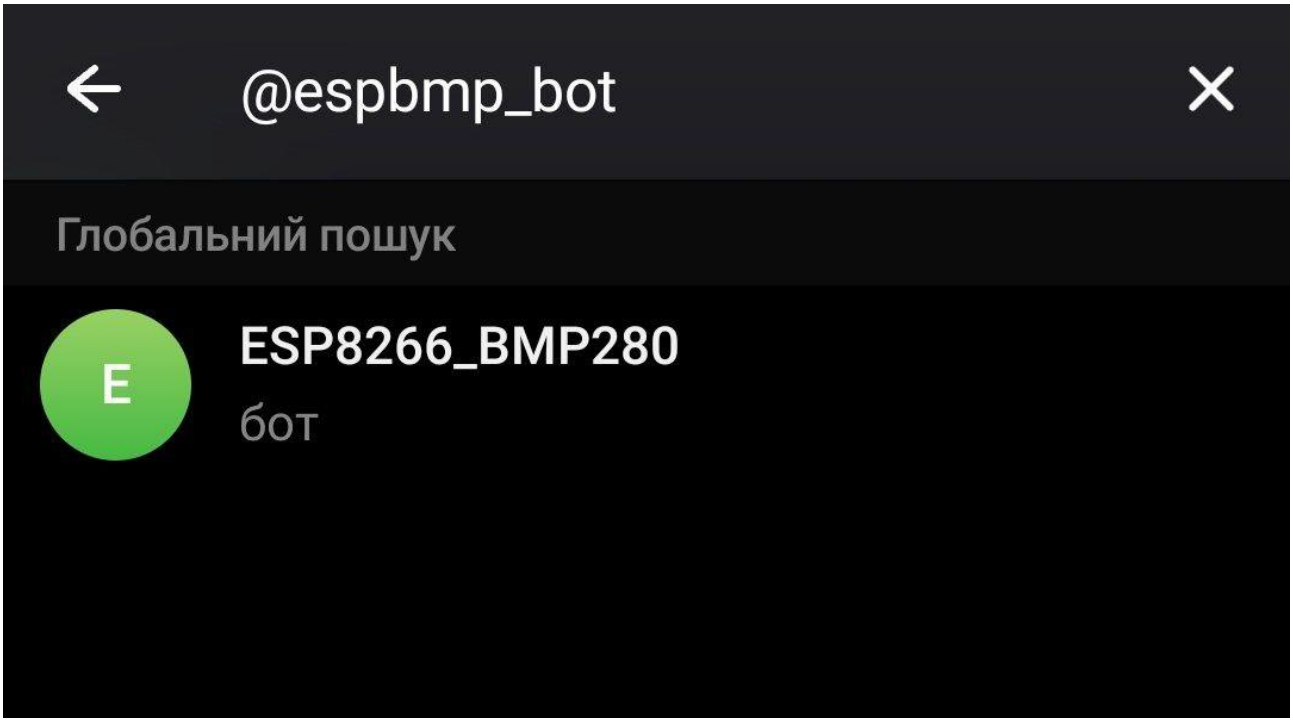

Рисунок 3.35 – Пошук бота за ім’ям в Telegram

Після чого необхідно перейти на бота та натиснути кнопку START

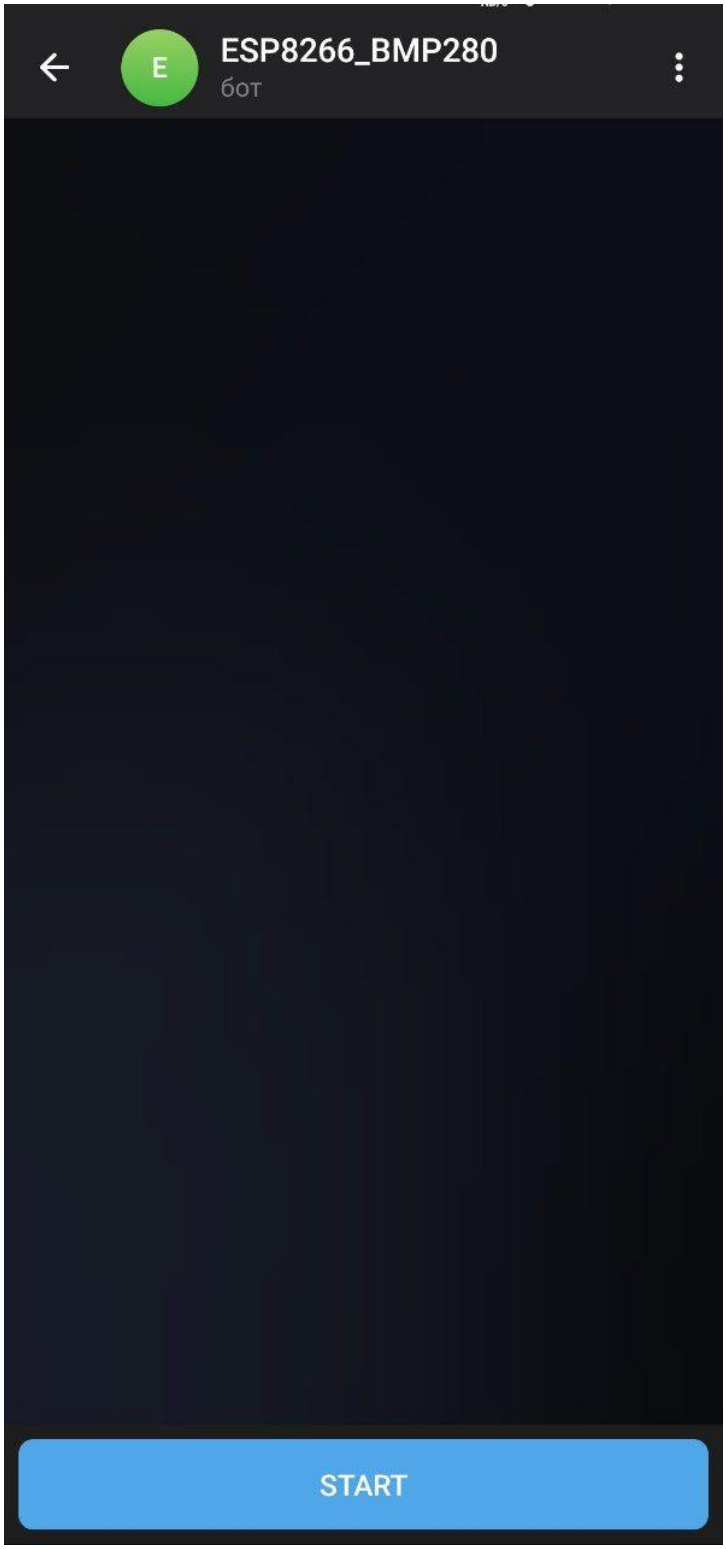

Рисунок 3.36 – Зображення при початковому вході до бота

Після натискання кнопки START, з’являється кнопка Меню яка вміщає в собі всі команди які може виконати система

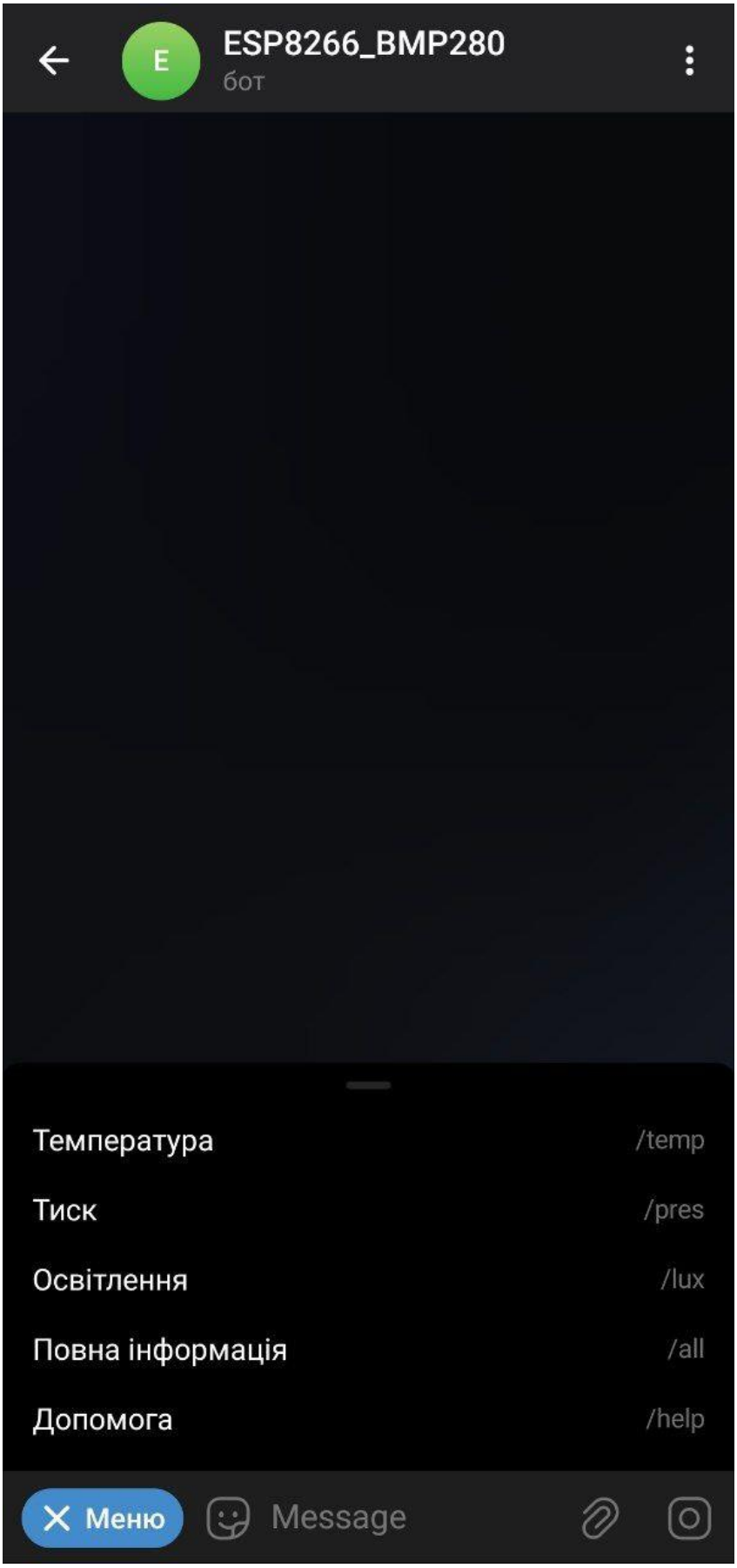

Рисунок 3.37 – Головне меню бота

При натисканні будь-якої кнопки в меню буде відбуватися відповідний параметричний запит до системи, відповіді до яких закладені в системі

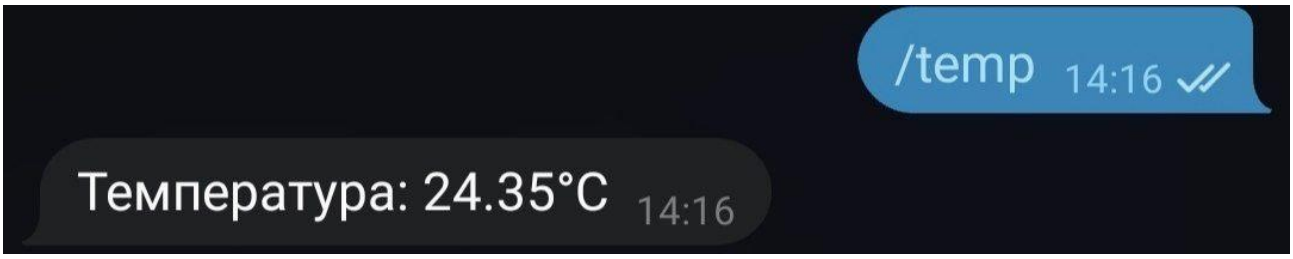

Рисунок 3.38 – Запит Температура та відповідь системи на запит

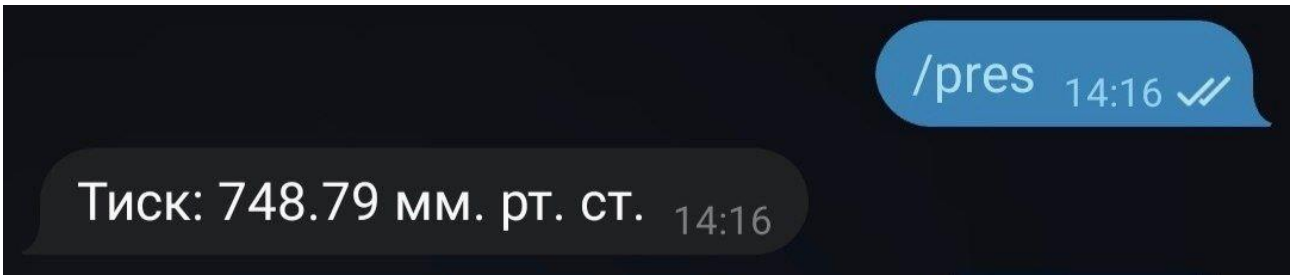

Рисунок 3.39 – Запит Тиск та відповідь системи на запит

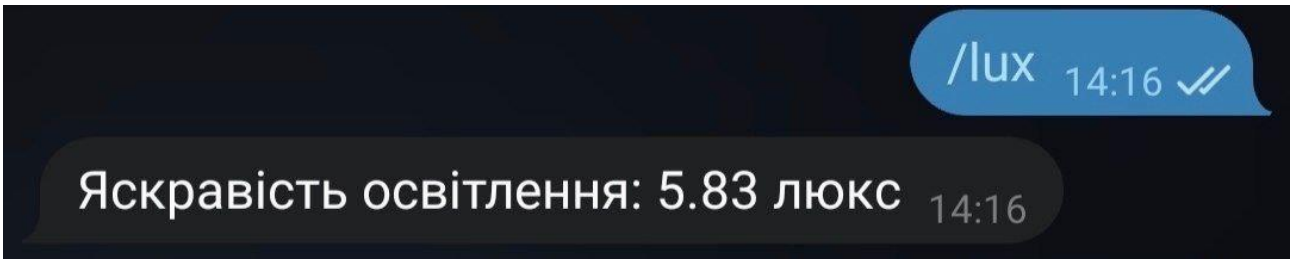

Рисунок 3.40 – Запит Освітлення та відповідь системи на запит

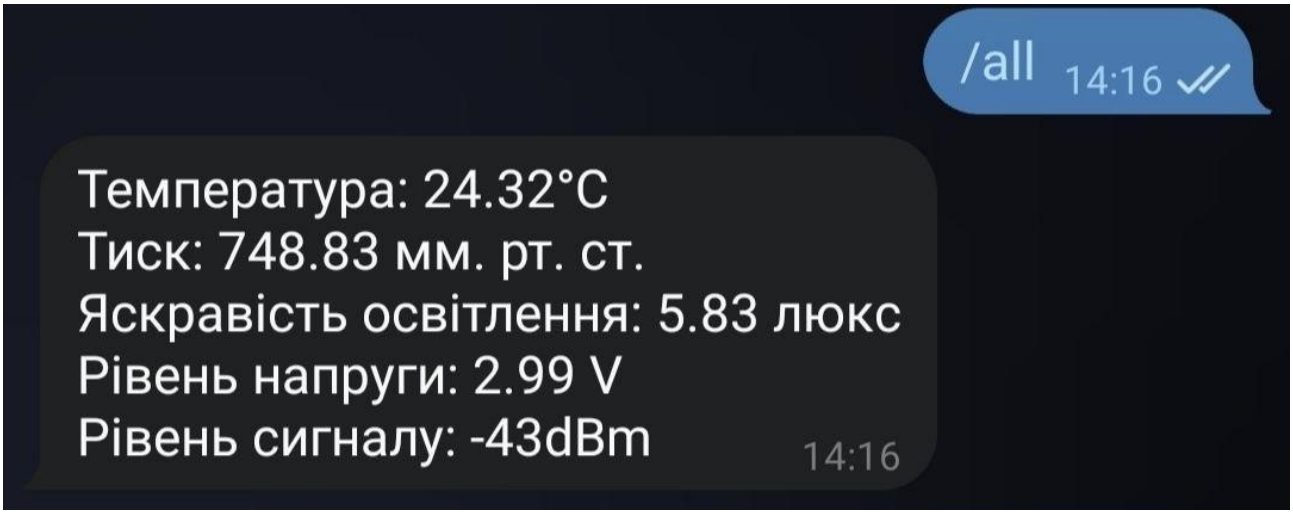

Рисунок 3.41 – Запит Повна інформація та відповідь системи на запит

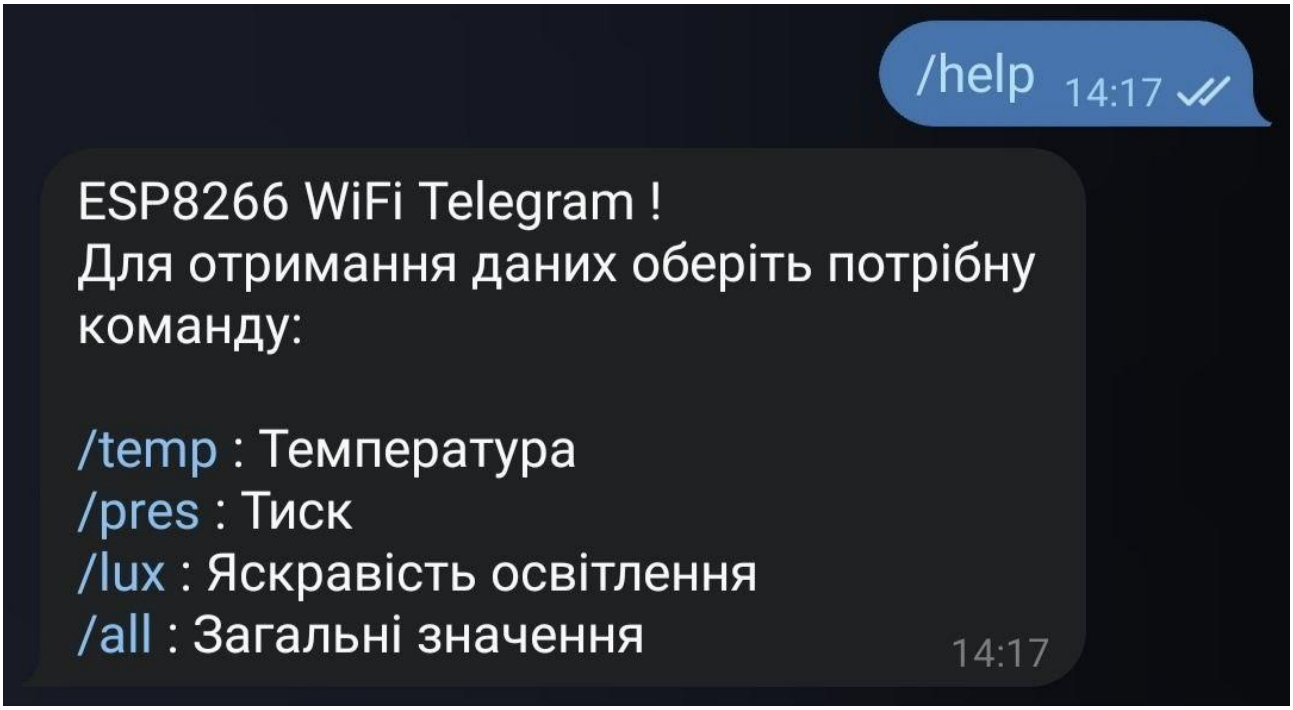

Рисунок 3.42 – Запит Допомога та відповідь системи на запит

Також можливий перегляд даних за допомогою сервісу Dweet.io. Необхідно перейти за посилання до якого підключена система: https://dweet.io/follow/ESP8266_BMP280

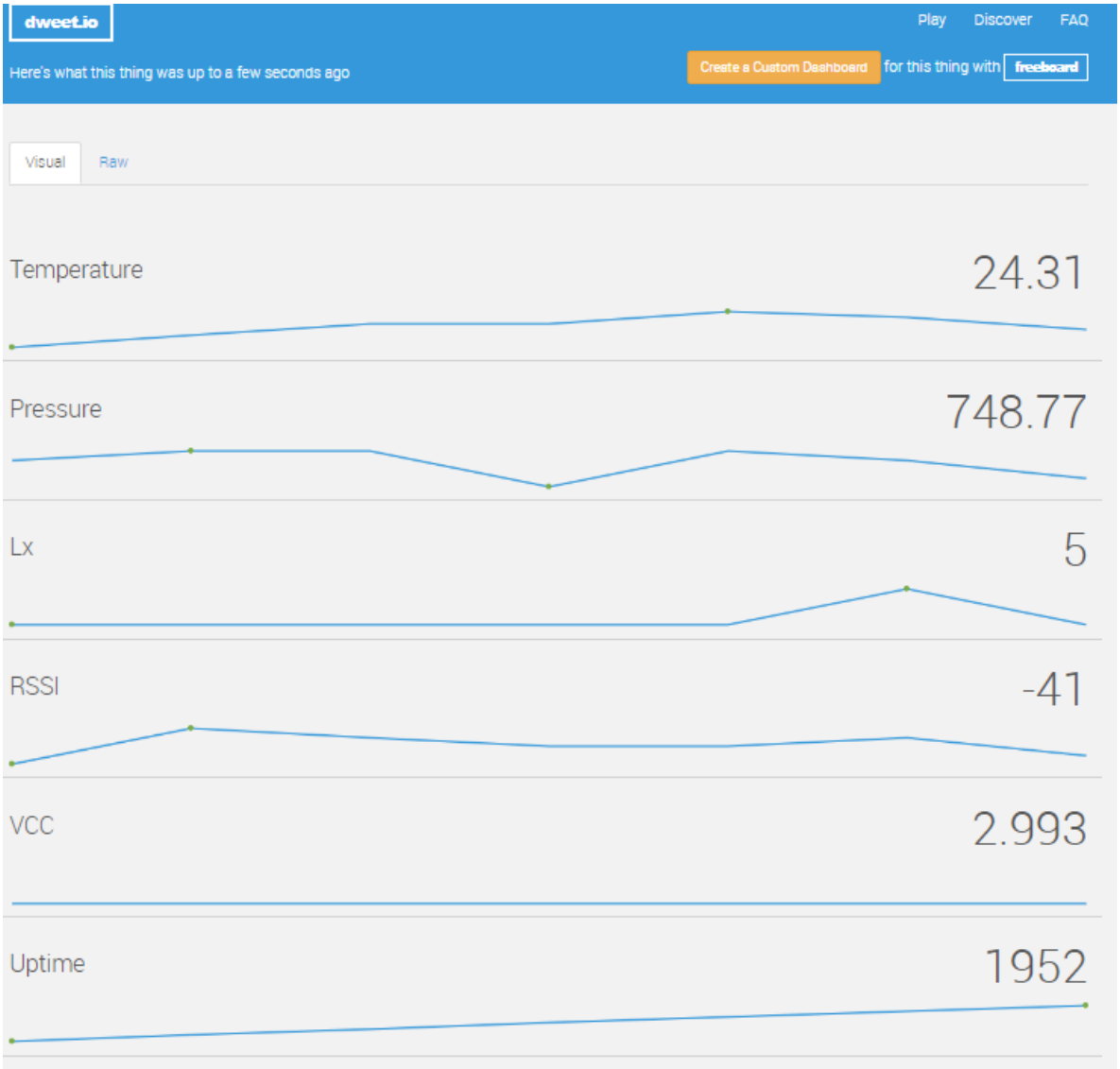

Рисунок 3.43 – Відображення метеоданих в сервісі Dweet.io

3.2.2. Стаціонарний режим роботи

Для комфортного спаціонарного режиму необхідно встановити та запустити сердовище розробки Arduino IDE.

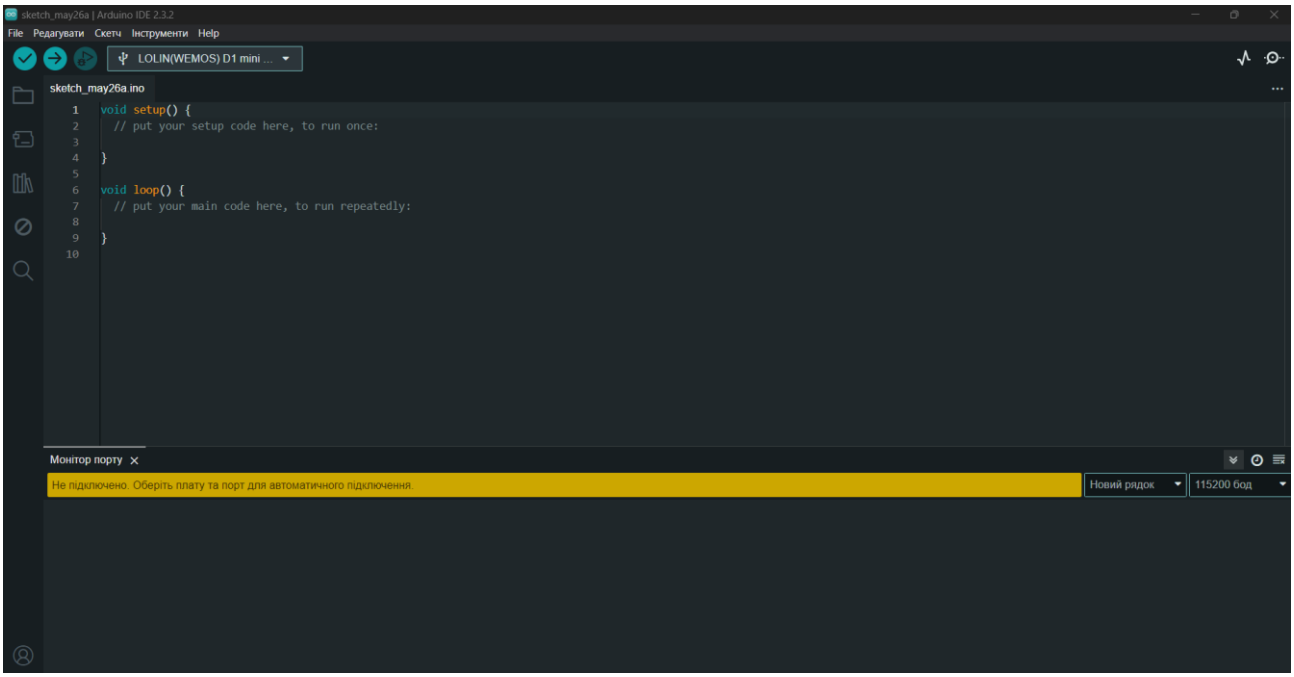

Рисунок 3.44 – Середовище розробки Arduino IDE

Після чого необхідно під’єднати метеостанцію до ПК за допомогою кабелю USB-A – microUSB та вибрати необхідні налаштування в середовищі розробки

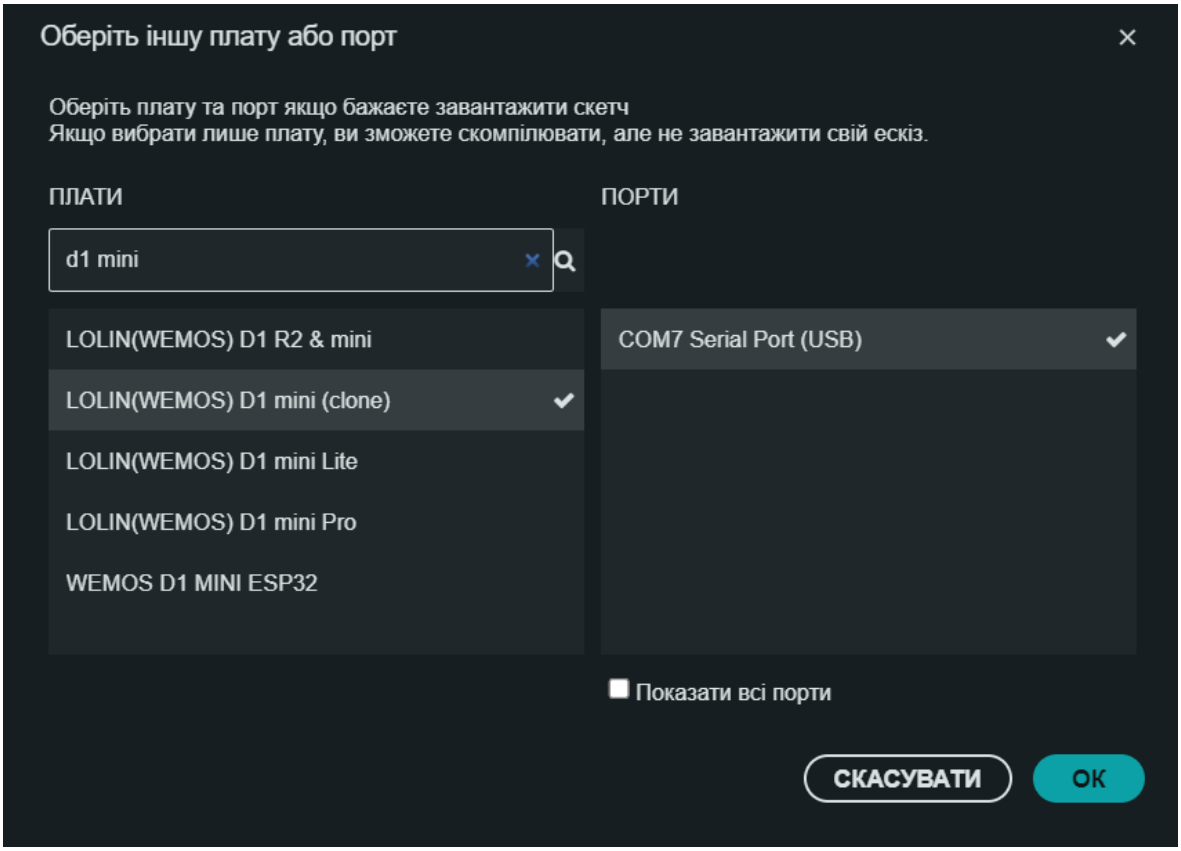

Рисунок 3.45 – Вибір параметрів метостанції для коректної роботи

Якщо параметри точки доступу задані відподно до параметрів системи то в моніторі порту з’явиться повідомлення про це і система почне успішно працювати

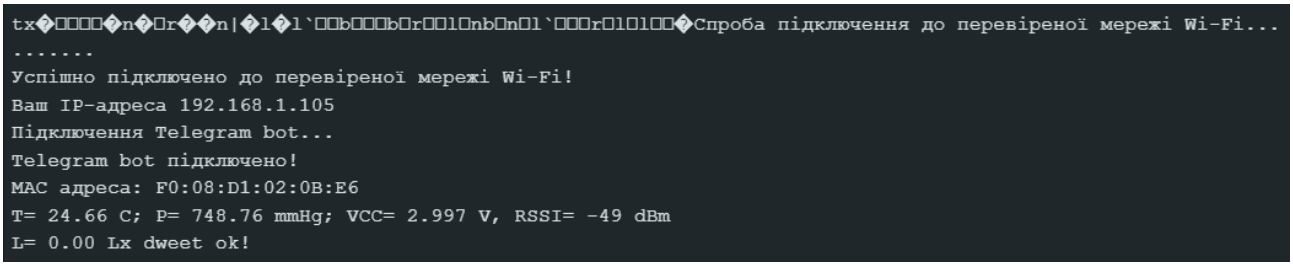

Рисунок 3.46 – Повідомлення в моніторі порта про початок роботи

Якщо параметки точки доступу не будуть відповідати заданим, система розпочне сканування мережі після чого дасть можливість підключитися до обраної користувачем мережі

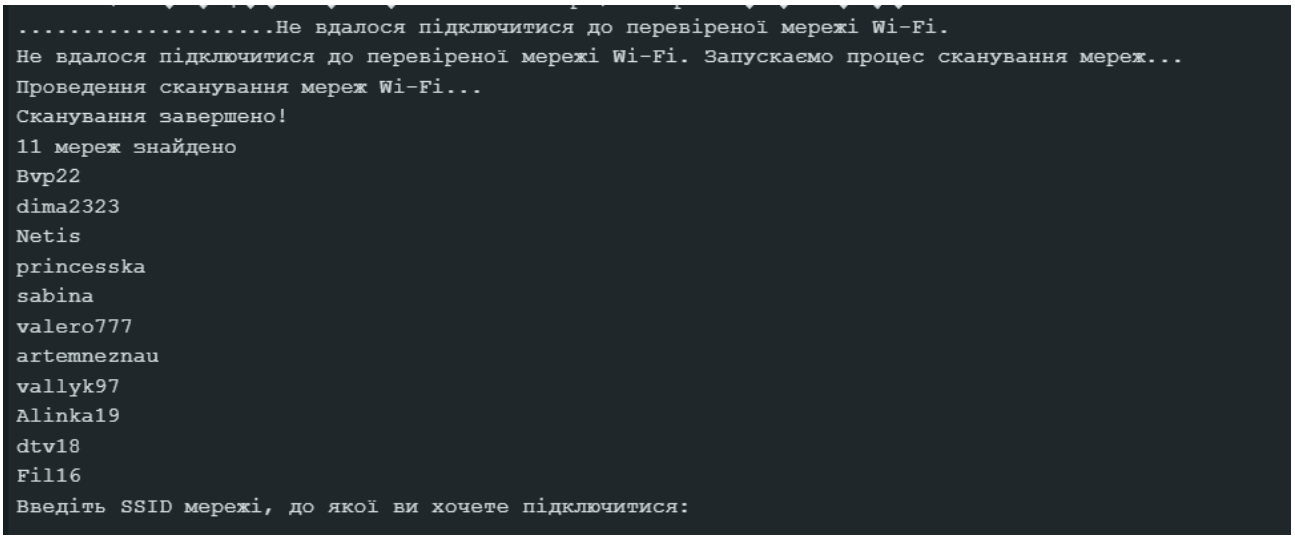

Рисунок 3.47 – Повідомлення в моніторі порта про сканування мережі

Після сканування необхідно увести назву мережі до строки

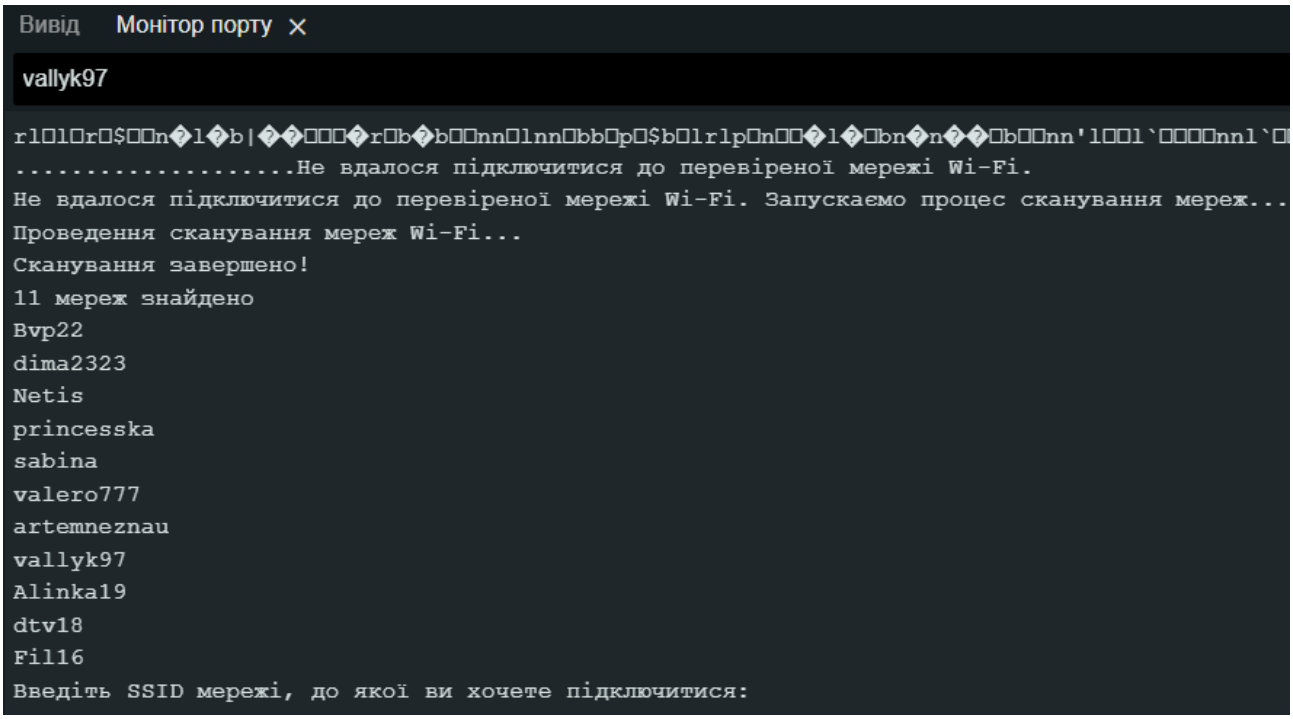

Рисунок 3.48 – Введення в строку необхідної мережі

Після чого система попросить ввести пароль від данної мережі

Рисунок 3.49 – Введеня в строку паролю від мережі

Якщо назву мережі та пароль буде введено вірно система почне свою роботу

```
Введіть SSID мережі, до якої ви хочете підключитися:
Введіть пароль:
Підключення до vallyk97
......
Успішно підключено до мережі Wi-Fi!
Ваш IP-адреса 192.168.1.105
Підключення Telegram bot...
Telegram bot підключено!
MAC адреса: F0:08:D1:02:0B:E6
T= 28.50 C; P= 748.68 mmHg; VCC= 2.993 V, RSSI= -55 dBm
L= 8.33 Lx Sending data on narodmon ...
Requesting: narodmon ok!
dweet ok!
```

Рисунок 3.50 – Розпочаток роботи після успішного підключення

Якщо у випадку дані від мережі будуть введні не правильно, система напише про це після чого цикл підключення до мережі почнеться знову

```
Введіть SSID мережі, до якої ви хочете підключитися:
Введіть пароль:
Підключення до vallyk97
...................Час очікування підключення вичерпано!
Помилка підключення, повторна спроба...
```

Рисунок 3.51 – Помилка підключення до мережі

Відслідковувати дані можна буде в моніторі порта

```
T= 27.46 C; P= 748.64 mmHg; VCC= 2.993 V, RSSI= -58 dBm
L= 43.33 Lx dweet ok!
T= 27.37 C; P= 748.62 mmHg; VCC= 2.994 V, RSSI= -48 dBm
L= 31.67 Lx dweet ok!
T= 27.37 C; P= 748.65 mmHg; VCC= 2.995 V, RSSI= -54 dBm
L= 31.67 Lx dweet ok!
T= 27.21 C; P= 748.65 mmHg; VCC= 2.993 V, RSSI= -58 dBm
L= 31.67 Lx dweet ok!
T= 27.09 C; P= 748.65 mmHg; VCC= 2.993 V, RSSI= -50 dBm
L= 31.67 Lx dweet ok!
T= 26.94 C; P= 748.64 mmHg; VCC= 2.993 V, RSSI= -50 dBm
L= 31.67 Lx dweet ok!
T= 26.78 C; P= 748.66 mmHg; VCC= 2.994 V, RSSI= -48 dBm
L= 31.67 Lx dweet ok!
T= 26.63 C; P= 748.68 mmHg; VCC= 2.993 V, RSSI= -46 dBm
L= 32.50 Lx dweet ok!
```

Рис 3.52 – Перегляд данних в моніторі порта

Або данні можна переглядати за допомогою Telegram та Dweet.io, інструкція до перегляду розписана в розділі 3.2.1

РОЗДІЛ 4. ОХОРОНА ПРАЦІ

Безпека праці при використанні інструментів

Робота з електроприладами вимагає дотримання особливих правил і дотримання вимог безпеки праці. Нижче наведені основні принципи безпеки при використанні електроприладів відповідно до державних стандартів України.

Електрична безпека

1. Перевірте стан електричного приладу: перед початком роботи ви повинні перевірити наявність пошкоджень, вирівнювання і правильне підключення електричного приладу до електричної мережі.

2. Використання захисного спорядження: при використанні електроприладів слід використовувати захисні рукавички, захисні окуляри та інше захисне спорядження для запобігання травм.

3. Безпечне відключення: Перед проведенням технічного обслуговування або ремонту електроприладів необхідно відключити їх від джерела живлення і переконатися у відсутності напруги .

4. Заземлення: електричні пристрої повинні бути належним чином заземлені для захисту від перенапруги та запобігання небезпечним ситуаціям .

Пожежна безпека при використанні електроприладів

Робота з електроприладами також пов'язана з пожежонебезпекою. Нижче наведені основні правила пожежної безпеки при використанні електроприладів відповідно до державних стандартів України.

Правила пожежної безпеки

1. Відповідність вимогам електромережі: електромережа повинна відповідати стандартам безпеки і бути обладнана захисним заземленням для запобігання загоряння .

2. Відстань до легкозаймистих матеріалів: електричні пристрої повинні бути встановлені на безпечній відстані від легкозаймистих матеріалів, щоб уникнути небезпеки пожежі.

3. Перевірте тепловий режим: перед використанням електричних приладів перевірте тепловий режим і уникайте перегріву.

4. Пожежогасіння: для швидкого реагування в разі пожежі необхідна наявність протипожежного обладнання, такого як Вогнегасники, на робочому місці.

5. Розвиток людських ресурсів: співробітники повинні бути навчені діяти в разі пожежі і мати можливість використовувати Протипожежне обладнання .

Забезпечити безпечне встановлення та обслуговування електричного обладнання та систем. (ДСТУ EN 60204-1:2010).

Зберігайте легкозаймисті та небезпечні матеріали безпечно та відповідно до нормативних вимог. ДСТУ ISO 6309:2007. Пожежна безпека. Загальні вимоги.

Встановити засоби протипожежного захисту та пожежогасіння відповідно до вимог нормативних документів. (ДСТУ Б.В.2.6-173:2010).

Всі ці правила і вимоги, що стосуються охорони праці та пожежної безпеки, спрямовані на забезпечення безпеки працівників і запобігання можливих негативних наслідків при роботі з електроприладами.

ВИСНОВКИ

В кваліфікаційні роботі була успішно створена система для обробки метеоданих портативної метеостанції. Всі поставлені задачі, визначені у технічному завданні виконані. Розроблена система демонструє надійність та ефективність в отримані та передачі метеоданих що спрощує контроль метрологічних даних для власного використання.

В процесі розробки було враховано повністю технічне завдання та здійснено повне його виконання.

Був проведений аналіз впливу метеоданих в житті людини та їхнє використання в інших сферах діяльності. Для поширення даних в загальний доступ були використанні сервіси які надають можливість це зробити. Для власного використання був створений Telegram Bot, доступ до якого можна отримати в будь-який час.

Було проведе ретельне тестування яке дало змогу виявити недоліки перед публікацією роботи. Результатом є стабільна система яка відповідає всім вимогам.

Отже, в бакалаврській роботі було розроблено систему обробки метеоданих портативної метеостанції на базі ESP8266.

Результати дослідження та розробки можуть бути використані для подальших наукових робіт. Система дуже варіативна тому має можливість змінюватися і покращуватися в залежності від сфери її використання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Arduino IDE[Електронний ресурс] - https://www.arduino.cc/en/software
2. Telegram API[Електронний ресурс] - https://core.telegram.org/
3. Narodmon[Електронний ресурс] - https://eu.narodmon.com/
4. Dweet[Електронний ресурс] - https://dweet.io/
5. BMP280[Електронний ресурс] - https://www.arduino.cc/reference/en/libraries/adafruit-bmp280-library/
6. BH1750[Електронний ресурс] - https://www.arduino.cc/reference/en/libraries/bh1750/
7. UniversalTelegramBot[Електронний ресурс] - https://github.com/witnessmenow/Universal-Arduino-Telegram-Bot
8. ESP8266[Електронний ресурс] - https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WiFi
9. ДСТУ EN 60204-1:2015 Безпечність машин. Електрообладнання машин. Частина 1. Загальні вимоги (EN 60204-1:2006; A1:2009; AC:2010, IDT). Зі зміною
10. ДСТУ ISO 6309:2007. Пожежна безпека. Загальні вимоги
11. The most popular HTML, CSS, and JS library in the world. [Електронний ресурс] - https://getbootstrap.com/docs/4.6/getting-started/introduction/
12. Термопара [Електронний ресурс] - https://uk.wikipedia.org/wiki/%D0%A2%D0%B5%D1%80%D0%BC%D0%BE%D0%BF%D0%B0%D1%80%D0%B0
13. Терморезистор [Електронний ресурс] - https://uk.wikipedia.org/wiki/%D0%A2%D0%B5%D1%80%D0%BC%D0%BE%D1%80%D0%B5%D0%B7%D0%B8%D1%81%D1%82%D0%BE%D1%80
14. ДСТУ 3008-2015. Документація. Звіти у сфері науки і техніки. Структура та правила оформлювання.
15. ДСТУ Б В.2.5-82:2016 Електробезпека в будівлях і спорудах. Вимоги дозахисних заходів від ураження електричним струмом.

ДОДАТКИ

Додаток А. Функціональна схема

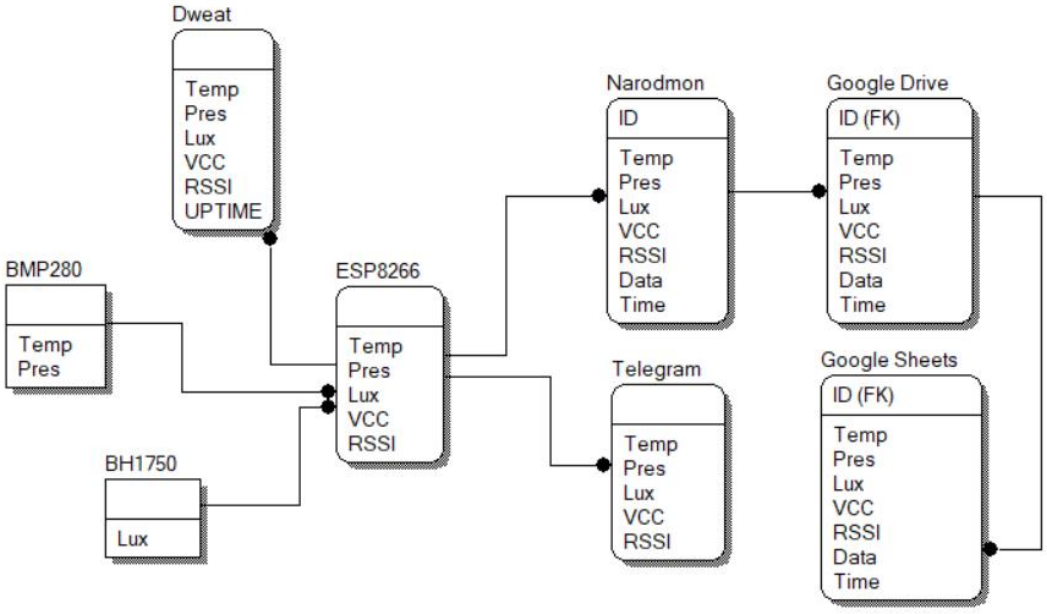

Рисунок А.1 – Функціональна схема

Додаток Б. Інтерфейс користувача

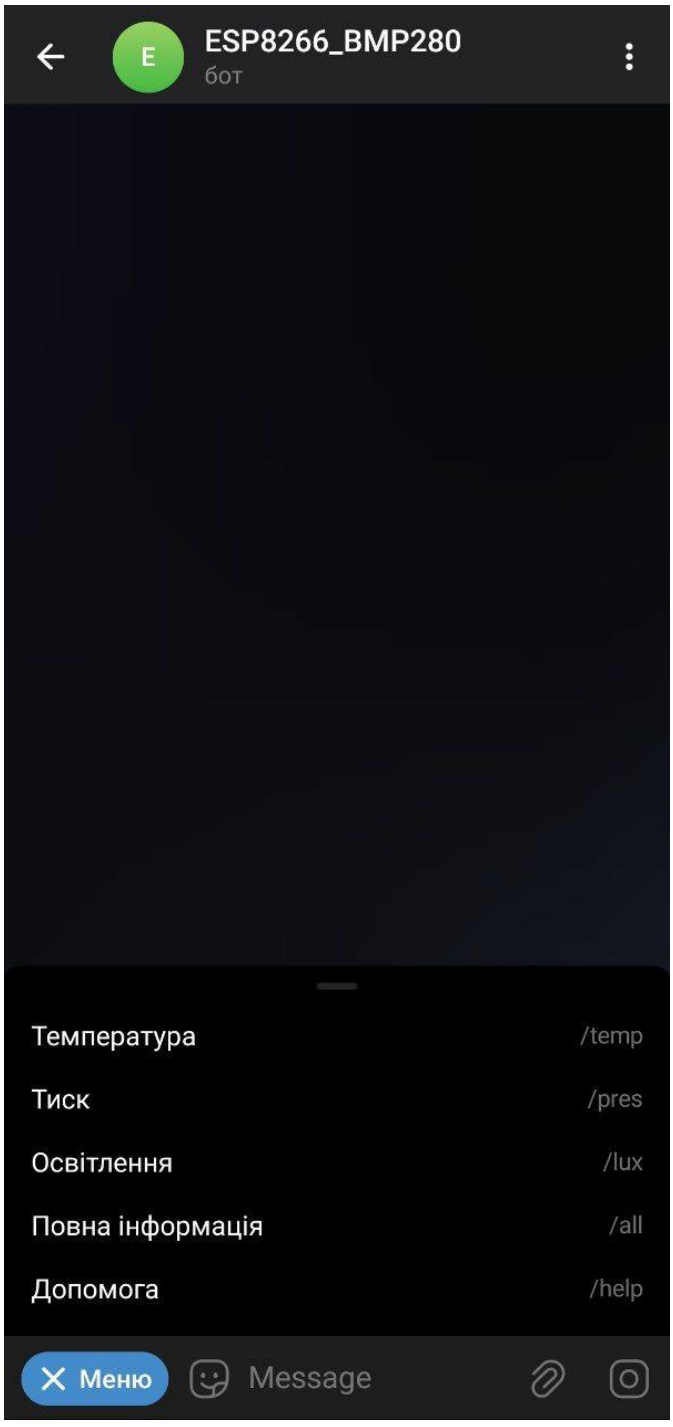

Рисунок Б.1 – Головне меню бота

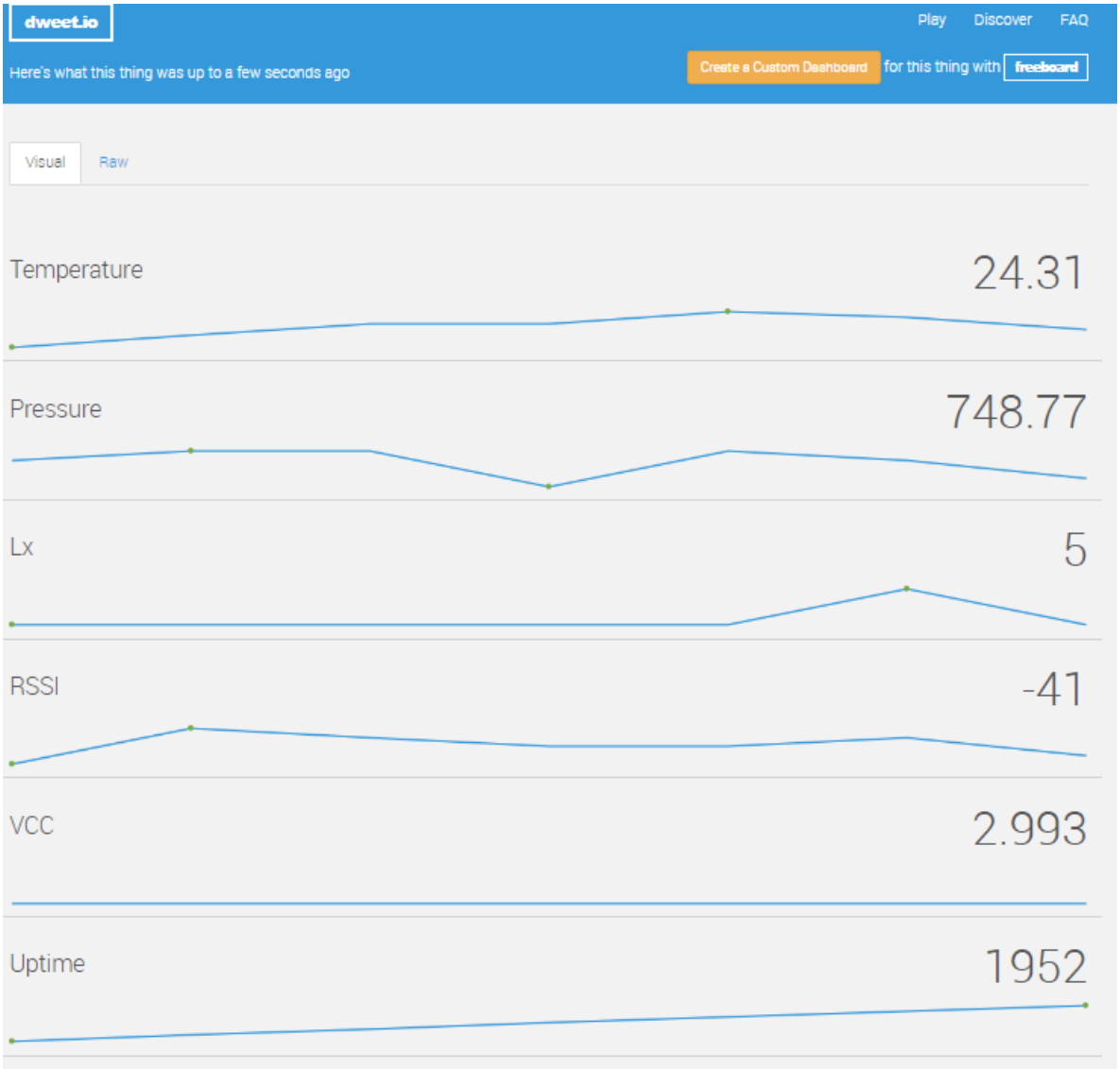

Рисунок Б.2 – Відображення метеоданих в сервісі Dweet.io

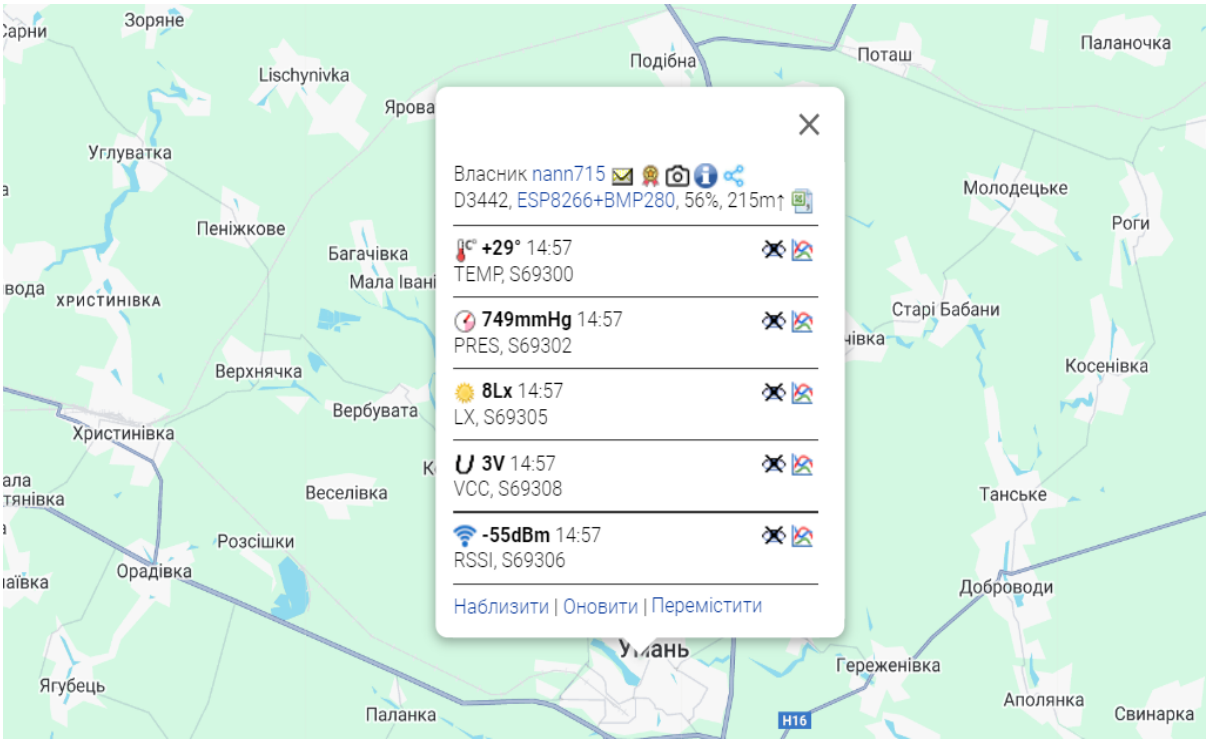

Рисунок Б.3 – Відображення метеоданих в сервісі Narodmon

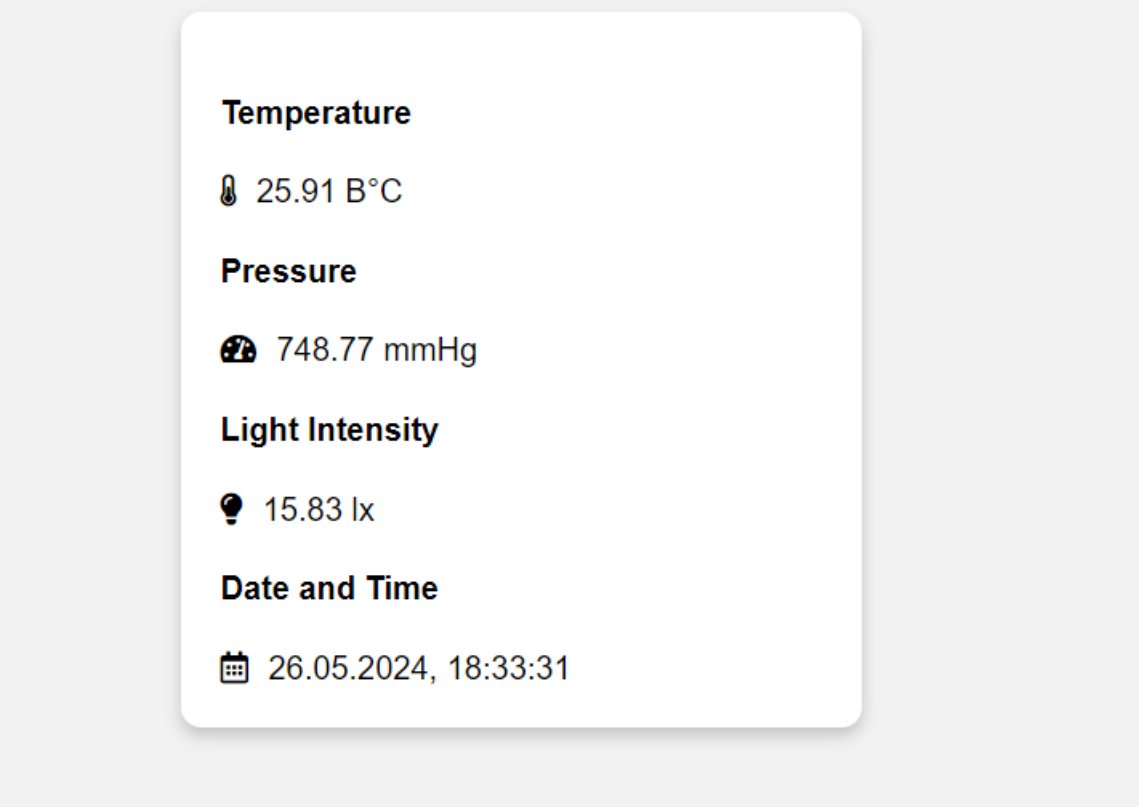

Рисунок Б.4 – Відображення метеоданих на веб – сторінці

Додаток В. Програмний код

```
#include <ESP8266WiFi.h>
#include <Wire.h>
#include "WiFiClient.h"
#include "ESP8266WebServer.h"
#include "ESP8266HTTPClient.h"
#include "ESP8266mDNS.h"
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP280.h>
#include <BH1750.h>
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>
#include <ArduinoJson.h>

ADC_MODE(ADC_VCC);
Adafruit_BMP280 bmp; // I2C
BH1750 lightMeter;
WiFiClient client;
ESP8266WebServer server(80);

#define host "eu.narodmon.com"
#define httpPort 8283
#define dweet "dweet.io"
#define dweetPort 80
#define interval 5000
#define BOT_TOKEN "6561845762:AAFul7SqNUqrJwoxJeY8SibZ9rGgM3PZ4is"

X509List cert(TELEGRAM_CERTIFICATE_ROOT);
WiFiClientSecure secured_client;
```

```
UniversalTelegramBot bot(BOT_TOKEN, secured_client);

float vcc, T, P, L;
int rssi;
unsigned long lasttime;
unsigned long lastNarodmonSendTime = 0;
const unsigned long narodmonSendInterval = 5 * 60 * 1000; // 5 хвилин у мілісекундах

String generateWebpage() {
  String webpage = "<!DOCTYPE html>";
  webpage += "<html>";
  webpage += "<head>";
  webpage += "<link rel=\"stylesheet\" href=\"https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.4/css/all.min.css\">";
  webpage += "<style>";
  webpage += "body {";
  webpage += "  font-family: Arial, Helvetica, sans-serif;";
  webpage += "  background-color: #f2f2f2;";
  webpage += "}";
  webpage += ".card {";
  webpage += "  background-color: white;";
  webpage += "  border-radius: 10px;";
  webpage += "  box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);";
  webpage += "  transition: 0.3s;";
  webpage += "  width: 300px;";
  webpage += "  margin: auto;";
  webpage += "  padding: 20px;";
  webpage += "}";
  webpage += ".container {";
```

```
webpage += "  text-align: left;";
webpage += "}";
webpage += ".icon {";
webpage += "  float: left;";
webpage += "  margin-right: 10px;";
webpage += "}";
webpage += "</style>";
webpage += "<script>";
webpage += "function updateTime() {";
webpage += "  var now = new Date();";
webpage += "  var dateElement = document.getElementById('date-time');";
webpage += "  dateElement.innerHTML = now.toLocaleString();";
webpage += "}";
webpage += "setInterval(updateTime, 1000);"; // Оновлюємо кожну секунду
webpage += "function fetchData() {";
webpage += "  fetch('/data')";
webpage += "    .then(response => response.json())";
webpage += "    .then(data => {";
webpage += "                document.getElementById('temp').innerText = data.temperature.toFixed(2) + ' °C';";
webpage += "                 document.getElementById('pres').innerText = data.pressure.toFixed(2) + ' mmHg';";
webpage += "                 document.getElementById('lux').innerText = data.light.toFixed(2) + ' lx';";
webpage += "    });";
webpage += "}";
webpage += "setInterval(fetchData, 1000);"; // Оновлюємо кожну секунду
webpage += "</script>";
webpage += "</head>";
```

```
  webpage += "<body onload=\"updateTime(); fetchData();\">";
  webpage += "<div class=\"card\">";
  webpage += "  <div class=\"container\">";
  webpage += "   <h4><b>Temperature</b></h4>";
  webpage += "      <div class=\"icon\"><i class=\"fas fa-thermometer-half\"></i></div>";
  webpage += "   <span id=\"temp\">Loading...</span><br>";
  webpage += "   <h4><b>Pressure</b></h4>";
  webpage += "      <div class=\"icon\"><i class=\"fas fa-tachometer-alt\"></i></div>";
  webpage += "   <span id=\"pres\">Loading...</span><br>";
  webpage += "   <h4><b>Light Intensity</b></h4>";
  webpage += "      <div class=\"icon\"><i class=\"fas fa-lightbulb\"></i></div>";
  webpage += "   <span id=\"lux\">Loading...</span><br>";
  webpage += "   <h4><b>Date and Time</b></h4>";
  webpage += "      <div class=\"icon\"><i class=\"far fa-calendar-alt\"></i></div>";
  webpage += "   <span id=\"date-time\"></span>";
  webpage += "  </div>";
  webpage += "</div>";
  webpage += "</body>";
  webpage += "</html>";
  return webpage;
 }

 void handleData() {
  StaticJsonDocument<200> data;

  data["temperature"] = T;
```

```
  data["pressure"] = P;
  data["light"] = L;

  String jsonData;
  serializeJson(data, jsonData);

  server.send_P(200, "application/json", jsonData.c_str());
}

void handleWebpage() {
  server.send(200, "text/html", generateWebpage());
}

void connectToPreferredWiFi() {
  // Введіть дані вашої перевіреної мережі
  const char* preferredSSID = "vallyk97";
  const char* preferredPassword = "48973592";

  Serial.println("Спроба підключення до перевіреної мережі Wi-Fi...");

  WiFi.begin(preferredSSID, preferredPassword);

  unsigned long startAttemptTime = millis(); // Початок спроби підключення
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
    if (millis() - startAttemptTime > 10000) { // Якщо пройшло більше 10 секунд, вважаємо це невдалою спробою
      Serial.println("Не вдалося підключитися до перевіреної мережі Wi-Fi.");
```

```
        return; // Виходимо з функції, оскільки не вдалося підключитися до переданої мережі
      }
    }

    // Якщо підключено успішно, виводимо IP-адресу та продовжуємо виконання програми
    Serial.println("");
    Serial.println("Успішно підключено до перевіреної мережі Wi-Fi!");
    Serial.println("Ваш IP-адреса " + WiFi.localIP().toString());
  }

  void connectWiFi() {
    // Спочатку спробуємо підключитися до заданої мережі
    connectToPreferredWiFi();

    // Якщо не вдалося підключитися до заданої мережі, виконаємо процес сканування та підключення до іншої мережі
    if (WiFi.status() != WL_CONNECTED) {
      Serial.println("Не вдалося підключитися до перевіреної мережі Wi-Fi. Запускаємо процес сканування мереж...");
      while (true) {
        boolean wifiFound = false;
        int i, n;

        WiFi.mode(WIFI_STA);
        WiFi.disconnect();
        delay(100);
        Serial.println("Проведення сканування мереж Wi-Fi...");
        int nbVisibleNetworks = WiFi.scanNetworks();
```

```
Serial.println(F("Сканування завершено!"));
if (nbVisibleNetworks == 0) {
  Serial.println("Мережі Wi-Fi не знайдено!");
  continue;
}
Serial.print(nbVisibleNetworks);
Serial.println(" мереж знайдено");

// Виводимо список доступних мереж
for (i = 0; i < nbVisibleNetworks; ++i) {
  Serial.println(WiFi.SSID(i));
}

// Введення імені мережі та пароля через Serial Monitor
Serial.println("Введіть SSID мережі, до якої ви хочете підключитися:");
while (!Serial.available()) {} // Очікування вводу
String ssidInput = Serial.readStringUntil('\n');
Serial.println("Введіть пароль:");
while (!Serial.available()) {} // Очікування вводу
String passwordInput = Serial.readStringUntil('\n');

// Перевірка, чи знайдено введену мережу в списку доступних
for (i = 0; i < nbVisibleNetworks; ++i) {
  if (WiFi.SSID(i) == ssidInput) {
    wifiFound = true;
    break;
  }
}
if (!wifiFound) {
  Serial.println("Введена мережа не знайдена!");
```

```
        continue;
      }

      // Підключення до введеної мережі
      Serial.print("Підключення до ");
      Serial.println(ssidInput);
      WiFi.begin(ssidInput.c_str(), passwordInput.c_str());
      unsigned long startAttemptTime = millis(); // Початок спроби підключення
      while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
        if (millis() - startAttemptTime > 10000) { // Якщо пройшло більше 10 секунд, вважаємо це невдалою спробою
          Serial.println("Час очікування підключення вичерпано!");
          break; // Виходимо з циклу
        }
      }
      if (WiFi.status() == WL_CONNECTED) { // Якщо підключено успішно, виходимо з циклу
        Serial.println("");
        Serial.println("Успішно підключено до мережі Wi-Fi!");
        Serial.println("Ваш IP-адреса " + WiFi.localIP().toString());
        break;
      } else { // Якщо підключення не вдалося
        Serial.println("Помилка підключення, повторна спроба...");
      }
    }
  }
}
```

```
void checkWiFiConnection() {
  if (WiFi.status() != WL_CONNECTED) {
    Serial.println("Втрачено з'єднання з Wi-Fi. Повторна спроба підключення...");
    connectWiFi(); // Повторне підключення до Wi-Fi
  }
}

void setupTelegram() {
  Serial.println("Підключення Telegram bot...");
  secured_client.setTrustAnchors(&cert); // Додати кореневий сертифікат для api.telegram.org

  configTime(0, 0, "pool.ntp.org"); // Отримати час UTC через NTP
  time_t now = time(nullptr);
  while (now < 24 * 3600) {
    delay(100);
    now = time(nullptr);
  }

  Serial.println("Telegram bot підключено!");

}

void handleNewMessages(int numNewMessages) {
  Serial.print("handleNewMessages ");
  Serial.println(numNewMessages);

  for (int i = 0; i < numNewMessages; i++) {
```

```
    String text = bot.messages[i].text;

    if (text == "/temp") {
      String message = "Температура: " + String(T) + "°C\n";
      bot.sendMessage(bot.messages[i].chat_id, message, "");
    } else if (text == "/pres") {
      String message = "Тиск: " + String(P) + " мм. рт. ст.\n";
      bot.sendMessage(bot.messages[i].chat_id, message, "");
    } else if (text == "/lux") {
      String message = "Яскравість освітлення: " + String(L) + " люкс\n";
      bot.sendMessage(bot.messages[i].chat_id, message, "");
    } else if (text == "/all") {
      String message = "Температура: " + String(T) + "°C\n";
      message += "Тиск: " + String(P) + " мм. рт. ст.\n";
      message += "Яскравість освітлення: " + String(L) + " люкс\n";
      message += "Рівень напруги: " + String(vcc) + " V\n";
      message += "Рівень сигналу: " + String(WiFi.RSSI()) + "dBm\n";
      bot.sendMessage(bot.messages[i].chat_id, message, "Markdown");
    } else if (text == "/help") {
      String welcome = "ESP8266 WiFi Telegram !\n";
      welcome += "Для отримання даних оберіть потрібну команду:\n\n";
      welcome += "/temp : Температура\n";
      welcome += "/pres : Тиск\n";
      welcome += "/lux : Яскравість освітлення\n";
      welcome += "/all : Загальні значення\n";
      bot.sendMessage(bot.messages[i].chat_id, welcome, "Markdown");
    }
  }
}
```

```
void measure() {
  Wire.begin();
  vcc = ((float)ESP.getVcc()) / 1000.0;
  rssi = WiFi.RSSI();
  T = bmp.readTemperature();
  P = bmp.readPressure();
  P = P / 133.322;
  Serial.print("T= ");
  Serial.print(T, 2);
  Serial.print(" C; P= ");
  Serial.print(P , 2);
  Serial.print(" mmHg; VCC= ");
  Serial.print(vcc, 3);
  Serial.print(" V, RSSI= ");
  Serial.print(WiFi.RSSI());
  Serial.println(" dBm ");
}

void luxt() {
  Wire.begin();
  lightMeter.begin();
  L = lightMeter.readLightLevel();
  Serial.print("L= ");
  Serial.print(L, 2);
  Serial.print(" Lx ");
}

void narodmonSend() {
  unsigned long currentTime = millis();
```

```
// Перевіряємо, чи минуло вже narodmonSendInterval мілісекунд
if (currentTime - lastNarodmonSendTime >= narodmonSendInterval) {
  if (!client.connect(host, httpPort)) {
    Serial.println("Connection to narodmon failed");
    return;
  }
  Serial.println("Sending data on narodmon ...");
  client.print("#");
  client.print(WiFi.macAddress());
  client.print("#");
  client.print("ESP8266+BMP280");
  client.println();
  client.print("#temp#");
  client.println(T);
  client.print("#pres#");
  client.println(P);
  client.print("#lx#");
  client.println(L);
  client.print("#rssi#");
  client.println(rssi);
  client.print("#vcc#");
  client.println(vcc);
  client.println("##");
  delay(10);
  Serial.print("Requesting: ");
  while (client.available()) {
    String line = client.readStringUntil('\r');
    Serial.println(line);
  }
  client.stop();
```

```
    Serial.println("narodmon ok!");

    // Оновлюємо час останнього надсилання
    lastNarodmonSendTime = currentTime;
  }
 }

 void dweetSend() {
  if (!client.connect(dweet, dweetPort)) {
    Serial.println("connection dweet failed");
    return;
  }
  client.print(String("GET  /dweet/for/ESP8266_BMP280?Temperature=") +
String(T, 2)
          + "&Pressure=" + String(P, 2)
          + "&Lx=" + String(L, 2)
          + "&RSSI=" + String(rssi)
          + "&VCC=" + String(vcc, 3)
          + "&Uptime=" + String(millis() / 1000)
          + " HTTP/1.1\r\n" +
          "Host: " + dweet + "\r\n" +
          "Connection: close\r\n\r\n");
  delay(10);
  while (client.available()) {
    String line = client.readStringUntil('\r');
    Serial.print(line);
  }
  client.stop();
  Serial.println("dweet ok!");
 }
```

```
void setup() {
  Serial.begin(115200);
  bmp.begin(0x76);
  connectWiFi();
  setupTelegram();
  Serial.print("MAC адреса: ");
  Serial.println(WiFi.macAddress());
  server.on("/", HTTP_GET, handleWebpage);
  server.on("/data", HTTP_GET, handleData);
  server.begin();
}

void loop() {
  if (millis() > lasttime + interval) {
    lasttime = millis();
    measure();
    luxt();
    checkWiFiConnection();
    narodmonSend();
    dweetSend();
    int numNewMessages = bot.getUpdates(bot.last_message_received + 1);
    while (numNewMessages) {
      Serial.println("got response");
      handleNewMessages(numNewMessages);
      numNewMessages = bot.getUpdates(bot.last_message_received + 1);
    }
    lasttime = millis();
  }
  server.handleClient();}
```