МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ПРАТ «ПВНЗ «ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ»

Кафедра комп'ютерної інженерії

ДО ЗАХИСТУ ДОПУЩЕНА

Зав.кафедри _____

д.т.н., проф. Переверзєв А.В.

МАГІСТЕРСЬКА ДИПЛОМНА РОБОТА

ОРГАНІЗАЦІЯ ВБУДОВАНИХ СИСТЕМ ЕКО-МОНІТОРИНГУ З ВИКОРИСТАННЯМ КОМП'ЮТЕРНИХ МЕРЕЖ

Виконав
ст. гр. КІ-210М _____ З. Каціа

Керівник
_____ С. О. Сабанов

Запоріжжя

2022

ПРАТ «ПВНЗ «ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ»

Кафедра комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Зав. кафедрою
д.т.н., професор
Переверзєв А.В.

_____

15.03.2022 р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ ДИПЛОМНУ РОБОТУ

Студенту гр. *КІ – 210М*, спеціальності *«Комп'ютерна інженерія»*

Кація Заза

1.Тема: *Організація вбудованих систем еко-моніторингу з використанням комп'ютерних мереж*

затверджена наказом по інституту 06.2-15-3 від 15 березня 2022 р.

2. Термін здачі студентом закінченої роботи: 19 червня 2022 р.

3. Перелік питань, що підлягають розробці

1. Провести огляд інформаційних джерел стосовно вбудованих систем.

2. Розглянути сучасні методи моніторингу стану навколишнього середовища

3. Виконати огляд найбільш популярних мікроконтролерів

4. Здійснити вибір та огляд сенсорів та датчиків, що здійснюють моніторинг навколишнього середовища.

5. Розробити структуру системи моніторингу, аналізу та передачі даних стану навколишнього середовища

6. Протестувати систему в різних умовах

7. Проаналізувати отримані результати

8. Оформити звіт за результатами роботи

Дата видачі завдання: 15.03.2022 р.

Керівник магістерської роботи _____ С.О. Сабанов
(підпис) (прізвище та ініціали)

Завдання отримав до виконання _____ З. Каціа
(підпис студента) (прізвище та ініціали)

РЕФЕРАТ

Магістерська дипломна робота містить 120 сторінок, 57 рисунків, 2 таблиці, 2 додатки, 45 посилань.

Об'єкт розробки – технологій роботи з даними за допомогою вбудованих систем та мережевої взаємодії.

Предмет розробки – проект системи еко-моніторингу з використанням комп’ютерних мереж різного рівня.

Мета роботи – розробка методології проектування систем моніторингу екологічних параметрів середовища задля їхньої кластеризації та взаємодії з користувачами за допомогою мережевих технологій.

Методи дослідження – аналіз, порівняння, узагальнення, декомпозиція, проектування, моделювання та програмування системи.

У дипломній роботі було розглянуто сучасний стан проблематики систем моніторингу навколишнього середовища різного ореолу використання та взаємодії.

Здійснено огляд програмно-апаратних засобів та технологій розробки, що можуть ефективно реалізовувати поставлені задачі у режимі реального часу.

Запропоновано багатошарову кластерну систему на базі інтеграції пристроїв за допомогою технології мережевої взаємодії. Здійснено практичну реалізацію запропонованої системи з відповідними ефективними результатами.

EMBEDDED SYSTEM, SENSORS, MCU, NETWORK, LAN, WAN, MQTT, WIRELESS SENSOR NETWORK

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Скорочення	**Повна назва/ Пояснення**
АЕС	Атомна електрична станція
АЦП	Аналого-цифровий перетворювач
БСМ	Бездротова сенсорна мережа
Гідромет	Державний департамент гідрометеорології
ГРЕ	Гамма радіаційна експозиція
ДСМД	Державна система моніторингу довкілля
ЕАД	Європейське екологічне агентство
ЄАД	Європейське агентство довкілля
ЄАОС	Європейське агентство довкілля
ЄС	Європейська спільнота
ООН	Організація об'єднаних націй
ОСРЧ	Операційна система реального часу
ПК	Персональний комп'ютер
РЧ	Радіочастотний
СКОПЕ	Науковий комітет з проблем навколишнього середовища
ШІМ	Широтно-імпульсна модуляція
ЮНЕП	United Nations Environment Programme
ЮНЕСКО	United Nations Educational, Scientific and Cultural Organization
ADU	Application Data Unit
ALU	Arithmetic logic unit
AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
CISC	Complex instruction set computer

CMOS	Complementary metal–oxide–semiconductor
CMT	Cable Mode Transition
CoAP	Constrained Application Protocol
CPU	Central Processing unit
CRC16	Cyclic redundancy check
DIP	Dual In-line Package
EDA	Electronic design automation
EEA	European Environment Agency
EEPROM	Electrically Erasable Programmable Read-Only Memory
Eionet	Європейська екологічна інформаційна та спостережна мережа
FIFO	First in first out
FTP	File Transfer Protocol
GEN	Global Ekolabelling Network
GPIO	General-purpose input/output,
HDMI	High Definition Media Interface
HTTP	Hyper text transfer protocol
I/O	input output
I2C	Inter-Integrated Circuit
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
ISO	International Organization for Standardization
JMS	Java Message Service
LAN	Local Access Network
LRC8	Longitudinal redundancy check
M2M	Machine-to-Machine
MAC	Media Access Control
MCU	Microcontroller Unit
MIPS	Microprocessor without Interlocked Pipeline Stages

MQTT	Message Queuing Telemetry Transport
OMS	Online Monitoring System
PADS	Personal Automated Design Solutions
PCB	Printed circuit board
PDU	Protocol data unit
QSPI	Quad serial peripheral interface
RAM	Random Access Memory
RF	radio frequincy
RISC	reduced instruction set computer
RTU	Remote Terminal Unit
SASL	Simple Authentication and Security Layer
SMD	Surface Mounted Device
SMTP	Simple Mail Transfer Protocol
SoC	System on Crystal
SPI	Serial Peripheral Interface
SRAM	static random access memory
SSL	Secure Sockets Layer
TCP/IP	Transfer Control protocol / internet protocol
TLS	Transport Layer Security
UART	Universal Asynchronous Receiver-Transmitter,
UDP	User Datagram Protocol
URL	Uniform Resource Locator
USB	Universal Serial Bus
USI	Universal Serial Interface
WAN	Wide Area Network
Wi-Fi	Wireless Fidelity
WSN	Wireless Sensor Networks
WWF	World Wildlife Fund
XMPP	Extensible Messaging and Presence Protocol

ВСТУП

У наш час питання екологічного стану оточуючого середовища є вельми актуальною задачею. Надлишкові викиди шкідливих газів, світовий підйом температурних показників та інші техногенні фактори можуть спричинити погіршення стану біосфери планети та негативно вплинути на життя людства [1]. Тому системи спостереження, обробки та аналізу даних про стан екології є важливими задачами, які вирішуються багатьма світовими науковими установами та вченими різними шляхами, у тому числі і з використанням комп'ютерного інжинірингу [2].

Одним з механізмів ефективної організації системи еко-моніторингу є використання вбудованих систем (embedded system). За матеріалами вікіпедії це комбінація апаратного та програмного забезпечення, можливо, з механічними або іншими частинами, призначена для виконання окремої функції [3]. Саме тому правильна та чітка організація її механізмів можуть дати ефективну систему контролю, а використання комп'ютерних мереж різного рівня можуть надати цим системам легкого масштабування (від локального осередку до світового масштабу).

Однією за важливих задач є розробка системи моніторингу навколишнього середовища, в якій завдяки використанню різних технологій передачі даних досягнуто ряд переваг перед існуючими системами з позиції енергоспоживання та вартості обладнання. У роботі пропонується ряд організаційних заходів щодо кластеризації даних систем, виділення ефективних підсистем, їхню взаємодію для легкого масштабування в цілому.

На думку автора, перспективним напрямком дослідження є використання та організація у якості першої підсистеми зчитування та обробки даних показників бездротових сенсорних мереж (БСМ) або Wireless Sensor Networks (WSN) – розподілені мережі, що складаються з маленьких сенсорних вузлів, з інтегрованими функціями моніторингу навколишнього

середовища, обробки і передачі даних. Сенсорні мережі спеціального призначення характеризуються: обмеженістю ресурсів вузлів мережі (по продуктивності процесора, пам`яті, потужності передавача, енергії батареї), обмеженою дальністю та пропускною спроможністю каналів радіозв'язку між вузлами концентрацією трафіка біля шлюзу та ін [4]. Дана система має задовільні показники у використанні локального значення.

Однак існує ряд недоліків: мережа повинна працювати в особливих умовах, враховуючи можливість застосування деструктивних чинників, що призводить до істотного уповільнення доставки даних і зниження продуктивності, зменшення часу функціонування БСМ тощо.

Таким чином, щоб якісно ліквідувати недоліки та масштабувати систему, треба використовувати механізми комп'ютерних мереж більш високого рівня організації. Виходом у цьому може служити MQTT протокол, що ефективно використовується для обміну невеликими повідомленнями у мережах різного рівня [5]. Це спрощений мережевий протокол, що працює на TCP/IP. Використовується для обміну повідомленнями між пристроями за принципом видавець-підписник.

РОЗДІЛ 1
ОГЛЯД СИСТЕМ МОНІТОРИНГУ ТА КОНТРОЛЮ ЕКОЛОГІЧНОГО СЕРЕДОВИЩА

Екологічний моніторинг (моніторинг навколишнього середовища) — комплексні спостереження за станом навколишнього середовища, в тому числі компонентів природних середовищ, природних екологічних систем, для процесів, явлень, оцінка і прогноз змін стану навколишнього середовища [6].

Термін «моніторинг» спочатку з'явився в рекомендаціях спеціальної комісії СКОПЕ (науковий комітет з проблем навколишнього середовища) при ЮНЕСКО в 1971 р., а потім питання проведення екологічного моніторингу розглядалися на проходженні в 1972 р. у Стокгольмі конференції Організацій об'єднаних націй по проблемам навколишнього середовища. Пропозиції щодо екологічного моніторингу були озвучені в 1972 р. перед конференцією ООН членами спеціальної комісії Наукового комітету з проблем навколишнього середовища Міжнародного ради наукових союзів, створеної американськими вченими Гільбертом Уайтом і Томасом Малоном. Під екологічним моніторингом їх понімали «систематичні спостереження за станом навколишнього середовища, можливі зміни у зв'язку з антропогенною діяльністю, контроль таких змін і проведення заходів з упорядкування навколишнього середовища».

Можна виділити класифікацію видів екологічного моніторингу.

За просторовим принципом - виділяються: точковий, локальний, регіональний, національний та глобальний моніторинг. Останній передбачає екологічні дослідження взаємодії людини та природи у масштабах усієї біосфери. Національний, як правило, має на увазі організацію моніторингу в межах однієї держави. Досить складно однозначно визначити масштаби регіонального моніторингу. Локальний моніторинг включає вивчення

простору одного джерела за впливу сукупності підприємств промислової зони, муніципального освіти (міста, району).

По об'єкту стеження: фоновий (базовий), імпактний, тематичний, територіальний, акваторіальний.

У межах фонового моніторингу ведуться дослідження, створені задля виявлення природних закономірностей зміни природних компонентів і комплексів. Під імпактним моніторингом розуміється спостереження, оцінка та прогноз стану природного середовища в районах розташування небезпечних та потенційно небезпечних (АЕС) джерел антропогенного впливу. Тематичний моніторинг – моніторинг природних компонентів, об'єктів, наприклад, лісових або природних територій, що особливо охороняються. Значною мірою за явищами та способами вивчення відрізняється мережа спостережень на суші та у водному середовищі.

За природними компонентами виділяється геологічний, атмосферний, гідрологічний, геофізичний, ґрунтовий, лісовий, біологічний, геоботанічний, зоологічний. Моніторинг атмосферного повітря – система спостережень за станом атмосферного повітря, його забрудненням і за природними явищами, що відбуваються в ньому, а також оцінка та прогноз стану атмосферного повітря, його забруднення. Аналогічно можна визначити інші компонентні моніторинги.

За організаційними особливостями спостереження виділяють міжнародний, державний, місцевий, громадський та відомчий моніторинги. До міжнародного належать системи оцінки прогнозу, організовані міждержавними організаціями глобального характеру, наприклад, ООН, ЮНЕСКО, ЮНЕП тощо. Моніторинг може здійснюватися державними та муніципальними службами. Зрештою, промислові та сільськогосподарські підприємства, галузі ведуть відомчий моніторинг. Екологічний моніторинг може організувати окремі фізичні особи, громадські об'єднання громадян.

Розрізняються такі підсистеми екологічного моніторингу, як: геофізичний моніторинг (аналіз даних із забруднення, мутності атмосфери,

досліджує метеорологічні та гідрологічні дані середовища, а також вивчає елементи неживої складової біосфери, у тому числі об'єктів, створених людиною); кліматичний моніторинг (служба контролю та прогнозу коливань кліматичної системи. Охоплює ту частину біосфери, яка впливає на формування клімату: атмосферу, океан, крижаний покрив та ін. Кліматичний моніторинг тісно змикається з гідрометеорологічними спостереженнями.); біологічний моніторинг (заснований на спостереженні за реакцією живих організмів на забруднення довкілля); моніторинг здоров'я населення (система заходів щодо спостереження, аналізу, оцінки та прогнозу стану фізичного здоров'я населення) та ін.

У загальному вигляді процес екологічного моніторингу можна представити схемою: навколишнє середовище (або конкретний об'єкт навколишнього середовища) -> вимірювання параметрів різними підсистемами моніторингу -> збирання та передача інформації -> обробка та подання даних (формування узагальнених оцінок), прогнозування. Система екологічного моніторингу варта обслуговування систем управління якістю довкілля (далі «система управління»). Інформація про стан навколишнього середовища, отримана в системі екологічного моніторингу, використовується системою управління для запобігання або усунення негативної екологічної ситуації, для оцінки несприятливих наслідків зміни стану навколишнього середовища, а також для розробки прогнозів соціально-економічного розвитку, розробки програм у галузі екологічного розвитку та охорони довкілля.

У системі управління можна виділити три підсистеми: прийняття рішення (спеціально уповноважений державний орган), управління виконанням рішення (наприклад, адміністрація підприємств), виконання рішення за допомогою різних технічних чи інших засобів.

Підсистеми екологічного моніторингу різняться на об'єктах спостереження. Оскільки компонентами навколишнього середовища є повітря, вода, мінерально-сировинні та енергетичні ресурси, біоресурси,

ґрунти та ін., то виділяють відповідні підсистеми моніторингу. Однак підсистеми моніторингу не мають єдиної системи.

1.1 Стан функціонування системи моніторингу довкілля

Проблема захисту навколишнього середовища набуває пріоритетний характер в міжнародних відносинах, оскільки від переходу до нового типу розвитку, розумного з природоохоронної та гуманістичної точки зору, залежить збереження життя на землі [7].

1.1.1 Проблематика питання на державному рівні України

З метою мінімізації наслідків техногенної діяльності людини багато демократичних та цивілізованих країн вводять спеціальні законодавчі програми щодо моніторингу довкілля. Серед них перебуває й Україна, яка запровадила державну систему моніторингу довкілля. Законом України „Про охорону навколишнього природного середовища" (ст.20, 22) передбачено створення державної системи моніторингу довкілля (далі – ДСМД) та проведення спостережень за станом навколишнього природного середовища, рівнем його забруднення. Виконання цих функцій покладено на Мінприроди та інші центральні органи виконавчої влади, які є суб'єктами державної системи моніторингу довкілля, а також підприємства, установи та організації, діяльність яких призводить або може призвести до погіршення стану довкілля [8].

Функціонування ДСМД здійснюється на трьох рівнях, що розподіляються за територіальним принципом:

- загальнодержавний рівень, що охоплює пріоритетні напрямки та завдання моніторингу в масштабах всієї країни;
- регіональний рівень, що охоплює пріоритетні напрямки та завдання в масштабах територіального регіону;

- локальний рівень, що охоплює пріоритетні напрямки та завдання моніторингу в масштабах окремих територій з підвищеним антропогенним навантаженням.

Також згідно за цією програмою основними напрямками екологічного моніторингу є:

1. *Моніторинг якості повітря* здійснюються спостереження за забрудненням атмосферного повітря у 53 містах України на 162 стаціонарних, двох маршрутних постах спостережень та двох станціях транскордонного переносу. Ведуться спостереження за хімічним складом атмосферних опадів та за кислотністю опадів. Програма обов`язкового моніторингу якості атмосферного повітря включає сім забруднюючих речовин: пил, двоокис азоту (NO2), двоокис сірки (SO2), оксид вуглецю, формальдегід (H2CO), свинець та бенз(а)пірен. Деякі станції здійснюють спостереження за додатковими забруднюючими речовинами. Проводиться аналіз наявності забруднюючих речовин в опадах та сніговому покриві.

2. *Моніторинг стану вод суші* - отримуються дані по 46 параметрах, що дають можливість оцінити хімічний склад вод, біогенні параметри, наявність зважених часток та органічних речовин, основних забруднюючих речовин, важких металів та пестицидів. На водних об`єктах проводяться спостереження за хронічною токсичністю води. Визначаються показники радіоактивного забруднення поверхневих вод.

3. *Моніторинг прибережних вод* - проводяться вимірювання від 16 до 26 гідрохімічних параметрів вод та донних відкладів: щомісячні відбори проб та аналіз впливу джерел забруднення, які розташовані на узбережжі; моніторинг скидів з кораблів; забруднення від діяльності з пошуку та видобування нафти, газу і будівельних матеріалів на морському шельфі; нагляд за використанням живих ресурсів моря.

4. *Моніторинг стану ґрунтів* сільськогосподарських земель пестицидами та важкими металами у населених пунктах.

5. *Моніторинг показників біологічного різноманіття* - здійснюється тільки за видами, які представляють промисловий інтерес (дерева, риба, дичина).

6. Моніторинг радіаційного випромінювання - здійснює спостереження за радіоактивним забрудненням атмосфери шляхом щоденних замірів доз гамма-радіаційної експозиції (ГРЕ), осідання радіоактивних частинок з атмосфери та вмісту радіоактивного аерозолю в повітрі.

Для упорядкування процесу обміну інформацією за показниками та термінами надання екологічної інформації між Мінприроди та суб'єктами ДСМД укладено двохсторонні угоди про співробітництво у сфері моніторингу навколишнього природного середовища, до яких розроблені відповідні регламенти обміну екологічною інформацією.

Оперативна моніторингова інформація передається територіальними органами суб'єктів ДСМД до регіональних центрів моніторингу довкілля, або державних управлінь охорони навколишнього природного середовища в регіонах.

Узагальнена аналітична інформація надається міністерствами та відомствами-суб'єктами ДСМД Мінприроди.

Отримані дані передаються до Інформаційно - аналітичного центру Мінприроди та накопичується у банках екологічних даних.

Функціонування Інформаційно-аналітичного центру Мінприроди забезпечує інформаційний обмін з регіональними центрами моніторингу довкілля, суб'єктами державної системи моніторингу довкілля, створення уніфікованого банку екологічних даних, проведення комплексного аналізу стану довкілля, тощо.

Постановою Кабінету Міністрів України від 05.12.2007 № 1376 затверджено Державну цільову екологічну програму проведення моніторингу навколишнього природного середовища.

Програма спрямована на поєднання зусиль усіх суб`єктів системи моніторингу щодо виключення дублювання та включення додаткових

функцій з моніторингу, створення єдиної мережі спостережень після оптимізації її елементів та програм спостережень, вдосконалення технічного, методичного, метрологічного та наукового забезпечення функціонування єдиної мережі спостережень. З метою забезпечення інтеграції інформаційних ресурсів суб`єктів системи моніторингу довкілля передбачено створення та забезпечення функціонування єдиної автоматизованої підсистеми збору, оброблення, аналізу і збереження даних та інформації, отриманих в результаті здійснення моніторингу.

1.1.2 Проблематика питання на державному рівні Грузії

Важка економічна ситуація в Грузії привела до різкого скорочення фінансування моніторингу навколишнього середовища. За минуле десятиліття станцій і обладнання моніторингу виносилися і во багатьох випадках прийшли у неробочий стан. У цілому бюджетні асигнування покривають лише витрати на заробітну плату і мінімальні послуги в закладах, які займаються моніторингом; для проведення серйозного технічного обслуговування та ремонту або для придбання нового обладнання засобів практично не залишається.

Дванадцять регіональних департаментів міністерств навколишнього середовища здійснюють нагляд за самоконтролем компаній, який заснований головним чином на розрахунках балансу енергії та маси, випадків вимірювання фактичних викидів дуже рідко, оскільки відповідне обладнання або засторіло, або його взагалі немає в наявності. Аналогічні проблеми з фінансовими засобами затронули і державні установи, безпосередньо займаються моніторингом, такі як Державний департамент гідрометеорології (Гідромет), який відповідає за збір та аналіз даних про якість поверхневих вод, якості повітря і ґрунту. На даний час Гідромет здійснює безперервний контроль за забрудненням навколишнього повітря тільки в чотирьох містах, відстежуючи всього п'ять забруднюючих речовин. У Державному департаменті геології, який відповідає за мінеральні ресурси, діє всього 30 з

500 станцій моніторингу рівнів ґрунтових вод. Крім того, за пройшли десять років мало, що було зроблено для удосконалення методів роботи, керівничих принципів і протоколів.

Контроль якості даних моніторингу ненадійний [9].

Однак Грузія не стоїть на місці. У Грузії досі не існував набір національних екологічних показників Індикатори розроблені Робочою групою разом з ЄАОС сприяють створенню національного набору індикаторів Грузії. Ці індикатори та рекомендації Робочої Групи з підготовки оціночних доповідей ляжуть в основу для об'єктивної оцінки стану навколишнього середовища та розроблення ефективних заходів для її покращення.

Рекомендації посібника з удосконалення моніторингу навколишнього середовища підприємствами та покращення їх екологічної звітності активно впроваджуються у роботі екологічної інспекції Грузії. Це дозволить посилити контроль за дотриманням підприємствами природоохоронних нормативних актів та покращить збір даних для національних оціночних доповідей з охорони навколишнього середовища [10].

Також ряд учених (Гунія Г.С., Сванидзе З.С. та ін.) постійно займаються питанням моніторингу довкілля та отримують сучасні результати [11].

1.1.3 Неурядові та неофіційні організації з еко-моніторингу

Global Nest — міжнародна асоціація вчених, технологів, інженерів та інших зацікавлених груп, що беруть участь у всіх наукових та технологічних аспектах навколишнього середовища, а також у застосуванні методів, спрямованих на сталий розвиток. Головною метою організації є підтримка та сприяння поширенню інформації про найсучасніші методи для покращення якості життя на основі розробки та застосування технологій та політики, дружньої до навколишнього середовища [12].

Глобальна мережа екологічного маркування (англ. Global Ekolabelling Network (GEN)) — асоціація незалежних організацій із 36 країн, які впроваджують системи екологічного маркування відповідно до добровільного міжнародного стандарту ISO 14024 [13].

Рис. 1.1 – Логотип організації GEN

Екомаркування – це добровільний метод екологічної сертифікації та маркування, який практикується у всьому світі. Екомаркування ідентифікує продукти або послуги, які є екологічно переважними в певній категорії.

На відміну від довільних «зелених» символів або заяв, зроблених постачальниками, члени GEN пропонують надійні етикетки, сертифіковані неупередженою третьою стороною, для продуктів або послуг, які були незалежно визначені відповідними прозорим критеріям екологічного лідерства на основі міркувань життєвого циклу.

Екологічна маркування члена GEN на продукті чи послугі означає, що вони сертифіковані відповідно до науково обґрунтованого стандарту. Вимоги та категорії продуктів можуть відрізнятися, але всі стандарти вирішують численні проблеми довкілля та здоров’я, які можуть включати токсичність, якість повітря, використання енергії та води, переробку, використання природних ресурсів та інші проблемні сфери.

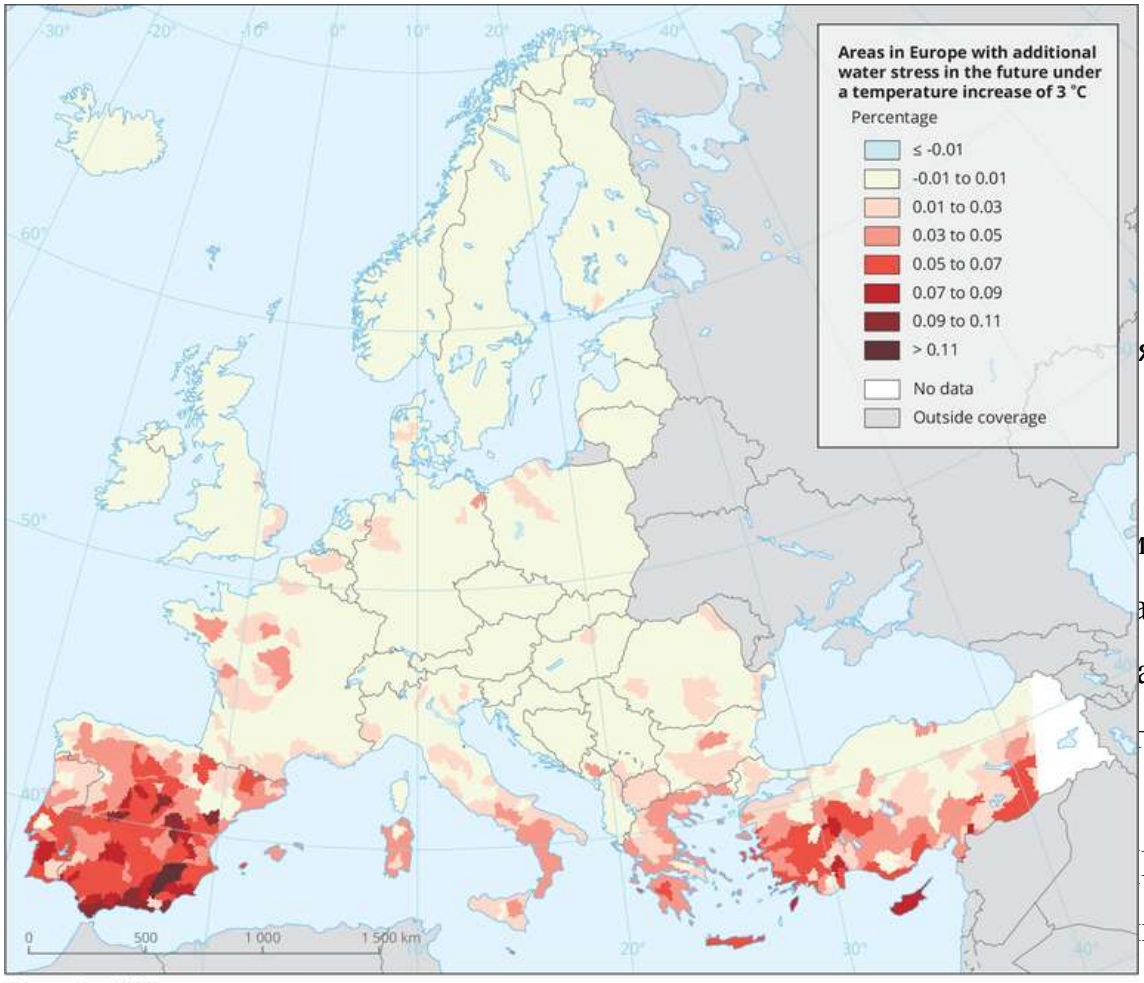
Програми-члени GEN отримали статус Типу 1 відповідно до ISO 14024:2018 Екологічні етикетки та Декларації — Екологічна маркування типу 1.

Європейське агентство довкілля (ЄАД) (англ. European Environment Agency (EEA) — агентство ЄС для забезпечення незалежною інформацією

про стан довкілля. Також зустрічаються назви — Європейське екологічне агентство (ЕАД), Європейське агентство з охорони навколишнього середовища. Розташований у Копенгагені (Данія) [14].

Рис. 1.2 – Логотип організації EEA

Матеріали ЄАД – це основна інформаційна база для тих, хто залучений до розвитку, прийняття, проведення та оцінювання екологічної політики, а також для громадськості (рис 1.3).



я його

ии.

андією,

аїн, які

говина,

Eionet)

ьна за

розвиток мережі та координація її дій. Для цього ЄАД тісно співпрацює з національними фокусними групами (focal points), як правило, це національні екологічні агенції або міністерства охорони навколишнього середовища.

Вони відповідальні за координування національних мереж, що включають багато установ (загалом близько 300).

Всесвітній фонд дикої природи (англ. World Wide Fund for Nature), до 1986 р. - Всесвітній фонд дикої природи (World Wildlife Fund, WWF) - міжнародна неурядова організація, що займається збереженням природи, дослідженнями та відновленням природного середовища. Офіційна назва організації була змінена з World Wildlife Fund на World Wide Fund for Nature, однак колишня назва залишається офіційною в багатьох країнах [15].

Це найбільша незалежна природоохоронна організація у світі, має близько 5 млн працівників та добровольців по всьому світу, працюючи у понад 120 країнах. Щорічно WWF здійснює понад 1200 екологічних проектів, привертаючи увагу мільйонів людей до проблем охорони навколишнього середовища та їх вирішення. Організація існує на добровільних внесках, приблизно 9% її бюджету надходить від приватних пожертвувань.

Місія WWF — запобігання деградації природного середовища планети, що наростає, і досягненню гармонії людини і природи. Головна мета – збереження біологічної різноманітності Землі. Символ Всесвітнього фонду дикої природи – гігантська панда.

Рис. 1.4 – Логотип організації WWF

1.2 Огляд існуючих систем моніторингу оточуючого середовища

1.2.1 Огляд персональних систем моніторингу

Метеостанція Bresser Weather Center 5-in-1 Wi-Fi Profi Sensor Color — професійна метеостанція з кольоровим екраном. Вона забезпечує вимірювання таких важливих метеопараметрів, як-от величина атмосферного тиску, швидкість і напрямок вітру, рівень опадів, температура та відносна вологість повітря, рівень комфорту в приміщенні. Дані збираються з бездротових датчиків, які можна під'єднувати до основного модуля за радіоканалом або через мережу Wi-Fi. Зв'язок підтримується на відстані до 150 метрів. Функціонал станції дає змогу переглядати дані на смартфоні, планшеті або комп'ютері. Можлива публікація даних на порталах: PWS Weather (www.pwsweather.com), Weathercloud (weathercloud.net), Weather Underground (www.wunderground.com) [16].

Рис. 1.5 - Bresser Weather Center

Метеостанція Bresser Weather Center 5-in-1 Wi-Fi Profi Sensor Color — це багатофункційний прилад для будинку, який допоможе вам бути в курсі

будь-яких погодних змін і допоможе в підтримці оптимального мікроклімату в будинку.

Сучасний лаконічний дизайн і найширший спектр функцій дасть змогу з однаковим успіхом використовувати цю метеостанцію як у квартирі, так і на присадибних ділянках.

Вся інформація про погоду виводиться на яскравий кольоровий екран метеостанції. Крім поточних показників, метеостанція будує графіки прогнозів. Невеликі стрілки вказують на те, чи будуть, наприклад, температура та вологість повітря підвищуватися найближчим часом чи, навпаки, варто очікувати їхнього зниження. Окремо розташована іконка прогнозу погоди на найближчі 24 години. Вона може мати кілька значень: сонячно, мінлива хмарність, хмарно, дощ, сніг.

Метеостанція цікава не тільки погодними вимірами. Є в ній і вбудований годинник і календар. У нижній частині екрана зображаються день тижня, поточна фаза Місяця, поточний час, число та місяць. Тут же з'являється і значок будильника, якщо він увімкнений.

Рис. 1.6 – Інформаційні можливості станції

Вбудована функція тривожних сповіщень попередить вас про вихід вимірюваних параметрів за вказані межі. Усі датчики скомпоновані в єдиному блоці, що набагато спрощує та прискорює процес установлення та монтажу.

MISOL WH1150-1 – сучасна, мультифункціональна метеостанція з безпровідною передачею даних від сенсора (до 100 метрів) для моніторингу погодних характеристик: температури, вологості, абсолютного та відносного атмосферного тиску, точки роси, відображення поточних погодних умов. Ця модель – відносно недорогий варіант домашньої метеостанції, що вирізняється простотою експлуатації, високою точністю та надійністю вимірювання [17].

Метеостанція оснащена великим РК-дисплеєм з підсвічуванням для чіткого відображення всіх показників вимірювання та бездротовим виносним сенсорним блоком з власним невеликим дисплеєм, на який також виводяться показання температури та вологості повітря. Робоча температура виносного сенсорного блоку від -40°C до +65°C, що дозволяє використовувати метеостанцію в наших кліматичних умовах протягом усього року.

еостанція MISOL WH1150-1

ика (відображення дати, дня тижня, місяця та
тареї, функції фіксації максимальних та
метрів вимірювання та звукова сигналізація
ратора.

з бездротовим датчиком MISOL WH 1150
зях сільського господарства, для планування
денної діяльності, яка залежить від погодних
опередження кризових станів здоров’я.

1.2.2 Огляд локальних рішень

The BRESSER 8-day 4CAST XL 7-in-1 Wi-Fi з датчиком на сонячній батареї BRESSER має сучасний дизайн із величезним ширококутним кольоровим екраном діагоналлю 48,3 см (19 дюймів) для відображення ваших метеорологічних даних. Ідеально підходить для офісів, шкіл і розсадників рослин – базова станція ідеально підходить для огляду погоди у великих приміщеннях. Окрім відображення даних із зовнішнього датчика, метеостанція Wi-Fi також може підключатися до Інтернету для відображення даних прогнозу з платформи ProWeatherLive, включаючи 8-денний прогноз погоди з максимальною/мінімальною температурою та ймовірністю дощу, а також місцева видимість та хмарність [18].

Рис. 1.8 - Метеостанція

Метеостанція оснащена екологічно чистим зовнішнім датчиком 7-в-1, який працює від сонячної енергії та вимірює температуру, вологість, швидкість вітру, напрямок вітру, кількість опадів, рівень ультрафіолету та інтенсивність світла (рис. 1.9). Оскільки датчик працює від сонячних батарей, не потрібно замінювати батареї. Просто вирівняйте велику сонячну

панель із сонцем для безперебійної роботи цілий рік. Коли світить сонце, вбудований акумулятор накопичує енергію для живлення датчика. Дані потім відображатимуться на вашій базовій станції, яка має радіус дії до 150 метрів.

На додаток до відображення вимірювань вашого зовнішнього датчика, базова станція 8-денного погодного центру 4CAST XL Wi-Fi також має багато інших функцій. Наприклад, він оснащений вбудованим термо/гігродатчиком, який вимірює температуру, вологість і тиск повітря в приміщенні. Є також максимальна/мінімальна пам’ять – ідеально підходить для моніторингу клімату в приміщенні та запобігання цвілі та надмірно сухого повітря. Максимальні/мінімальні значення для зовнішнього повітря, температури, кількості опадів та інших зовнішніх вимірювань також можна переглянути на дисплеї. Окрім даних про погоду, великий екран має достатньо місця для відображення часу, дати, дня тижня та фази місяця. Час сходу/сходу та заходу/заходу сонця також відображається у верхній частині екрана.

Рис. 1.9 – Сенсорні прилади метеостанції

Показання бездротового датчика на сонячній енергії можна опублікувати в Інтернеті кількома простими клацаннями, щоб ви могли отримати доступ до своїх даних віддалено за допомогою погодної платформи ProWeatherLive (за допомогою веб-браузера або програми). Ви також можете безкоштовно надіслати вимірювання зі своєї метеостанції в систему розумного дому за допомогою німецькомовного порталу AWEKAS. Просто перейдіть за посиланням нижче, щоб налаштувати параметри: https://www.awekas.at/for2/index.php?thread/17080-software-api-stations-api-beschreibung-beta/

Дані про погоду на сонячній батареї: відстежуйте місцеві погодні умови за допомогою 8-денного погодного центру BRESSER 4CAST XL 7-в-1 Wi-Fi із датчиком на сонячній енергії.

MISOL HP2550 – професійна метеостанція для самостійного відслідковування і прогнозування погодних умов. Висока точність сенсорів дозволяє отримувати прогноз, який не поступається точністю великим професійним метеостанціям.

Метеостанція складається з великого дисплею-консолі (приймача) 19,5x13,8x1,9 см, зовнішнього датчика та внутрішнього датчика (термогігрометра). Пристрій підтримує до 8 термогігрометричних датчиків (датчики продаються окремо).

На кольоровому дисплеї відображається швидкість та напрямок вітру, кількість опадів, температура, вологість, ультрафіолетове та сонячне випромінювання – від зовнішнього датчика бездротового зв'язку, барометричний тиск, вологість та температура – від внутрішнього датчика, а також відображаються параметри з розрахунковими даними: індекс холоду та тепла, точка роси, швидкість та напрямок вітру (в середньому один раз на секунду протягом 16 секунд), пориви вітру, 10-хвилинну середню швидкість та напрямок вітру, схід сонця, захід сонця, фазу місяця, прогноз тощо [19].

Для дисплея передбачено 2 фони (світлий і темний) для зручного зчитування інформації в денний та нічний час. Приймач (дисплей) зберігає дані вимірювання, а також максимальні/мінімальні значення даних із позначкою часу які можна скопіювати на карту TF (постачається опційно) у форматі (*.csv). Дані можна переглядати та аналізувати в програмі Excel на комп’ютері або ноутбуці.

ifi) MISOL HP2550

Fi та завантажувати дані на https://wow.metoffice.gov.uk/,

я внутрішньої та зовнішньої ABS, барометра REL, вітру, имка калібрування точності ганцією.

ISOL HP2550 широко використовують для моніторингу погодних умов як на відкритому ґрунті, так і в теплицях, прогнозування погодних умов, моніторингу сприятливих погодних умов для проведення сільськогосподарських робіт (посівні роботи, зрошення, внесення добрив, засобів захисту рослин, збирання врожаю, тощо), попередження про заморозки і засухи, моніторинг вологості і температури ґрунту, контроль мікроклімату на полях і в теплицях, контроль вентиляції тваринницьких ферм, для спостереження за тепловим режимом тварин, тощо.

1.2.3 Огляд глобальних рішень

Проект «Спільний моніторинг охорони довкілля в країнах BSB» спрямований на вирішення спільних проблем природних парків та

заповідних територій Чорноморського басейну (BSB). Загальна мета проекту – сприяти підвищенню рівня доступності транскордонних сумісних даних та інформації з моніторингу довкілля в природних парках та заповідних територіях у BSB. Основним результатом проекту стане створення інтелектуальної платформи для збору, обробки та аналізу даних про навколишнє середовище через веб-хмарний сервіс для автоматичного збору даних з бездротових сенсорних мереж та веб-хмарний сервіс для відеоконтенту. Онлайн-система моніторингу (OMS) для даних про навколишнє середовище в BSB інтегруватиме дані вимірювань і служитиме платформою для поширення зібраної інформації та даних. Розроблені розумні технології та інтелектуальні бездротові сенсорні мережі будуть також використовуватися для моніторингу стану природних місць існування та наявності інвазивних видів, а також для дистанційного спостереження територій, найбільш залежних від зміни клімату та антропогенних впливів [20].

Транскордонна команда дослідників розробить спільну методологію моніторингу стану природних середовищ існування та доступності інвазивних видів та проведе моніторинг. Методологія гарантує збір сумісних даних та актуальної інформації про розташування та розміри пошкоджених ділянок, типи тиску та оцінку потенційних джерел і форм загрози.

Рис 1.11 – Приклад роботи датчиків повітря

Для кожного природного парку чи заповідної території будуть визначені постійні місця контролю відбору проб, на яких експерти та волонтери проводитимуть щорічні польові спостереження. На основі зібраної інформації моніторингу буде розроблено звіт на основі оцінених існуючих та потенційних джерел і форм тиску на ключові території в межах заповідних територій Чорноморського басейну та Переліку заходів, які пропонуються для збереження забруднення та відновлення забруднення. моніторинг ключових областей.

1.3 Висновки за розділом

Здійснено огляд сучасного стану технологій та інформаційних систем з екологічного моніторингу у світі, Грузії та Україні зокрема. Установлено значимість та актуальність проблеми. Здійснено огляд сучасних систем-аналогів. Виявлено доцільність подальшої розробки багатошарової архітектури з мікроконтролерним керуванням та використанням мереж різного рівня.

РОЗДІЛ 2
ВИЗНАЧЕННЯ АРХІТЕКТУРИ СИСТЕМИ ЕКО-МОНІТОРИНГУ

2.1 Архітектурні особливості бездротових сенсорних мереж

Бездротові сенсорні мережі (Wireless Sensor Network WSN) – це новий клас бездротових мереж, які набувають популярність з величезною кількістю цивільних і військових застосувань. Це бездротова мережа, яка містить розподілений незалежний датчик та пристрої, призначені для моніторингу фізичних або екологічних умов. WSN складається з набору з'єднаних крихітних сенсорних вузлів, які спілкуються один з одним і обмінюються інформацією та даними. Ці вузли отримують інформацію про навколишнє середовище наприклад, температура, тиск, вологість або забруднювачі, і відправляють цю інформацію на базову станцію. Остання надсилає інформацію в дротову мережу або активує сигнал тривоги чи дію, залежно від типу та обсягу даних, які відстежуються [21].

Бездротові сенсорні мережі створюють нові програми та вимагають нетрадиційних парадигм для розробки протоколів через декілька обмежень. Через вимогу щодо низької складності пристрою разом із низьким споживанням енергії (тобто тривалим терміном служби мережі), необхідно знайти належний баланс між можливостями зв’язку та обробки сигналів/даних. Це спонукає до величезних зусиль у дослідницькій діяльності, процесі стандартизації та промислових інвестиціях у цій галузі з останнього десятиліття. В даний час більшість досліджень WSN зосереджено на розробці енерго- та обчислювально ефективних алгоритмів і протоколів, а область застосування була обмежена простими програмами моніторингу та звітності, орієнтованими на дані. Одним із рішень є алгоритм CMT (Cable Mode Transition), який визначає мінімальну кількість активних датчиків для підтримки K-покриття місцевості, а також K-з'єднання мережі. Зокрема, він

розподіляє періоди бездіяльності для кабельних датчиків, не впливаючи на покриття та вимоги до підключення мережі на основі лише локальної інформації.

Метою запропонованої структури мережі є мінімізувати затримки в процесах збору даних бездротових сенсорних мереж, що подовжує термін служби мережі. У [22] автори розглянули ретрансляційні вузли, щоб пом’якшити геометричні недоліки мережі, і використали алгоритми на основі оптимізації рою частинок (PSO), щоб знайти оптимальне розташування стоків щодо цих вузлів ретрансляції, щоб подолати проблему протягом усього життя.

Але сьогодні дослідники зосередилися на гетерогенних сенсорних мережах, де сенсорні вузли не схожі один на одного з точки зору їхньої енергоспоживання. Нові архітектури мережі з гетерогенними пристроями та останній прогрес у цій технології усувають поточні обмеження та значно розширюють спектр можливих застосувань для WSN, і все це дуже швидко змінюється [22].

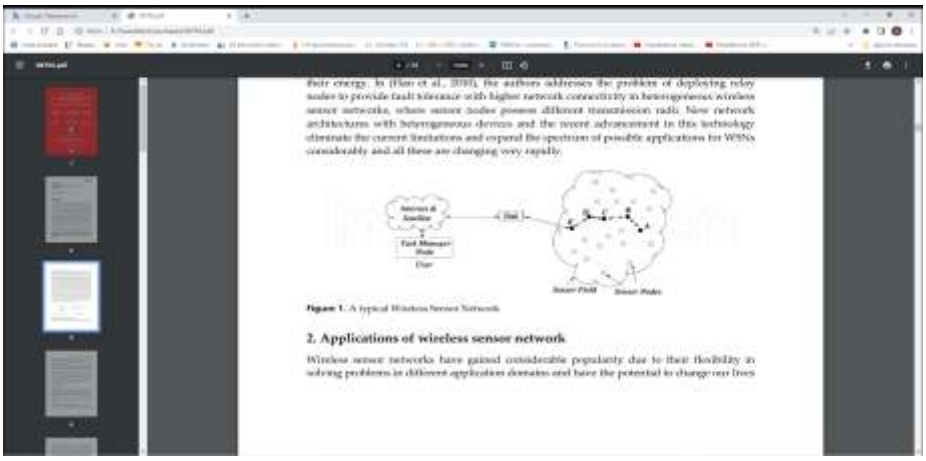


Рис. 2.1 – Типова WSN

2.1.1 Сфера використання WSN

Бездротові сенсорні мережі здобули значну популярність завдяки своїй гнучкості у вирішенні проблем у різних областях застосування та мають потенціал змінити наше життя багатьма різними способами.

Військове застосування: бездротові сенсорні мережі, ймовірно, є

невід’ємною частиною військового командування, управління, зв’язку, обчислювальної техніки, розвідки, спостереження на полі бою, розвідки та систем націлювання.

Моніторинг території: при моніторингу території сенсорні вузли розгортаються в регіоні, де потрібно відстежувати певне явище. Коли датчики виявляють подію, що контролюється (тепло, тиск тощо), про подію повідомляється на одну з базових станцій, яка потім приймає відповідні дії.

Транспорт: WSN збирає інформацію про дорожній рух у режимі реального часу, щоб згодом подавати моделі транспортування та сповіщати водіїв про затори та проблеми з трафіком.

Програми для охорони здоров’я: деякі програми охорони здоров’я для сенсорних мереж підтримують інтерфейси для інвалідів, інтегрований моніторинг пацієнтів, діагностика та адміністрування ліків у лікарнях, дистанційний моніторинг фізіологічних даних людини, а також відстеження та моніторинг лікарів або пацієнтів у лікарні.

Зондування навколишнього середовища. Термін екологічні сенсорні мережі розвинувся для охоплення багатьох застосувань WSN для наукових досліджень Землі. Це включає зондування вулканів, океанів, льодовиків, лісів тощо. Нижче наведено деякі інші основні області:

- Моніторинг забруднення повітря.
- Виявлення лісових пожеж.
- Моніторинг теплиць.
- Виявлення замлетрусів.

Моніторинг будівництва: бездротові датчики можна використовувати для моніторингу руху всередині будівель та інфраструктури, таких як мости, естакади, набережні, тунелі тощо, що дозволяє інженерним практикам дистанційно відстежувати активи без необхідності дорогого відвідування місця.

Промисловий моніторинг: бездротові сенсорні мережі були розроблені для технічного обслуговування на основі стану обладнання,

оскільки вони пропонують значну економію коштів і надають нові функції. У дротових системах установка достатньої кількості датчиків часто обмежується вартістю електропроводки.

Сільськогосподарський сектор: використання бездротової мережі звільняє фермера від обслуговування електропроводки в складних умовах. Автоматизація зрошення забезпечує більш ефективне використання води та зменшує відходи.

2.1.2 Поширені топології використання WSN

Структура бездротової сенсорної мережі включає різні топології для мереж радіозв'язку. Нижче наведено коротке обговорення мережевих топологій, які застосовуються до бездротових сенсорних мереж.

Мережа зірка – це топологія зв’язку, де одна базова станція може надсилати та/або отримувати повідомлення на декілька віддалених вузлів. Віддаленим вузлам заборонено надсилати повідомлення один одному. Перевага цього типу мережі для бездротових сенсорних мереж включає простоту, можливість мінімізувати споживання енергії віддаленим вузлом. Це також дозволяє здійснювати зв'язок з малою затримкою між віддаленим вузлом і базовою станцією.

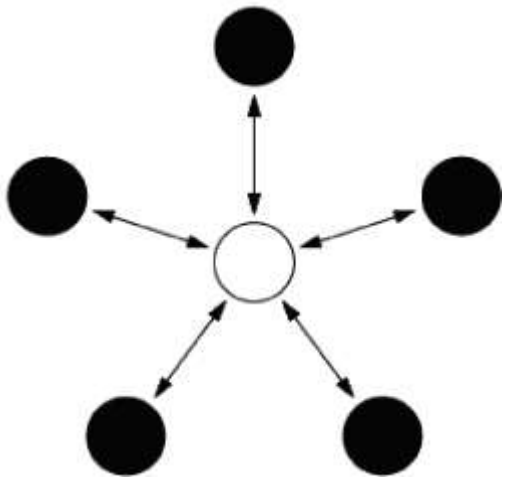


Рис. 2.2 – Зіркова топологія

Недоліком такої мережі є те, що базова станція повинна перебувати в

діапазоні радіопередавання всіх окремих вузлів і не є настільки надійною, як інші мережі, через її залежність від одного вузла для управління мережею.

Сітчаста мережа дозволяє передавати дані до одного вузла до іншого вузла мережі, що знаходиться в межах її діапазону радіопередачі. Це дозволяє здійснювати так званий багатострибковий зв’язок, тобто якщо вузол хоче надіслати повідомлення іншому вузлу, який знаходиться поза діапазоном радіозв’язку, він може використовувати проміжний вузол для пересилання повідомлення до потрібного вузла.

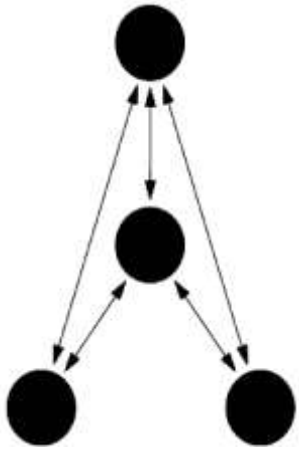


Рис. 2.3 – Сітчаста топологія

Ця топологія мережі має перевагу надлишковості та масштабованості. Якщо окремий вузол виходить з ладу, віддалений вузол все ще може спілкуватися з будь-яким іншим вузлом у своєму діапазоні, який, у свою чергу, може переслати повідомлення в потрібне місце. Крім того, діапазон мережі не обов'язково обмежується діапазоном між окремими вузлами; його можна просто розширити, додавши більше вузлів до системи. Недоліком такого типу мережі є те, що енергоспоживання вузлів, які реалізують багатострибковий зв’язок, зазвичай вище, ніж для вузлів, які не мають цієї можливості, що часто обмежує термін служби батареї. Крім того, у міру збільшення кількості стрибків зв’язку до пункту призначення збільшується час доставки повідомлення також збільшується, особливо якщо низька потужність роботи вузлів є вимогою.

Сітчасто-зіркова топологія. Гібрид між мережею «зірка» та мережевою мережею забезпечує надійну та універсальну комунікаційну мережу, зберігаючи при цьому можливість мінімізувати споживання електроенергії бездротовими сенсорними вузлами. У цій топології мережі сенсорні вузли з найменшою потужністю не мають можливості пересилати повідомлення. Це дозволяє підтримувати мінімальне споживання електроенергії. Однак інші вузли в мережі мають можливість перейти до кількох переходів, що дозволяє їм пересилати повідомлення від вузлів низької потужності до інших вузлів мережі. Як правило, вузли з можливістю кількох стрибків мають більшу потужність і, якщо можливо, часто підключаються до електричної мережі. Це топологія, реалізована новим стандартом сітчастої мережі, відомим як ZigBee.

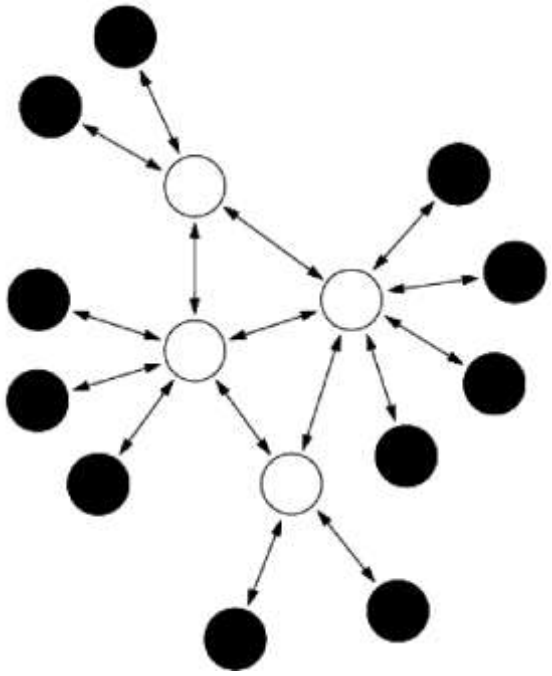


Рис. 2.3 – Проміжна топологія

2.2 Організація мережевого вузла системи

Сенсорний вузол складається з чотирьох основних компонентів, таких як сенсорний блок, процесор, блок приймача та блок живлення, який

показаний на рис. 2.4. Він також має додаткові компоненти, залежні від застосування, такі як система визначення місцезнаходження, генератор електроенергії та актуатор. Сенсорні блоки зазвичай складаються з двох субблоків: датчиків і аналого-цифрових перетворювачів (АЦП). Аналогові сигнали, вироблені датчиками, перетворюються в цифрові сигнали АЦП, а потім надходять на блок обробки.

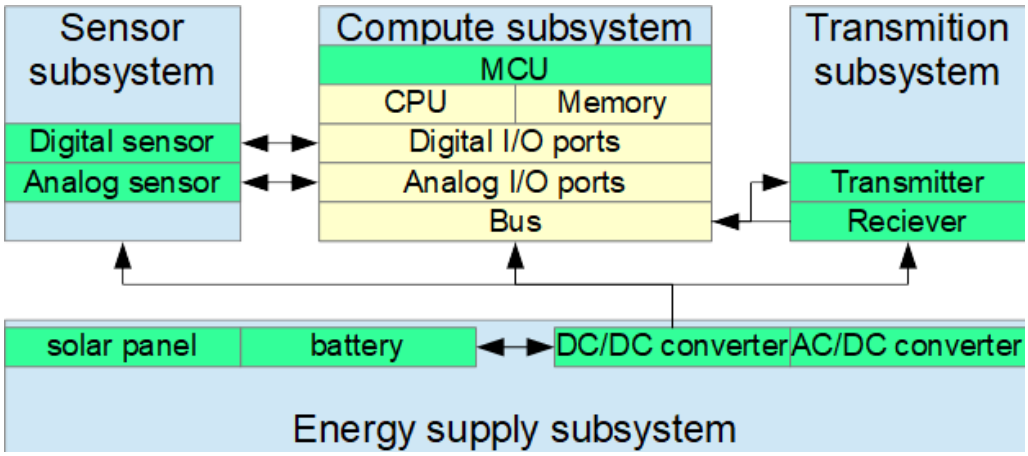


Рис. 2.4 – Компоненти сенсорного вузла

Блок обробки, як правило, пов’язаний з невеликим накопичувачем, і він може керувати процедурами, які змушують вузол датчика співпрацювати з іншими вузлами для виконання призначених завдань зондування. Приймач з'єднує вузол з мережею. Одним з найважливіших компонентів сенсорного вузла є блок живлення. Блоки живлення можуть підтримуватися блоком поглинання енергії, таким як сонячні батареї. Інші субодиниці вузла залежать від програми.

Функціональна блок-схема універсального бездротового сенсорного вузла представлена на рис. 2.5 Модульний підхід до проектування забезпечує гнучку та універсальну платформу для задоволення потреб широкого спектру застосувань. Наприклад, залежно від датчиків, які будуть розгорнуті, блок формування сигналу можна перепрограмувати або замінити. Це дає змогу використовувати широкий спектр різних датчиків із бездротовим сенсорним вузлом. Аналогічно, радіоканалу можна замінити відповідно до вимог

бездротового діапазону даної програми та потреби в двонаправленому зв’язку.

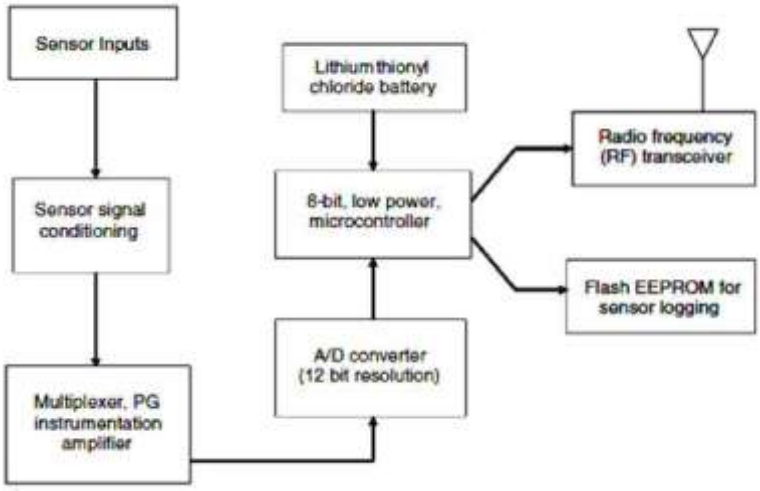


Рис. 2.5 – Функціональна блок-схема сенсорного вузла

Використовуючи флеш-пам’ять, віддалені вузли отримують дані за командою від базової станції або за подією, що відстежується одним або кількома входами до вузла. Крім того, вбудоване мікропрограмне забезпечення можна оновити через бездротову мережу в польових умовах.

Мікропроцесор виконує ряд функцій, серед яких:

- управління збором даних з датчиків;
- виконання функцій управління живленням;
- з'єднання даних датчика з фізичним радіорівнем;
- управління протоколом радіомережі.

Ключовим аспектом будь-якого бездротового сенсорного вузла є мінімізація енергії, споживаної системою. Зазвичай радіопідсистема вимагає найбільшої потужності. Тому дані надсилаються по радіомережі лише тоді, коли це необхідно. Алгоритм має бути завантажений у вузол, щоб визначити, коли надсилати дані на основі відчутної події. Крім того, важливо мінімізувати потужність, споживану самим датчиком. Тому апаратне забезпечення має бути сконструйовано так, щоб мікропроцесор міг розумно керувати живленням радіостанції, датчика та перетворювача сигналу

датчика.

2.3 Використання протоколів обміну інформацією

2.3.1 Архітектурна взаємодія вузлів у глобальній мережі

Шаблон «запит – відповідь» (request – response), це, ймовірно, найвідоміший шаблон обміну інформацією. Його реалізація передбачає наявність клієнта, або сторони, що викликає, який виконує запити до якоїсь служби, розташованої на сервері, що надає послуги. Сервер ще називають респондентом чи відповідачем.

Рис. 2.6 - Шаблон «запит – відповідь»

Цей шаблон використовує протокол HTTP. Він є основою сервісно-орієнтованих архітектур, веб-служб і REST-рішень. Це практичний шаблон, особливо якщо архітектура проекту передбачає наявність клієнтських і серверних частин або провідних і ведених сутностей [23].

Крім HTTP, шаблон «запит – відповідь» підтримують такі протоколи, як Constrained Application Protocol (CoAP) та Extensible Messaging and Presence Protocol (XMPP).

Основний недолік даного шаблону полягає в нерівності учасників обміну даними, що цілком очевидно проявляється у топології інтернету. Двонаправлений обмін даними, коли обидва учасники запитують дані один в одного, може бути складний у реалізації, якщо на шляху даних є мережні екрани.

Плануючи використовувати шаблон «запит – відповідь» у проекті, необхідно вирішити, які частини системи будуть клієнтами, а які –

серверами. Якщо, наприклад, якийсь датчик буде клієнтом, а IoT-шлюз - сервером, датчик сам вирішуватиме, коли йому передавати на сервер власні показання. Сервер ж, якщо йому знадобляться відомості з датчика, їх самостійно запитати не зможе. Якщо ж датчик зробити сервером, а шлюз - клієнтом, датчик можна буде опитувати будь-коли. Однак, тут є одна проблема: якщо датчик недостатньо захищений, будь-хто зможе до нього підключитися. Якщо ж у такому рішенні задіяна надійна система безпеки, то, як наслідок, ускладниться спосіб взаємодії клієнта та сервера та й сама система в цілому. Можливо, до проекту потрібно буде додати додаткові служби, датчики доведеться оснащувати потужнішим апаратним забезпеченням. Крім того, усім цим буде складніше керувати.

Шаблон "підписка на події". Цей шаблон (event subscription) дозволяє клієнту передплатити події заданого типу на сервері. Сервер сповіщає клієнта щоразу, коли відбувається цікава для нього подія. Як результат, відпадає потреба у постійному опитуванні сервера.

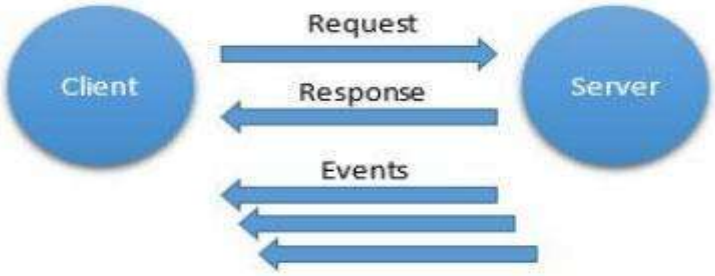


Рис. 2.7 - Шаблон "підписка на події"

Просунутий механізм підписки на події може включати вимоги, що залежать від клієнта, які стосуються того, які саме події і за яких умов його цікавлять. Переваги використання підписки на події перед раніше розглянутим шаблоном «запит – відповідь» полягають у тому, що для обміну даними між клієнтом та сервером потрібно приблизно вдвічі менше повідомлень. Крім того, дані передаються клієнту при виникненні якоїсь події, а не за запитом, що знижує до мінімуму час між виникненням якоїсь

ситуації, що цікавить клієнта, і моментом, коли він про це дізнається.

Протоколи, які підтримують цей шаблон, включають CoAP, XMPP і General Event Notification Architecture, який є частиною архітектури Universal Plug and Play, що базується на HTTP.

Шаблон "асинхронний обмін повідомленнями". Асинхронний обмін повідомленнями (asynchronous messaging) передбачає можливість надсилання повідомлень між рівноправними системами, що знаходяться на одному ступені ієрархії. Цей шаблон має на увазі двонаправлений обмін повідомленнями.

Рис. 2.8 - Шаблон "асинхронний обмін повідомленнями"

Якщо протокол підтримує асинхронний обмін повідомленнями, на його основі можна побудувати будь-які інші шаблони передачі даних.

Серед протоколів, які підтримують цей шаблон, можна назвати XMPP, Advanced Message Queuing Protocol (AMQP), і на рівні IP – User Datagram Protocol (UDP). Однак, у разі застосування UDP для реалізації цього шаблону, можливі проблеми з мережними екранами.

Шаблон «Надійна доставка повідомлень». Додаткам, які виконують критично важливі функції, необхідно знати, що повідомлення було доставлене отримувачу щонайменше один раз. Власне кажучи, при використанні асинхронного шаблону обміну даними ця вимога виконується. Повідомлення може загубитися в дорозі, але використання шаблону «запит – відповідь» дозволяє запитати надсилання повідомлення знову, доки не буде отримано підтвердження (або відповідь) від сторони, яка має отримати повідомлення. Тут треба враховувати, що може бути і повідомлення, і відповідь про його отримання, тому даний шаблон гарантує, що

повідомлення буде доставлено як мінімум один раз. Однак, те, що повідомлення буде доставлено одержувачу не більше одного разу (або не менше одного разу), не дуже підходить деяким додаткам, наприклад, таким, що залучають концепцію транзакцій або виконують підрахунок повідомлень.

Застосування шаблону надійної доставки повідомлень (reliable messaging) дає гарантію того, що повідомлення буде доставлене отримувачу точно один раз. Серед протоколів, що підтримують надійну доставку повідомлень, можна назвати Message Queuing Telemetry Transport (MQTT), AMQP. Крім того, завдяки відкритим розширенням, подібний функціонал підтримують HTTP та XMPP.

Шаблон «багатоадресна передача повідомлень». Попередній шаблон зайнятий обміном повідомлень між двома об'єктами. Іноді, однак, потрібен ефективніший підхід, якщо одну й ту саму інформацію потрібно в один і той же час надіслати кільком одержувачам. Найпростіший із шаблонів, що реалізують подібний функціонал – це «багатоадресна передача повідомлень» (multicasting). В рамках цього шаблону відправник надсилає одне повідомлення через проміжне ланка системи (це може бути брокер або маршрутизатор), після чого повідомлення пересилається кільком одержувачам, кожен з яких зареєструвався для отримання подібних повідомлень.

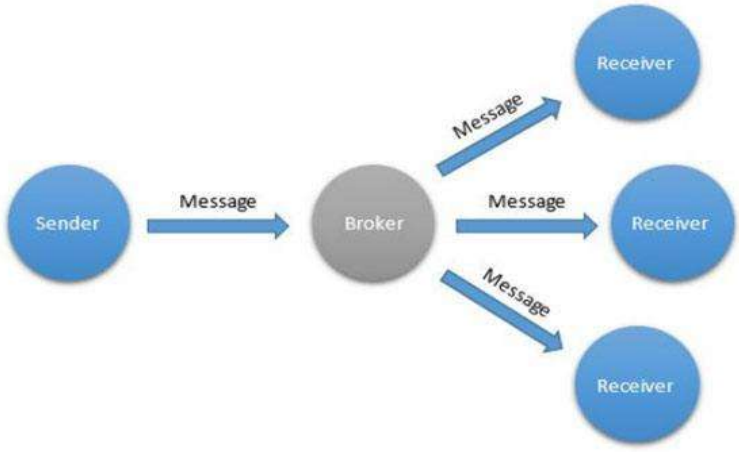


Рис. 2.9 - Шаблон «багатоадресна передача повідомлень»

Завдяки використанню даного шаблону можна знизити навантаження

на мережу, так як відправнику не потрібно самостійно надсилати те саме повідомлення кожному, хто його очікує. Насправді, відправник навіть не повинен знати, хто саме отримає повідомлення. Цей шаблон може бути дуже корисним у безлічі ситуацій. Наприклад, при синхронізації безлічі пристроїв або при розподілі тих самих даних між декількома одержувачами. Багатоадресну передачу повідомлень підтримують протоколи XMPP, AMQP та UDP.

Тут доречно висловити деякі застереження. Стосуються використання багатоадресної передачі повідомлень для реалізації інших схем зв'язку.

Так, хоча цей шаблон і можна використовувати для зниження навантаження на мережу, часто до нього звертаються як до способу обходу обмежень у протоколі, а також – для реалізації на базі якогось протоколу шаблону «підписка на події». Якщо, наприклад, використовувати багатоадресну передачу повідомлень для того, щоб зменшити затримки в мережах, де потрібно, але неможливо, реалізувати шаблон «підписка на події», цей шаблон призведе не до зниження, а підвищення навантаження на мережу. Крім того, систему з багатоадресною передачею даних складніше захистити.

Що стосується підвищення ефективності використання пропускної спроможності мережі при використанні багатоадресної передачі даних, то ресурси можна заощадити лише в тому випадку, якщо одержувачі споживають більшу частину даних. Якщо ж значна частка даних, що передаються таким чином, не використовується одержувачами – це привід розглянути інші шаблони взаємодії.

Шаблон «видавець – передплатник». Шаблон «видавець – передплатник» (publish – subscribe) є розширенням шаблону багатоадресної доставки повідомлень. Принципова різниця між ними полягає в тому, що надіслані повідомлення зберігаються на проміжному вузлі. Ці повідомлення, або посилання на них, потім розподіляються за зацікавленими у них передплатниками.

Особливості реалізації шаблону залежать від протоколу, що використовується, залежить від нього, а також від налаштувань проміжного вузла, і те, які саме повідомлення зберігаються. Це може бути тільки найсвіжіше повідомлення, або кількість повідомлень, або всі повідомлення.

Тут, крім того, важлива різниця у передачі самого повідомлення та посилання на нього, оскільки це впливає на потрібну смугу пропускання мережі, і, як наслідок, на продуктивність рішення.

Якщо передплатники використовують більшість повідомлень, то передача самих повідомлень ефективніша, як і у випадку з багатоадресною передачею даних. Якщо ж фактичне споживання даних одержувачами залежить від деяких додаткових чинників, ефективніше передавати посилання повідомлення. Вони менше, ніж самі повідомлення, а передплатники, найімовірніше, використовують лише невелика їх кількість у тому, щоб отримати ті повідомлення, куди вказують посилання. У такому випадку, для отримання повідомлення за посиланням, потрібно виконувати додаткові звернення до вузла зберігання повідомлень за моделлю «запит – відповідь».

Шаблон «видавець – передплатник» підтримує такі протоколи, як MQTT, AMQP, XMPP.

Шаблон "черга". Черги, а саме - черги FIFO - це шаблон обміну даними, який дозволяє одній або більшій кількості сутностей відправляти деякі повідомлення або завдання для обробки в чергу, після чого один або кілька одержувачів отримують ці повідомлення в порядку, в якому вони були поставлені в чергу.

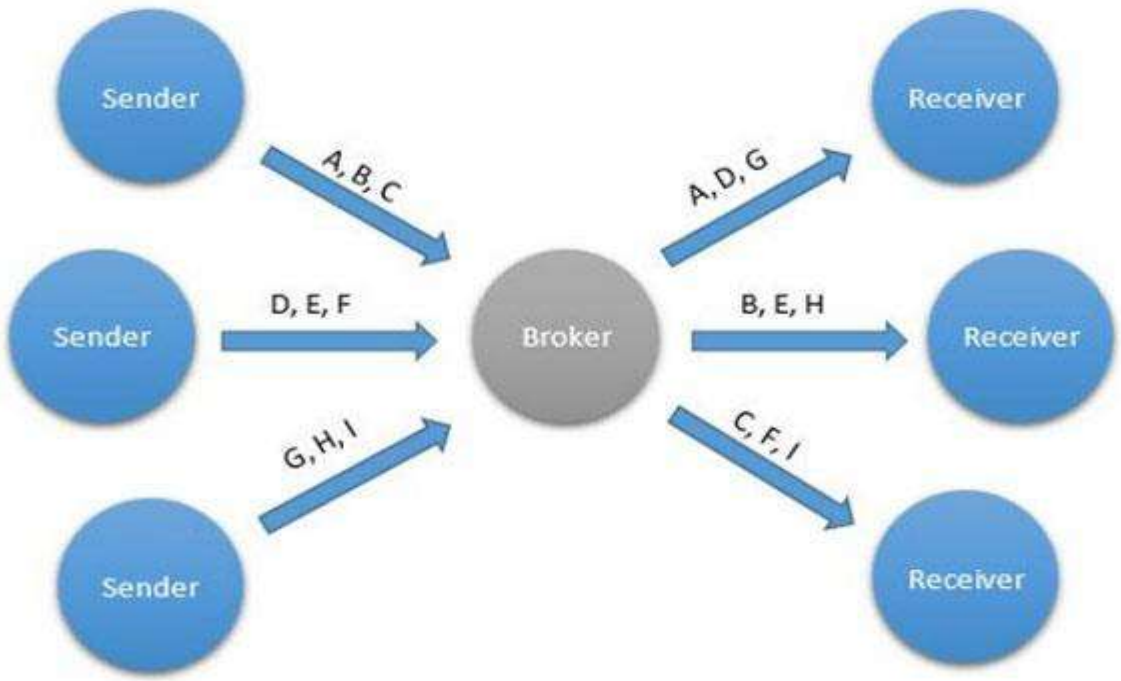


Рис. 2.10 - Шаблон "черга"

Черга зазвичай знаходиться на проміжному вузлі або в мережі, до якої підключені всі учасники обміну даними. Черги – це чудовий засіб для балансування навантаження. У чергу збирають завдання з різних джерел і розподіляють їх серед існуючих обробників, які, можливо, мають різну продуктивність.

Використовуючи чергу, можна уникнути жорсткого зв'язку між системами, що передають дані, та системами, які ці дані отримують і обробляють. В результаті, залежно від реального робочого навантаження на систему, можна збільшувати або зменшувати кількість приймачів та передавачів даних. Серед протоколів, про які ми вже говорили, лише AMQP має вбудовану підтримку черг.

Шаблон "брокери повідомлень". Брокери повідомлень (message brokers) є стандартизованими компонентами допоміжної мережевої інфраструктури IoT-проектів. Вони вирішують проблеми, викликані обмеженнями, які накладають мережеві екрани на двонаправлений обмін даними між пристроями. Брокер дозволяє сутності підключатися до нього, займаючись

передачею повідомлень між підключеними до нього клієнтами. Оскільки всі підключення виконані через брокера, тільки брокер має бути доступний з Інтернету. Не потрібно приймати або перенаправляти вхідні підключення до пристроїв, як було б потрібно при використанні протоколу, що забезпечує зв'язок рівноправних систем, жорстко обмеженого подібною моделлю обміну повідомленнями.

Крім керування повідомленнями, брокери можуть надавати додаткові служби підключеним клієнтам. Наприклад, брокер може бути посередником при реалізації шаблону багатоадресного обміну повідомленнями, шаблонів «видавець – передплатник» та «черга».

Крім того, брокери повідомлень зазвичай надають послуги автентифікації клієнтів. Це полегшує роботу в розподілених мережах, де автентифікація пристроїв може виявитися непростим завданням. Таким чином, якщо брокер може повідомляти про статус автентифікованих учасників системи, включених в обмін даними, інші учасники можуть використовувати цю інформацію для прийняття рішень у сфері безпеки. Це, до того ж, позбавляє необхідності реалізації власної схеми аутентифікації на кожному учаснику обміну даними.

Хоча обмін повідомленнями між рівноправними системами – це лише один із варіантів організації зв'язку, подібні рішення мають передбачати аутентифікацію клієнтів. Інакше серйозно постраждає безпека системи. Якщо ж використовується протокол, який включає брокери повідомлень, то, найімовірніше, не знадобиться розробляти власні допоміжні служби, які дозволять рішенню працювати надійно і безпечно.

Протоколи XMPP, AMQP і MQTT, у тій чи іншій формі, використовують цей шаблон.

Шаблон "федерація". "Федерація" (federation) - це важливий шаблон, в якому якась глобальна мережа розбивається на логічні частини. Це дозволяє здійснювати глобальне масштабування рішення та забезпечує все необхідне для його природного зростання.

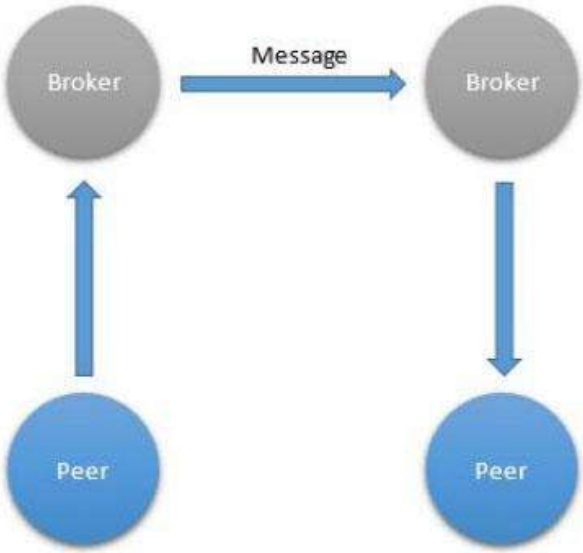


Рис. 2.11 - Шаблон "федерація"

Основна ідея тут – дозволити збільшувати розміри рішення, не обмежуючи при цьому продуктивність існуючої мережної інфраструктури, використовуючи підхід «розділяй та владарюй».

При здійсненні зв'язку без брокерів, наприклад, як із використанні протоколів HTTP і CoAP, федеративна структура є лише на рівні домену. Кожен домен вказує на власний набір IP-адрес, з ним пов'язаний власний веб-сервер. До системи можна додавати нові веб-сервери, в нових доменах, не обмежуючи доступ до існуючих систем. Такий підхід – один із основних ключів до успіху Всесвітньої павутини.

При використанні протоколів, що передбачають наявність брокерів і підтримують федерації, брокери з'єднуються між собою маршрутизації повідомлень. Кожен брокер управляє автентифікацією у своєму домені і знає, як підключатися до інших доменів для перенаправлення в них повідомлень. Крім того, федеративні мережі з брокерами надають зручне вирішення проблеми глобальної ідентифікації учасників обміну даними.

Найбільш широко відомий протокол, що використовує брокерів та федерації – Simple Mail Transfer Protocol (SMTP). Серед протоколів, про які ми говоримо у цьому матеріалі, що вміють працювати з брокерами,

федерацію підтримує лише XMPP.

Шаблон "виявлення". Розглянемо умовний приклад. Нехай, ми маємо якийсь виготовлений на заводі пристрій, який планується використовувати в IoT-системі. Якщо його, наприклад, планується застосовувати в системі з багатоадресною передачею даних, ми відразу зіткнемося з деякими складнощами, пов'язаними з інтеграцією пристроїв у систему.

Полягають вони в тому, що «речі» обізнані лише про власну ідентифікаційну інформацію (це може бути щось на зразок MAC-адреси), але нічого не знають ні про те, як вони будуть «видні» в мережі, до якої їх планується підключити , ні про якийсь головний мережевий пристрій, з яким їм доведеться взаємодіяти.

Після встановлення та налаштування (що більше автоматизоване налаштування – тим краще), «речі» дізнаються про свою мережеву ідентифікаційну інформацію, але не про те, як підключитися до головного пристрою. У свою чергу, головному пристрої відома власна мережна адреса, а також заводські дані «речей» (які, наприклад, можна швидко ввести в систему, відсканувавши наклейки на коробках), але не мережеві дані інших пристроїв.

Шаблон обміну повідомленнями "виявлення" (discovery) дозволяє створити механізм, за допомогою якого здійснюється зіставлення мережевих ідентифікаційних даних підпорядкованих пристроїв з мережними даними головного вузла. Робиться це з використанням загальних знань про вихідні ідентифікаційні параметри підпорядкованих пристроїв.

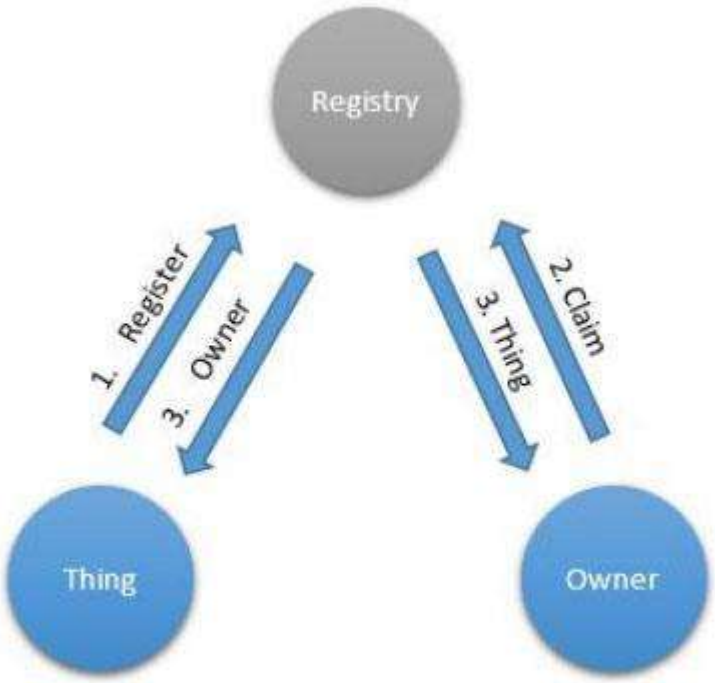


Рис. 2.12 - Шаблон "виявлення"

Даний шаблон реалізується з використанням "реєстру речей" (Thing Registry), доступного по мережі як самим "речам", так і головного пристрою. Клієнти реєструються в реєстрі, а головний пристрій звертається до них через реєстр, використовуючи лише заводські ідентифікатори. Якщо запит успішний, то мережеві ідентифікаційні дані кожного з учасників обміну даними надсилаються іншому, і обидва, таким чином, знають, як взаємодіяти один з одним. Існує розширення XMPP, яке підтримує цей шаблон.

2.3.2 Огляд протоколу MQTT

Протокол Message Queuing Telemetry Transport (MQTT) використовується протягом багатьох років, але зараз він особливо актуальний завдяки вибуховому зростанню IoT: і споживчі, і промислові пристрої впроваджують розподілені мережі та граничні обчислення (edge computing), а пристрої з постійною трансляцією даних стають частиною повсякденного життя [23].

Це означає, що легкі, відкриті та доступні протоколи з часом стануть

ще важливішими. У цій статті наводиться концептуальне занурення в MQTT: як він працює, як використовується зараз і як використовуватиметься в майбутньому.

MQTT — це протокол обміну повідомленнями за шаблоном видавець-підписник (pub/sub). Початкову версію в 1999 опублікували Енді Стенфорд-Кларк з IBM і Арлен Ніппер з Cirrus Link. Вони розглядали MQTT як спосіб підтримки зв'язку між машинами в мережах з обмеженою пропускною здатністю або непередбачуваним зв'язком. Одним з перших варіантів його використання було забезпечення контакту фрагментів нафтопроводу один з одним та з центральними ланками через супутники.

З урахуванням серйозних умов експлуатації протокол створено невеликим і легким. Він ідеальний для пристроїв слабкої потужності та з обмеженим часом автономної роботи. До них зараз відносяться і всюдисущі смартфони, і число датчиків і підключених пристроїв, що постійно зростає.

Таким чином, MQTT став протоколом потокової передачі даних між пристроями з обмеженою потужністю CPU і/або часом автономної роботи, а також для мереж з дорогою або низькою пропускною здатністю, непередбачуваною стабільністю або високою затримкою. Саме тому MQTT відомий як ідеальний транспорт для IoT. Він побудований на протоколі TCP/IP, але є відгалуження MQTT-SN для роботи з Bluetooth, UDP, ZigBee та інших мережах IoT, відмінних від TCP/IP.

MQTT - не єдиний у своєму роді протокол обміну повідомленнями pub/sub в реальному часі, але він вже набув широкого поширення в різних середовищах, які залежать від міжмашинного зв'язку. Серед його однолітків — Web Application Messaging Protocol, Streaming Text-Oriented Messaging Protocol та Alternative Message Queueing Protocol.

MQTT — логічний вибір для розробників, які хочуть створювати програми з надійною функціональністю та широкою сумісністю з підключеними до Інтернету пристроями та програмами, включаючи браузери, смартфони та пристрої IoT.

Система зв'язку, побудована на MQTT, складається з сервера-видавця, сервера-брокера та одного або кількох клієнтів. Видавець не вимагає жодних налаштувань за кількістю або розташуванням передплатників, які отримують повідомлення. Крім того, передплатникам не потрібне налаштування на конкретного видавця. У системі може бути кілька брокерів, які розповсюджують повідомлення.

MQTT надає спосіб створення ієрархії каналів зв'язку - свого роду гілка з листям. Щоразу, коли видавець має нові дані для поширення серед клієнтів, повідомлення супроводжується приміткою контролю доставки. Клієнти вищого рівня можуть отримувати кожне повідомлення, тоді як клієнти нижчого рівня можуть отримувати повідомлення, що стосуються лише одного або двох базових каналів, що «відгалужуються» у нижній частині ієрархії. Це полегшує обмін інформацією розміром від двох байт до 256 мегабайт.

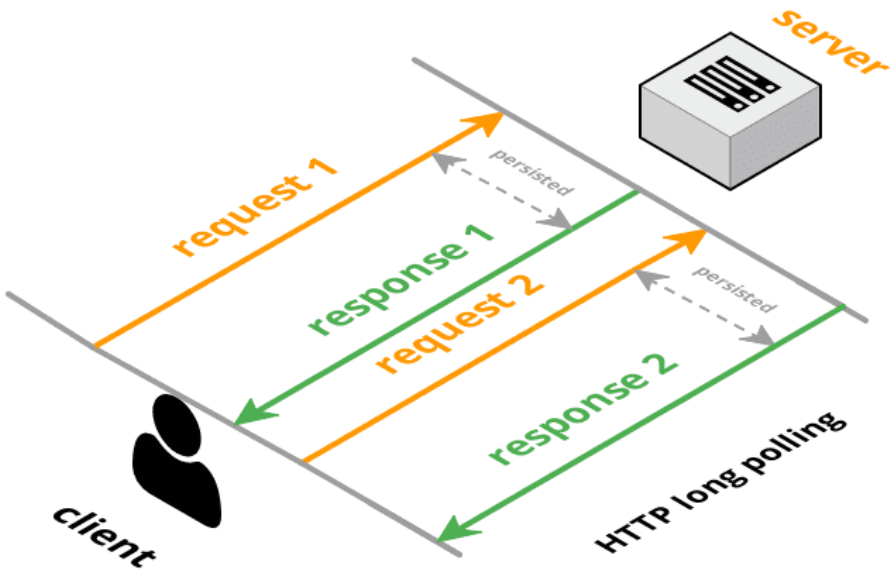


Рис. 2.13 – Схема клієнт-серверної взаємодії

Приклад того, як можна налаштувати клієнт для підключення через брокер MQTT:

var options = {

```
  keepalive: 60,
  username: 'FIRST_HALF_OF_API_KEY',
  password: 'SECOND_HALF_OF_API_KEY',
  port: 8883
};
var client = mqtt.connect('mqtts:mqtt.ably.io', options);
```

Будь-які дані, опубліковані або отримані брокером MQTT, будуть закодовані у двійковому форматі, оскільки MQTT є бінарним протоколом. Це означає, що для отримання вихідного вмісту необхідно інтерпретувати повідомлення. Ось як це виглядає за допомогою Ably та JavaScript:

```
var ably = new Ably.Realtime('REPLACE_WITH_YOUR_API_KEY');
var decoder = new TextDecoder();
var channel = ably.channels.get('input');
channel.subscribe(function(message) {
  var command = decoder.decode(message.data);
});
```

Брокери MQTT іноді можуть накопичувати повідомлення, пов'язані з каналами, які не мають поточних передплатників. У цьому випадку повідомлення будуть відкинуті, або збережені, залежно від інструкцій в повідомленні. Це корисно в тих випадках, коли новим абонентам може знадобитися остання записана точка даних, замість чекати наступної відправки.

Примітно, що MQTT передає облікові дані безпеки відкритим текстом, інакше підтримується автентифікація або функції безпеки. Ось де вступає в гру фреймворк SSL, допомагаючи захистити інформацію від перехоплення або іншої підробки.

Крім того, у MQTT можна використовувати автентифікацію Ably на токенах, якщо ви взагалі не хочете розкривати свій ключ API фактичному клієнту MQTT (у випадку MQTT без SSL токени обов'язкові, щоб запобігти передачі ключів API відкритим текстом).

MQTT має клієнт-серверну архітектуру. Обмін повідомленнями відбувається через центральний сервер, який називається брокером. У звичайних умовах клієнти можуть спілкуватися безпосередньо друг з одним, і весь обмін даними відбувається через брокера.

Клієнти можуть у ролі постачальників даних (Publisher) й у ролі одержувачів даних (Subscriber). У російському перекладі ці терміни часто перекладають як видавець і передплатник, але щоб уникнути плутанини, ми будемо використовувати лише оригінальну термінологію.

Для ідентифікації сутностей у MQTT застосовуються топики, у перекладі їх ще називають каналами. Топики складаються з UTF8-символів і мають деревоподібну структуру, схожу на файлову систему в UNIX. Це зручний механізм, що дозволяє називати сутності у людинозрозумілому вигляді [24].

Приклад топіків в MQTT

Датчик температури на кухні

home/kitchen/temperature

Датчик температури у спальні

home/sleeping-room/temperature

Датчик освітленості на вулиці

home/outdoor/light

Такий підхід дозволяє наочно бачити, які дані передаються, та зручно розробляти та налагоджувати код, без необхідності запам'ятовувати цифрову адресу розміщення даних, як це зроблено в Modbus.

Топики також передбачають wildcard-синтаксис, добре знайомий тим, хто працював із файловою системою UNIX. Wildcard може бути однорівневим та багаторівневим.

Однорівневий wildcard позначається символом "+".

Наприклад, щоб отримати дані з температурних датчиків у всіх приміщеннях у будинку, передплатнику потрібно передплатити такий топік:

home/+/temperature

В результаті він підпишеться на отримання даних із таких датчиків:

home/kitchen/temperature

home/sleeping-room/temperature

home/living-room/temperature

home/outdoor/temperature

Багаторівневий wildcard позначається символом "#".

Приклад отримання даних з усіх датчиків у всіх кімнатах у будинку:

home/#

Підписка на такий топік дозволить отримувати дані з таких датчиків:

home/kitchen/temperature

home/kitchen/humidity

home/kitchen/light

home/sleeping-room/temperature

home/sleeping-room/humidity

home/sleeping-room/light

....

Для контролю доступу в MQTT передбачено автентифікацію клієнтів, на відміну від протоколу Modbus, який не має такої функції. Для контролю доступу використовуються такі поля:

ClientId - (обов'язкове поле) - унікальний ідентифікатор клієнта. Має бути унікальним для кожного клієнта. Поточна версія стандарту MQTT 3.1.1 дозволяє використовувати порожнє поле ClientId, якщо не потрібне збереження стану підключення.

Username — (опціональне поле) логін для автентифікації у форматі UTF-8. Може бути не унікальним. Наприклад, група клієнтів може авторизуватися з тим самим логіном/паролем.

Password — (опціональне поле) може надсилатися лише разом із полем Username, причому Username може передаватися без поля Password. Максимум 65 535 байт. Важливо знати, що ім'я та пароль передаються у відкритому вигляді, тому якщо дані передаються по публічних мережах,

необхідно використовувати SSL для шифрування підключення.

Як уже говорилося вище, у протоколі MQTT підключення завжди ініціюють клієнти, незалежно від того, чи вони є одержувачами (Subscriber) або постачальниками (Publisher) даних. Розберемо пакет із встановленням з'єднання, перехоплений за допомогою програми Wireshark [25].

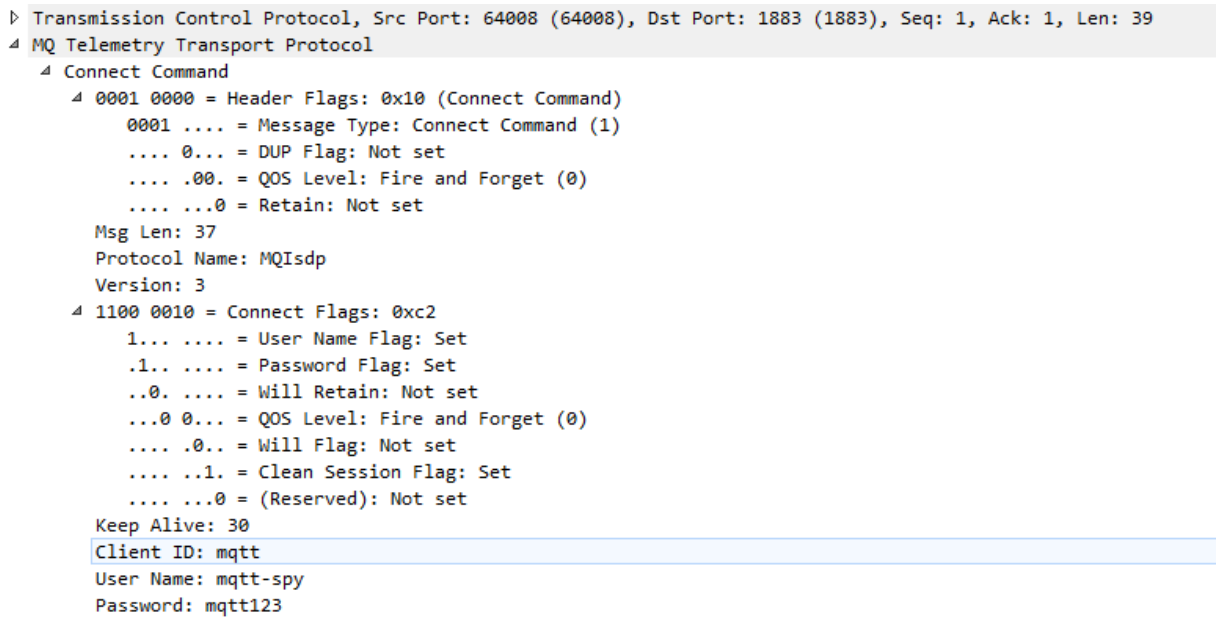


Рис 2.14 - Пакет з опцією MQTT, переданий нешифрованим каналом

У TCP заголовку видно, що пакет переданий портом 1883, тобто шифрування немає, отже у відкритому вигляді доступні всі дані, зокрема логін і пароль.

Тип повідомлення - Connect (команда 0x0001), встановлення з'єднання з брокером. Основні команди: Connect, Disconnect, Publish, Subscribe, Unsubscribe. Існують також команди підтвердження отримання, keep alive, і т.д.

Прапор DUP означає, що повідомлення передається повторно, використовується тільки в типах повідомлень PUBLISH, SUBSCRIBE, UNSUBSCRIBE, PUBREL, для випадків, коли брокер не отримав підтвердження отримання попереднього повідомлення.

Рівень QoS – прапор Quality of Service. Ми розберемо цю тему докладніше.

Retain – дані, опубліковані з прапором retain, зберігаються на брокері. При наступній підписці на цей топік, брокер одразу надішле повідомлення із цим прапором. Використовується лише у повідомленнях із типом Publish.

2.3.3 Огляд протоколу Modbus

Протокол Modbus – найпоширеніший промисловий протокол для M2M-взаємодії. Є стандартом де-факто та підтримується майже всіма виробниками промислового обладнання [26].

Завдяки універсальності та відкритості стандарт дозволяє інтегрувати обладнання різних виробників. Modbus використовується для збору показання з датчиків, управління реле та контролерами, моніторингу, і т.д.

Modbus був представлений 1979 року компанією Modicon (нині Schneider Electric). Це був відкритий стандарт, який працює за інтерфейсом RS-232. Пізніше з'явилася продаж протоколу для інтерфейсів RS-485 і Modbus TCP. Протокол швидко набрав популярності, і багато виробників стали впроваджувати його у своїх пристроях.

Пізніше права на протокол були передані некомерційній організації Modbus Organization, яка досі має стандарт.

В описі стандарту Modbus використовують термінологію, успадковану від мов релейної логіки. Приміром, деякі регістри називаються котушками (англ. coil).

Рис. 2.15 – Апаратна взаємодія протоколу Modbus

На фізичному рівні - RS-232/422/485 - послідовні інтерфейси, широко поширені в промисловості. Інтерфейси RS-422/485 забезпечують дальність сигналу до 1200 метрів. Використовуються протоколи Modbus RTU/ASCII. Мережі TCP/IP - фізичним каналом передачі можуть будь-які ethernet-інтерфейси. Використовується протокол Modbus TCP.

Modbus ASCII. Дані кодуються символами з таблиці ASCII та передаються у шістнадцятковому форматі. Початок кожного пакета позначається символом двокрапки, а кінець - символами повернення каретки та перенесення рядка. Це дозволяє використовувати протокол на лініях з великими затримками та обладнанням із менш точними таймерами.

Modbus RTU. У протоколі Modbus RTU дані кодуються у двійковий формат, і роздільником пакетів служить часовий інтервал. Цей протокол критичний до затримок і може працювати, наприклад, на модемних лініях. При цьому, накладні витрати на передачу даних менше, ніж Modbus ASCII, так як довжина повідомлень менше.

Modbus TCP. Структура пакетів схожа з Modbus RTU, дані також кодуються в двійковий формат, і упаковуються в стандартний TCP-пакет, передачі по IP-мережам. Перевірка цілісності, що використовується Modbus RTU, не застосовується, оскільки TCP вже має власний механізм контролю цілісності.

Всі пристрої Modbus взаємодіють, слідуючи моделі master-slave. Запити може ініціювати тільки master-пристрій, slave-пристрої можуть відповідати на запити, і можуть самостійно починати передачу даних. Залежно від реалізації протоколу заголовки пакета різняться. Ось основні складові пакета, які важливо знати:

ADU (Application Data Unit) - пакет Modbus повністю, з усіма заголовками, PDU, контрольною сумою, адресою та маркерами. Відрізняється залежно від реалізації протоколу.

PDU (protocol data unit) — основна частина пакету, однакова всім реалізацій протоколу. Містить сам платіж.

Адреса пристрою – адреса одержувача, тобто slave-пристрою. В одному сегменті Modbus-мережі можуть бути до 247 пристроїв. Тільки slave-пристрої мають адреси, що відрізняються, master-пристрій не має адреси. Адреса «0» використовується для широкомовних запитів від master, при цьому slave-пристрої не можуть відповідати на ці широкомовні пакети.

Контрольна сума – алгоритми перевірки цілісності пакетів. У Modbus RTU та ASCII використовується 2 байти контрольної суми. У Modbus RTU застосовується алгоритм CRC16, Modbus ASCII — більш простий і менш надійний LRC8. Modbus TCP контрольна сума не додається в ADU, так як цілісність перевіряється на рівні TCP.

У спрощеному вигляді структура запитів Modbus складається з коду функції (читання/запис), і даних, які потрібно рахувати або записати. При цьому коди функції відрізняються для різних типів даних. Розберемо, які бувають регістри, та функції для роботи з ними.

Discrete Inputs – дискретні входи пристрою, доступні лише для читання. Діапазон адрес регістрів: з 10001 до 19999. Мають функцію «02» - читання групи регістрів

Coils — дискретні виходи пристрою або внутрішні значення. Доступні для читання та запису. Діапазон адрес регістрів: з 20001 по 29999. Має функції: "01" - читання групи регістрів, "05" - запис одного регістру, "15" - запис групи регістрів

Input Registers – 16-бітові входи пристрою. Доступні лише для читання. Діапазон адрес регістрів: з 30001 до 39999. Мають функцію: «04» - читання групи регістрів

Holding Registers - 16-бітні виходи пристрою, або внутрішні значення. Доступні для читання та запису. Діапазон адрес регістрів: з 40001 до 49999.

Незважаючи на назви, входи і виходи можуть бути внутрішніми змінними, зберігати лічильники, прапори, або бути керуючими тригерами. Існують також інші діапазони регістрів, але в переважній більшості пристроїв вони не використовуються, тому ми розглянемо чотири основних

типи регістрів. У різних пристроях можуть бути задіяні різні діапазони регістрів, або все відразу.

2.3.4 Огляд протоколу COAP

CoAP (Constrained Application Protocol) було розроблено на вирішення завдання взаємодії пристроїв з обмеженими ресурсами. Майже всі кінцеві пристрої інтернету речей можна віднести до цього визначення. І саме такі пристрої виконують основну роботу. Концепція CoAP значно відрізняється від MQTT і орієнтована на взаємодію точки-точки (клієнт-сервер). Клієнт звертається до сервера і посилає йому прості команди типу PUT, POST, GET, DELETE, зміст яких зрозумілий з назви та аналогічний HTTP. З цієї точки зору можна сказати "зі спрощеннями", що COAP - це такий спрощений HTTP, у якого мало ресурсів. В результаті, варто відзначити легку і просту інтеграцію CoAP с HTTP. Звичайний користувач через браузер може інтегрувати систему управління пристроями інтернету речей у звичайну web-додаток і не помічати цього, а в деяких випадках навіть не здогадуватися про це. З цього погляду протокол відповідає вимогам RESTfull [27].

Розглянемо роботу з CoAP практично. Одним із перших питань, що виникають у процесі освоєння CoAP є практично повна відсутність графічних інструментів для роботи з ним. Існує досить багато реалізацій під різні мови програмування та платформи. Однак практично всі вони представляють рішення у вигляді бібліотек, що відповідають стандарту RFC 7252. Серед інструментів варто відзначити плагін Copper для Firefox. Однак починаючи з 56 версії та переходу на технологію WebExtension, він перестав працювати. Як альтернатива можна назвати Cu4Cr (Correr4Chrome) призначений, як не важко здогадатися, для браузера від Google. Проте, його установка не така проста і очевидна, як була раніше для Firefox. Необхідно виконати дії згідно з інструкцією. Після цього можна буде взаємодіяти з сервером CoAP прямо з браузера.

Модель взаємодії CoAP схожа на модель клієнт/сервер HTTP. Однак,

як правило, виникають міжмашинні взаємодії у реалізації CoAP, що виконує роль як клієнта, так і сервера. А запит CoAP еквівалентний запиту HTTP і надсилається клієнтом до запитати дію (за допомогою коду методу) на ресурсі (ідентифікований за URI) на сервері. Потім сервер надсилає відповідь з Відповіддю код; ця відповідь може включати подання ресурсу [28].

На відміну від HTTP, CoAP має справу з цими обмінами асинхронно через а транспорт, орієнтований на дейтаграми, такий як UDP. Це зроблено логічно використовуючи рівень повідомлень, який підтримує додаткову надійність. CoAP визначає чотири типи повідомлень: Підтверджується, Не підтверджується, Підтвердження, Скидання.

Коди методів і коди відповіді, включені в деякі з цих повідомлень, змушують їх переносити запити чи відповіді. Основні обміни чотирьох типів повідомлення дещо ортогональні запиту/відповіді взаємодії; запити можуть передаватися в підтверджуваних і непідтверджувані повідомлення, і в них також можна переносити відповіді як контрейнерний у повідомленнях підтвердження.

Можна було б логічно думати про CoAP як про використання двошарового підходу, рівень обміну повідомленнями CoAP використовується для роботи з UDP та асинхронним характером взаємодій та взаємодії запит/відповідь використовуючи коди методу та відповіді (див. рисунок 2.16). Однак CoAP є єдиним протокол, з обміном повідомленнями та запитом/відповіддю як лише функціями заголовка CoAP.

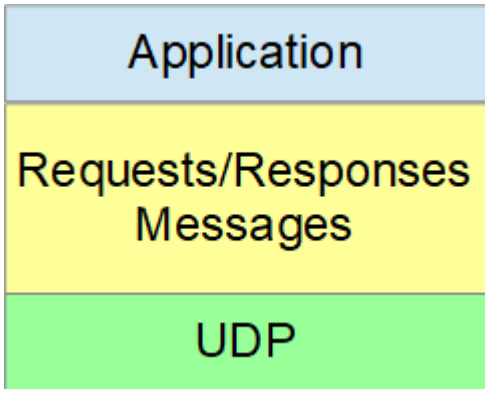


Рис. 2.16 – Шари протоколу CoAP

2.3.5 Огляд протоколу AMQP

Розширений протокол черги повідомлень (Advanced Message Queuing Protocol - AMQP) — це відкритий стандартний протокол прикладного рівня для проміжного програмного забезпечення, орієнтованого на повідомлення. Визначальними характеристиками AMQP є орієнтація повідомлень, черга, маршрутизація (включаючи «точка-точка» та опублікування та підписки), надійність та безпека [29].

AMQP визначає поведінку постачальника повідомлень і клієнта в тій мірі, в якій реалізації різних постачальників є взаємосумісними, так само, як SMTP, HTTP, FTP тощо створили взаємодіючі системи. Попередні стандартизації проміжного програмного забезпечення відбувалися на рівні API (наприклад, JMS) і були зосереджені на стандартизації взаємодії програміста з різними реалізаціями проміжного програмного забезпечення, а не на забезпеченні взаємодії між кількома реалізаціями. На відміну від JMS, який визначає API та набір поведінки, які має забезпечити реалізація обміну повідомленнями, AMQP є протоколом на рівні проводів. Протокол рівня проводів — це опис формату даних, які надсилаються по мережі у вигляді потоку байтів. Отже, будь-який інструмент, який може створювати та інтерпретувати повідомлення, які відповідають цьому формату даних, може взаємодіяти з будь-яким іншим сумісним інструментом, незалежно від мови реалізації.

AMQP — це двійковий протокол прикладного рівня, розроблений для ефективної підтримки широкого спектру програм обміну повідомленнями та шаблонів зв'язку. Він забезпечує контрольований потоком зв’язок, орієнтований на повідомлення, з гарантіями доставки повідомлень, наприклад, щонайменше один раз (коли кожне повідомлення доставляється один раз або ніколи), щонайменше один раз (де кожне повідомлення обов’язково буде доставлено, але може робити це кілька разів) і точно один раз (де повідомлення обов’язково прийде і тільки один раз), і аутентифікація та/або шифрування на основі SASL та/або TLS. Він передбачає наявність

базового надійного протоколу транспортного рівня, такого як протокол керування передачею (TCP).

Специфікація AMQP визначається на кількох рівнях: (i) система типів, (ii) симетричний, асинхронний протокол для передачі повідомлень від одного процесу до іншого, (iii) стандартний, розширюваний формат повідомлення і (iv) набір стандартизованих, але розширюваних «можливостей обміну повідомленнями».

AMQP визначає схему кодування, що самоописує, що дозволяє сумісне представлення широкого діапазону часто використовуваних типів. Це також дозволяє додавати до введених даних додаткове значення[17], наприклад, певне значення рядка може бути анотовано, щоб його можна було зрозуміти як URL-адресу. Так само значення карти, що містить пари ключ-значення для "ім'я", "адреса" тощо, може бути анотовано як подання типу "клієнт".

Система типів використовується для визначення формату повідомлення, що дозволяє виражати і розуміти стандартні та розширені метадані об'єктами обробки. Він також використовується для визначення комунікаційних примітивів, через які обмінюються повідомленнями між такими об’єктами, тобто тілами кадру AMQP.

Основною одиницею даних в AMQP є кадр. Визначено дев’ять тіл кадру AMQP, які використовуються для ініціації, контролю та припинення передачі повідомлень між двома одноранговими. Це:

- відкрити (з'єднання);
- розпочати (сесію);
- прикріпити (посилання);
- передача;
- потік;
- диспозиція;
- від'єднати (посилання);
- закінчити (сесію);
- закрити (з'єднання).

Протокол з'єднання лежить в основі AMQP. Тіло прикріпленого кадру надсилається, щоб ініціювати нове посилання; від'єднати, щоб розірвати посилання. Посилання можуть бути встановлені для отримання або надсилання повідомлень.

Повідомлення надсилаються за встановленим посиланням за допомогою кадру передачі. Повідомлення по посиланню надходять лише в одному напрямку.

Перекази підпадають під дію кредитної схеми контролю потоків, яка керується за допомогою кадрів потоків. Це дозволяє процесу захистити себе від перевантаження занадто великою кількістю повідомлень або, простіше кажучи, дозволити посиланню для підписки отримувати повідомлення за бажанням.

Кожне передане повідомлення в кінцевому підсумку має бути розраховано. Розрахунок гарантує, що відправник і одержувач узгоджують стан переказу, забезпечуючи гарантії надійності. Зміни в стані та розрахунках за переказ (або набір переказів) повідомляються між одноранговими за допомогою фрейму розміщення. Різні гарантії надійності можуть бути застосовані таким чином: якнайбільше один раз, принаймні один раз і точно один раз.

Кілька посилань в обох напрямках можна згрупувати разом у сеансі. Сеанс — це двонаправлена послідовна розмова між двома одноранговими користувачами, яка починається початковим кадром і завершується кінцевим кадром. З’єднання між двома одноранговими партнерами може мати кілька сеансів, мультиплексованих через нього, кожен з яких логічно незалежний. З'єднання ініціюються відкритим фреймом, в якому виражені можливості однорангового партнера-відправника, і завершується закритим фреймом.

AMQP визначає як оголене повідомлення ту частину повідомлення, яка створюється програмою-відправником. Це вважається незмінним, оскільки повідомлення передається між одним або кількома процесами.

Забезпечення незмінності повідомлення, надісланого програмою,

дозволяє наскрізне підписувати та/або шифрувати повідомлення та гарантує, що будь-які перевірки цілісності (наприклад, хеші або дайджести) залишаються дійсними. Повідомлення можуть бути анотовані посередниками під час передачі, але будь-які такі анотації зберігаються окремо від незмінного відкритого повідомлення. Анотації можуть бути додані до або після голого повідомлення.

Заголовок — це стандартний набір анотацій, пов’язаних з доставкою, які можна запитати або вказати для повідомлення, і включає час життя, довговічність, пріоритет.

Саме відкрите повідомлення структурується як додатковий список стандартних властивостей (ідентифікатор повідомлення, ідентифікатор користувача, час створення, відповідь, тема, ідентифікатор кореляції, ідентифікатор групи тощо), необов'язковий список властивостей програми (тобто розширені властивості) і тіло, яке AMQP називає даними програми.

Властивості вказуються в системі типів AMQP, як і анотації. Дані програми можуть бути будь-якої форми та в будь-якому кодуванні, яке вибирає програма. Одним з варіантів є використання системи типу AMQP для надсилання структурованих даних, що самоописуються.

2.4 Вибір апаратного комплексу розробки системи

2.4.1 Вибір мікроконтролеру вузлу сенсорної мережі

У першу чергу необхідно визначитись з мікроконтролером, що буде забезпечувати роботу вузлу сенсорної мережі. Вибір необхідно робити з урахуванням ціни та функціональної можливості чипів. У наш час існує багато лінійок мікроконтролерів різної розрядності, характеристик та цінової політики.

У якості найдешевшого та енергоефективного варіанту (як вузол з одним сенсором та радіообміном) доцільним є розглянути мікроконтролер ATtiny85, що вартує приблизно 1,5$.

За даними офіційної документації [30] основні характеристики наступні:

• Високопродуктивний 8-розрядний мікроконтролер AVR® малої потужності

• Розширена архітектура RISC:

– 120 потужних інструкцій – Виконання найбільшого тактового циклу.

– Робочі реєстри загального призначення 32 x 8.

– Повністю статична робота.

• Енергонезалежна пам'ять програм і даних:

– 8 Кбайт внутрішньосистемної програмованої пам’яті програм.

• Витривалість: 10 000 циклів запису/стирання:

– 512 байт внутрішньосистемна програмована EEPROM.

• Витривалість: 100 000 циклів запису/стирання:

– 512 байт внутрішньої SRAM.

– Блокування програмування для самопрограмованої флеш-програми та безпеки даних EEPROM.

• Периферійні функції:

– 8-бітний таймер/лічильник з попереднім масштабуванням і двома каналами ШІМ.

– 8-розрядний високошвидкісний таймер/лічильник.

• 2 високочастотних ШІМ виходи з окремими регістрами порівняння вихідних сигналів.

• Програмований генератор мертвого часу:

– USI – універсальний послідовний інтерфейс з детектором умов запуску.

– 10-розрядний АЦП.

• 4 односторонні канали:

• 2 пари диференціальних каналів АЦП з програмованим посиленням (1x, 20x).

• Вимірювання температури:

– Програмований сторожовий таймер з окремим вбудованим осцилятором.

– Вбудований аналоговий компаратор.

• Спеціальні функції мікроконтролера:

– Вбудована система налагодження debugWIRE.

– Внутрішньосистемне програмування через порт SPI.

– Зовнішні та внутрішні джерела переривань.

– Режими холостого ходу з низьким енергоспоживанням, зменшення шуму АЦП та режими вимкнення живлення.

– Покращена схема скидання увімкнення живлення.

– Програмована схема виявлення прогоряння.

– Внутрішній калібрований генератор.

• I/O та пакети:

– Шість програмованих ліній вводу/виводу.

– 8-контактний PDIP, 8-контактний SOIC, 20-контактний QFN/MLF і 8-контактний TSSOP (тільки ATtiny45/V).

• Робоча напруга:

– 1,8 - 5,5 В для ATtiny25 В/45 В/85 В.

– 2,7 - 5,5 В для ATtiny25/45/85.

• Оцінка швидкості:

– ATtiny25V/45V/85V: 0 – 4 МГц при 1,8 – 5,5 В, 0 – 10 МГц при 2,7 – 5,5 В.

– ATtiny25/45/85: 0 – 10 МГц при 2,7 – 5,5 В, 0 – 20 МГц при 4,5 – 5,5 В.

• Промисловий температурний діапазон.

• Низьке споживання електроенергії.

- Активний режим:

• 1 МГц, 1,8 В: 300 мкА:

– Режим вимкнення живлення:

• 0,1 мкА при 1,8 В.

Постачається у корпусах двох типів – 8pin DIP та SOC SMD (20 pin):

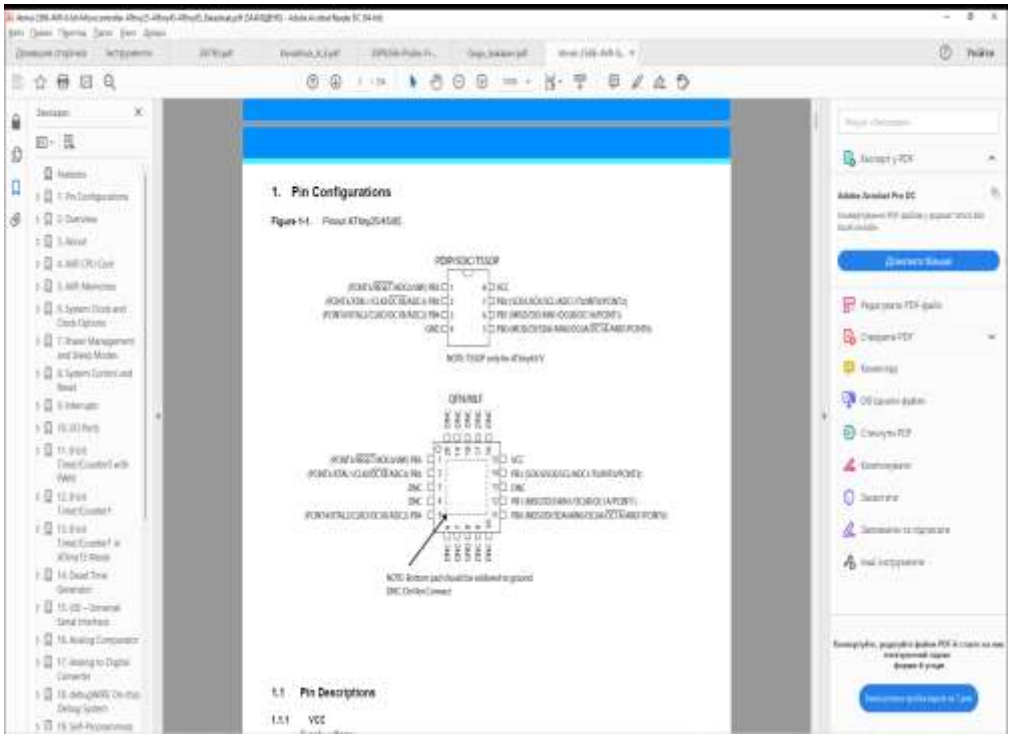


Рис. 2.17 – Розпіновка контактів мікроконтролера

ATtiny85 - це 8-розрядний CMOS мікроконтролер з низьким енергоспоживанням на основі вдосконаленої архітектури RISC AVR. Виконуючи потужні інструкції за один тактовий цикл, ATtiny25/45/85 досягає пропускної здатності, що наближається до 1 MIPS на МГц, що дозволяє розробнику системи оптимізувати споживання енергії порівняно зі швидкістю обробки.

Ядро AVR поєднує в собі багатий набір інструкцій з 32 робочими регістрами загального призначення. Усі 32 регістри безпосередньо підключені до арифметичного логічного блоку (ALU), що дозволяє отримати доступ до двох незалежних регістрів в одній команді, що виконується за один такт. Отримана в результаті архітектура є більш ефективною для коду, досягаючи пропускної здатності в десять разів швидше, ніж звичайні мікроконтролери CISC.

ATtiny85 надає такі функції: 8 Кбайт внутрішньосистемного програмованого Flash, 512 байт EEPROM, 512 байт SRAM, 6 ліній вводу-

виводу загального призначення, 32 робочих регістра загального призначення, один 8-розрядний таймер/лічильник з режимами порівняння, один 8-розрядний високошвидкісний таймер/лічильник, універсальний послідовний інтерфейс, внутрішні та зовнішні переривання, 4-канальний 10-розрядний АЦП, програмований сторожовий таймер з внутрішнім осцилятором і три програмно вибираних режими енергозбереження. Режим очікування зупиняє ЦП, дозволяючи SRAM, таймеру/лічильнику, АЦП, аналоговому компаратору та системі переривань продовжувати функціонувати. Режим вимкнення зберігає вміст реєстру, відключення всіх функцій мікросхеми до наступного переривання або апаратного скидання. Режим шумозаглушення АЦП зупиняє ЦП і всі модулі вводу/виводу, крім АЦП, щоб мінімізувати шум перемикання під час перетворення АЦП.

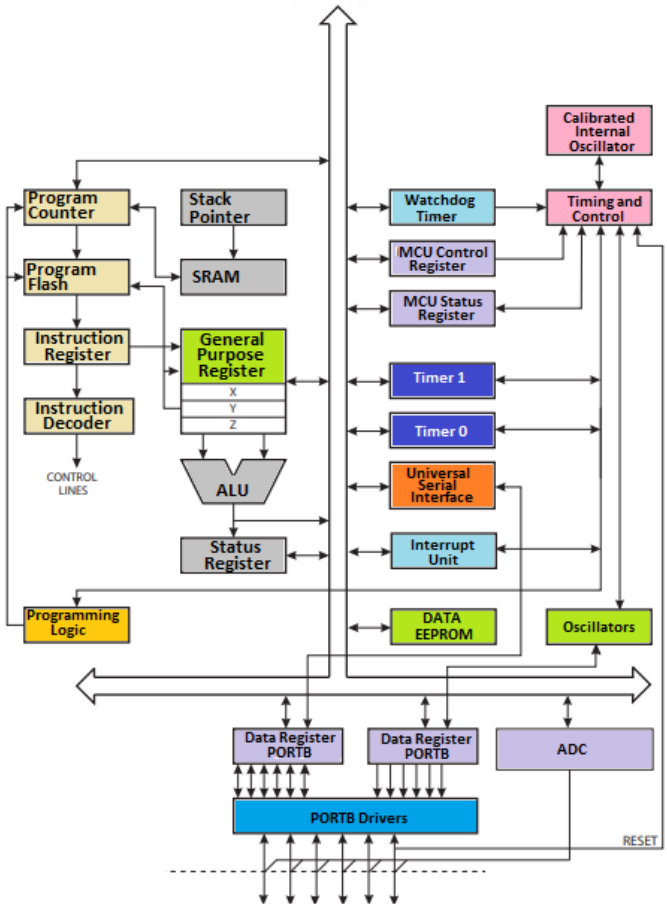


Рис. 2.18 – Блок схема архітектури мікроконтролера

Пристрій виготовлено з використанням технології енергонезалежної

пам’яті Atmel високої щільності. Вбудований ISP Flash дозволяє перепрограмувати пам’ять програми в системі через послідовний інтерфейс SPI, за допомогою звичайного програматора енергонезалежної пам’яті або за допомогою коду завантаження на чіпі, що працює на ядрі AVR.

ATtiny25/45/85 AVR підтримується повним набором інструментів для розробки програм і системи, включаючи: компілятори C, ассемблери макросів, налагоджувач програм/симулятори та набори для оцінки.

Якщо перший рівень локальної мережі організовувати через WiFi, то доцільним буде використання мікроконтролеру ESP8266, який коштує 2$, та має вбудований модуль.

ESP8266 — це недорогий мікрочіп Wi-Fi з вбудованим програмним забезпеченням для роботи в мережі TCP/IP та можливістю мікроконтролера, вироблений Espressif Systems у Шанхаї, Китай.

Чіп був популяризований в англомовній спільноті виробників у серпні 2014 року за допомогою модуля ESP-01, виготовленого стороннім виробником Ai-Thinker. Цей невеликий модуль дозволяє мікроконтролерам підключатися до мережі Wi-Fi і встановлювати прості TCP/IP-з’єднання за допомогою команд у стилі Хейса. Однак спочатку англомовної документації щодо чіпа та команд, які він приймав, майже не було. Дуже низька ціна та той факт, що на модулі було дуже мало зовнішніх компонентів, що свідчило про те, що він в кінцевому підсумку може бути дуже недорогим за обсягом, привернули багатьох хакерів до вивчення модуля, чіпа та програмного забезпечення на ньому, а також щоб перекласти китайську документацію.[31]

Загальні характеристики мікроконтролеру:

Процесор: L106 32-розрядне ядро мікропроцесора RISC на основі Tensilica Diamond Standard 106Micro, що працює на частоті 80 МГц.

Пам'ять:

Оперативна пам'ять з інструкціями 32 КБ.

32 Кбайт кеш-пам'яті інструкцій.

80 КБ оперативної пам’яті з даними користувача.

Оперативна пам'ять системних даних 16 КБ ETS.

Зовнішній флеш-пам’ять QSPI: підтримується до 16 МБ (як правило, від 512 КБ до 4 МБ).

IEEE 802.11 b/g/n Wi-Fi.

Аутентифікація WEP або WPA/WPA2 або відкриті мережі.

17 контактів GPIO.

Шина послідовного периферійного інтерфейсу (SPI).

I²C (програмна реалізація).

I²S інтерфейси з DMA (спільний доступ до GPIO).

UART на виділених контактах, а також UART тільки для передачі можна ввімкнути на GPIO2.

10-розрядний АЦП (АЦП послідовного наближення) [32].

Окрім самого мікроконтролеру існує багато варіантів їхної інтеграції до плат розробки, серед яких можна виділити NodeMCU ESP8266.

NodeMCU — це прошивка та плата для розробки на базі Lua з відкритим вихідним кодом, спеціально призначена для додатків на основі IoT. Він включає в себе мікропрограму, яка працює на ESP8266 Wi-Fi SoC від Espressif Systems, і апаратне забезпечення, яке базується на модулі ESP-12.

Плата розробника NodeMCU ESP8266 поставляється з модулем ESP-12E, що містить чіп ESP8266 з 32-розрядним мікропроцесором Tensilica Xtensa LX106 RISC. Цей мікропроцесор підтримує ОСРЧ і працює на частоті від 80 МГц до 160 МГц. NodeMCU має 128 КБ оперативної пам’яті та 4 МБ флеш-пам’яті для зберігання даних і програм. Його висока обробна потужність із вбудованим Wi-Fi / Bluetooth і функціями глибокого сну робить його ідеальним для проектів IoT [33].

NodeMCU можна живити за допомогою роз’єму Micro USB і контакту VIN (зовнішнього живлення). Він підтримує інтерфейс UART, SPI та I2C.

Рис. 2. 19 – Зовнішній вигляд плати розширення

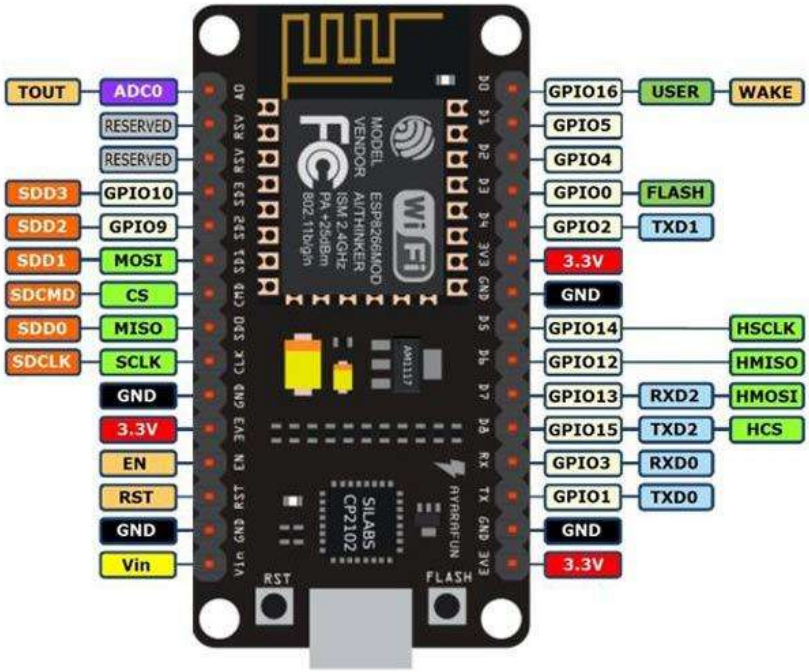


Рис. 2.20 – Виведення контактів плати розширення

Плату розширення NodeMCU можна легко запрограмувати за допомогою Arduino IDE, оскільки вона проста у використанні.

Програмування NodeMCU за допомогою Arduino IDE навряд чи займе 5-10 хвилин. Все, що вам потрібно, це Arduino IDE, USB-кабель і сама плата NodeMCU.

2.4.2 Вибір датчиків та сенсорів

Датчик тиску и вологості повітря (BME280). Модуль датчика BME280

(рис 2.21) (температура, вологість, тиск) - нове покоління датчиків тиску, що дозволяють вимірювати не тільки значення атмосферного тиску, а й температуру і вологість. Датчик характеризується високою точністю вимірювання, високою швидкодією інтерфейсу та надмалим споживанням. Для підключення використовується I2C [34].

Рисунок 2.21 – Датчик вимірювання тиску і вологості BME280

Характеристики:

- Інтерфейси підключення: I2C;
- Максимальна швидкодія інтерфейсу: I2C до 3.4МГц;
- Межі вимірювання температури: від -40 до 85 градусів;
- Точність вимірювання температури: від 0.5 до 1 градуса;
- Межі вимірювання вологості: від 0 до 100%;
- Точність вимірювання вологості: 3%;
- Межі вимірювання тиску: від 300 до 1100 гПа;
- Точність вимірювання тиску: 1гПа;
- Напруга живлення: від 1.8 до 5 В;
- Струм в режимі вимірювання тиску: 714 мкА;
- Струм в режимі вимірювання вологості: 340 мкА;
- Споживаний струм в режимі вимірювання температури: 350 мкА;
- Струм в режимі сну: від 0.1 мкА до 0.5 мкА;
- Розміри модуля: 15 х 12 х 3 мм.

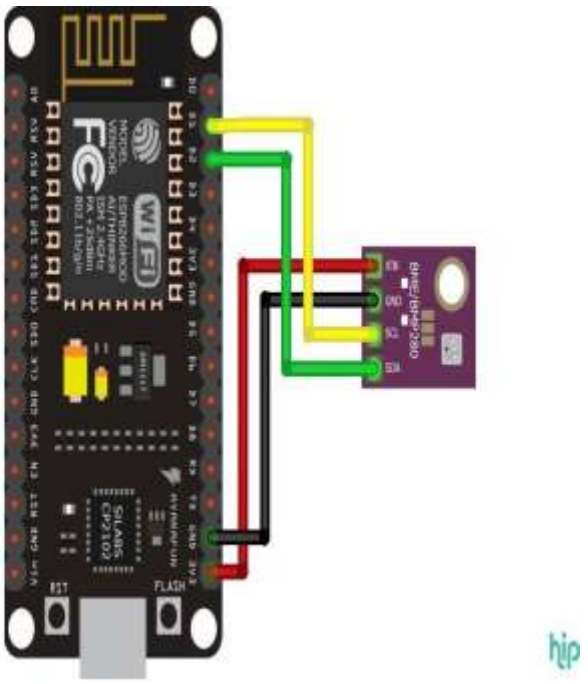


Рисунок 2.22 – Підключення BME280 до ESP8266

Датчик вимірювання температури (DS18BB20). DS18B20 це цифровий вимірювач температури, з дозволом перетворення 9 - 12 розрядів і функцією тривожного сигналу контролю за температурою (рис. 2.23). Параметри контролю можуть бути задані користувачем і збережені в енергонезалежній пам'яті датчика.

Рисунок 2.23 – Датчик температури DS18B20

DS18B20 обмінюється даними з мікроконтролером за допомогою однодротової лінії зв'язку, використовуючи протокол інтерфейсу 1-Wire.

Живлення датчик може отримувати безпосередньо від лінії даних, без використання зовнішнього джерела. В цьому режимі живлення датчика походить від енергії, запасеної на паразитної ємності.

Діапазон вимірювання температури становить від -55 до +125 С. Для діапазону від -10 до +85 С похибка не перевищує 0,5 С.

У кожної мікросхеми DS18B20 є унікальний серійний код довжиною 64 розряду, який дозволяє декільком датчикам підключатися на одну загальну лінію зв'язку. Тобто через один порт мікроконтролера можна обмінюватися даними з декількома датчиками, розподіленими на значній відстані. Режим вкрай зручний для використання в системах екологічного контролю, моніторингу температури в будівлях, вузлах устаткування [35].

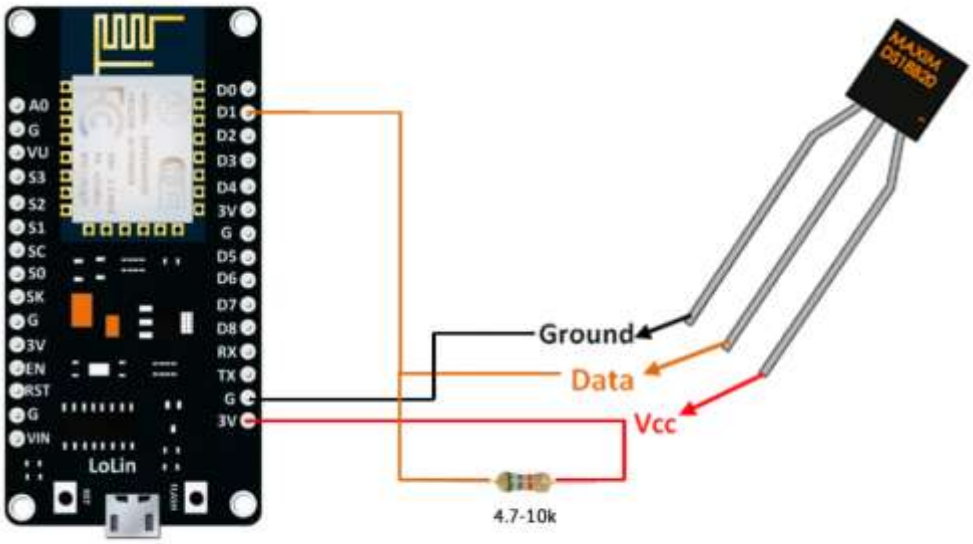


Рисунок 2.24 - Підключення DS18B20 до ESP8266

Датчик DHT-11 та DHT-22. Датчик температури та вологості DHT11 має комплекс датчиків температури та вологості з каліброваним цифровим вихідним сигналом. Використовуючи ексклюзивну техніку отримання цифрового сигналу та технологію датчика температури та вологості, він забезпечує високу надійність і чудову довгострокову стабільність. Цей датчик включає в себе компонент вимірювання вологості резистивного типу та компонент вимірювання температури NTC, а також підключається до

високопродуктивного 8-розрядного мікроконтролера, забезпечуючи чудову якість, швидку реакцію, захист від перешкод і економічну ефективність [36].

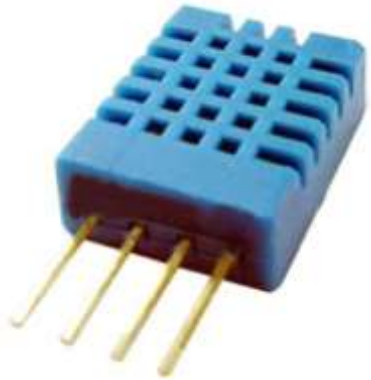


Рис. 2.25 – Датчик вимірювання DHT-11

Технічні характеристики DHT11

Робоча напруга: від 3,5 до 5,5 В

Робочий струм: 0,3 мА (вимірювання) 60 мкА (режим очікування)

Вихід: послідовні дані

Температурний діапазон: від 0°C до 50°C

Діапазон вологості: від 20% до 90%

Роздільна здатність: температура та вологість 16-бітні

Точність: ±1°C і ±1%

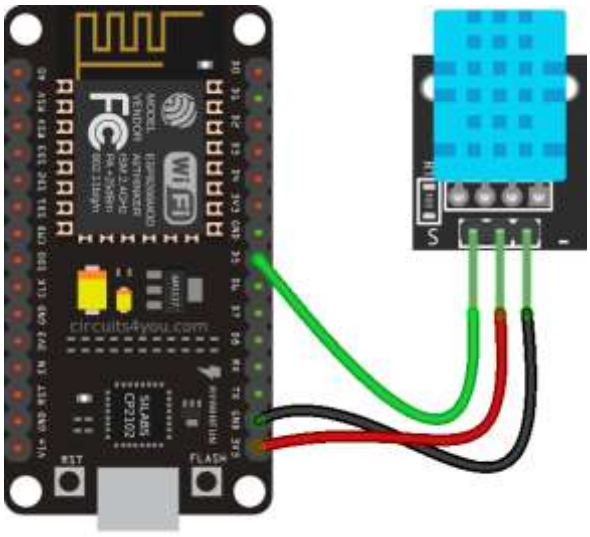


Рис. 2. 26 - Підключення DHT-11 до ESP8266

DHT22 використовує ексклюзивну техніку збору цифрового сигналу та технологію визначення вологості та може видавати калібрований цифровий сигнал. Невеликі розміри, низьке споживання та велика відстань передачі (20 метрів) дозволяють AM2303 бути придатним для всіх видів жорстких застосувань [37].

Напруга живлення має бути від 3,3 В до 6 В постійного струму. Коли на датчик подається живлення, не надсилайте йому жодних інструкцій протягом однієї секунди, щоб передати нестабільний статус. Для фільтрації хвиль між VDD і GND можна додати один конденсатор ємністю 100 нФ.

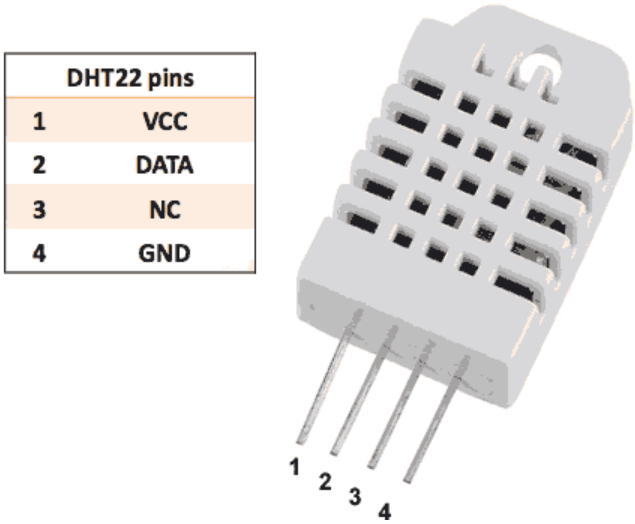


Рис. 2.25 – Датчик вимірювання DHT-22

Характеристики DHT22:

джерело живлення: 3,3 В – 6 В постійного струму;

вихідний сигнал: одношинний;

Чутливий елемент: полімерний конденсатор вологості & DS18B20;

Діапазон вимірювання: вологість 0-100% RH / температура -40°C – 125°C;

точність: вологість ±2% / температура ±0,2°C;

період сприйняття: ~2c.

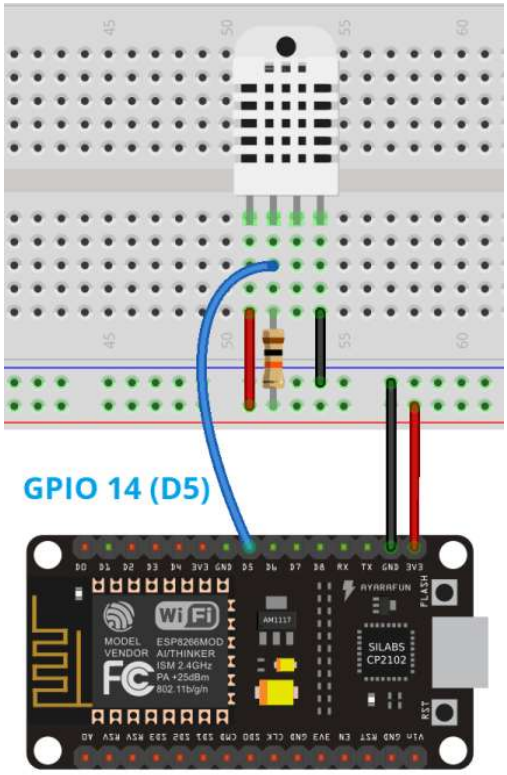


Рис. 2. 26 - Підключення DHT-11 до ESP8266

Датчик газу MQ-2. Датчик MQ-2 відноситься до напівпровідникових приладів. .(Рис 2.7) Принцип роботи датчика заснований на зміні опору тонкоплівкового шару діоксиду олова SnO2 при контакті з молекулами визначається газу.

Рисунок 2.27 – Датчик газу MQ-2

Чутливий елемент датчика складається з керамічної трубки з покриттям Al2O3 і нанесеного на неї чутливого шару діоксиду олова. Всередині трубки проходить нагрівальний елемент, який нагріває чутливий шар до температури, при якій він починає реагувати на визначений газ. Чутливість до різних газів досягається варіюванням складу домішок в чутливому шарі [38].

Діапазон вимірювань:

- Пропан: 200-5000 ppm;
- Бутан: 300-5000 ppm;
- Метан: 500-20000 ppm;
- Водень: 300-5000 ppm.

Характеристики:

- Напруга живлення нагрівача: 5 В;
- Напруга живлення датчика: 3,3-5 В;
- Струм: 150 мА;
- Габарити: 25,4 × 25, 4 мм.

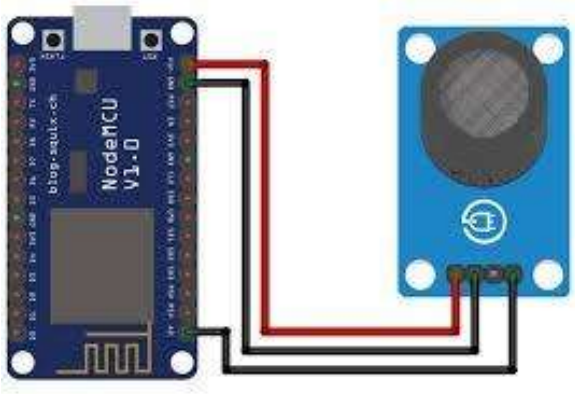


Рисунок 2.28 - Підключення MQ2 до ESP8266

2.4.3 Вибір засобів зв’язку

315/433 МГц РЧ передавач-приймач. У багатьох випадках електронні пристрої необхідно підключати бездротовим способом. У таких випадках

використовується радіочастотне або радіочастотне обладнання. РЧ-модулі включають всі радіохвилі, які можуть проходити різні відстані і досягати приймача відповідно до їх частоти та амплітуди [39].

Модуль радіопередавача-приймача 315/433 МГц включає в себе передавач і приймач, які можуть працювати на частотах 433 МГц і 315 МГц.

Цей модуль випускається в двох різних типах: 315 МГц і 433 МГц.

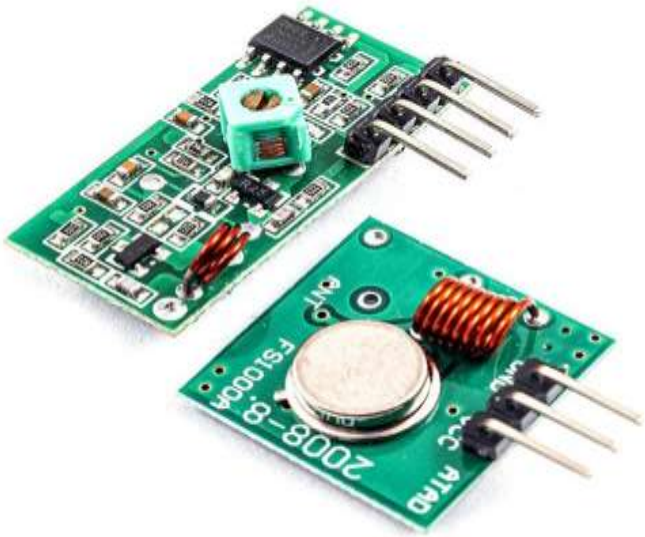


Рис. 2.29 – Радіочастотний приймач/передавач

Ці радіочастотні модулі дуже популярні серед майстрів. Частота 433 МГц використовується в широкому спектрі програм, які потребують бездротового керування. Ці модулі дешеві, і їх можна використовувати з будь-яким мікроконтролером (MCU).

Технічні характеристики Приймач RF 433MHz

- Частотний діапазон: 433 МГц
- Модуляція: ASK
- Вхідна напруга: 5 В

Технічні характеристики RF 433MHz передавач

- Частотний діапазон: 433 МГц
- Вхідна напруга: 3-12В.

2.4.4 Вибір одноплатного комп’ютера

Для розгортання брокера Mosquitto вибирається простий варіант одноплатного комп'ютера, наприклад, одна з першої версії Raspberry Pi Model B 512 МБ RAM. Данний вибір продиктовано буквально тем, що було під рукою. Звичайно, для реального проекту слід окремо вивчити питання про надійність і вартість плати і, крім вибору оптимальних апаратних рішень, також слід розробити питання: надійність корпусу, якість блоку живлення, безпека системи в цілому і т.п. Плату Raspberry Pi ще було цікаво взяти, щоб остаточно розв’язати міф про труднощі старту проекту на базі відкритих апаратних рішень та операційної системи Linux. Звичайно, від розробника вимагаються базові знання, але сучасні платформи стали виключно дружніми з точки зору програмного інтерфейсу і при цьому добре документовані [40].

Рис. 2.30 - Зовнішній вид Raspberry Pi Model B 512 МБ RAM

Raspberry Pi — це комп’ютер розміром із кредитну картку, заснований на системі на чіпі (SoC) BCM2835, який включає процесор ARM11 і потужний графічний процесор. Raspberry Pi підтримує різні дистрибутиви Linux, включаючи Debian, Fedora та Arch Linux. Цей елемент — Raspberry Pi Model B, версія 2.0.

Raspberry Pi був розроблений Raspberry Pi Foundation, щоб забезпечити

доступну платформу для експериментів і навчання комп’ютерного програмування. Raspberry Pi можна використовувати для багатьох речей, які виконує звичайний настільний ПК, включаючи текстову обробку, електронні таблиці, відео високої чіткості, ігри та програмування. USB-пристрої, такі як клавіатури та миші, можна підключати через два USB-порти плати. Завдяки своєму заголовку GPIO на відстані 0,1 дюйма і невеликому розміру Raspberry Pi також працює як програмований контролер у широкому спектрі робототехнічних та електронних додатків. Понад два мільйони Raspberry Pi було продано, і багато ресурсів для Raspberry Pi доступні в Інтернеті.

Особливості:

512 МБ оперативної пам'яті;

порт Ethernet;

Два USB порти;

Два варіанти відеовиходу: HDMI або композитний;

3,5 мм аудіо вихід;

26-контактний роз'єм GPIO з 0,1-дюймовими штифтами, які сумісні з нашими штекерними роз'ємами 2×13 і роз'ємами наших провідників преміум-класу.

2.5 Вибір програмного комплексу проекту

2.5.1 Вибір середовища проектування та розробки

Proteus Design Suite — це власний набір програмних засобів, що використовується в основному для автоматизації електронного проектування. Програмне забезпечення використовується в основному інженерами та техніками електронного дизайну для створення схем та електронних відбитків для виготовлення друкованих плат [41].

Proteus Design Suite — це програма для Windows для створення схеми, моделювання та розробки макета друкованої плати (друкована плата). Його можна придбати в багатьох конфігураціях, залежно від розміру конструкцій,

що виготовляються, і вимог до моделювання мікроконтролера. Усі продукти PCB Design включають автомаршрутизатор і базові можливості моделювання SPICE в змішаному режимі.

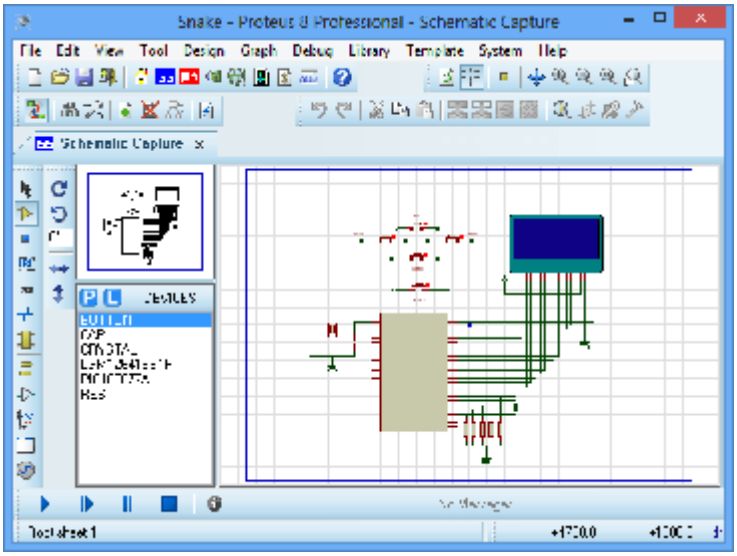


Рис. 2.31 – Інтерфейс програми

Схематичний знімок. Збір схеми в Proteus Design Suite використовується як для моделювання проектів, так і як етап проектування проекту компонування друкованої плати. Тому він є основним компонентом і входить до всіх конфігурацій продукту.

Симуляція мікроконтролера. Симуляція мікроконтролера в Proteus працює шляхом застосування шістнадцяткового файлу або файлу налагодження до частини мікроконтролера на схемі. Потім він моделюється разом з будь-якою аналоговою та цифровою електронікою, підключеною до неї. Це дозволяє використовувати його в широкому спектрі прототипів проектів у таких областях, як керування двигуном, контроль температури та дизайн інтерфейсу користувача. Він також знаходить застосування в загальній спільноті любителів і, оскільки не потрібно обладнання, його зручно використовувати як навчальний або навчальний інструмент. Підтримка доступна для спільного моделювання:

Мікроконтролери Microchip Technologies PIC10, PIC12, PIC16, PIC18,

PIC24, dsPIC33.

Мікроконтролери Atmel AVR (і Arduino), 8051 і ARM Cortex-M3.

Мікроконтролери NXP 8051, ARM7, ARM Cortex-M0 і ARM Cortex-M3.

Мікроконтролери Texas Instruments MSP430, PICCOLO DSP і ARM Cortex-M3.

Parallax Basic Stamp, мікроконтролери Freescale HC11, 8086.

EasyEDA — це веб-набір інструментів EDA, який дозволяє інженерам-технікам проектувати, моделювати, ділитися — публічно та приватно — та обговорювати схеми, моделювання та друковані плати. Інші функції включають створення переліку матеріалів, файлів Gerber, а також файли вибору й розміщення та документальних виходів у форматах PDF, PNG та SVG [42].

EasyEDA дозволяє створювати та редагувати принципові схеми, SPICE моделювання змішаних аналогових і цифрових схем, а також створення та редагування макетів друкованих плат і, за бажанням, виготовлення друкованих плат.

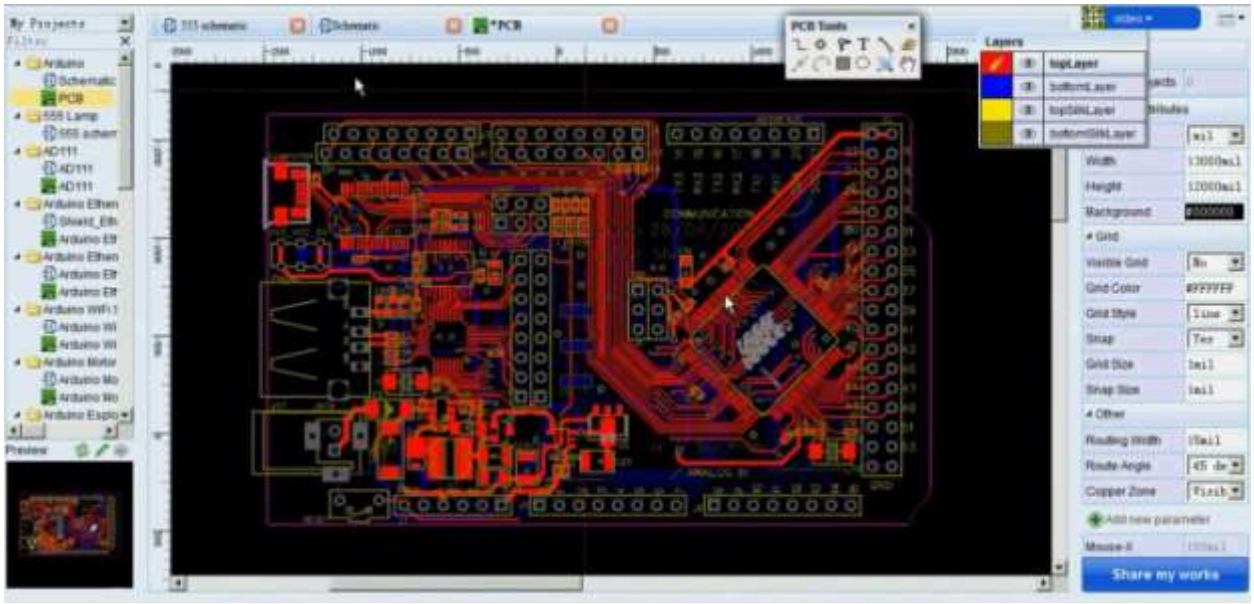


Рис. 2.32 – Інтерфейс програми

Безкоштовне членство пропонується для загальнодоступних та

обмеженої кількості приватних проектів. Кількість приватних проектів можна збільшити, вносячи високоякісні публічні проекти, схематичні символи та посадки друкованих плат та/або сплачуючи щомісячну передплату.

Зареєстровані користувачі можуть безкоштовно завантажувати файли Gerber з інструменту; але за окрему плату EasyEDA пропонує послугу виготовлення друкованих плат. Ця служба також може приймати введення файлів Gerber від сторонніх інструментів.

EasyEDA — це інтегрований браузерний інструмент для захоплення схем, моделювання схеми SPICE на основі Ngspice та компоновки друкованої плати.

Підтримується імпорт із форматів файлів Altium Designer, CircuitMaker, Eagle, Kicad і LTspice, а також загальних списків мереж SPICE. Списки мереж SPICE можна експортувати до інструментів моделювання сторонніх розробників, а також підтримується експорт списків мереж PCB у форматах Altium, PADS і FreePCB.

Можливість імпорту схем і символів LTspice забезпечує корисний спосіб перенесення схем на компоновку друкованої плати без необхідності перемалювати їх з нуля.

Після того, як файли Gerber із готовою конструкцією друкованої плати були завантажені та перевірені - за допомогою стороннього засобу перегляду Gerber - користувач може вибрати виробника друкованої плати або, за певну плату, він може подати Gerbers безпосередньо в EasyEDA для виробництва. Крім того, виведення зображення шару PCB для друку також підтримується у форматах PDF, PNG і SVG для домашнього травлення друкованих плат.

Інструмент також включає в себе функції спільного доступу та спільної роботи, а також вичерпні частини та бібліотеку моделей SPICE, що розширюється.

2.5.2 Вибір мови та середовища програмування

Arduino розробляє, виробляє та підтримує електронні пристрої та програмне забезпечення, що дозволяє людям у всьому світі легко отримувати доступ до передових технологій, які взаємодіють із фізичним світом. Наші продукти прості, прості та потужні, готові задовольнити потреби користувачів від студентів до розробників і аж до професійних розробників [43].

Arduino IDE 1.8.19. Програмне забезпечення Arduino (IDE) з відкритим кодом дозволяє легко писати код і завантажувати його на плату. Це програмне забезпечення можна використовувати з будь-якою платою Arduino.

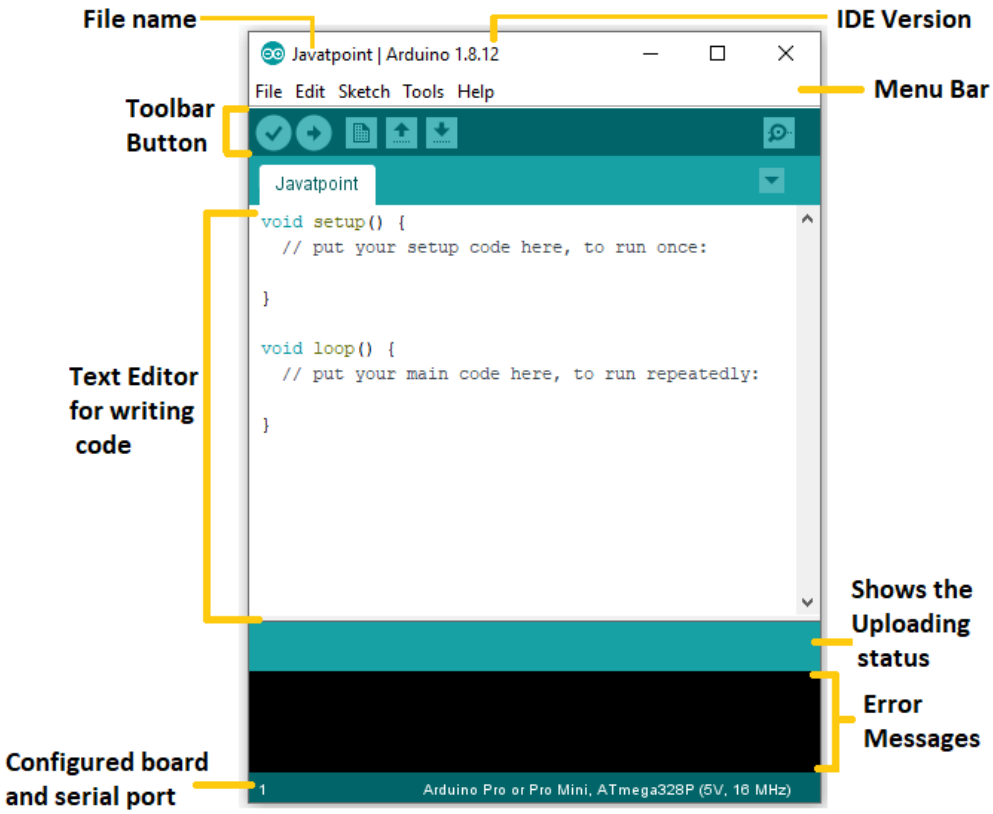


Рис. 2.33 – Інтерфейс програми

Програма IDE підходить для різних операційних систем, таких як Windows, Mac OS X і Linux. Він підтримує мови програмування C і C++. Тут IDE означає інтегроване середовище розробки.

Програму або код, написану в Arduino IDE, часто називають скетчем.

Нам потрібно підключити плату Genuino і Arduino до IDE, щоб завантажити скетч, написаний у програмному забезпеченні Arduino IDE. Скетч зберігається з розширенням ".ino".

2.5.3 Вибір MQTT локального та глобального серверу

Eclipse Mosquitto — це брокер повідомлень з відкритим вихідним кодом (ліцензований EPL/EDL), який реалізує протокол MQTT версії 5.0, 3.1.1 та 3.1. Mosquitto легкий і підходить для використання на всіх пристроях від малопотужних одноплатних комп'ютерів до повноцінних серверів [44].

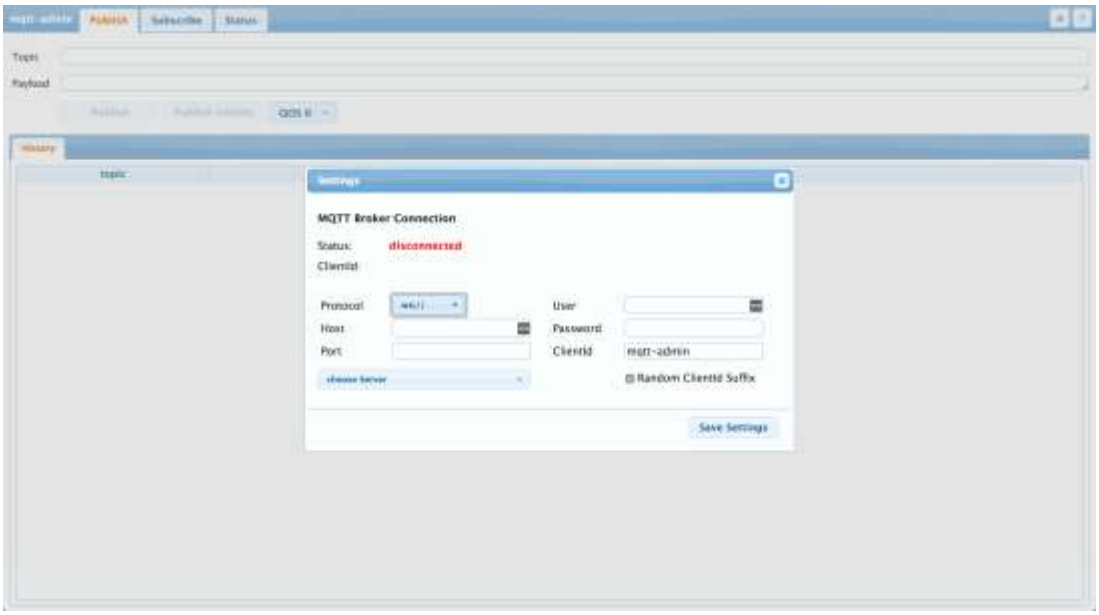


Рис. 2.34 – Інтерфейс програми

Протокол MQTT забезпечує легкий метод обміну повідомленнями за допомогою моделі публікації/підписки. Це робить його придатним для обміну повідомленнями в Інтернеті речей, наприклад із датчиками низької потужності або мобільними пристроями, такими як телефони, вбудовані комп'ютери чи мікроконтролери.

Проект Mosquitto також надає бібліотеку C для реалізації клієнтів MQTT, а також дуже популярні клієнти командного рядка mosquitto_pub і mosquitto_sub MQTT.

Mosquitto є частиною Eclipse Foundation, це проект iot.eclipse.org і спонсорується cedalo.com.

У якості хмарних рішень можна використовувати EMQX MQTT Broker.

EMQX — це масштабуємий, високодоступний, розширений брокер MQTT з повністю відкритим кодом для інтернет-речей, міжмашинної взаємодії та мобільних додатків, який підтримує мільйони одночасних підключень [45].

Основна з релізою 3.0, брокер EMQX повністю підтримує протокол MQTT версії 5.0, і зворотно сумісний з версіями 3.1 і 3.1.1, а також протоколами MQTT-SN, CoAP, LwM2M, WebSocket і STOMP. Основна реліза 3.0, брокер EMQX може масштабуватися до більш ніж 10 мільйонів одночасних MQTT, об’єднаних в одному кластері.

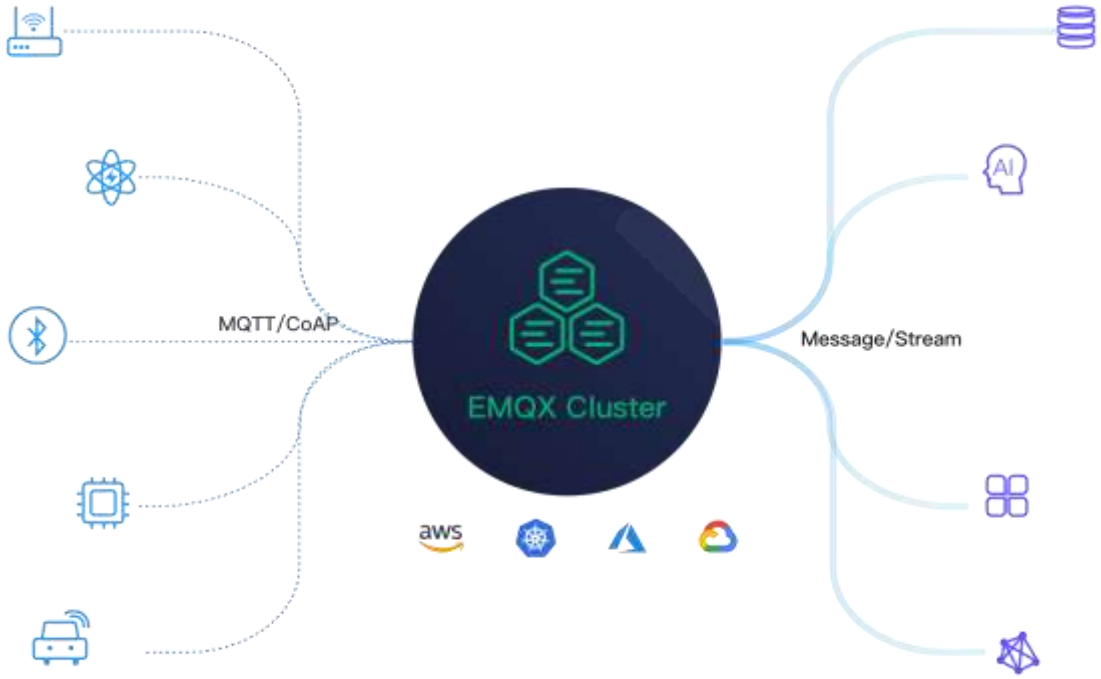


Рис. 2.35 – Блок схема взаємодії брокера

2.6 Висновки за розділом

Було здійснено огляд архітектурних особливостей організації бездротових сенсорних мереж та доведена можливість їхньої інтеграції до

структури локальних та глобальних комп’ютерних мереж. Проаналізовані сучасні протоколи, що можуть бути використані при формуванні бездротової сенсорної мережі та обрано протокол MQTT.

Здійснено підбір апаратних комплектуючих та програмних засобів програмування та проектування. Обрано MQTT брокери локального та глобального користування.

РОЗДІЛ 3
ПРАКТИЧНА РЕАЛІЗАЦІЯ СИСТЕМИ ЕКОЛОГІЧНОГО МОНІТОРИНГУ

3.1 Проектування пристроїв та архітектури мережі

Перш за все необхідно визначитись з моделлю мережі та вузлів з яких вона складається. Основним фактором впливу на побудову є призначення конкретної системи еко-моніторингу та загальні вимоги до її функціоналу. Щоб систематизувати підхід до побудови таких систем можна розглянути різні випадки організації та обґрунтувати доцільність використання.

Для початку переглянемо можливі конструкції вузлів WSN. Першою конструкцію є автономні енергоефективні вузли на базі мікроконтролеру, зображені на рисунку 3.1.

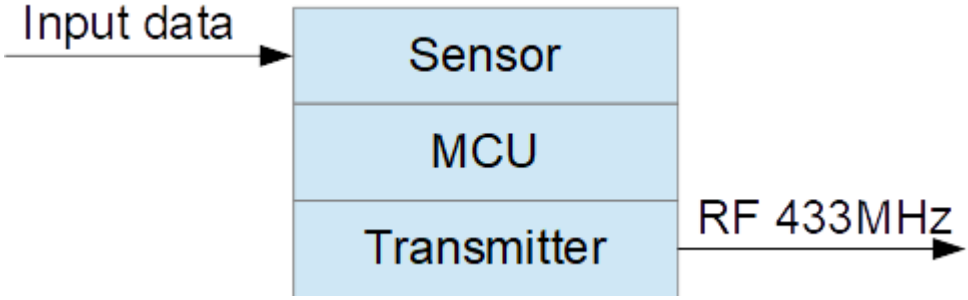


Рис. 3.1 – Схема сенсорного вузла тип А

Такі сенсорні вузли можуть складатись з 1-3 різних датчиків (температури, вологості, тиску, концентрації газів тощо), енергоефективного та недорогого мікроконтролера (8 bit Atmel), бездротової системи передачі інформації (радіочастотної).

Основне призначення вузлів даного типу автономне та енергоефективне зчитування еко-показників та трансляції даних на пристрій – хазяїн. Такі вузли можуть встановлюватись у будинках, приміщеннях та на вулиці. Область встановлення лімітується потужністю радіочастотного

передавача. У пункті 3.2 цього розділу буде детально описано приклад побудови даного вузлу.

Наступний тип вузлу (тип Б) є більш потужною системою, яка може містити цілий комплекс датчиків та сенсорів, систему радіоприймача для отримання даних з вузлів типу А, бездротову систему передачі даних WiFi (для обміном інформацією між вузлами цього ж рівня або до серверу) (рис. 3.2). Пропонується використання більш потужного 32 бітного мікроконтролеру з підтримкою бездротової мережі.

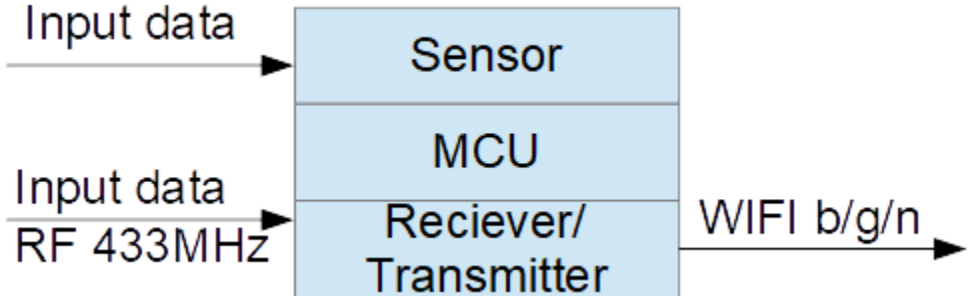


Рис. 3.2 - Схема сенсорного вузла тип Б

Даний тип вузла має більш розширений функціонал та складнішу конструкцію. Він має власний блок датчиків/сенсорів, може приймати дані від вузлів типу А. Оброблювані дані можуть надаватись через бездротову мережу WiFi 802.11 b/g/n.

Третім типом вузла (ТИП В - серверний) може бути система на базі SoC мікропроцесора для отримання, зберігання, обробки інформації еко-параметрів, надання оброблених результатів користувачам, або іншим програмам – «підписникам».

На базі даного вузла можливо організація локальної мережі з центральним вузлом, вихід до мереж наступного рівня.

Наступним кроком є варіанти організації локальної бездротової сенсорної мережі з вузлів різного типу.

Розглянемо першу організацію системи, наведену на рисунку 3.3

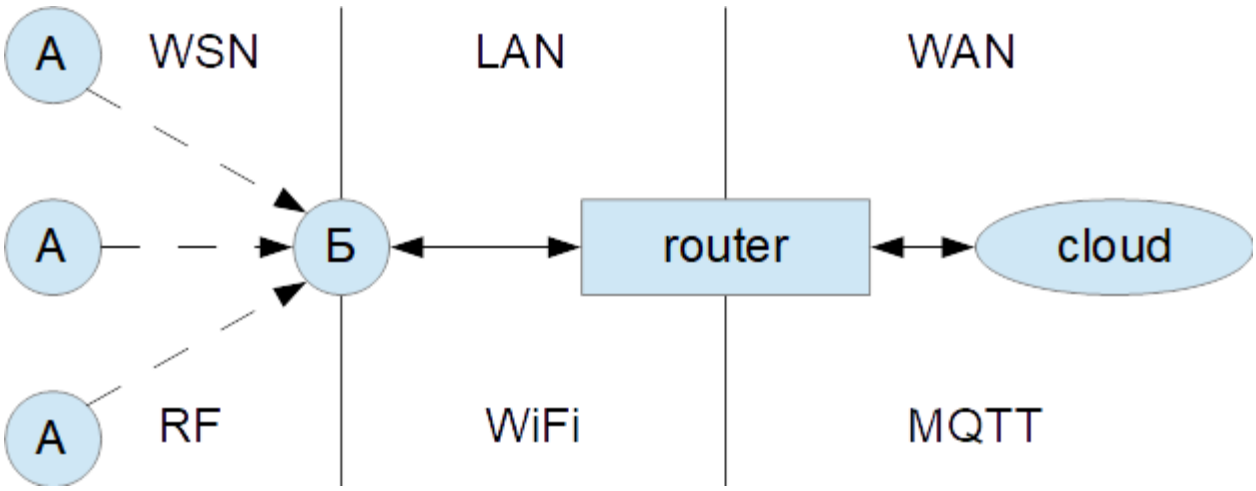


Рис. 3.3 – Організація мережі

Дана мережа складається з декількох вузлів типу А, що за допомогою радіочастотної передачі надають дані до одного вузла Б. У цьому випадку вузол Б відіграє роль сервера що отримує дані та оброблює їх. Потім Вузол Б має можливість надати результати обробки даних всередині локальної мережі, або через маршрутизатор вийти до глобальної та зберігати інформацію до хмарних брокерів.

Система є найбільш простою реалізацію, та може бути корисна невеликим підприємствам, приватному сектору та ін.

Наступна модель є більш складною та структурованою та зображена на рисунку 3.4.

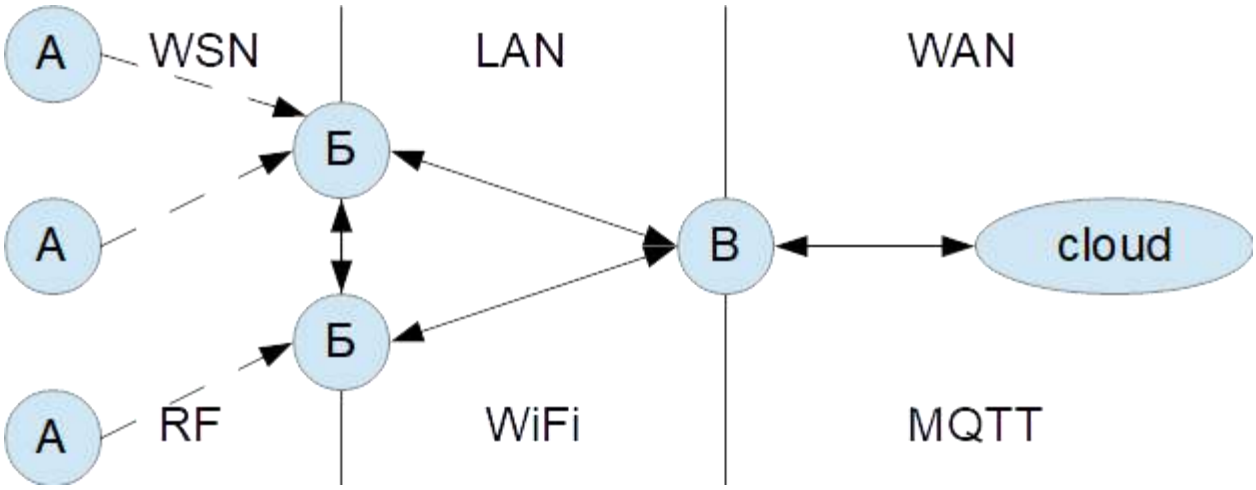


Рис. 3.4 – Організація мережі

На відміну від попередньої схеми ми маємо більш розвинуту структуру вузлів типу Б що можуть організувати власну мережу з обміном інформацією, або надавати її до вузлу типу В, де можна встановити локальний MQTT брокер та надавати інформацію до глобальної мережі.

Дана модель стане в нагоді при масштабному проектуванні на великих підприємствах, організаціях тощо.

3.2 Розробка вузла сенсорної мережі на базі ATtiny85

Даний тип пристрою необхідний для встановлення віддалених автономних точок сенсорів та датчиків, які будуть з’єднуватись з вузлом модуля ESP8266 за допомогою радіочастотного приймача/передача.

У якості центрального вузла було обрано мікроконтролер Attiny85 через його енергоефективність та простоту використання. У якості прикладу сенсором буде датчик температури/вологості DHT 22. Також до схеми необхідно додати модуль радіочастотного передавача на частоті 433 МГц. Для індикації роботи приладу додаємо світлодіод з резистором на 220 ОМ. Також між піном передачі даних та живленням температурного датчику необхідно встановити резистор на 10 кОм. Живлення необхідно брати 4-6В (підходить послідовно з’єднати три пальчикові батарейки). Принципова електрична схема зображена на рисунку 3.5.

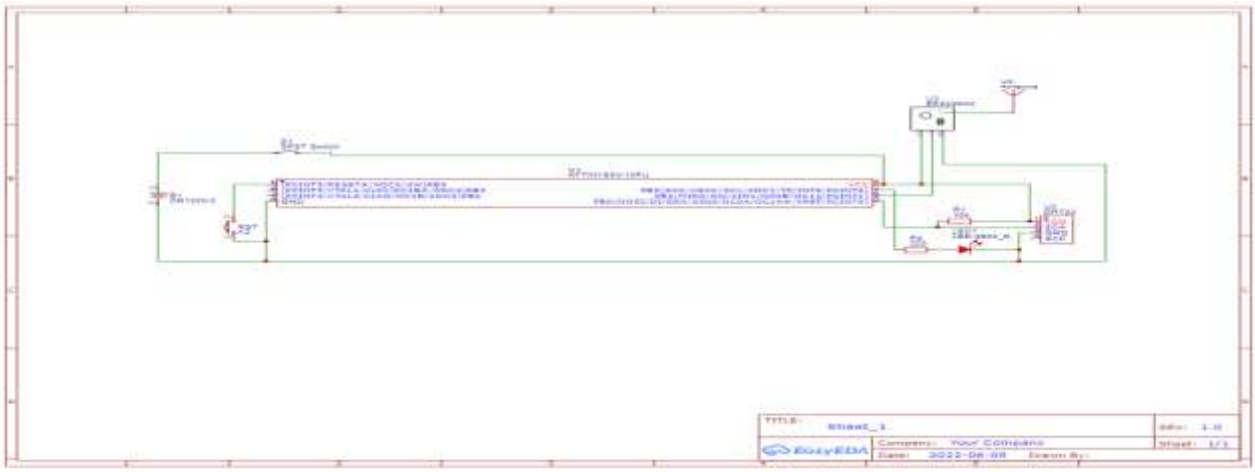


Рис. 3.5 – Принципова схема пристрою

Також до схеми додаємо кнопку перевантаження роботи мікроконтролера та перемикач включення/виключення приладу.

До цього приладу можна долучити мінімум ще два прилади що мають один інформаційний пін підключення.

За допомогою середовища проектування EasyEDA у режимі автоконфігурації отримаємо друковану плату (рис. 3.6).

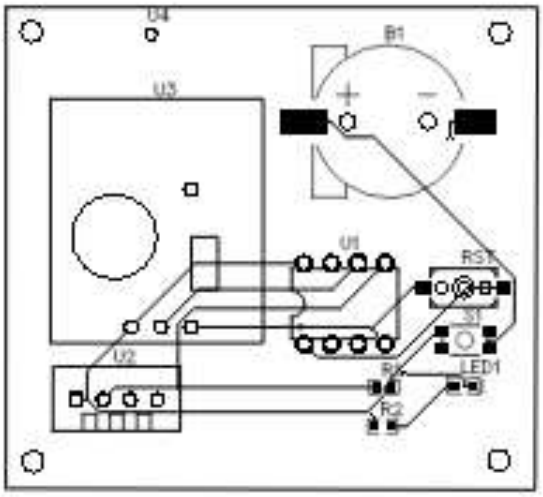


Рис. 3.6 – Друкована плата приладу

Плата розмірами 50x50мм з чотирьома отворами для кріплення. Вона має два шари та зручне розташування елементів. Монтаж SMD компонентів та роз’єми для впайки датчику та модулю. Для корпусу живлення відведено позицію B1 куди рекомендовано впаяти тримач двох батарейок CR2032.

Програмування мікроконтролеру проводимо у Arduino IDE. Особливу увагу необхідно надати енергоефективності приладу – для цього будемо використовувати бібліотеки роботи з живленням:

```
#include <avr/sleep.h>
#include <avr/power.h>
#include <avr/wdt.h>
```

Також необхідно підключити бібліотеку роботи з датчиком:

```
#include <dht.h>
```

dht DHT;

Було створено блок констант. Тут можна налаштувати сигнал і властивості передачі.

// НАЛАШТУВАННЯ

#define INTVAL 60 // частота надсилання даних у секундах.

#define CHAN 1 // встановити біт каналу (0=CH1,1=CH2,2=CH3,)

#define REP 7 // сигнал повторюється (за замовчуванням=7x)

#define ID 1318 // ідентифікатор пристрою (1280-1535) [якщо вимкнено, випадковий ідентифікатор під час запуску]

Некалібровані показання VCC дуже неточні, тому це рекомендується робити. Виміряйте напругу на виводі VCC і помножте значення на 100, а потім розкоментуйте цей рядок у коді з цим значенням.

#define VCC1 424 // Значення калібрування батареї

Для роботи пристрою було створено функції, наведені у таблиці 3.1.

Таблиця 3.1 – Функції для роботи приладу

Назва	Пояснення
void ledBlink(byte n)	Проста функція для блимкатіння світлодіоду. Параметр – кількість повторів.
void sendBit(byte b)	Передача біту через РЧ трансміттер.
void sendSync() {	Передача коригуючого сигналу.
void sendMessage(byte repeats)	Передача 36-бітного повідомлення. Параметр – кількість повторів передавання.
void sleep()	Функція – обгортка бібліотек живлення для переходу у режим сну.
void setup_watchdog()	налаштувати таймер на запуск переривання, що пробуджує attiny кожні 8 секунд
void initMessage()	Ініціалізує пристрій на передавання інформації

void updateMessage(byte mode)	Оновлює дані з сенсору для відправки повідомлення
void convertBit(int decValue, byte binIndex)	Конвертує значення даних та заповнює масив повідомлення.
void updateDHT()	Оновлює дані з датчику до масиву
int readVcc()	Вимірює напругу живлення
void checkBattery()	Перевіряє заряд батареї живлення
void calibration()	Калібрування значення живлення

Стартова функція налаштування містить у собі вибір режиму роботи контактів:

```
pinMode(2, OUTPUT);  // STATUS LED
pinMode(1, OUTPUT);  // RF TRANSMITTER
pinMode(0, INPUT);  // DHT SENSOR
```

та налаштування системи енергоефективності та радіочастотної передачі:

```
calibration();
initMessage();
updateMessage(1);
sendMessage(REP);
setup_watchdog();
```

У нескінченому циклі loop перевіряємо на необхідність виходу зі сну. У разі необхідності оновлюємо інформацію зі сенсору та відправляємо її та засинаємо.

У підсумку ми маємо автономний енергоефективний модуль сповіщення даних до приймача за радіочастотною передачею.

3.3 Розробка вузла сенсорної мережі на базі ESP8266NodeMCU

Даний тип пристрою необхідний для організації більш потужного вузлу категорії Б, де передбачено організацію .

У якості центрального вузла було обрано мікроконтролер ESP8266 через його розповсюдженість, дешевизну, наявність модулю FiWi. До модулю під’єднаємо сенсори MQ-2, BME280, DHT-22 та DS18B20. Також під’єднаємо радіоприймач HC-12. До схеми треба додати підтягувані резистори номіналом 4,7 кОм (для датчиків температури). Живлення буде розподілятись за допомогою розводки модуля NodeMCU. Принципова електрична схема зображена на рисунку 3.7

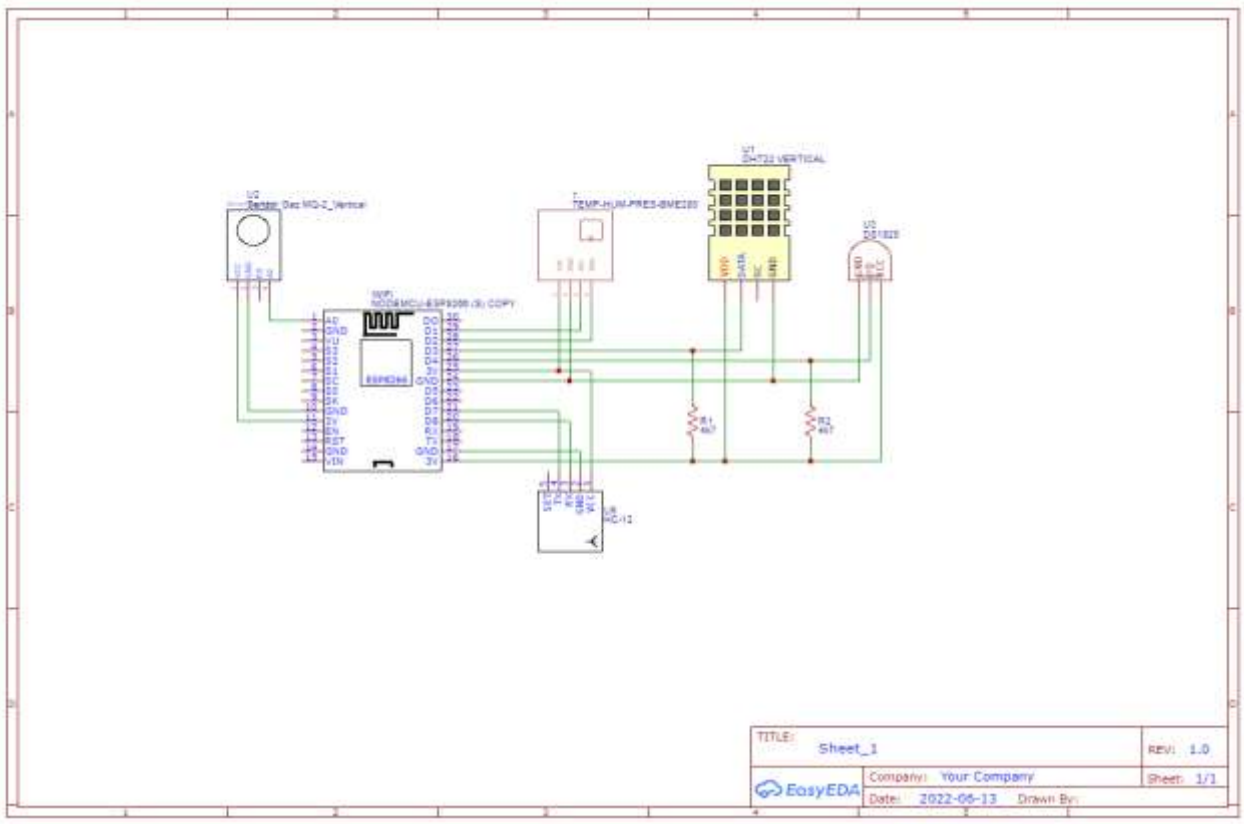


Рис. 3.7 – Принципова схема пристрою

Датчики підключені до портів за наступним принципом:

A0 – MQ-2 (A0);

D1 – BME280 (SCL);

D2 - BME280 (SDA);

D3 – DHT22 (Data);

D4 – DS18B20 (D0);

D7 – HC-12 (TX);

D8 – HC-12 (RX).

За допомогою середовища проектування EasyEDA у режимі автоконфігурації отримаємо друковану плату (рис. 3.8).

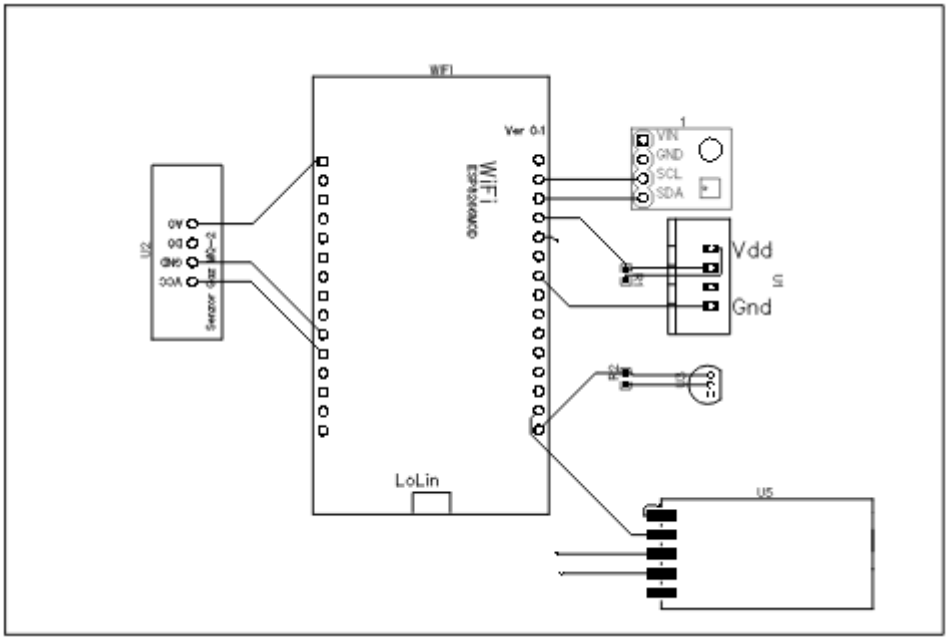


Рис. 3.8 – Друкована плата приладу

Програмування мікроконтролеру проводимо у Arduino IDE.

Підключимо наступні бібліотеки:

#include <ESP8266WiFi.h> - для роботи з WiFi;

#include <HTTPClient.h> - для використання протоколу HTTP;

#include <Wire.h> - для роботи з датчиком DS18B20 по протоколу OneWire;

#include <DallasTemperature.h> - для роботи з датчиком DS18B20;

#include <Adafruit_BME280.h> - для роботи з BME280;

#include <Adafruit_Sensor.h> - для роботи сенсорів датчику;

#include <PubSubClient.h> - для використання протоколу MQTT;

#include "htu21d.h" – для роботи датчику DHT22;

#include <ArduinoOTA.h> - завантажує скетч по мережі на плату Arduino з бібліотеками WiFi або Ethernet;

#include <SoftwareSerial.h> - для роботи з радіочастотним приймачем.

Наступним кроком є створення основних змінних:

SoftwareSerial mySerial(7,8); //RX, TX

Adafruit_BME280 bme;

HTU21D myHumidity;

DallasTemperature sensors(&oneWire);

WiFiClient wifiClient;

PubSubClient client(wifiClient);

Для роботи пристрою було створено функції, наведені у таблиці 3.1.

Таблиця 3.2 – Функції для роботи приладу

Назва	Пояснення
bool ConnectToWiFi(int timeout)	Для під’єднання до мережі
bool DisconnectFromWiFi(int timeout)	Для роз’єднання від мережі
void ReadSensors()	Зчитування сенсорів
void reconnect()	Для оновлення MQTT серверу
void callback(char* topic, byte* payload, unsigned int length)	Для встановлення даних на підписку MQTT серверу

Загальний алгоритм програми, наведеної у додатку Б наступний:

Ініціалізуємо сенсори та приймач;

Підключаємо пристрій до локальної мережі;

Підключаємо пристрій до MQTT брокеру;

Надсилаємо дані до публікації.

У підсумку ми маємо модуль, що отримує дані від сенсорів (у тому числі й дистанційно через радіоприймач) та надає дані до брокеру (локального, чи глобального).

3.4 Встановлення Eclipse MQTT Broker

В якості операційної системи можна зробити вибір в сторону офіційної прошивки Raspbian Stretch з робочим столом або Raspbian Stretch Lite. Це дві одиничні операційні системи на базі відомого дистрибутива Debian Stretch, які модифікували для роботи з Raspberry Pi. Відмінно, тільки в наявності графічного робочого столу, що для початку ознайомлення цілком виправдано, але для сервера, звичайно, графічна оболонка не до чого. Для установки виберете всі варіанти з графічним робочим столом і все це встановлюється на недорогу флешку на 8 ГБ SD-карту. Фактично, залишається завантажити образ операційної системи та за допомогою дуже зручної кроссплатформної утиліти Etcher перенести образ на фізичний накопичувач.

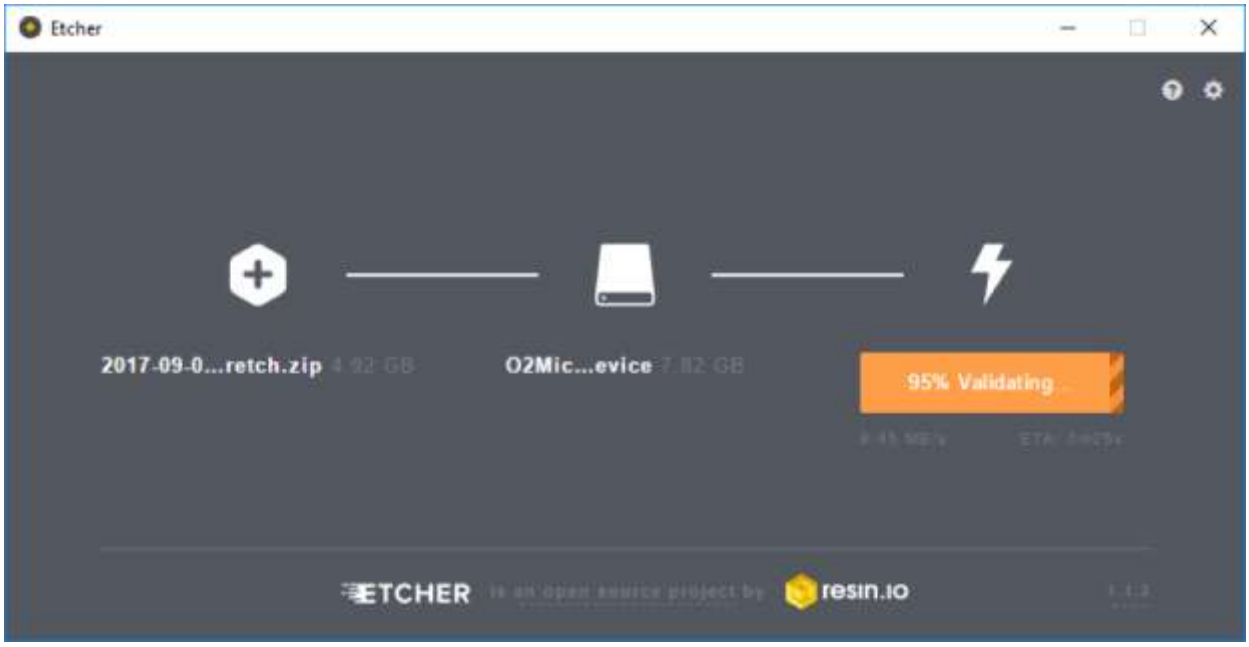


Рис. 3.9 – Встановлення дистрибутиву за допомогою Etcher

Тепер залишається помістити прошиту флешку в плату Raspberry Pi і подати живлення. Є одна особливість, якщо потрібно отримати видалений доступ до командного рядка Raspbian Stretch за протоколом ssh, при цьому

немає клавіатури, фізично підключеної платі, щоб можна було записати пустий файл з іменем ssh у завантажувальний розділ флешки, що дозволить активувати відповідний доступ. Після включення Raspberry Pi доступ до операційної системи дозволеного користувача «pi» під паролем «raspberry». Користувач за умовчанням «pi» може отримати права root, виконавши команду sudo. Оскільки в Raspbian все підготовлено для комфортної роботи непідготовленого користувача, то перша команда в новій системі може стати викликом меню системи конфігурації:

```
$ sudo raspi-config
```

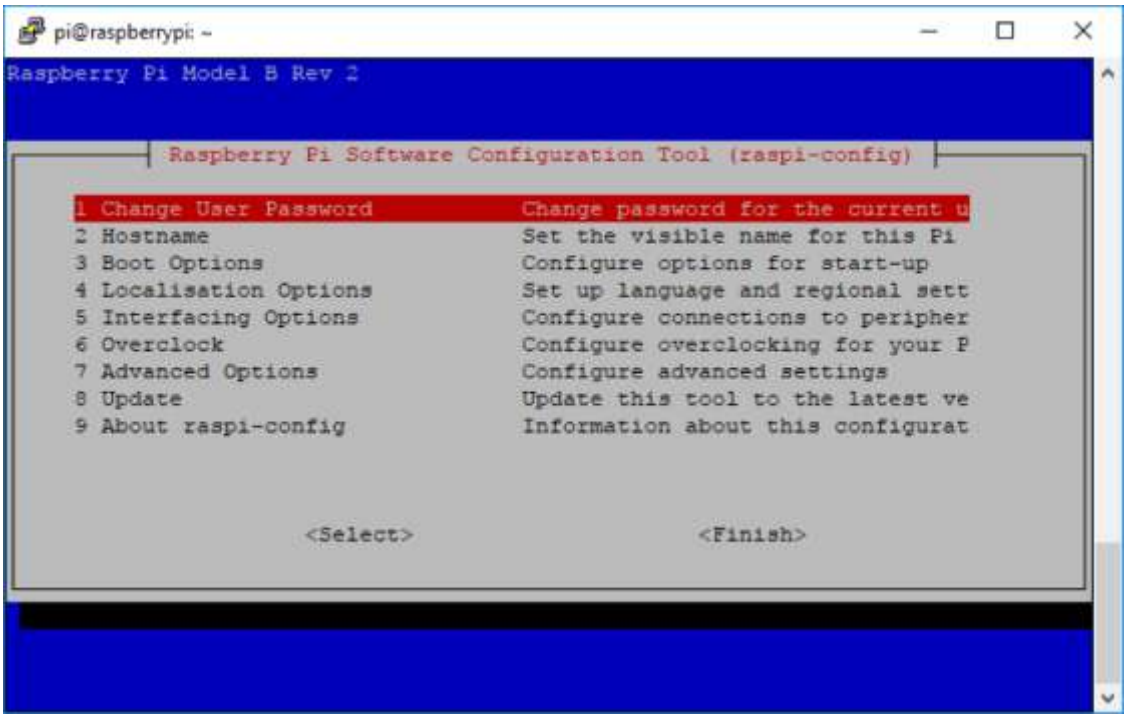


Рис. 3.10 - Віддалене підключення до Raspberry Pi через SSH і запуск raspi-config

Утиліта raspi-config дозволяє задати новий пароль користувачеві і, наприклад, у меню Interfacing Options можна налаштувати поведінку інтерфейсів системи, дозволити віддалений робочий стіл за протоколом VNC і т.п. Також у меню: «Boot Options» можна встановити параметри за замовчуванням для завантаження системи, наприклад вибрати тільки режим командного рядка, а в підменю «Resolution» з меню «Advanced Options»

встановити потрібну роздільну здатність графічного екрана, наприклад, вибравши необхідний розмір "DMT" - для цифрового інтерфейсу монітора або "CEA" - аналогового входу тощо.

Тепер можна оновити пакети системи:

$ sudo apt-get update

$ sudo apt-get upgrade

І встановити брокер Mosquitto та клієнта MQTT, наприклад, для тестування системи:

$ sudo apt-get install mosquitto

$ sudo apt-get sudo mosquitto-clients

Тепер можна перевірити локально роботу брокера MQTT, для чого підписатися, наприклад, на подію:

$ mosquitto_sub -h localhost -t "sensor/temperature"

І опублікувати якесь значення для відповідного каналу:

$ mosquitto_pub -h localhost -t "sensor/temperature" -m 21.0

Переглядати підписки та робити публікації за протоколом MQTT, а також виконати тестування навантажувальної здатності брокера, можна в середовищі браузера Chrome та операційних систем: Linux, Mac, Windows завдяки кроссплатформенному додатку MQTTBox.

Справді, протокол MQTT зараз став настільки поширеним, що знайти зручне рішення своїх завдань просто неможливо. Також в рамках проекту Eclipse Paho активно розвивається розробка відкритих програмних рішень та бібліотек, практично на всіх мовах програмування, для реалізації протоколу обміну повідомленнями MQTT та, тим самим, вдосконалення рішень для Інтернету речей.

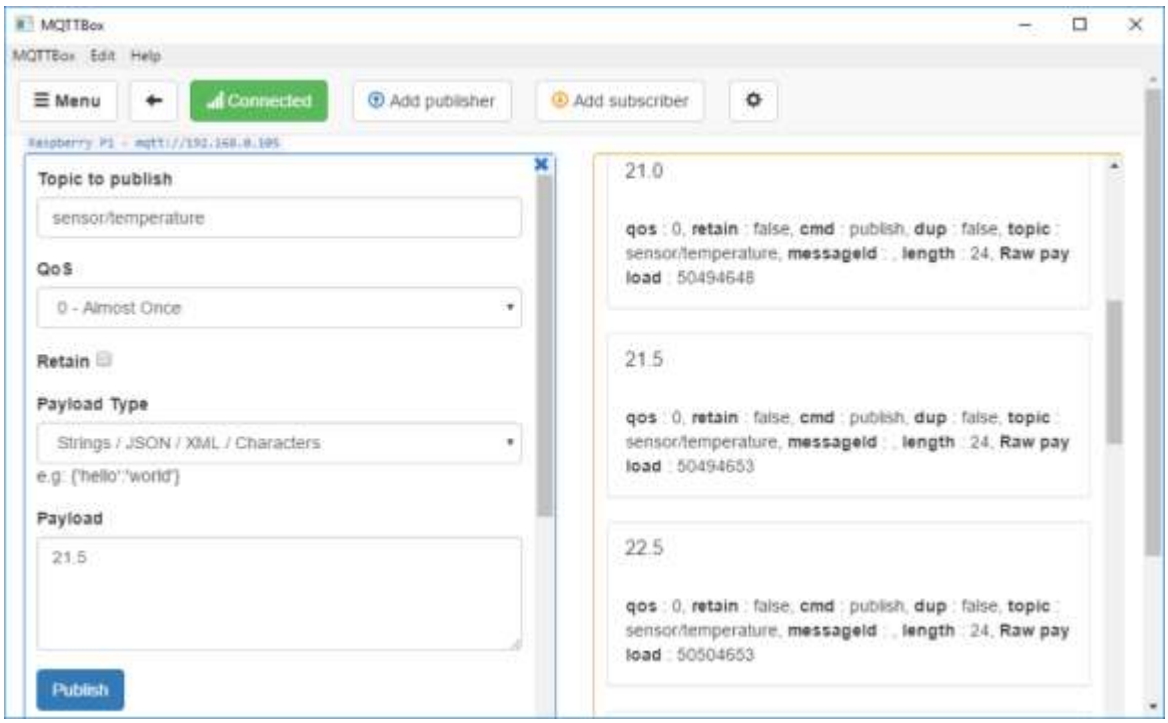


Рис. 3.11 – Інтерфейс роботи MQTTBox

Побудова рішень на основі брокера MQTT стає вже очевидним завданням. Варто лише розгорнути систему, підключитися до хмари, наприклад, використовуючи Node-RED або спеціалізоване рішення і залишається справа за «малим» – підключити інтелектуальні датчики та виконавчі механізми. Для розгляду нижнього рівня систем IoT слід написати окрему статтю, але для даного прикладу можна обійтися розглядом прикладу на базі фактичного стандарту для Інтернету речей – плати NodeMCU на базі SoC ESP8266 компанії Espressif.

Під’єднання до вузлів сенсорної мережі можливо за допомогою середовища розробки Arduino IDE для роботи з NodeMCU DEVKIT, втім, як і з іншими платами на базі ESP8266. Все робиться майже моментально з урахуванням готових бібліотек і прикладів коду. У параметрах "File" - "Preferences" - "Addition Boards Manager URLs:" вказати посилання на проект Arduino core for ESP8266 WiFi chip: arduino.esp8266.com/stable/package_esp8266com_index.json Після цього в "Tools" - "Boards Manager » залишається завантажити та встановити підтримку для плат ESP8266 Community. До речі, не слід забувати про те, що

для роботи з платою NodeMCU потрібно встановити драйвер інтерфейсного перетворювача CP210x або його ще називають міст UART до USB.

Для роботи з MQTT можна встановити бібліотеку PubSub Arduino Library. Для цього в меню "Sketch" - "Include Library" - "Manage Libraries" вибрати відповідну бібліотеку та встановити. Ознайомитися з можливостями програмної платформи можна на базі прикладу коду, наприклад, Basic ESP8266 MQTT example, де на брокері публікується канал outTopic, а також встановлюється підписка на inTopic, і коли приходить на inTopic значення 1, то на платі включається вбудований світлодіод.

3.5 Висновки за розділом

Було здійснено проектування бездротової сенсорної мережі різного призначення, що може якісно задовільнити вимоги до систем еко-моніторингу.

Запроєктовано та реалізовано сенсорні вузли на базі мікроконтролерного управління.

Здійснено налаштування MQTT брокеру локального призначення для публікації повідомлень еко-моніторингу.

ВИСНОВКИ

Відповідно до завдання було дано загальний огляд області, до якої належить тема дипломної роботи, проведений аналіз сенсорних мереж, розглянуто їх будову і застосування.

Здійснено огляд сучасного стану технологій та інформаційних систем з екологічного моніторингу у світі, Грузії та Україні зокрема. Установлено значимість та актуальність проблеми. Здійснено огляд сучасних систем-аналогів. Виявлено доцільність розробки багатошарової архітектури з мікроконтролерним керуванням та використанням мереж різного рівня.

Було здійснено огляд архітектурних особливостей організації бездротових сенсорних мереж та доведена можливість їхньої інтеграції до структури локальних та глобальних комп'ютерних мереж. Проаналізовані сучасні протоколи, що можуть бути використані при формуванні бездротової сенсорної мережі та обрано протокол MQTT.

Здійснено підбір апаратних комплектуючих та програмних засобів програмування та проектування. Обрано MQTT брокери локального та глобального користування.

Було здійснено проектування бездротової сенсорної мережі різного призначення, що може якісно задовільнити вимоги до систем еко-моніторингу. Запроєктовано та реалізовано сенсорні вузли на базі мікроконтролерного управління. Здійснено налаштування MQTT брокеру локального призначення для публікації повідомлень еко-моніторингу.

У підсумку дана система може бути використана організаціями, підприємствами для моніторингу довкілля, а також може інтегруватись до глобальних систем з еко-моніторингу.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гелен Бріггс, Бекі Дейл та Нассос Стиліану. Екологічна катастрофа: головні загрози нашій планеті у п'яти графіках [Електронний ресурс]. Режим доступу URL:https://www.bbc.com/ukrainian/features-48169628 - 04.01.22
2. Екологічний моніторинг та його види [Електронний ресурс]. Режим доступу https://buklib.net/books/24813/ - 04.01.22
3. Вбудована система [Електронний ресурс]. Режим доступу https://uk.wikipedia.org/wiki/Вбудована_система - 04.01.22
4. Wireless sensor network [Електронний ресурс]. Режим доступу https://en.wikipedia.org/wiki/Wireless_sensor_network - 05.01.22
5. MQTT [Електронний ресурс]. Режим доступу https://uk.wikipedia.org/wiki/MQTT - 05.01.22
6. Екологічний_моніторинг [Електронний ресурс]. Режим доступу https://uk.wikipedia.org/wiki/Екологічний_моніторинг - 06.01.22
7. Програма ООН з навколишнього середовища (ЮНЕП) [Електронний ресурс]. Режим доступу https://mepr.gov.ua/content/programa-oon-z-navkolishnogo-seredovishcha-yunep.html - 06.01.22
8. Екологічний моніторинг довкілля [Електронний ресурс]. Режим доступу https://mepr.gov.ua/content/ekologichniy-monitoring-dovkillya.html - 06.01.22
9. Системы мониторинга окружающей среды [Електронний ресурс]. Режим доступу https://unece.org/fileadmin/DAM//env/europe/monitoring/EnvMonRep/ru/chapter1.pdf - 07.01.22
10. ПИТАННЯ для членів Робочої групи з моніторингу та оцінки навколишнього середовища [Електронний ресурс]. Режим доступу https://unece.org/fileadmin/DAM/env/europe/monitoring/8thMeeting/Georgia.pdf - 07.01.22

11. Гуния Г. С. Результаты мониторинга экологической нагрузкиприродных сред антропогенного воздействия ряда районов грузии [Електронний ресурс]. Режим доступу https://www.researchgate.net/publication/356391478_REZULTATY_MONITORINGA_EKOLOGICESKOJ_NAGRUZKI_PRIRODNYH_SRED_ANTROPOGENNOGO_VOZDEJSTVIA_RADA_RAJONOV_GRUZII - 07.01.22
12. the Global Network on Environmental Science and Technology [Електронний ресурс]. Режим доступу https://www.gnest.org/ - 07.01.22
13. Global Ecolabelling Network [Електронний ресурс]. Режим доступу https://www.globalecolabelling.net/ - 07.01.22
14. European Environmental Agency [Електронний ресурс]. Режим доступу https://www.eea.europa.eu/ - 07.01.22
15. World Wide Fund for Nature [Електронний ресурс]. Режим доступу https://wwf.org/ - 07.01.22
16. Bresser group of companies [Електронний ресурс]. Режим доступу https://www.bresser.de/en/Weather-Time/BRESSER-WIFI-color-weather-center-with-5in1-profi-sensor.html - 10.01.22
17. Домашня метеостанція з бездротовим датчиком MISOL WH1150-1 [Електронний ресурс]. Режим доступу https://simvolt.ua/domashnya-meteostantsiya-z-bezdrotovim-datchikom-misol-wh1150-1/ - 10.01.22
18. The BRESSER 8-day 4CAST XL 7-in-1 Wi-Fi [Електронний ресурс]. Режим доступу https://www.bresser.de/en/Weather-Time/WLAN-Weather-Stations-Centers/BRESSER-8-day-4CAST-XL-7-in-1-Wi-Fi-weather-centre-with-solar-powered-sensor.html - 10.01.22
19. Професійна метеостанція (Wifi) MISOL HP2550 [Електронний ресурс]. Режим доступу https://simvolt.ua/profesiyna-meteostantsiya-wifi-misol-hp2550/ - 10.01.22
20. BSB ECO MONITORING [Електронний ресурс]. Режим доступу https://bsbecomonitoring.net/ - 12.01.22
21. M.A. Matin and M.M. Islam Overview of Wireless Sensor Network.

[Електронний ресурс]. Режим доступу https://www.intechopen.com/chapters/38793 DOI: 10.5772/49376

22. Mohammad S. Obaidat, Sudip Misra. Principles of Wireless Sensor Networks Cambridge University Press 2014 [Електронний ресурс]. Режим доступу https://doi.org/10.1017/CBO9781139030960
23. Шаблоны взаимодействия для интернета вещей [Електронний ресурс]. Режим доступу https://habr.com/ru/company/intel/blog/397241/
24. Протокол MQTT: концептуальное погружение [Електронний ресурс]. Режим доступу https://habr.com/ru/post/463669/
25. Как общаются машины — протокол MQTT [Електронний ресурс]. Режим доступу https://habr.com/ru/company/advantech/blog/452904/
26. Как общаются машины: протокол Modbus [Електронний ресурс]. Режим доступу https://habr.com/ru/company/advantech/blog/450234/
27. COAP [Електронний ресурс]. Режим доступу https://r-iot.org/2018/05/14/coap/
28. Z. Shelby, K. Hartke, C. Bormann. The Constrained Application Protocol (CoAP) [Електронний ресурс]. Режим доступу https://datatracker.ietf.org/doc/html/rfc7252
29. Advanced Message Queuing Protocol [Електронний ресурс]. Режим доступу https://en.wikipedia.org/wiki/Advanced_Message_Queuing_Protocol
30. ATtiny25/V / ATtiny45/V / ATtiny85/V Datasheet. Atmel. Rev. 2586Q–AVR–08/2013. [Електронний ресурс]. Режим доступу https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2586-AVR-8-bit-Microcontroller-ATtiny25-ATtiny45-ATtiny85_Datasheet.pdf 234 pp
31. ESP8266 [Електронний ресурс]. Режим доступу https://en.wikipedia.org/wiki/ESP8266
32. ESP8266. Technical Reference v1.7 2020.10 [Електронний ресурс]. Режим доступу https://www.espressif.com/sites/default/files/documentation/esp8266-technical_reference_en.pdf 117pp
33. NodeMCU ESP8266 [Електронний ресурс]. Режим доступу

https://components101.com/development-boards/nodemcu-esp8266-pinout-features-and-datasheet

34. BME280 - arduino.ua [Електронний ресурс]. Режим доступу: https://arduino.ua/prod1930-bme280-5v-i2c-datchik-temperatyri-vlajnosti-davleniya
35. DS18B20 - mypractic.ru [Електронний ресурс]. Режим доступу: WWW. URL - http://mypractic.ru/ds18b20-datchik-temperatury-s-interfejsom-1-wire-opisanie-na-russkom-yazyke.html#1
36. DHT11 Humidity & Temperature Sensor [Електронний ресурс]. Режим доступу https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf
37. AM2302 / DHT22 Datasheet [Електронний ресурс]. Режим доступу: https://www.electroschematics.com/am2302-dht22-datasheet/
38. Датчик газу MQ-2 - wiki.amperka.ru [Електронний ресурс]. Режим доступу: http://wiki.amperka.ru
39. Interfacing 315/433 MHz RF Transmitter-Receiver Module with Arduino [Електронний ресурс]. Режим доступу https://electropeak.com/learn/interfacing-315-433-mhz-rf-transmitter-receiver-module-with-arduino/
40. Платформа ARM и брокер MQTT, как современная основа решений для Интернета вещей [Електронний ресурс]. Режим доступу https://habr.com/ru/company/unet/blog/407867/
41. Proteus Design Suite [Електронний ресурс]. Режим доступу https://en.wikipedia.org/wiki/Proteus_Design_Suite
42. EasyEDA [Електронний ресурс]. Режим доступу https://en.wikipedia.org/wiki/EasyEDA
43. About Arduino [Електронний ресурс]. Режим доступу https://www.arduino.cc/en/about
44. Eclipse Mosquitto™. An open source MQTT broker [Електронний ресурс]. Режим доступу https://mosquitto.org/

45. An Open-Source, Cloud-Native, Distributed MQTT Broker for IoT [Електронний ресурс]. Режим доступу https://www.emqx.io/

Додаток А. Вихідний код мікроконтролеру Attiny85

```
// SETTINGS
#define INTVAL 60  // frequency of sending the data in seconds.
#define CHAN 1  // set channel bit (0=CH1,1=CH2,2=CH3, )
#define REP 7  // signal repeats (default=7x)
//#define ID 1318  // device id (1280-1535) [when disabled, random id on start]

// sleep function libraries
#include <avr/sleep.h>
#include <avr/power.h>
#include <avr/wdt.h>

// initialize DHT sensor library
#include <dht.h>
dht DHT;

// timing values of the signal
const unsigned short PULSE_LENGTH = 272;
const unsigned short LONG = PULSE_LENGTH*14;
const unsigned short SHORT = PULSE_LENGTH*7;
const unsigned short SEP = PULSE_LENGTH*2;
const unsigned short SYNC = PULSE_LENGTH*34;

// variables of sleep code
volatile bool watchdogActive = false;
byte sleepCycles = 0;

// buffer to store message ready-to-send by bits
byte message[36];

long scale_constant = 112640L; // default scale_constant (1.1*1024*100)

// define watchdog timer interrupt action
ISR(WDT_vect) {
  watchdogActive = true;  // set flag
}

void setup() {
  //OSCCAL = 0x4E;  // internal oscillator calibration value (important! see manual...)

  pinMode(2, OUTPUT);  // STATUS LED
  pinMode(1, OUTPUT);  // RF TRANSMITTER
  pinMode(0, INPUT);  // DHT SENSOR

  calibration(); // calibrate battery readings if VCC1 present

  initMessage();  // set constant message bits (id,ch)
  updateMessage(1);  // refresh message data (bat,txmode,temp,hum)
  sendMessage(REP);  // send the message on rf transmitter for 'REP' times

  setup_watchdog();  // set 8s watchdog timer interrupt
}
```

```
void loop() {
  if (watchdogActive) {  // if there was a watchdog wakeup
    watchdogActive = false;  // reset the flag
    sleepCycles += 1;

    if (sleepCycles >= INTVAL/8) {
      updateMessage(0);
      sendMessage(REP);

      sleepCycles = 0;  // reset counter
    }
  }

  sleep();  // go to sleep!
}


// A very simple led blinking function for feedback
void ledBlink(byte n) {
  for (n; n>0; n--) {
    digitalWrite(2, HIGH);
    delay(80);
    digitalWrite(2, LOW);
    delay(80);
  }
}
// replay the value b on tx-pin
void sendBit(byte b) {
  if (b == 0) {              // LOW
    PORTB = 1 << PB1;
    delayMicroseconds(SHORT);
    PORTB = 0 << PB1;
  }
  else if (b == 1) {         // HIGH
    PORTB = 1 << PB1;
    delayMicroseconds(LONG);
    PORTB = 0 << PB1;
  }
  delayMicroseconds(SEP);
}

// SYNC signal
void sendSync() {
  PORTB = 1 << PB1;
  delayMicroseconds(SYNC);
  PORTB = 0 << PB1;
}

// send the message by bits
void sendMessage(byte repeats) {
  noInterrupts();

  for (repeats; repeats > 0; repeats--) {
    // begin with a separator
```

```
   PORTB = 0 << PB1;
   delayMicroseconds(SEP);

   for (byte i = 0; i < 36; i++) {
    sendBit(message[i]);
   }
   sendSync();    // send the SYNC signal at the end
  }

  interrupts();
  ledBlink(2);
}
void sleep() {
  set_sleep_mode(SLEEP_MODE_PWR_DOWN);  // Set sleep mode to full power down, most power-
saving
  power_adc_disable();  // Turn off the ADC while asleep.
  sleep_enable();  // Enable sleep and enter sleep mode.
  sleep_mode();  // System sleeps here
  // CPU is now asleep and program execution completely halts!
  // Once awake, execution will resume at this point.
  sleep_disable();    // When awake, disable sleep mode...
  power_all_enable();  // ...and turn on all devices.
}

// set wdt timer to run an interrupt waking the attiny every 8 sec
void setup_watchdog() {
  noInterrupts();  // timing critical part, disable interrupts
  MCUSR &= ~(1<<WDRF);  // set the watchdog reset bit in the MCU status register to 0
  WDTCR |= (1<<WDCE) | (1<<WDE);  // set WDCE and WDE bits in the watchdog control register
  WDTCR = (1<<WDP0) | (1<<WDP3);  // set watchdog clock prescaler bits to a value of 8 seconds
  WDTCR |= (1<<WDIE);  // enable watchdog as interrupt only (no reset)
  interrupts();  // enable interrupts again.
}

/ Function used to set constant message bits at start
void initMessage() {
  #if not defined (ID)
    randomSeed(analogRead(A3)); // generate random ID on every restart...
    const int ID = random(1280,1535); // ...if there is no defined ID
  #endif

  // write to message array
  convertBit(ID, 0);
  convertBit(CHAN-1, 14);
}

// helper function to update message data
void updateMessage(byte mode) {
  checkBattery();
  message[13] = mode;  // 0:auto, 1:forced
  updateDHT();
}

// Convert data values into bits and fill to message array
```

```
void convertBit(int decValue, byte binIndex) {
  int data_length;

  switch (binIndex) {
    case 0:  // ID
      data_length = 12;
      break;
    case 12:  // BAT
      data_length = 1;
      break;
    case 13:  //TXMOD
      data_length = 1;
      break;
    case 14:  // CH
      data_length = 2;
      break;
    case 16:  // TEMP
      data_length = 12;
      break;
    case 28:  // HUM
      data_length = 8;
      break;
  }

  for (data_length -= 1; data_length>=0; data_length--) {
    int binValue = decValue >> data_length;
      if (binValue & 1) {
        message[binIndex] = 1;
        binIndex += 1;
      }
      else {
        message[binIndex] = 0;
        binIndex += 1;
      }
  }
}

// read and convert TEMP and HUM to bits and fill to message array
void updateDHT() {
  int tempValue,humValue;
  int chk = DHT.read22(0);

  while (chk != DHTLIB_OK) {
    delay(2200);
    chk = DHT.read22(0);
  }
  tempValue = DHT.temperature * 10;
  humValue = DHT.humidity;

  // minus grades
  if (tempValue < 0) tempValue += 1023;

  // convert values into message array
  convertBit(tempValue, 16);
```

```
  convertBit(humValue, 28);
}

// function to measure voltage
int readVcc() {
  ADMUX = _BV(MUX3) | _BV(MUX2);  // set reference to vcc and measurement to int. 1.1v reference.
  delay(2); // wait for vref to settle
  ADCSRA |= _BV(ADSC); // start conversion
  while (bit_is_set(ADCSRA,ADSC)); // wait here while measuring

  uint8_t low  = ADCL; // must read ADCL first - it then locks ADCH
  uint8_t high = ADCH; // unlocks both

  long vcc_reading = (high<<8) | low;
  int result = round(scale_constant / vcc_reading); // calculate VCC (in mV)
  return result;
}

void checkBattery() {
  int voltage = readVcc();

  if (voltage < 370) message[12] = 0;  // low
  else message[12] = 1;  // charged
}

// Improve accuracy of VCC readings with calibration
void calibration() {  //... if VCC1 is available
  #if defined (VCC1)
    unsigned int readings = 0;

    for (byte i=0;i<10;i++) {  // get 10 initial readings
      readings += readVcc();
      delay(25);  // gives more stable results
    }

    int vcc2 = readings / 10;  // calculate the average

    // ###### ###### could check here those readings later...
    int refvolt = 110L*VCC1/vcc2;
    scale_constant = refvolt * 1024L;
  #endif
}
```

Додаток Б. Вихідний код мікроконтролеру ESP8266

```
#include "htu21d.h"

#include <WiFi.h>
#include <HTTPClient.h>
#include <Wire.h>
#include <esp_adc_cal.h>
#include <driver/adc.h>
#include <driver/gpio.h>
#include <esp_err.h>
#include <apps/sntp/sntp.h>
#include <inttypes.h>

//'''''''''''''''global''''''''''''''''''''''''''
esp_adc_cal_characteristics_t characteristics;
RTC_DATA_ATTR static int bootCount = 0;
time_t now = 0;
//frequency of wake cycles
const long samplingFrequency = 300;
//'''''''''''''''''''''''''''''''''''''''''''''

bool ConnectToWiFi(int timeout)
{
  if (WiFi.status() == WL_CONNECTED)
    return true;

  unsigned long max = timeout * 1000;
  unsigned long t = millis();
  const char *ssid = "";
  const char *password = "";

  IPAddress local_IP(192, 168, 23, 135);
  IPAddress gateway(192, 168, 23, 1);
  IPAddress subnet(255, 255, 0, 0);
  IPAddress dns(192, 168, 23, 1);

  if (!WiFi.config(local_IP, gateway, subnet, dns, dns))
  {
    Serial.println("STA Failed to configure");
    return false;
  }

  WiFi.begin(ssid, password);
  while (max > millis() - t)
  {
    //digitalWrite(13, !digitalRead(13));
    ledcWrite(1, (millis() - t) * 4);
    if (WiFi.status() == WL_CONNECTED)
    {
      ledcWrite(1, 16384);
      break;
    }
    delay(50);
```

```
  }

  Serial.printf("After Connection loop: %d, duration: %d \n", WiFi.status(), (millis() - t));
  return (WiFi.status() == WL_CONNECTED);
}

bool DisconnectFromWiFi(int timeout)
{
  if (WiFi.status() == WL_NO_SHIELD)
    return true;
  unsigned long max = timeout * 1000;
  unsigned long t = millis();
  WiFi.disconnect(true);

  while (max > millis() - t)
  {
    if (WiFi.status() == WL_NO_SHIELD)
    {
      digitalWrite(13, LOW);
      break;
    }
    timeout--;
    delay(5);
  }

  Serial.printf("After disconnect loop: %d, duration: %d \n", WiFi.status(), (millis() - t));
  return (WiFi.status() == WL_NO_SHIELD);
}

//Syncs with ntp server, provided the time passed in is not a valid time or the forced flag is set.
long obtainTime(time_t *t, bool forced)
{
  const char *sntpServer = "pool.ntp.org";
  const long minEpoch = 1465345377; //seconds for Jun  8 2016 00:22:57, ESP32 default timestamp
  time(t);

  long oldTime = *t > minEpoch ? *t : minEpoch;

  if (*t < minEpoch || forced)
  {
    sntp_setoperatingmode(SNTP_OPMODE_POLL);
    sntp_setservername(0, (char *)sntpServer);
    sntp_init();
    // wait for time to be set
    unsigned int timeout = 10 * 1000;
    unsigned int m = millis();
    while (*t < minEpoch && (timeout > millis() - m))
    {
      delay(5);
      time(t);
    }
    Serial.printf("Time received %d\n", *t);
  }
  return *t - oldTime;
```

```
}

float readBatteryVoltage()
{
 // Read ADC and obtain result in mV, its from a voltage devider, so should be multiplied by 2
 return adc1_to_voltage(ADC1_CHANNEL_7, &characteristics) * 2.0 / 1000.0;
}

bool PostToInflux(String post)
{
 HTTPClient http;
 http.begin("http://raspberrypi:8086/write?db=environment&precision=s");
 http.addHeader("Content-Type", "text/json");

 int httpResponseCode = http.POST(post);
 http.end();

 Serial.printf("%d : %s \n", httpResponseCode, post);
 return httpResponseCode == 204;
}

void setup()
{
 unsigned long startTime = millis();

 //used for battery reading
 //https://esp-idf.readthedocs.io/en/latest/api-reference/peripherals/adc.html#adc-calibration
 // ESP32 0x4485 chip read 1090mV
 // Configure ADC
 adc1_config_width(ADC_WIDTH_12Bit);
 adc1_config_channel_atten(ADC1_CHANNEL_6, ADC_ATTEN_11db);

 // Calculate ADC characteristics i.e. gain and offset factors
 esp_adc_cal_get_characteristics(1090, ADC_ATTEN_DB_11, ADC_WIDTH_BIT_12, &characteristics);

 Serial.begin(115200);
 pinMode(13, OUTPUT); // set the LED pin mode

 ledcAttachPin(13, 1);
 ledcSetup(1, 12000, 16);

 Serial.printf("Boot count %d \n", ++bootCount);
 char value[256];
 float vBat = readBatteryVoltage();
 Serial.printf("Battery Voltage: %f V\n", vBat);
 if (vBat > 3.4)
 {
  ConnectToWiFi(10);

  int drift = obtainTime(&now, bootCount % (24 * 3600 / samplingFrequency) == 0);
  uint64_t chipid = ESP.getEfuseMac();

  Serial.printf("Time Drift: %d sec\n", drift);
```

```
  if (drift != 0 && bootCount > 0)
  {
    sprintf(value, "rtc,board=0x%04X%08X drift=%f,bootcount=%di %d", (uint16_t)(chipid >> 32),
(uint32_t)chipid, drift, bootCount, now);
    PostToInflux(value);
  }
  HTU21D htu;
  bool b = htu.begin(25, 26);
  if (b)
  {
    Serial.printf("Temp:%f, Humid:%f, dew:%f, HI%f \n", htu.getTemperature(), htu.getHumidity(),
htu.getDewPoint(), htu.getHeatIndex());
    sprintf(value,
"weather,board=0x%04X%08X,sensor=htu21d,region=outside,variable=Temperature,unit=Celsius
value=%f %d", (uint16_t)(chipid >> 32), (uint32_t)chipid, htu.getTemperature(), now);
    PostToInflux(value);
    sprintf(value,
"weather,board=0x%04X%08X,sensor=htu21d,region=outside,variable=Relative_Humidity,unit=Percent
age value=%f %d", (uint16_t)(chipid >> 32), (uint32_t)chipid, htu.getHumidity(), now);
    PostToInflux(value);
    sprintf(value,
"weather,board=0x%04X%08X,sensor=htu21d,region=outside,variable=Dew_Point,unit=Celsius
value=%f %d", (uint16_t)(chipid >> 32), (uint32_t)chipid, htu.getDewPoint(), now);
    PostToInflux(value);
    sprintf(value,
"weather,board=0x%04X%08X,sensor=htu21d,region=outside,variable=Heat_Index,unit=Celsius
value=%f %d", (uint16_t)(chipid >> 32), (uint32_t)chipid, htu.getHeatIndex(), now);
    PostToInflux(value);
  }

  sprintf(value, "battery,board=0x%04X%08X voltage=%f,bootcount=%di %d", (uint16_t)(chipid >> 32),
(uint32_t)chipid, vBat, bootCount, now);

  PostToInflux(value);
  DisconnectFromWiFi(10);
  Serial.printf("Start: %u, finish: %u, sleep: %u ", startTime, millis(), (samplingFrequency * 1000 - (millis()
- startTime)));

  esp_deep_sleep(1000LL * (samplingFrequency * 1000 - (millis() - startTime)));
 }
 else
 {
  esp_deep_sleep(1000000LL * 3600);
 }
}

void loop()
{
}
```