

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

Факультет комп'ютерних наук та технологій
Кафедра комп'ютерних систем та мереж

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавра

(ступінь вищої освіти (освітній ступінь))

на тему «РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
МІКРОКОНТРОЛЕРІВ НА ОСНОВІ ESP8266 З ПІДТРИМКОЮ
БЕЗДРОТОВОЇ МЕРЕЖІ WI-FI»

Виконав: студент 4 курсу, групи КНТ-520
спеціальності _____

123 Комп'ютерна інженерія

Освітня програма (спеціалізація)

Комп'ютерна інженерія

(код і назва спеціальності)

МУСІЧЕНКО О.Є.

(прізвище та ініціали)

Керівник ІЛЬЯШЕНКО М.Б.

(прізвище та ініціали)

Рецензент КОЗИНА Г.Л.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування вищого навчального закладу)

Факультет Комп'ютерних наук і технологій
Кафедра «Комп'ютерні системи та мережі»
Ступінь вищої освіти (освітній ступінь) бакалаврський
Спеціальність 123 Комп'ютерна інженерія
(код і назва)
Освітня програма (спеціалізація) Комп'ютерна інженерія
(назва)

ЗАТВЕРДЖУЮ

Зав. кафедри

Кудерметов Р.К.
“ ” 2024 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ СТУДЕНТУ

МУСІЧЕНКО Олега Євгеновича

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) «Розробка програмного забезпечення мікроконтролерів на основі ESP8266 з підтримкою бездротової мережі Wi-Fi»

керівник проекту (роботи) ІЛЬЯШЕНКО М.Б., к. т. н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від “01” квітня 2024 року № 105

2. Строк подання студентом проекту (роботи) 01.06. 2024 року

3. Вихідні дані до проекту (роботи) MQTT, модуль ESP8266, датчик температури та вологості DHT11, датчик DHT, NodeMCU, LUA, C++

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

1) Огляд та характеристика розробки програмного модуля _____

2) Розробка модуля керування _____

3) Реалізація програми _____

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

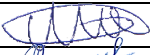
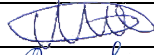
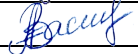
Схема підключення пристроїв

Схема підключення модуля до USB to UART під час прошивання

Прошивка плати

Завантаження скриптів у пам'ять мікроконтролера за допомогою програми ESPlorer

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-3	ІЛЬЯШЕНКО М.Б., к.т.н.. доцент		
Нормоконтроль	ПОЛЬСЬКА О.В., ст. викладач		

7. Дата видачі завдання 01.04. 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Огляд програмного модуля	05.04.2024	
2	Огляд мікроконтролера	10.05.2024	
3	Вибір мови розробки	17.04.2024	
4	Обрання датчиків температури та вологості	23.04.2024	
5	Розробка модуля керування	18.05.2024	
6	Реалізація розробки	25.05.2024	
7	Оформлення отриманих результатів у ПЗ	26.05.2024	
8	Оформлення графічного матеріалу	29.05.2024	
9	Захист роботи		

Студент


(підпис)

Олег МУСІЧЕНКО

(ініціали та прізвище)

Керівник проекту (роботи)


(підпис)

Матвій ІЛЬЯШЕНКО

(ініціали та прізвище)

РЕФЕРАТ

ПЗ: 70 с., 15 рис., 16 джерел, 8 дод.

МІКРОКОНТРОЛЕР, ДАТЧИК, МОДУЛЬ, ПРОТОКОЛ MQTT, ВЕБ-СЕРВЕР

Метою даної роботи є розробка програмного забезпечення (ПЗ) мікроконтролерів на основі ESP8266 з підтримкою бездротової мережі Wi-Fi. Об'єкти можуть входити в загальну мережу "інтернету речей" (IoT) з можливістю автоматичного розпізнавання та початкового налаштування, а також мати веб-інтерфейс для остаточного налаштування.

У рамках цієї роботи було розроблено програмне забезпечення мікроконтролера ESP8266, з підтримкою протоколів обміну з системою управління, протоколу MQTT. Застосування розробки можливе в найширших сферах автоматизації, наприклад, - системи "розумного будинку", автоматизація тепличних господарств або промислових об'єктів. Розробка може бути сумісна з роботами: "Мережа IoT на основі контролерів ESP8266" і "Система управління IoT мережею".

ЗМІСТ

Вступ	7
1 Огляд та характеристика розробки програмного модуля	9
1.1 Огляд наявних мереж IoT	9
1.2 Огляд мікроконтролера ESP8266	15
1.2.1 Технічні характеристики ESP8266	17
1.2.2 Різновиди модуля ESP8266	19
1.2.3 Огляд прошивок модуля ESP8266	19
1.2.4 Початкове завантаження та оновлення прошивки	20
1.2.5 Типові сценарії використання модуля ESP8266	20
1.3 Програмні засоби розробки	25
1.3.1 Мова програмування LUA	25
1.3.2 Протоколи та стандарти зв'язку	27
1.4 Датчик температури та вологості	29
1.5 Типи використовуваних реле	30
2 Розробка модуля керування	35
2.1 Розробка електричної схеми модуля керування електроприладами	36
2.2 Конфігурація модуля	37
2.2.1 Опис протоколу взаємодії із системою керування	38
2.3 Розробка програмного забезпечення модуля	39
2.3.1 Опис модулів програмного забезпечення контролера	42
3 Реалізація розробки	43
3.1 Алгоритм роботи пристрою	43
3.2 Підготовка модуля до роботи	44
3.3 Опис режимів роботи	47
3.4 Загальний опис роботи та взаємодія програмних модулів	49
Висновки	51
Перелік джерел посилання	52
Додаток А. Модуль ініціалізації	54
Додаток Б. Модуль конфігурації контролера	55
Додаток В. Налаштування модуля виводів	56
Додаток Г. Модуль підключення до мережі wifi	58

Додаток Д. Модуль створення web-серверу	60
Додаток Е. Модуль керування входами	62
Додаток Ж. Основний логічний модуль програми.....	64
Додаток К. Модуль роботи з датчиком dht11	67

ВСТУП

Інтернет речей - це технологічна революція, яка є майбутнім обчислень і комунікацій, і її розвиток залежить від технічних інновацій у кількох важливих галузях.

По-перше, для того, щоб під'єднати речі до інтернету, необхідна проста й ефективна за витратами система передавання інформації та ідентифікації речей. Таку можливість і необхідний функціонал надають різні пасивні й активні бездротові технології: RFID, NFC, Wi-Fi тощо.

По-друге, користь від зібраних даних про речі проявиться, тільки якщо є можливість відстежувати зміни у фізичному статусі речей (у просторі та в часі). Для цього використовуються різні сенсорні технології.

По-третє, чип у будь-який продукт буде вбудовуватися вже на етапі виробництва, і вбудований у речі "інтелект" ("розумні" речі) дасть змогу ухвалювати складні і навіть незалежні рішення.

І нарешті, мініатюризація і нанотехнології забезпечують дедалі дрібнішим і дрібнішим речам можливість економічної комунікації та взаємодії між собою.

Комбінація всіх цих технологій і призведе до створення інтернету речей, який з'єднає об'єкти оточуючого нас фізичного і віртуального світів [1].

На основі Інтернету речей можуть бути реалізовані всілякі "розумні" smart додатки в різних сферах діяльності та життя людини:

- "розумна планета" - людина зможе буквально "тримати руку на пульсі" планети - своєчасно реагувати на упущення в плануванні господарств, забруднення та інші екологічні проблеми, а значить, ефективно розпоряджатися невідновлюваними ресурсами;
- "розумне місто" - міська інфраструктура та супутні муніципальні послуги, як-от освіта, охорона здоров'я, громадська безпека, ЖКГ, стануть більш пов'язаними та ефективними;
- "розумний дім" - система розпізнаватиме конкретні ситуації, що

відбуваються в будинку, і реагуватиме на них відповідним чином, що забезпечить мешканцям безпеку, комфорт і ресурсозбереження;

- "розумна енергетика" - буде забезпечено надійне та якісне передавання електричної енергії від джерела до приймача в потрібний час і в необхідній кількості;

- "розумний транспорт" - переміщення пасажирів з однієї точки простору в іншу стане зручнішим, швидшим і безпечнішим;

- "розумна медицина" - лікарі та пацієнти зможуть отримати віддалений доступ до дорогого медичного обладнання або до електронної історії хвороби будь-де, буде реалізовано систему віддаленого моніторингу здоров'я, автоматизовано видачу лікарських препаратів хворим і багато іншого [1].

1 ОГЛЯД ТА ХАРАКТЕРИСТИКА РОЗРОБКИ ПРОГРАМНОГО МОДУЛЯ

Мета - розробка програмного забезпечення (далі ПЗ) WIFI модуля на базі мікроконтролера ESP8266, для реалізації елемента управління IoT мережі.

Завданням цієї розробки є :

- початкове проведення аналізу мереж IoT;
- варіантів ідентифікації об'єктів та їх початкового конфігурування;
- розробка способів застосування мікроконтролера ESP8266;
- вибір протокол взаємодії мікроконтролера з елементами керування;
- розробка структури програмного забезпечення контролера;
- розробка та тестувати програмного забезпечення.

1.1 Огляд наявних мереж IoT

Як приклад, одне з таких рішень - це система "Розумного будинку" HomeKit компанії Apple. Завдяки HomeKit можна підключити, об'єднати і контролювати зі смартфона всю "розумну" техніку в будинку, на якій є мітка "Works with Apple HomeKit" (Сумісно з Apple HomeKit). Такими аксесуарами можуть бути світильники, замки, термостати, інтелектуальні розетки та багато іншого.

Усі сумісні з HomeKit пристрої можуть бути об'єднані і працювати разом за заданим користувачем сценарієм або за допомогою інтелектуального помічника Siri. Керування приладами відбувається з пристроїв iOS. Перевага Apple HomeKit у тому, що простота встановлення та налаштування системи не вимагають від власника спеціальних знань і навичок [2].

Розумний офіс. Microsoft, а точніше, технологічний центр (Microsoft Technology Center), створений для демонстрації та тестування передових

технологій, є реальним досвідом проектування та реалізації проекту "Розумного офісу". Роботу інфраструктури будівлі контролюють сотні датчиків і виконавчих механізмів, які забезпечують управління такими параметрами як:

- температура і вологість у приміщеннях;
- електропостачання;
- охолодження;
- відеоспостереження;
- освітлення;
- аудіо-відео та презентаційне обладнання;
- розклад, конфігурація і завантаження приміщень;
- клімат;
- система доступу;
- підвищення бізнес-продуктивності та об'єднаних комунікацій.

Система має безліч цікавих функцій, які, наприклад, дають змогу ідентифікувати співробітників або гостей офісу за смарт-картками або за допомогою NFC у телефоні, забезпечує навігацію офісом для гостей, має розумного електронного секретаря, уміє керувати проектором, освітленням та іншим обладнанням у переговорній кімнаті голосом зі смартфона та багато іншого [3].

Інтернет речей (IoT) являє собою мережу фізичних об'єктів, які оснащені датчиками, програмним забезпеченням та іншими технологіями з метою підключення та обміну даними з іншими пристроями і системами через Інтернет. Нижче представлено огляд основних мереж IoT, які використовуються сьогодні.

LPWAN – це тип бездротової телекомунікаційної мережі, призначений для забезпечення довготривалого з'єднання пристроїв з низьким споживанням енергії. Основні переваги LPWAN включають великий радіус дії, низьку вартість обслуговування і низьке енергоспоживання. До основних технологій LPWAN відносяться:

- LoRaWAN, використовується для комунікацій на довгі дистанції з низьким енергоспоживанням;

- NB-IoT забезпечує широке покриття і підтримує велику кількість одночасних з'єднань;

- Sigfox підходить для простих пристроїв з низьким споживанням енергії, які передають невеликі обсяги даних.

Мобільні мережі (3G, 4G, 5G) також використовуються для підключення IoT-пристроїв, забезпечуючи високу пропускну здатність і надійність. Основні варіанти включають:

- LTE-M (LTE Cat-M1) оптимізована для IoT з низьким енергоспоживанням і високою пропускну здатністю;

- NB-IoT призначена для обміну даними в умовах слабого сигналу і для пристроїв з низьким енергоспоживанням.

Ці мережі підходять для підключення пристроїв на короткі відстані з високою швидкістю передачі даних. Найпоширеніші технології включають:

- Wi-Fi , використовується для підключення пристроїв в межах однієї будівлі або прилеглої території.

- Bluetooth, підходить для зв'язку на короткі відстані, особливо в системах з низьким енергоспоживанням (Bluetooth Low Energy);

- Zigbee, застосовується в автоматизації будівель і промислових системах завдяки низькому енергоспоживанню і надійній роботі в умовах перешкод.

Mesh-мережі використовують бездротову інфраструктуру, де кожен вузол мережі діє як точка доступу і ретранслятор сигналу. Це дозволяє забезпечити ширше покриття і підвищену надійність з'єднань. До найбільш поширених mesh-технологій належать:

- Zigbee, часто використовується для автоматизації будинків і промислових додатків;

- Z-Wave, поширена в системах автоматизації житла і безпеки.

Сучасні IoT-мережі пропонують різноманітні рішення для підключення пристроїв, що дозволяє вибирати найбільш оптимальну технологію в залежності від потреб конкретного додатка. Вибір відповідної мережі IoT залежить від вимог

до енергоспоживання, радіусу дії, пропускної здатності та вартості обслуговування.

Цей огляд дає загальне уявлення про існуючі мережі IoT та їхні основні характеристики.

Розумне місто. Поширення Інтернету речей не обмежується тільки масштабами будинків і окремих офісів. Цілі міста з успіхом впроваджують у свою інфраструктуру "розумні сервіси". Масштабний інфраструктурний проект, спеціалізацією якого є сфера високих та інформаційних технологій. Основною метою проекту є створення унікальної міської екосистеми, що забезпечує якісну підготовку, високий рівень життя та ефективну роботу кваліфікованих фахівців у сфері IT.

Усі наявні інфраструктурні системи міста (будівлі, транспортні об'єкти, інженерні комунікації тощо) об'єднано в єдину "розумну" мережу "Smart + Connected Communities, S + CC", розроблену компанією Cisco. Впроваджене рішення передбачає єдину міську мережу Wi-Fi і відеоспостереження, контроль міського середовища та екологічної обстановки, інтелектуальну транспортну систему (комплекс датчиків транспортних потоків, центр обробки і керування світлофори) і багато іншого. Усю інформацію, отриману від усіх смарт-об'єктів міста, обробляють у центрах обробки даних Іннополіса. На їх підставі системи життєзабезпечення міста оптимізують роботу всієї міської інфраструктури, що дає змогу місту ефективніше використовувати ресурси і скоротити фінансові витрати [4].

Розумне страхування. Страхова компанія дає можливість повернути частину грошей за поліс КАСКО, завдяки програмі "Розумне страхування". Суть програми в тому, що протягом півроку моніторинговий пристрій, встановлений в автомобілі автовласника, збирає інформацію про якість його водіння (перевищення швидкості, різкі прискорення і гальмування тощо) і передає дані на сервер. На підставі цих даних система розраховує рівень акуратності водія, його стиль водіння і поведінки за кермом. За результатами оцінки якості, автовласникові повертається частина суми, яку він заплатив за поліс. Усі

фіксовані дані доступні в додатку на смартфоні, а також в особистому кабінеті на сайті компанії, що дає змогу відстежувати зміну оцінки [5].

Розумна медицина. "Інтернет речей" відкриває для медицини безпрецедентні можливості. Американська компанія Proteus Digital Health розробила продукт Proteus Discover, який дає змогу лікарям оцінити здоров'я пацієнта та контролювати курс лікування хворого. Proteus Discover - це система, яка складається з датчиків і програмного забезпечення.

Суть роботи системи в тому, що пацієнт приймає ліки, наприклад, у вигляді таблетки, на якій встановлено маленький, розміром з піщинку, датчик-сенсор. Другий датчик у вигляді пластиру клеїться на пацієнта і фіксує сигнали від проковтнутого сенсора. Коли датчик-сенсор потрапляє в шлунок, він посиляє сигнал другому датчику про те, коли пацієнт прийняв ліки. Далі, у міру взаємодії сенсора з навколишнім середовищем, він передає інформацію про його стан і про те, в якій ділянці організму він наразі перебуває. Датчик-приймач оснащений бездротовим модулем зв'язку, який передає отримані дані на сервер. Через застосунок на мобільному телефоні або на порталі Discover Portal лікар може простежити, приймав чи не приймав його підопічний ліки вчасно, де перебуває проковтнута пігулка на даний момент, і оцінити стан здоров'я пацієнта. Крім цього, Proteus Discover має детектор якості сну і датчик рівня фізичної активності користувача, що дає змогу виміряти додаткові параметри організму. Система нагадає пацієнтові через додаток на смартфоні, коли йому необхідно прийняти чергову порцію ліків згідно з курсом лікування [6].

Компанія REDMOND перша ринку представила лінійку унікальних побутових приладів з технологією дистанційного керування Ready for Sky. Прилади з технологією R4S управляються одним зручним мобільним додатком. Для передачі даних між приладами використовуються сучасні технології, що істотно скорочують енерговитрати. Створення та впровадження революційного стандарту управління технікою R4S стало можливим завдяки роботі міжнародної команди hardware&software-розробників.

Зручний та інтуїтивно зрозумілий мобільний застосунок Ready for Sky для

пристроїв на платформах iOS та Android об'єднує всі необхідні користувачеві дані, сортуючи їх за видом приладу. Так, у застосунку міститься понад 3000 рецептів для мультиварки, калькулятори підрахунку харчової цінності продуктів для кухонних ваг, лічильник кількості кроків або витрачених калорій для фітнес-браслета, варіанти дієт і графіки зміни ваги для підлогових ваг та багато іншого. Крім того, завдяки додатку R4S можна зберігати будь-які вподобані рецепти і ділитися ними з іншими користувачами.

Для управління побутовою технікою з R4S достатньо під'єднати прилади до маршрутизатора SkyBox або станції SkyTVbox, і встановити додаток на свій смартфон. І тоді, де б користувач не перебував і чим би не займався, він завжди буде підтримувати невідомий зв'язок зі своїм будинком. Управління розумною технікою Redmond безпосередньо зі смартфона або планшета можливе на відстані до 50 метрів (максимальна відстань до "далекобійність" Bluetooth 4.0), залежно від товщини стін та інших перешкод на шляху сигналу відстань може варіюватися. На жаль, техніка Redmond Smart Home оснащена тільки Bluetooth-модулями. Тож для контролю через інтернет знадобиться ще один смартфон або планшет, який виступатиме в ролі "гейтвея". Апарат-передавач все час має залишатися вдома, адже він прийматиме ваші команди через Інтернет і транслюватиме їх через Bluetooth вашим чайникам, праскам та іншій розумній техніці. Для перетворення гаджета на "гейтвей" необхідно встановити на нього інший застосунок - R4S Gateway. Вимоги до смартфона-передавача невисокі: Android 4.3+, підтримка Bluetooth 4.0 і доступ в інтернет (Wi-Fi, 4G/3G або хоча б 2G/EDGE). Програма R4S Gateway працює у фоновому режимі, займає мало пам'яті, не заважає працювати іншим додаткам і вкрай економно витрачає трафік. Додаток Ready for Sky використовується не тільки для управління і контролю за розумною побутовою технікою, з його допомогою можна оновлювати ПЗ пристроїв. У розумних мультиварках і прасках може оновлюватися прошивка так само, як у смартфонах і планшетах. Причому з оновленнями не просто виправляються якісь баги і недоробки - апарати набувають нових функцій і режимів роботи [7].

1.2 Огляд мікроконтролера ESP8266

Компанія Espressif Systems постійно здійснює науково-дослідницьку та проєктну діяльність у галузі Wi-Fi та Bluetooth-технологій. В результаті чого, продукція Espressif широко відома в усьому світі і використовується в мобільних пристроях, побутовій техніці та промислових додатках.

Високоінтегровані Espressif Wi-Fi чіпсети і модулі створені таким чином, щоб займати мінімальне місце на платі, універсальні в застосуванні, економічні за споживанням і вигідні за ціною.

Приблизно в серпні 2014 року на торговому майданчику aliexpress з'явилися дешеві (приблизно 4\$) WI-FI модулі ESP8266 китайського розробника.

Модуль продається із завантаженою прошивкою, яка утворює WI-FI--UART міст для під'єднання до іншого мікроконтролера, зокрема й до Arduino. Налаштування та обмін даними відбуваються за допомогою AT команд [8].

Мікроконтролер ESP8266, розроблений компанією Espressif Systems, здобув велику популярність завдяки своїм компактним розмірам, високій продуктивності та інтегрованому Wi-Fi модулю, що робить його ідеальним для створення пристроїв Інтернету речей (IoT).

Оснoву ESP8266 становить 32-бітний процесор Tensilica L106 Diamond, який працює на частоті до 160 МГц. Цей мікроконтролер оснащений внутрішньою флеш-пам'яттю об'ємом від 512 КБ до 4 МБ та має SRAM об'ємом 64 КБ для інструкцій і 96 КБ для даних. Завдяки цим характеристикам ESP8266 забезпечує високий рівень обробки даних і дозволяє зберігати достатньо інформації для багатьох застосувань.

Однією з ключових особливостей ESP8266 є вбудований Wi-Fi модуль, який підтримує стандарти Wi-Fi 802.11 b/g/n. Модуль може працювати в режимах точки доступу (AP), клієнта (STA) або обох одночасно (AP+STA), що надає гнучкість у створенні різних мережевих конфігурацій. Він також підтримує сучасні протоколи безпеки WPA/WPA2, забезпечуючи захищений зв'язок.

ESP8266 має до 17 загальних цифрових входів/виходів (GPIO), які можна використовувати для підключення різних периферійних пристроїв. Він підтримує інтерфейси SPI, I2C та UART, що дозволяє легко інтегрувати його з іншими мікроконтролерами та сенсорами. Крім того, він має один аналоговий вхід (ADC) з роздільною здатністю 10 біт, що дозволяє вимірювати напругу в діапазоні від 0 до 1 В.

Енергоспоживання ESP8266 є дуже низьким завдяки кільком режимам енергозбереження, включаючи deep sleep mode. Це робить його ідеальним для автономних проектів, що працюють на батарейках, оскільки він може значно знижувати споживання енергії під час простою.

Програмування ESP8266 є досить зручним, оскільки він підтримує використання популярного середовища розробки Arduino IDE, а також інших інструментів, таких як PlatformIO та NodeMCU для програмування на Lua. Велика спільнота розробників та широкий спектр бібліотек і прикладів роблять початок роботи з ESP8266 відносно простим навіть для початківців.

Мікроконтролер ESP8266 існує в кількох варіантах модулів, включаючи ESP-01, ESP-07, ESP-12 та інші. Найбільш популярним є модуль ESP-12, який має компактні розміри та великий набір GPIO, що робить його дуже універсальним.

ESP8266 знайшов широке застосування в багатьох сферах завдяки своїм можливостям та доступній ціні. Його використовують для створення розумних будинків, управління освітленням, температурою, системами безпеки, а також для портативних та носимих пристроїв, таких як трекери активності та смарт-годинники. У промислових IoT системах ESP8266 застосовують для моніторингу та управління виробничими процесами. Завдяки простоті використання, цей мікроконтролер ідеально підходить для розробки та тестування нових ідей у різних експериментальних проектах.

Загалом, ESP8266 є потужним і універсальним інструментом для створення різноманітних IoT пристроїв, надаючи розробникам можливість швидко і ефективно реалізовувати свої проекти.

1.2.1 Технічні характеристики ESP8266

Мікропроцесор ESP8266 має наступні характеристики:

- процесор: одноядерний Tensilica L106 частотою до 160 MHz;
- підтримувані стандарти WI-FI: 802.11 b/g/n;
- підтримувані типи шифрування: WEP, WPA, WPA2;
- підтримувані режими роботи: Клієнт (STA),
- точка доступу (AP), Клієнт+Точка доступу (STA+AP);
- напруга живлення: 1.7..3.6 В;
- струм споживання: до 215мА залежно від режиму роботи;
- кількість GPIO: 16 (фактично до 11)., доступно на модулях: ESP-01 - 4, ESP-03 - 7+1, включаючи UART. Існують й інші варіанти модулів;Інтерфейси: 1 ADC, I2C. UART, SPI, PWM;
- зовнішня Flash пам'ять може бути встановлена від 512кб до 4мб;
- RAM даних 80 кб, RAM інструкцій - 64 кб.

Мікроконтролер, представлений на рисунку 1.1, не має на кристалі користувацької енергонезалежної пам'яті.

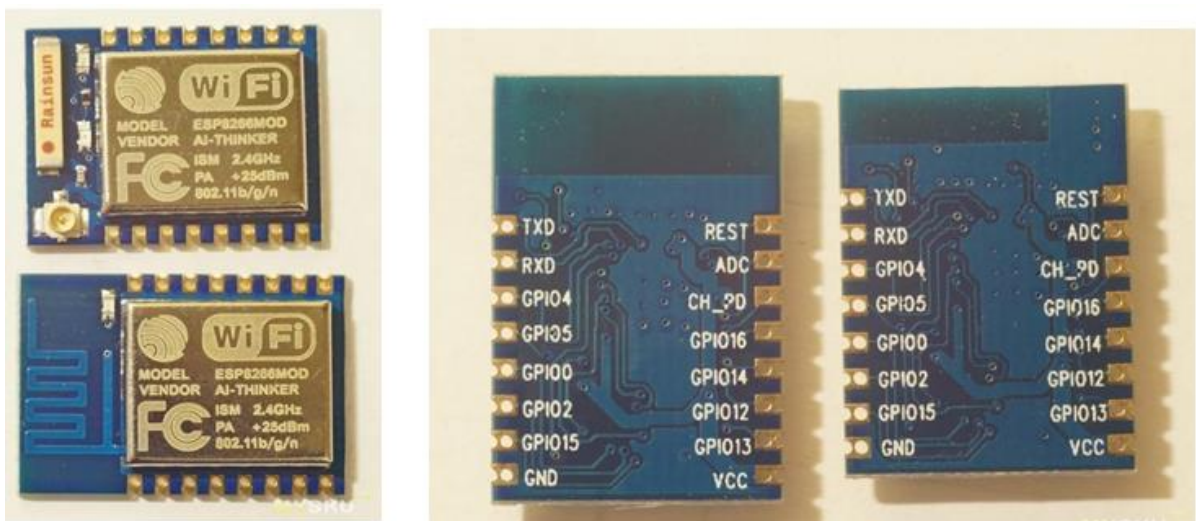


Рисунок 1.1 - Модуль ESP8266-07 і ESP8266-12

Виконання програми ведеться із зовнішньої SPI ПЗП шляхом динамічного підвантаження необхідних ділянок програми в кеш інструкцій. Підтримується до

16 МБ зовнішньої пам'яті програм. Можливий Standard, Dual або Quad SPI інтерфейс.

Чип ESP8266 є одним із найбільш високоінтегрованих рішень для роботи з WiFi [9].

Типова обв'язка чипа показана на рисунку 1.2. Вона складається всього з декількох елементів. Відповідно менше елементів - менша ціна компонентів, менша вартість пайки, менша площа розміщення, менша вартість друкованої плати. Керує всім цим розширена версія 32-бітного процесора Tensilica's L Diamond series.

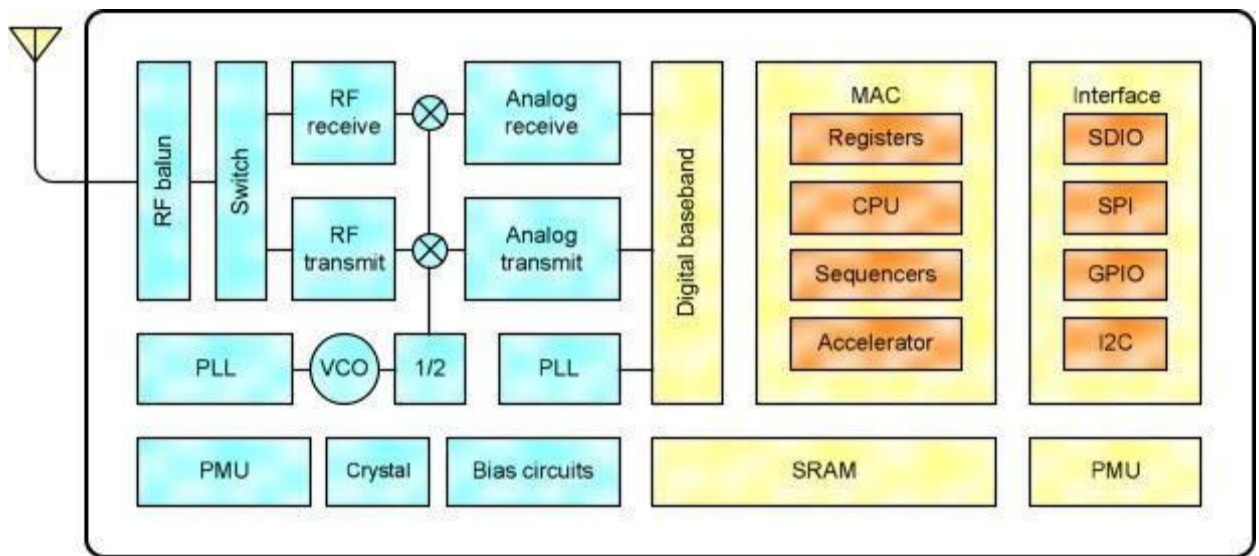


Рисунок 1.2 - Обв'язка

Енергоспоживання - одна з найважливіших характеристик інтернету речей. Споживачі не готові міняти батареї в датчиках кожні два-три місяці. Тому "вихід у мережу" доводилося забезпечувати мостами, підключеними до постійної електрики. І тепер з появою цього модуля, Wi-Fi можна використовувати навіть в автономних датчиках, що працюють на невеликих батареях. Завдяки використанню просунутих механізмів управління енергоспоживанням. Якщо подивитися на характеристики споживання чипа, 215mA в режимі передавання, близько 60mA в режимі глибокого сну (з годинником реального часу, який працює) і менше ніж 1.0mA (DTIM=3) або менше ніж 0.5mA (DTIM=10) в режимі

підтримки зв'язку з точкою доступу Wi-Fi [9].

1.2.2 Різновиди модуля ESP8266

Зараз у продажу можна знайти готові модулі 12 видів: з підключенням зовнішньої антени, з керамічною антеною, з антеною з доріжок, в екранованому корпусі. Так само буває виведено різну кількість GPIO.

Модулі ESP8266 ESP-01, ESP-02, ESP-03, ESP-04, ESP-05, ESP-06, ESP-07, ESP-08, ESP-09, ESP-10, ESP-11, ESP-12, ESP-13, WROOM, WROOM-02

1.2.3 Огляд прошивок модуля ESP8266

Щоб спростити використання мікроконтролера в типових проєктах, можливе використання готових бінарних файлів, придатних до прямої заливки в ПЗП модулів (так званих прошивок). Готові прошивки можна розділити на кілька груп відповідно до концепції їх використання:

а) прошивки для роботи під управлінням зовнішнього контролера. У цих прошивках реалізовано певний функціонал, якому зовнішній контролер через UART задає параметри роботи. До таких прошивок належить - прошивка з керуванням AT-командами з SDK Espressif;

б) прошивки з вбудованими інтерпретаторами різноманітних мов високого рівня. Ці прошивки дають змогу довантажувати через UART і виконувати скрипти розробника пристрою:

1) NodeMCU - проєкт на основі скриптової мови Lua. Прошивка вміє виконувати Lua-скрипти як з UART (аналогічно AT-командам), так і з внутрішньої flash пам'яті. Для завантаження скриптів у flash пам'ять підтримується файлова система SPIffs. З боку Wi-Fi є вбудовані MQTT протокол і HTTP сервер. Вбудована графічна оболонка, що дає змогу підключати до ESP8266 графічні індикатори;

в) прошивки для інтернету речей. Цей клас прошивок дає змогу з одного боку під'єднати до ESP8266 набір датчиків і виконавчих пристроїв, а з іншого боку надає необхідний мережевий функціонал для роботи в інфраструктурі Internet of Things:

1) ESP Easy - прошивка для домашньої автоматизації. Виконана як скетч

Arduino. Підтримує безліч датчиків і виконавчих пристроїв, підтримка яких є в Arduino (наприклад, датчики температури, вологості, освітленості, дисплеї, реле тощо). Є клієнт MQTT;

2) WiFi-IoT.ru - онлайн конструктор IoT-прошивок для ESP8266 від Максима Мікліна для проєкту домашньої автоматизації homes-smart.ru. Дає змогу прямо на сайті створити прошивку ESP8266 для роботи з однією з безлічі підтримуваних IoT систем верхнього рівня. Варіант необхідної прошивки створюється шляхом вибору опцій конфігурації. Проєкт комерційний, але є безкоштовні обмежені можливості [10].

1.2.4 Початкове завантаження та оновлення прошивки

Для керування процесом оновлення прошивки існує безліч утиліт:

- XTCOM - консольна утиліта з Espressif SDK;
- NodeMCU-Flasher - віконна утиліта під Win;
- esptool_ck - консольна утиліта, написана на Cі;
- esp_tool - консольна утиліта, написана на C++;
- esptool.py - консольна утиліта, написана на Python[11].

1.2.5 Типові сценарії використання модуля ESP8266

ESP8266 можна використовувати в розумних розетках, mesh-мережах, IP-камерах, бездротових сенсорах, переносній електроніці тощо.

Передбачено два варіанти використання чипа:

- у вигляді моста UART-WIFI, коли модуль на базі ESP8266 під'єднується до наявного рішення на базі будь-якого іншого мікроконтролера і керується AT-командами, забезпечуючи зв'язок рішення з інфраструктурою Wi-Fi;
- реалізуючи нове рішення, що використовує сам чип ESP8266 як керуючий мікроконтролер [11].

Основне застосування ESP8266 знаходить в управлінні різноманітними побутовими приладами через бездротові мережі. Існує кілька популярних реалізацій концепції IoT в плані обміну даними по мережі:

- HTML сервер на ESP8266. Контроль і керування пристроєм ведеться через браузер. Важковагове рішення, підходить для автономних пристроїв

автоматики;

- AllJoyn - відкритий IoT протокол великого альянсу виробників цифрової техніки "Allseen", що набирає популярності;
- MQTT. Це простий протокол поверх TCP/IP. Дуже популярне рішення. Існує велика кількість IoT застосунків верхнього рівня для Android, iOS та інших платформ, що підтримують цей протокол;
- SNMP. Розширюваний протокол керування мережевими пристроями. Основний недолік у тому, що в більшості мереж фаїрволи блокують проходження SNMP.

Модулі ESP8266 - це дуже дешеве рішення для побудови мережі розумного будинку та іншої домашньої автоматизації з використанням WiFi. Але для роботи в якості самостійного контролера ESP8266 має замало ресурсів. Загалом ESP8266 мають великі перспективи [12].

Протокол MQTT (Message Queuing Telemetry Transport) — це легкий, відкритий протокол обміну повідомленнями, який розробили для надійної передачі даних між пристроями в умовах низької пропускну здатності мережі, обмежених ресурсів і високої затримки. Спочатку створений IBM у 1999 році, MQTT став стандартом OASIS і широко використовується в Інтернеті речей (IoT), мобільних додатках, сенсорних мережах та інших системах, де важливі швидкість і ефективність передачі даних.

Основна модель роботи MQTT полягає в публікації повідомлень відправниками (паблішерами) та отриманні цих повідомлень підписниками (сабскрайберами). Центральним елементом цієї моделі є брокер MQTT, який отримує повідомлення від паблішерів і доставляє їх сабскрайберам. Це дозволяє розподіляти повідомлення між багатьма пристроями без необхідності прямих з'єднань між ними.

MQTT використовує мінімум мережевого трафіку, оскільки його заголовки дуже компактні. Це важливо для пристроїв з обмеженими ресурсами та мереж з низькою пропускну здатністю. Протокол також підтримує три рівні якості обслуговування (QoS), що дозволяють визначити, наскільки надійно

повідомлення буде доставлено. Ці рівні включають QoS 0, де повідомлення доставляється не більше одного разу і без підтвердження, QoS 1, де повідомлення доставляється принаймні один раз і підтверджується, і QoS 2, де повідомлення доставляється тільки один раз і підтверджується з гарантією.

MQTT дозволяє зберігати останнє опубліковане повідомлення для кожної теми, щоб нові підписники могли негайно отримати актуальну інформацію при підписці. Повідомлення в MQTT організовані в ієрархію тем (topics), які представляють собою рядки, що описують шлях до певного повідомлення, наприклад, "датчики/температура/кімната1". Крім того, MQTT дозволяє клієнтам вказувати повідомлення, яке буде автоматично опубліковано брокером, якщо клієнт відключиться некоректно. Це корисно для виявлення проблем з підключенням та інших несправностей.

Протокол MQTT широко використовується в різних сферах. У Інтернеті речей (IoT) він застосовується для збору даних від датчиків і управління пристроями в реальному часі. Це включає моніторинг домашніх систем автоматизації, розумних міст, промислових процесів і багато іншого. У системах моніторингу та управління MQTT використовується для моніторингу віддалених систем, таких як станції енергетики, нафтові платформи та транспортні засоби. У мобільних додатках MQTT використовується для передачі повідомлень між мобільними додатками та серверами, наприклад, для обміну миттєвими повідомленнями або нотифікацій. У розумних будинках MQTT застосовується для управління освітленням, системами опалення, вентиляції та кондиціонування, безпековими системами та іншими домашніми пристроями.

Серед переваг MQTT можна виділити легкість і ефективність, простоту у реалізації та використанні, високу надійність завдяки різним рівням QoS і гнучку систему тем і підписки, що полегшує масштабування і управління. Проте MQTT також має певні недоліки, зокрема потребу в брокері для роботи, що додає додаткову точку відмови, і не завжди підходить для додатків, що потребують високої пропускної здатності або низької затримки.

Таким чином, MQTT залишається одним із ключових протоколів у сфері

IoT завдяки своїй простоті, надійності та ефективності. Його здатність працювати в умовах обмежених ресурсів робить його ідеальним вибором для багатьох сучасних додатків, де необхідний надійний і легкий обмін повідомленнями.

MQTT (Message Queuing Telemetry Transport) - легкий і простий протокол обміну повідомленнями, що реалізує модель "публікація/підписка" (publish/subscribe) і призначений для зв'язку комп'ютеризованих пристроїв, під'єднаних до локальної або глобальної мережі, між собою та різноманітними публічними або приватними веб - сервісами. Схему роботи протоколу MQTT наведено на рисунку 1.3.

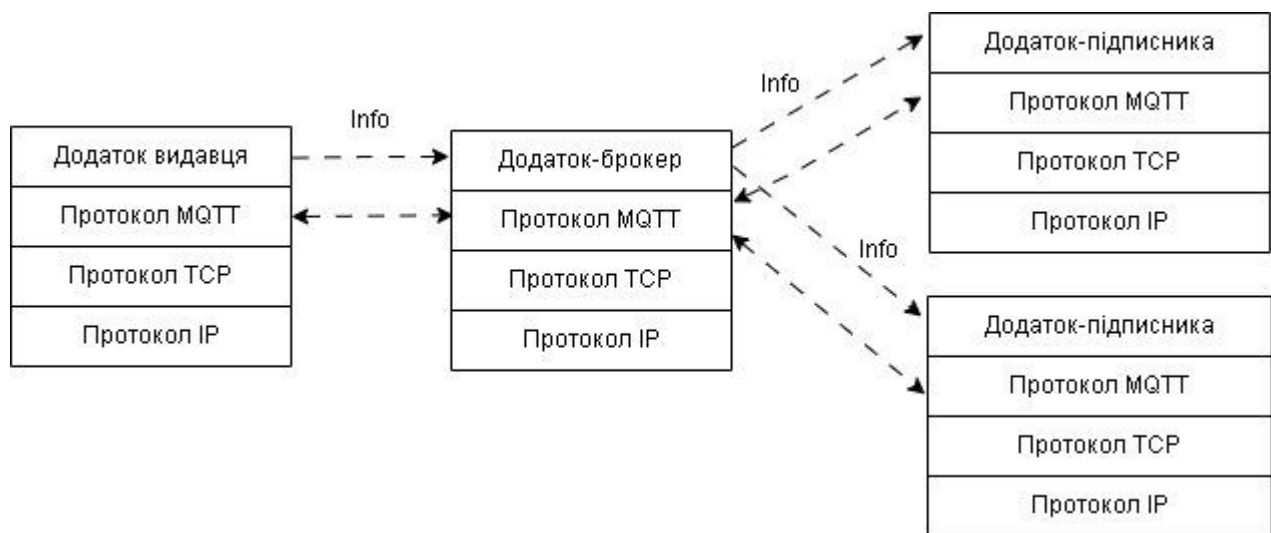


Рисунок 1.3 - Схема роботи протоколу MQTT

Протокол MQTT спочатку був створений для датчиків, що відстежують стан труб, проте пізніше сферу його діяльності було розширено, і він знайшов своє застосування в безлічі вбудовуваних рішень, зокрема в смартфонах. Так соціальна мережа Facebook застосовує цей протокол для обміну повідомленнями (Facebook Messenger).

У мережі на базі протоколу MQTT розрізняють 3 об'єкти:

- видавець (publisher) - MQTT-клієнт, який у разі виникнення певних подій передає інформацію про нього в брокер;
- брокер (broker) - MQTT-сервер, який приймає інформацію від

видавців і передає її відповідним передплатникам, у складних системах може виконувати також різні операції, пов'язані з аналізом і обробленням даних, що надійшли;

- передплатник (subscriber) - MQTT-клієнт, який після підписки у відповідного брокера більшу частину часу "слухає" його і постійно готовий до приймання та обробки вхідного повідомлення від брокера.

Брокери можуть бути сконфігуровані мостом, як показано на рисунку 1.3. Він може перенаправляти повідомлення, що прийшли на певний канал, іншому брокеру. Таким чином можна масштабувати мережу управління.

Специфікація MQTT відкрита і доступна в Інтернеті. Наразі є два варіанти специфікації MQTT:

- MQTT v3.1 - основна специфікація для мереж на базі TCP/IP;
- MQTT-S v1.2 для датчиків і вбудованих пристроїв у мережах, відмінних від TCP/IP, наприклад ZigBee.

MQTT вже використовується в роботі супутників, а також у медицині та деяких промислових сферах.

Основні переваги протоколу MQTT:

- невеликі накладні витрати на транспортному рівні (заголовок фіксованого розміру довжиною 2 байти);
- протокол обміну зведений до мінімуму для зменшення мережевого трафіку;
- вбудований механізм контролю з'єднання.

Протокол MQTT має низку переваг, порівняно з протоколом HTTP:

- менші накладні витрати на передачу даних і менша смуга пропускання;
- для своєї роботи він не вимагає постійного з'єднання між клієнтом і сервером (як у випадку HTTP).

MQTT також добре адаптований до роботи каналами зв'язку з низькою пропускнуою здатністю [13].

1.3 Програмні засоби розробки

Програмний комплект розробника SDK складається з:

- компілятора - має відкриті вихідні тексти, то в різних SDK можуть міститися різні збірки цього компілятора, що трохи відрізняються підтримуваними опціями;
- бібліотек для роботи з периферією контролера, стеків протоколів WiFi, TCP/IP;
- засобів завантаження виконуваного файлу в пам'ять програм мікроконтролера;
- опціональної IDE.

Espressif вільно поширює свій комплект розробника. У цей комплект входить компілятор GCC, бібліотеки Espressif і завантажувальна утиліта XTCOM. Бібліотеки поставляються у вигляді скомпільованих бібліотек, без вихідних текстів.

Espressif підтримує дві версії SDK: одна на основі RTOS, інша на основі зворотних викликів (callback) [14].

1.3.1 Мова програмування LUA

Lua – це потужна, легка, вбудовувана скриптова мова програмування, яка була розроблена в 1993 році в Технічному інституті Pontifical Catholic University of Rio de Janeiro в Бразилії. Lua, що означає "місяць" португальською мовою, спочатку була створена як мова для конфігурації програмного забезпечення, але швидко здобула популярність у багатьох інших областях завдяки своїй гнучкості та ефективності.

Lua розроблена як проста і легка мова, яка легко інтегрується в інші додатки. Це означає, що вона може бути вбудована в програмне забезпечення, написане іншими мовами, такими як C, C++ або Java. Така інтеграція дозволяє розробникам використовувати Lua для розширення функціональності своїх програм без необхідності переписувати їх основні компоненти.

Однією з ключових особливостей Lua є її простота та легкість у навчанні. Синтаксис мови мінімалістичний і інтуїтивно зрозумілий, що робить її доступною навіть для новачків у програмуванні. Незважаючи на свою простоту, Lua є дуже потужною мовою, що підтримує такі функції, як першокласні функції, метатаблиці (які дозволяють змінювати поведінку таблиць, надаючи їм нові можливості), корутини (які підтримують багатозадачність), та автоматичне управління пам'яттю через збірку сміття.

Lua часто використовується у відеоіграх для створення сценаріїв, що керують поведінкою персонажів, рівнів, подій та інших аспектів гри. Відомі ігрові движки, такі як Unity, CryEngine і Unreal Engine, підтримують Lua для створення модів та скриптів. Lua також знайшла своє застосування в робототехніці, де вона використовується для управління рухом роботів і обробки даних з датчиків, вбудованих системах, таких як мережеве обладнання, де вона допомагає в налаштуванні та автоматизації, а також у веб-розробці, де Lua використовується для створення динамічних веб-сторінок і обробки запитів на сервері.

Середовище виконання Lua, яке називається Lua interpreter, легко інтегрується з іншими мовами і дозволяє виконувати Lua-скрипти. Lua підтримує різні типи даних, включаючи nil, boolean, number, string, function, userdata, thread, і найважливіше - таблиці. Таблиці є універсальним типом даних у Lua, який використовується для масивів, асоціативних масивів (словників) і об'єктів. Таблиці в Lua дуже гнучкі і дозволяють створювати складні структури даних.

Ще одна важлива особливість Lua – це її переносимість. Інтерпретатор Lua написаний на ANSI C, що дозволяє легко компілювати його на багатьох платформах без модифікацій. Lua також має малий розмір виконуваного файлу, що робить її ідеальною для вбудованих систем з обмеженими ресурсами.

Попри свої численні переваги, Lua має деякі обмеження. Наприклад, вона не підтримує багатопоточність на рівні мови і не має вбудованої підтримки для роботи з мережею або файловою системою (хоча ці можливості можна додати через бібліотеки). Проте ці обмеження рідко стають критичними, оскільки Lua зазвичай використовується разом з іншими мовами, які можуть забезпечити

необхідну функціональність.

У підсумку, Lua є універсальною та ефективною мовою програмування, яка підходить для широкого спектру завдань. Її простота, легкість інтеграції та потужні можливості роблять її відмінним вибором для розробки скриптів, ігор, вбудованих систем і багатьох інших додатків.

Lua є типовою процедурною мовою програмування. Вона надає широкі можливості для об'єктно-орієнтованої та функціональної розробки. Lua створювалася як потужна і проста мова, що володіє всіма необхідними виразними засобами. Бібліотека функцій мови Lua написана мовою C. Будучи розширюваною мовою, Lua не має поняття "main" програми: вона працює в середовищі виконання, яке скорочено називають хост. Хост-програма дає змогу запускати частини коду, написані Lua, модифікувати змінні Lua і реєструвати C-функції для використання безпосередньо в коді Lua. Завдяки можливості розширення за допомогою C-функцій, Lua може застосовуватися для вирішення широкого кола завдань. Таким чином, ми можемо створювати спеціалізовані бібліотеки, використання яких нічим не відрізняється від використання стандартних засобів мови. Іменами (ідентифікаторами) в Lua можуть бути будь-які рядки з літер, цифр і символу підкреслення, що не починаються з цифри. Це правило типове для більшості мов програмування. (Поняття букви залежить від поточної локалі: будь-який символ з алфавіту поточної локалі може бути використаний. у складі ідентифікатора). Ідентифікатори використовуються для іменування змінних і таблиць значень. Lua є мовою, чутливою до регістру символів. Lua являє собою мову з динамічним визначенням типів даних. Змінна мови може містити значення будь-якого типу. Можливості визначення користувацьких типів даних відсутні. Усі значення в Lua можуть зберігатися у змінних, використовуватися як аргументи під час виклику функцій і повертатися у вигляді результату їхнього виконання. У Lua можна використовувати функції, написані на Lua і на C [15].

1.3.2 Протоколи та стандарти зв'язку

1-Ware - основною магістраллю для передавання даних тут виступає

двоспрямована шина, яка має вигляд двожильного дроту. Один дріт використовується для живлення і передачі даних, інший - для заземлення. Топологія мережі - загальна шина. Головна перевага цього стандарту - дешевизна і невибагливість.

X-10 - для передачі сигналу використовується електропроводка будівлі. Також передбачена можливість використання трансиверів, які ловлять радіосигнал від бездротових пристроїв, перетворюють його в потрібний формат і передають в електричну мережу. Ця функція використовується для взаємодії з датчиками і пультами ДУ. Серйозним недоліком X10 є низька швидкість передачі даних.

KNX - дорогий варіант, що користується популярністю в Європі. Як середовище для передавання даних протокол KNX може використовувати шину (виту пару), електричну мережу або радіоканал. Стандарт передбачає різні варіанти топології мережі. Система зобов'язана мати власне джерело живлення, але водночас у ній може бути відсутній центральний контролер, тобто KNX дає змогу створювати децентралізовані рішення, у яких сенсори та виконавчі модулі взаємодіють безпосередньо.

WIFI - стандарт протоколу IEEE 802.11 на обладнання для широкопasmового радіозв'язку, призначеного для організації локальних бездротових мереж.

ZigBee - протокол зв'язку по радіоканалу. Стандарт дає змогу створювати датчики з низьким енергоспоживанням і чудовою чуйністю. ZigBee підтримує комірчасту топологію мережі, за якої окремі компоненти можуть виступати як посередник, що передає сигнал від одного пристрою до іншого. Ніздрювата топологія також дає змогу істотно збільшити область покриття бездротової мережі.

Z-Wave - протокол бездротового зв'язку, який багато в чому схожий на ZigBee - тут і низьке енергоспоживання, і підтримка пористої топології мережі. Ці стандарти також можна порівняти за вартістю обладнання. Z-Wave працює в діапазоні частот до 1 ГГц і оптимізована для передавання простих керуючих

команд з малими затримками (наприклад, увімкнути/вимкнути, змінити гучність, яскравість тощо).

1.4 Датчик температури та вологості

DHT - цифровий датчик температури та вологості, що представлений на рисунку 1.4, є складеним датчиком, який містить калібрований цифровий вихідний сигнал з показаннями температури та вологості. Датчик містить резистивний сенсор вологості та компоненти NTC структури для вимірювання температури. Датчик DHT складається з ємнісного датчика вологості та термістора. Також, датчик містить у собі простенький АЦП для перетворення аналогових значень вологості та температури.

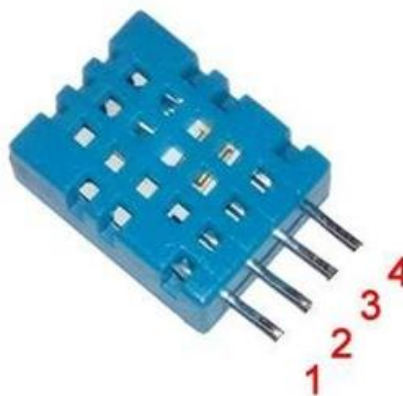


Рисунок 1.4 - Датчик температури та вологості DHT11

Розглянемо два види недорогих цифрових датчиків температури та вологості DHT11 і DHT22.

За цюколюванням ці датчики ідентичні, проте трохи різняться характеристиками.

DHT11:

- дуже низька вартість;

- живлення та I/O 3-5 В;
- визначення вологості 20-80% з 5% точністю;
- визначення температури 0-50 град. з 2% точністю;
- частота опитування не більше 1 Гц (не більше разу на 1 сек.);
- розміри 15.5мм x 12мм x 5.5мм;
- 4 виводи з відстанню між ніжками 0.1" .

DHT22:

- низька вартість;
- живлення та I/O 3-5 В;
- визначення вологості 0-100% з 2-5% точністю;
- визначення температури -40 - 125 град. з точністю ± 0.5 град;
- частота опитування не більше 0.5 Гц (не більше разу на 2 сек.);
- розміри 15.1мм x 25мм x 7.7мм;
- 4 виводи з відстанню між ніжками 0.1" .

Сенсор DHT22 має кращі характеристики. Обидва датчики повільні і зчитувати з них інформацію можливо не частіше, ніж раз на 1 або 2 сек. [16].

1.5 Типи використовуваних реле

Контактне реле SRD-05VDC-SL-C (рисунок 1.5) .

Реле SRD-05VDC-SL-C — це електромеханічне реле з нормально відкритими (NO) і нормально закритими (NC) контактами, що широко використовується в різних електронних і електротехнічних проектах для керування високовольтними ланцюгами з низьковольтними сигналами. Це реле має номінальну напругу котушки 5 В постійного струму (DC).

Основні характеристики котушка (Coil):

- номінальна напруга: 5 В DC;

- номінальний струм котушки: близько 89 мА;
- опір котушки: 56 Ом.

Контакти (Contacts):

- конфігурація контактів, SPDT (Single Pole Double Throw) або 1 форма С (1С). Це означає, що реле має три контакти: загальний (COM), нормально відкритий (NO) і нормально закритий (NC);
- максимальний комутований струм, 10 А при 250 В АС або 10 А при 30 В DC;
- максимальна комутована напруга: 250 В АС або 30 В DC.

Робочі параметри:

- час спрацьовування (Operate Time) 10 мс;
- час відпускання (Release Time) 5 мс;
- електрична довговічність до 100,000 циклів (при номінальних навантаженнях);
- механічна довговічність до 10,000,000 циклів.

Розміри і монтаж:

- розміри 19 мм x 15 мм x 15 мм;
- тип монтажу призначене для монтажу на друковану плату (PCB).

Особливості:

- універсальність, завдяки своїй конфігурації SPDT, реле SRD-05VDC-SL-C може бути використано для перемикання різних типів навантажень, включаючи лампи, двигуни та інші електричні пристрої;
- надійність, висока електрична і механічна довговічність забезпечує тривалий термін служби в різних умовах експлуатації;
- швидкодія, низький час спрацьовування та відпускання забезпечує швидке перемикання ланцюгів.

Реле SRD-05VDC-SL-C використовується в багатьох областях завдяки своїй універсальності та надійності:

- автоматизація та управління, у системах автоматизації для керування

електромагнітними клапанами, двигунами, освітленням та іншими електричними навантаженнями;

- побутова техніка, в різних побутових приладах для перемикання електричних ланцюгів.
- електроніка, у схемах, що потребують відокремлення низьковольтних керуючих сигналів від високовольтних навантажень.
- розумні будинки, у системах розумного дому для автоматичного керування освітленням, опаленням, вентиляцією та іншими системами.

Таким чином, контактне реле SRD-05VDC-SL-C(рис 1.5) є ефективним і надійним рішенням для широкого спектру застосувань, де необхідно керувати високовольтними ланцюгами за допомогою низьковольтних сигналів.



Рисунок 1.5 - Контактне реле SRD(T73) 5V (SRD-05VDC-SL-C)

Твердотільне реле Omron зображено на рисунку 1.6.



Рисунок 1.6 - Твердотільне малогабаритне реле Omron

Основні характеристики. Конструкція і принцип роботи:

- твердотільні реле складаються з напівпровідникових компонентів, таких як тиристори, транзистори або симістори, які забезпечують комутацію електричних ланцюгів без використання механічних контактів;
- SSR перетворюють вхідний сигнал низької потужності (наприклад, від мікроконтролера) у вихідний сигнал, який може управляти високопотужним навантаженням.

Електричні характеристики:

- номінальна напруга: SSR Omron можуть працювати з різними напругами, від низьковольтних (до 24 В) до високовольтних (до 600 В і більше);
- номінальний струм: Від декількох ампер до сотень ампер, залежно від моделі.

Вхідні характеристики:

- управляючий сигнал: SSR можуть приймати різні типи сигналів, включаючи постійний (DC) і змінний (AC) струм.
- напруга управління: зазвичай 3-32 В DC, що робить їх сумісними з багатьма мікроконтролерами і ПЛК.

Вихідні характеристики:

- тип виходу: AC або DC вихід, що дозволяє використовувати реле для управління різними типами навантажень;
- фазне управління: SSR можуть забезпечувати фазове управління для плавного регулювання потужності.

Захисні функції:

- вбудовані захисти від перенапруги, перевантаження і короткого замикання забезпечують безпеку роботи реле і навантаження;
- деякі моделі мають захист від перегріву, що додатково підвищує їх надійність.

Переваги твердотільних реле Omron:

- безконтактність, відсутність механічних частин забезпечує високу

надійність і довговічність;

- безшумність, твердотільні реле працюють без шуму, що важливо для використання у побутових і медичних приладах;
- швидкість перемикання, SSR мають дуже короткий час перемикання, що дозволяє використовувати їх у високочастотних застосуваннях.
- висока стійкість до вібрацій і ударів, завдяки відсутності механічних контактів, SSR менш вразливі до вібрацій і ударів.

Твердотільні реле Omron знаходять широке застосування в різних галузях:

- промислова автоматизація, управління електродвигунами, нагрівальними елементами, освітленням і іншими навантаженнями в промислових процесах.
- побутова техніка, управління побутовими приладами, такими як пральні машини, посудомийні машини, кондиціонери;
- медична техніка, управління медичними приладами, де важливо забезпечити безшумну і надійну роботу;
- інформаційні технології, використання в серверних кімнатах для управління охолоджувальними системами і системами живлення.

Характеристики:

- серія - G3MB;
- канали - SPST-NO (1 Form A);
- тип виходу - AC, Zero Cross;
- струм навантаження -: 2A;
- напруга керуючого входу - 4 ~ 6 VDC;
- напруга – навантаження - 75 ~ 264 V.

Таким чином, твердотільні реле Omron є надійним рішенням для багатьох завдань, де важливо забезпечити безконтактне, безшумне і швидке перемикання електричних навантажень.

2 РОЗРОБКА МОДУЛЯ КЕРУВАННЯ

Модуль ESP8266 може використовуватися як елемент керування електроприладами. Дистанційно вмикати або вимикати будь-який електроприлад зі смартфона, ноутбука, ПК, планшета. З'єднання з пристроями здійснюється бездротовим зв'язком стандарту WIFI.

Модуль ESP8266 – це потужний та дешевий Wi-Fi модуль, який використовується для підключення мікроконтролерів до бездротових мереж. Завдяки своїй універсальності, невеликому розміру та широким можливостям, він став популярним вибором для проектів Інтернету речей (IoT). Нижче наведено кілька основних способів використання модуля ESP8266.

Модуль ESP8266 можна використовувати як веб-сервер для обробки запитів з мережі та управління підключеними до нього пристроями. Наприклад, можна створити веб-інтерфейс для управління освітленням, системами опалення, або іншими приладами.

ESP8266 можна використовувати для збору даних з різних датчиків та передачі їх на віддалений сервер або хмарну платформу. Це може бути корисним для створення систем моніторингу навколишнього середовища, розумних будинків, або промислових додатків.

ESP8266 можна використовувати для створення розумних пристроїв, які можна керувати через інтернет. Наприклад, можна створити розумні розетки, лампи, термостати, або інші пристрої, якими можна керувати через мобільний додаток або веб-інтерфейс.

ESP8266 можна інтегрувати з різними IoT-платформами, такими як ThingSpeak, Blynk, або Adafruit IO. Це дозволяє створювати складні системи моніторингу та керування пристроями, використовуючи готові інструменти для візуалізації даних, віддаленого керування та автоматизації.

ESP8266 може бути використаний для автоматизації різних процесів у домашніх або промислових умовах. Наприклад, можна створити систему

автоматичного поливу рослин, яка буде вмикатися в залежності від показників вологості ґрунту, або систему керування освітленням, яка буде вмикатися і вимикатися в залежності від часу доби або рівня освітленості.

Модуль ESP8266 є потужним інструментом для створення різноманітних IoT-рішень, завдяки своїй універсальності, доступності та простоті використання.

2.1 Розробка електричної схеми модуля керування електроприладами

Розглянемо систему дистанційного керування звичайної розетки. У вигляді навантаження може виступати будь-який електроприлад, що не перевищує максимальне навантаження обраного реле. Реле можна використовувати як контактне так і твердотільне. Перевага твердотільного реле в тому, що воно має компактні розміри порівняно з контактним.

Електричну схему з'єднання показано на рисунку 2.1.

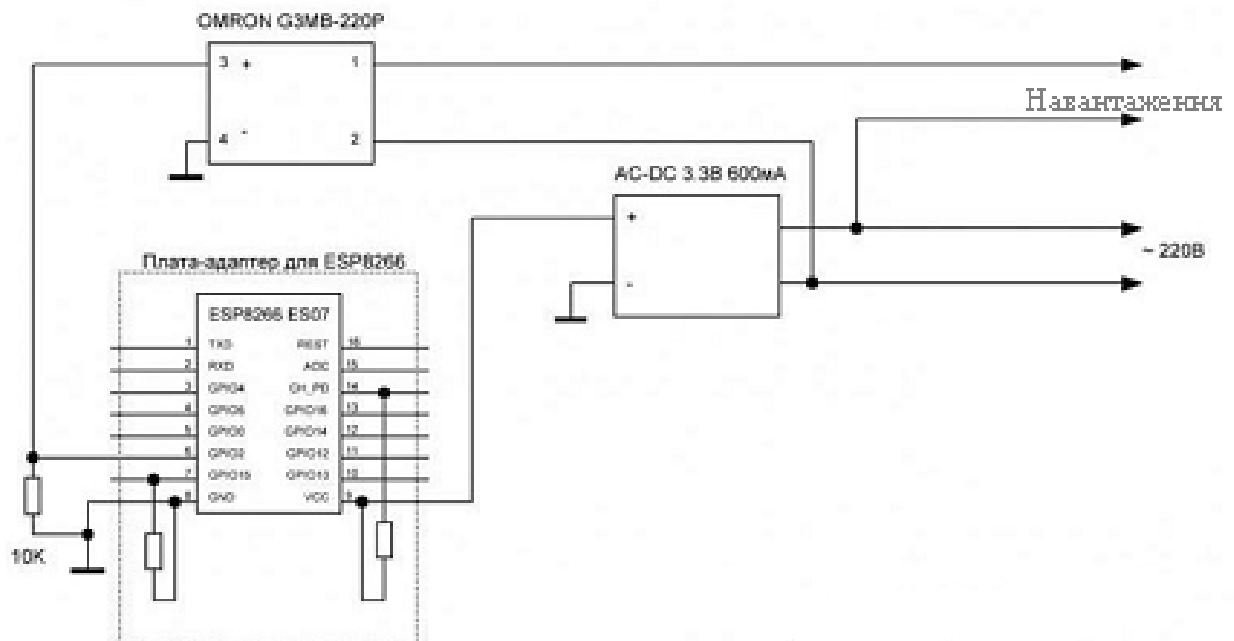


Рисунок 2.1 – Електрична схема з'єднання

Схема складається з:

- перетворювача напруги (1) (що складається зі знижувального трансформатора і випрямляча з виведенням напруги на виході 3.3 вольт, для живлення модуля ESP);

- твердотільного реле Omron G3MB-220P (2) (має компактні розміри, максимальний струм навантаження 2А);

- датчика температури та вологості DHT11;

- модуля ESP8266 (3);

- резисторів R1,R2,R3 (резистори R1, R3 - можуть бути номіналом від 4,7 кОм до 50 кОм, а резистор R2 - номіналом до 10 кОм). Підключення цих резисторів необхідна вимога виробника для забезпечення стабільної роботи пристрою.

Конфігурація виводів модуля може бути різною, все залежить від того, які завдання виконуватиме цей модуль.

У цьому проєкті модуль буде сконфігурований таким чином - виводи GPIO0, GPIO2, GPIO4, GPIO5, GPIO14, GPIO16 будуть призначені для підключення реле, а виводи GPIO12, GPIO13, GPIO15 - для підключення датчиків DHT. На схемі реле підключено до виводу GPIO2 модуля. Аналогічно можна підключити реле до інших виводів. Датчик DHT11 під'єднано до виводу GPIO12.

2.2 Конфігурація модуля

Конфігурацію виводів контролера налаштовано таким чином.

Піни для реле:

- module.IO0 = 0 -- GPIO16;

- module.IO1 = 1 -- GPIO5;

- module.IO2 = 2 -- GPIO4;

- module.IO3 = 3 -- GPIO0;

- module.IO4 = 4 -- GPIO2;

- module.IO5 = 5 -- GPIO14.

Піни для датчиків dht11:

- module.IO6 = 6 -- GPIO12;

- module.IO7 = 7 -- GPIO13.

GPIO0-GPIO16 - позначення виводів контролера, а цифри від 0 до 7 - призначення номера виводу для керування. (наприклад позначення виводу 0 відповідає маркуванню виводу контролера GPIO16 тощо) Під час увімкнення модуля або рестарту за замовчуванням усі виводи перебувають у стані увімкнено.

2.2.1 Опис протоколу взаємодії із системою керування

Інформація про датчик DHT і стан виводів контролера надсилається у форматі об'єкта json.

Датчик dht11 працює за інтерфейсом 1-wire (тобто одна лінія для передавання даних в ту і в іншу сторону за часом).

Формат передавання даних, який шле датчик:

- 8 біт - вологість ціла частина;
- 8 біт - вологість дробова частина;
- 8 біт - температура ціла частина;
- 8 біт - температура дробова частина;
- 8 біт контрольна сума.

Таким чином одна посилка даних з датчика - це 40 біт. потім їх розбираєш по порядку, перевіряєш контрольну суму, яка рахується як сума перших 4х байт, тобто: вологість ціла, вологість дробова, температура ціла, температура дробова і порівнюється з останнім байтом контрольної суми.

Формат об'єкта json має такий вигляд:

```
{"id" : 726864, "state" : "1|1|1|1|1|1|1|1|1|1|29,27|0,0"},
```

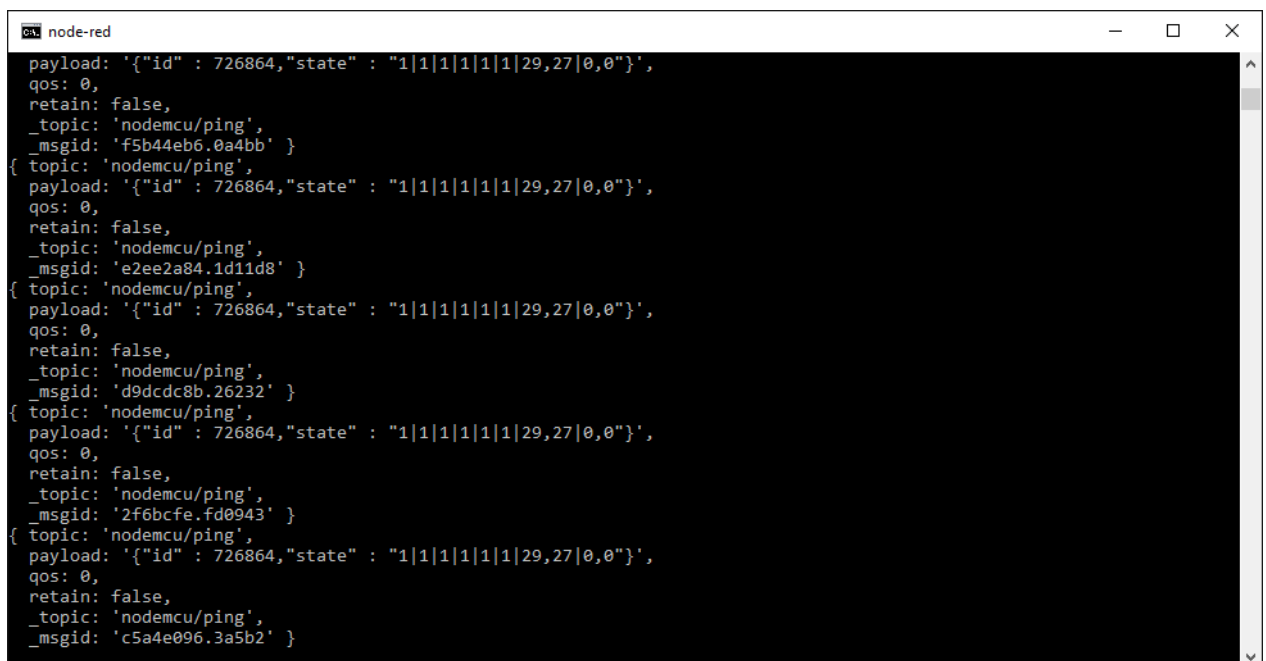
де "id": 726864 - постійний ідентифікаційний номер пристрою, присвоєний йому виробником;

1|1|1|1|1|1|1|1| - стан виводів контролера, налаштованих під реле, нумерація виводів здійснюється зліва на право, тобто 0,1,2,3,4,5 (1 означає увімкнено, 0 - вимкнено);

- виходи 6 і 7 сконфігуровані під приймання інформації з датчиків температури та вологості DHT;

- |29,27| - 29 це відображення температури, 27 - показник вологості з датчика DHT. На 7 виводі відображаються нулі, оскільки до цього виходу не під'єднано датчик.

На рисунку 2.2 показано приклад приймання повідомлень від контролера mqtt клієнтом node-red, який є оброблювачем даних формату json, системи управління.



```
node-red
payload: '{"id": 726864,"state": "1|1|1|1|1|1|29,27|0,0"}',
qos: 0,
retain: false,
_topic: 'nodemcu/ping',
_msgid: 'f5b44eb6.0a4bb' }
{ topic: 'nodemcu/ping',
  payload: '{"id": 726864,"state": "1|1|1|1|1|1|29,27|0,0"}',
  qos: 0,
  retain: false,
  _topic: 'nodemcu/ping',
  _msgid: 'e2ee2a84.1d11d8' }
{ topic: 'nodemcu/ping',
  payload: '{"id": 726864,"state": "1|1|1|1|1|1|29,27|0,0"}',
  qos: 0,
  retain: false,
  _topic: 'nodemcu/ping',
  _msgid: 'd9dc8b.26232' }
{ topic: 'nodemcu/ping',
  payload: '{"id": 726864,"state": "1|1|1|1|1|1|29,27|0,0"}',
  qos: 0,
  retain: false,
  _topic: 'nodemcu/ping',
  _msgid: '2f6bcfe.fd0943' }
{ topic: 'nodemcu/ping',
  payload: '{"id": 726864,"state": "1|1|1|1|1|1|29,27|0,0"}',
  qos: 0,
  retain: false,
  _topic: 'nodemcu/ping',
  _msgid: 'c5a4e096.3a5b2' }
```

Рисунок 2.2 - Приймання повідомлень від контролера

2.3 Розробка програмного забезпечення модуля

Існує три варіанти реалізації цього завдання:

Розроблення програмного забезпечення модуля мовою C++ з використанням бібліотек SDK фірми EspressifSystem і бібліотек сторонніх розробників.

Розробка програмного забезпечення на скриптовій мові Lua з використанням готової операційної системи NodeMSU, яка має свою файлову систему SPIFFS, вбудований інтерпритатор вміє виконувати Lua скрипти з пам'яті пристрою.

Розробка програмного забезпечення модуля мовою C++ з використанням багатозадачної операційної системи реального часу FreeRTOS, написаної мовою C, з відкритим вихідним кодом. Це рішення дає змогу створювати кілька процесів, що виконують свої завдання.

У своїй роботі я вибрав варіант з використанням операційної системи NodeMsu і мови програмування Lua. Оскільки цей варіант є найбільш швидким і простим у плані реалізації поставленого завдання. Але у нього є свої мінуси. Він не підійде для реалізації проєктів, у яких потрібно забезпечити багатозадачність цього модуля, через брак пам'яті.

Виконання поставлених завдань передбачає розробку внутрішнього програмного забезпечення контролера з підтримкою протоколу MQTT. Для реалізації обміну даними між кінцевим WIFI пристроєм і сервером.

Найоптимальнішою і найзручнішою базою для розробки ПЗ є скриптова мова LUA. Необхідно створити ПЗ для цього мікроконтролера, що складається зі скриптів, написаних мовою LUA. Але для того щоб ці скрипти виконувалися, потрібно прошити модуль прошивкою з вбудованим інтерпретатором мов високого рівня. Ця прошивка дає змогу довантажувати через UART і виконувати скрипти розробника пристрою.

У цьому проєкті я буду використовувати прошивку NodeMCU. Прошивка вміє виконувати Lua-скрипти як з UART (аналогічно AT-командам), так і з внутрішньої flash пам'яті. Для завантаження скриптів у flash-пам'ять підтримується файлова система spiffs. Можна створити безліч скриптів на LUA і записати їх у файлову систему. Після чого через WIFI можна викликати потрібні скрипти на виконання.

Основний недолік у тому, що VM LUA виконує LUA скрипти, розміщені в оперативній пам'яті кристала. А цієї пам'яті для скриптів усього 20 Кбайт. Цього

обсягу пам'яті вистачає на виконання скрипта приблизно в 110 рядків. Тому створення порівняно великих скриптів для цього модуля має свою специфіку:

- необхідно алгоритм розділити на лінійні блоки;
- записати ці блоки в окремі файли файлової системи модуля;
- виконувати ці модулі за допомогою оператора dofile.

Під час написання модулів треба дотримуватися таких правил:

- наприкінці кожного модуля потрібно викликати збирач сміття;
- для обміну даними між модулями використовувати глобальні змінні, а для обчислень усередині модулів – локальні.

Приймання та передавання даних від модуля до сервера здійснюється засобами мережі WIFI за протоколом MQTT. Схему під'єднання пристроїв показано на рисунку 2.3.

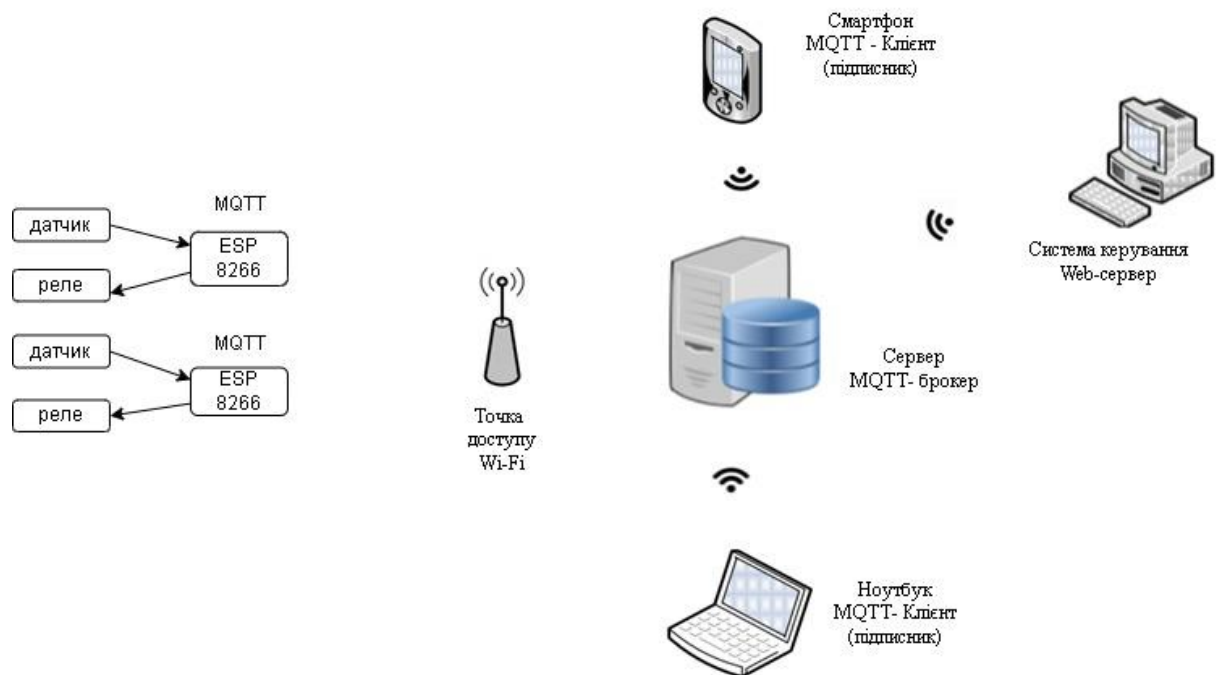


Рисунок 2.3 - Схема підключення пристроїв

Основні функції, які має виконувати модуль:

- ініціалізація пристрою;
- приймання команд від сервера (брокера);
- подача сигналу на відповідні виводи модуля для керування елементами

типу реле;

- читання даних, отриманих з кінцевих пристроїв;
- передача даних (стану) на сервер (брокер);
- періодичне інформування сервера (брокера) про функціонування пристрою.

2.3.1 Опис модулів програмного забезпечення контролера

Програмне забезпечення модуля складається з 8 файлів (скриптів):

Найголовніший файл прошивки - це файл ініціалізації `init.lua` (Додаток А). Він починає виконуватися після подачі живлення або рестарту модуля. У ньому описуються основні дії, які треба виконати перед виконанням основного блоку і, відповідно, передається управління на основний блок. Інші файли можуть бути названі довільно:

- `config.lua` - конфігураційний файл, у якому вказується SSID точки доступу, її пароль, IP-адреса брокера, порт брокера, гілки MQTT, куди надсилаємо і з яких зчитуємо інформацію, вказуються виводи, до яких під'єднуємо реле, та виводи, до яких під'єднуємо датчики dht (Додаток Б);

- `setup.lua` - початкове налаштування виходів модуля під час увімкнення всі виходи перебувають у режимі увімкнено (Додаток В);

- `wifi_connect.lua` - модуль, який здійснює підключення до існуючої WiFi мережі (Додаток Г);

- `webserver.lua` - модуль, який розгортає веб-сервер на базі мікроконтролера для можливості керувати виводами через веб-інтерфейс за протоколом HTTP за допомогою get-запитів (Додаток Д);

- `io_module.lua` - модуль керування цифровими входами тобто модуль, що здійснює ввімкнення або вимкнення виводів контролера (Додаток Е);

- `application.lua` - модуль, що виконує основну логіку програми і MQTT протоколу. Надсилає пінг і свій id брокеру, стан виводів контролера, дані з датчиків температури і вологості, чекає повідомлень від брокера (Додаток Ж);

- `dht11.lua` - модуль роботи з датчиком. У цьому модулі проводиться опитування датчика Датчик dht11 працює за інтерфейсом 1-wire (тобто одна лінія

для передачі даних в інший бік за часом) (Додаток К).

3 РЕАЛІЗАЦІЯ РОЗРОБКИ

3.1 Алгоритм роботи пристрою

Якщо команди немає, пристрій продовжує перебувати в режимі очікування. Блок-схему алгоритму роботи модуля наведено на рисунку 3.1.



Рисунок 3.1 - Блок-схема алгоритму роботи модуля

Під час увімкнення живлення відбувається ініціалізація пристрою, на цьому етапі здійснюється під'єднання модуля до наявної WIFI-мережі, запускається mqtt-клієнт, який здійснює пінг брокера, надсилення стану виводів з інтервалом часу 2 с, і прослуховування порту 1884 для приймання повідомлень від брокера. Інтервал часу 2с виставлено як тестовий, надалі його можна змінити залежно від розв'язуваного завдання.

При появі команди від брокера відбувається її виконання і пристрій повертається в режим очікування команди.

3.2 Підготовка модуля до роботи

Порядок прошивки модуля має наступні кроки:

- необхідно завантажити прошивку NodeMCU;
- завантажити програму для зміни прошивки XTCOM_UTIL або ESP8266Flasher;
- підключити модуль через перетворювач USB-COM до ПК (для заливки прошивки вивід GPIO0 має бути "підтягнутий" до GND);
- запускаємо програму XTCOM_UTILITY, у меню Tools вибираємо Config Device.

Вибираємо COM порт і швидкість 115200. Програма підтримує максимально COM6.

Якщо перетворювач визначився як COM9 або 8, то треба його замінити в Диспетчері пристроїв.

Диспетчер пристроїв - Параметри порту - Додатково.

Схема підключення представлена на рисунку 3.2.

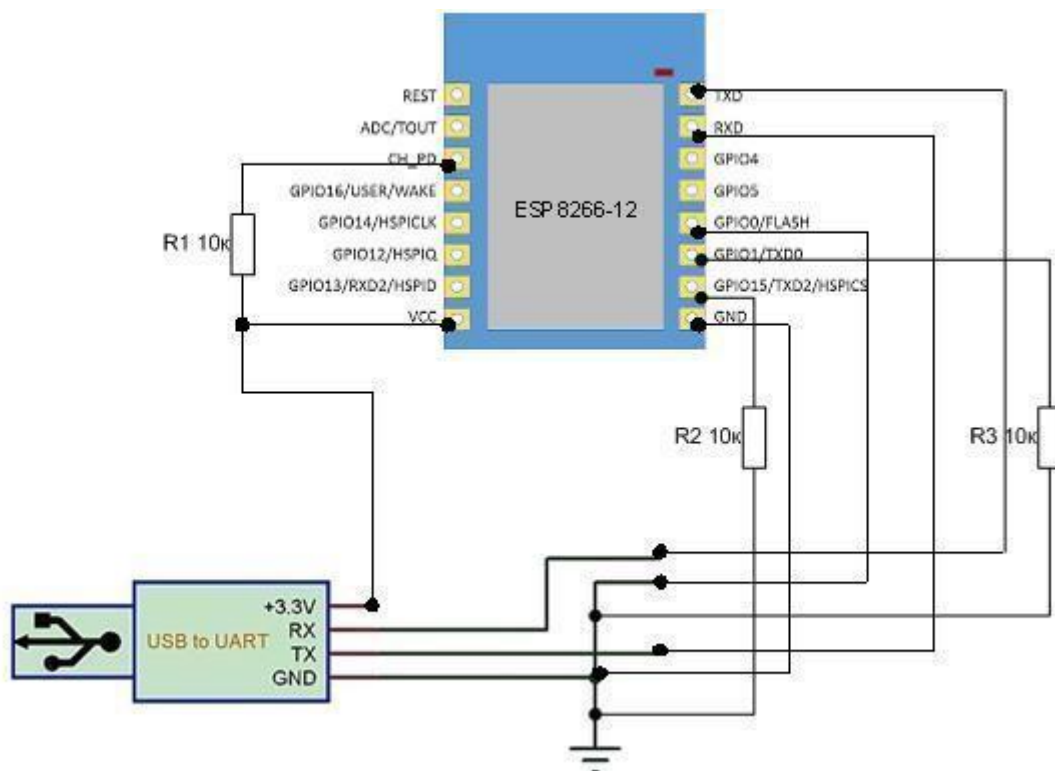


Рисунок 3.2 - Схема підключення модуля до USB to UART під час прошивання

Успішне з'єднання з контролером показано на рисунку 3.3.

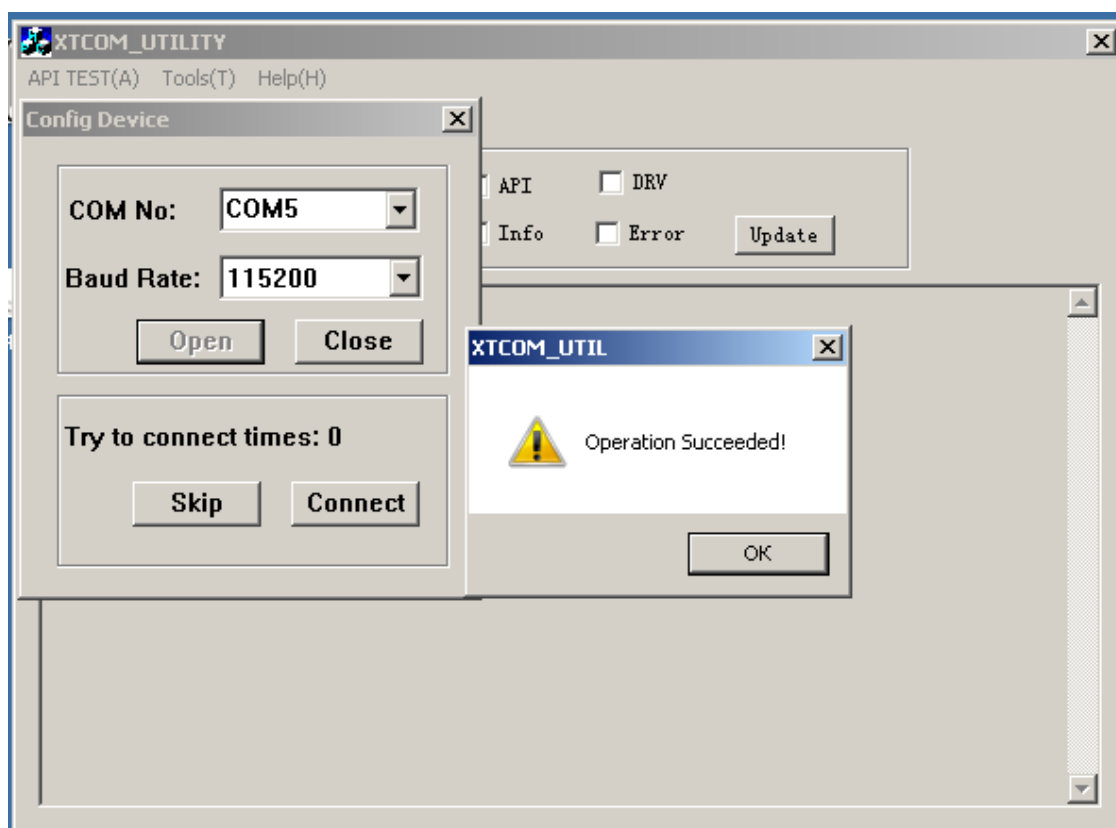


Рисунок 3.3 - З'єднання з контролером

Далі в API TEST(A) вибираємо (4)Flash Image Download. Вказуємо шлях до файлу прошивки, адресу залишаємо 0x00000, тиснемо Download і чекаємо закінчення процесу прошивки (рис. 3.4).

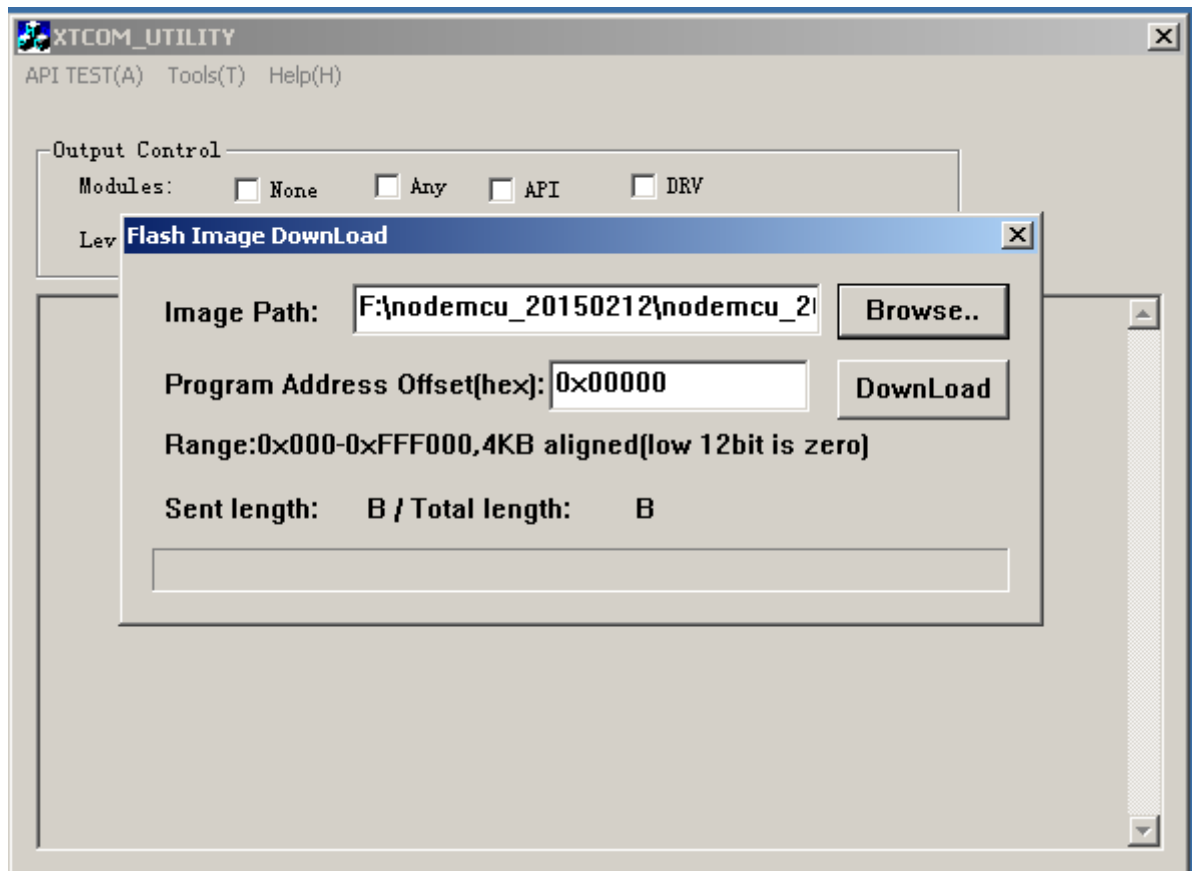


Рисунок 3.4 – Початок прошивки

Тепер від'єднаємо GPIO0 від GND. Перевантажимо модуль живленням. Далі можна приступити до написання скриптів.

Середовище розробки може бути звичайним блокнотом, FAR-менеджером, але набагато зручніше використовувати спеціальну програму ESPlorer, що дає змогу писати скрипти і завантажувати їх на флеш-пам'ять контролера. Але перед її використанням потрібно встановити JAVA, оскільки програма працює з використанням цього середовища. Інтерфейс роботи програми представлений на рисунку 3.5.

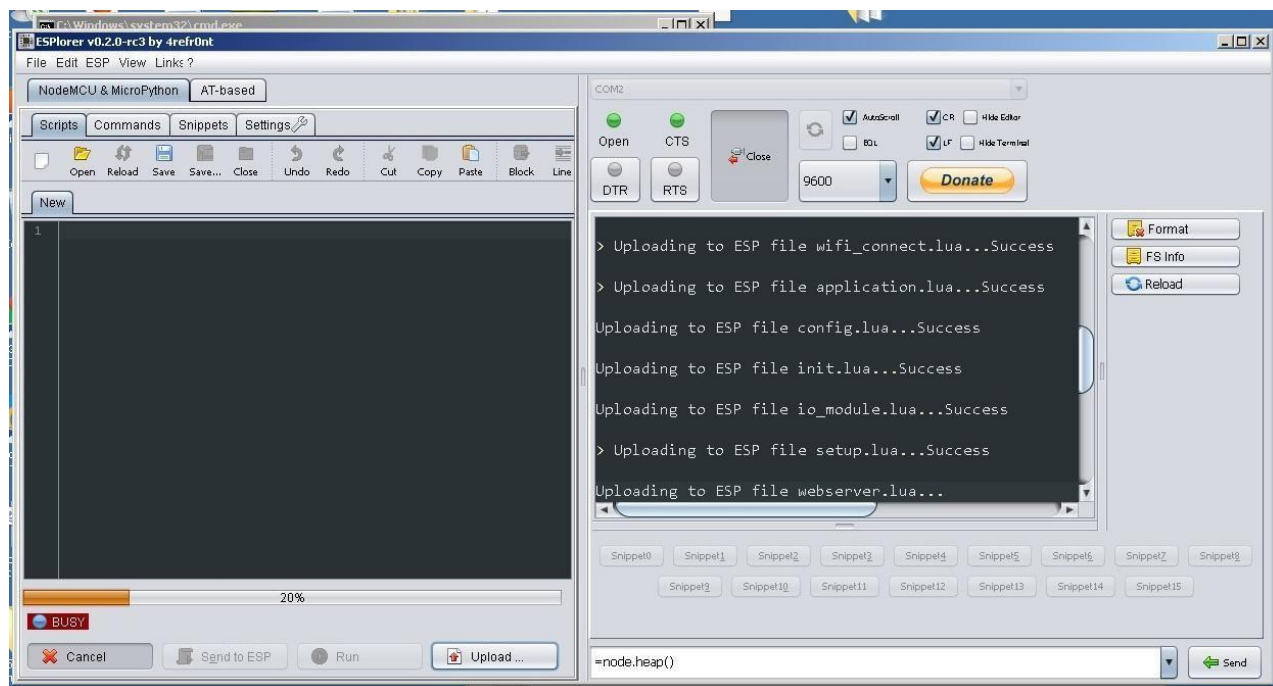


Рисунок 3.5 - Завантаження скриптів у пам'ять мікроконтролера за допомогою програми ESPlorer

Праворуч зверху вибираємо COM порт і натискаємо Open. Нижче розташовано монітор порту, де видно команди, що надсилаються в модуль, і його відповіді. Ще нижче список деяких команд, які можна вибрати і послати в модуль кнопкою Send. Зліва закладки з файлами прошивки. Їх можна додавати, видаляти, коригувати. Під файлами розташовані кнопки для керування модулем.

3.3 Опис режимів роботи

Контролер може працювати як самостійний пристрій при використанні вбудованого веб-сервера, який розгортається в пам'яті пристрою. Керування виводами здійснюється через веб інтерфейс за протоколом HTTP, за допомогою get-запитів, підключившись безпосередньо до самого модуля за його IP-адресою. Під час під'єднання до модуля відкривається вебсторінка, на якій відображено найменування виводів контролера, а на протилежному боці праворуч

розташовуються 2 кнопки ON і OFF відповідно увімкнути і вимкнути. Схема підключення показана на рисунку 3.6.

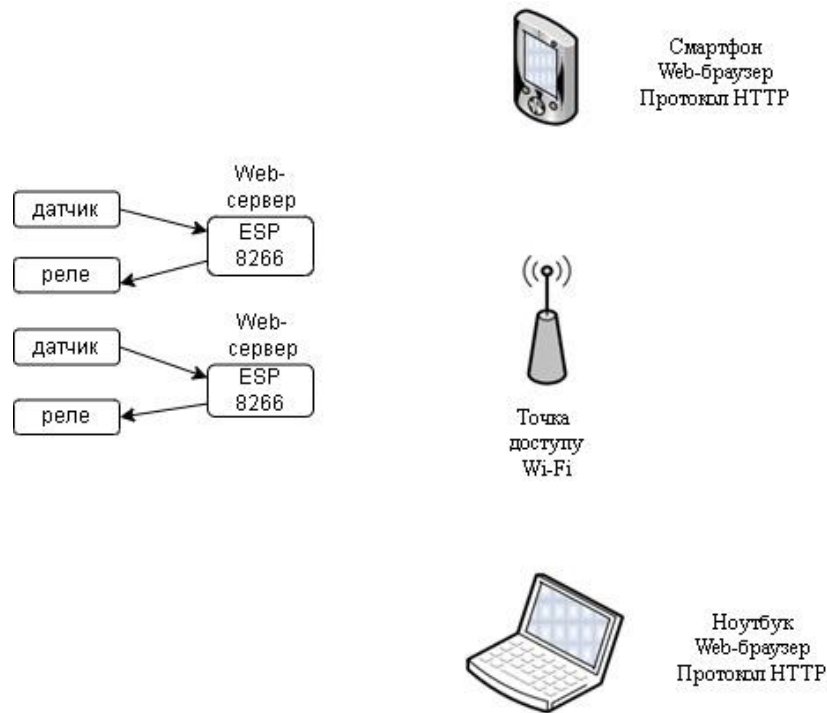


Рисунок 3.6 - Схема підключення пристроїв без брокера

Або контролер може бути елементом системи управління розглянутої вище, в якій приймання та передавання даних здійснюється через MQTT-брокер за протоколом MQTT. У цьому разі модуль постійно в заданий інтервал часу надсилає на сервер свій ідентифікаційний номер, присвоєний йому виробником, стан виводів (у вигляді 1 або 0 увімкнути або вимкнути), та інформацію з датчика температури і вологості у форматі об'єкта json. І здійснює прослуховування порту 1884 для приймання повідомлень від сервера. Керування здійснюється як із системи керування через веб інтерфейс, команда надсилається через брокер у вигляді повідомлення в гілку /NodeMsu/ 4ON або 4OFF (4 - номер виводу, ON/OFF - увімкнути, вимкнути), так і з застосунку під андроїд зі смартфона або планшета.

3.4 Загальний опис роботи та взаємодія програмних модулів

Під час подачі живлення на модуль починає виконуватися файл ініціалізації `init.lua`, у якому описано основні дії, які потрібно виконати перед виконанням основного блоку. Цей скрипт реалізує сценарій увімкнення видалених файлів. Далі виконується скрипт `setup.lua`, в якому описано початкову конфігурацію виходів GPIO контролера, потім стартує модуль підключення до WiFi мережі.

Здійснюється підключення до наявної WiFi мережі відповідно до параметрів, заданих у конфігураційному файлі.

У конфігураційному файлі зазначено параметри точки доступу, до якої під'єднується модуль (`ssid` і пароль), а також налаштування для під'єднання до брокера (його IP-адреса, порт і гілки, за якими він буде відправляти й отримувати повідомлення від брокера).

Далі паралельно стартують веб-сервер і застосунок, що виконує основну логіку програми та MQTT протоколу `application.lua` (запускається MQTT-клієнт, створюється дві гілки приймання й передавання даних, модуль починає пінгувати MQTT-брокер за IP-адресою, зазначеною у файлі конфігурації, та надсилати йому свій ідентифікаційний номер ID, інформацію про стан виходів та інформацію, отриману з датчика DHT, у форматі json, а друга гілка переходить у режим прослуховування порту 1884, для приймання повідомлень від брокера).

Команда від системи управління надходить у форматі повідомлення на гілку (топiк) `/NODEMSU/`.

Формат повідомлення такий `/NODEMSU/2ON` або `/NODEMSU/2OFF`. Цифра означає номер виводу контролера, `ON/OFF` - `ON/OFF` (увімкнути/вимкнути).

Для роботи з модулем через Web-сервер в адресному рядку браузера набираємо IP-адресу модуля, відкривається HTML-сторінка, на якій відображено всі сконфігуровані виводи контролера. Відображатися вони можуть по-різному,

все залежить від того, як ви їх назвете. Праворуч від назви виводу дві кнопки ON і OFF увімкнути і вимкнути. Дизайн, тло сторінки та оформлення кнопок за потреби можна змінити у файлі конфігурації Web-сервера.

Web-сервер передбачено на той випадок, якщо неможливо з якихось причин під'єднатися через систему управління.

ВИСНОВКИ

Під час виконання цієї роботи було вирішено всі поставлені завдання.

Результатом цієї роботи є внутрішнє програмне забезпечення модуля ESP8266, написаного скриптовою мовою LUA.

ПЗ було протестовано з використанням локального брокера, розгорнутого на локальному ПК (MQTT-брокер Mosquitto), MQTT-клієнта у вигляді стандартного застосунку під Android (MyMqtt), встановленого на смартфон, самого модуля ESP8266, датчика температури та вологості DHT11, контактного реле керування навантаженням SRD-05VDC-SL-C, як навантаження - лампа розжарювання потужністю 100 ват напругою 220 вольт.

Для реалізації складніших і багатозадачних рішень необхідно використовувати варіанти розроблення програмного забезпечення на мові C++ з використанням бібліотек SDK, або операційної системи з відкритим вихідним кодом FreeRTOS.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Грінгард С. Інтернет речей. Харків: Книжковий Клуб «Клуб Сімейного Дозвілля», 2018. – 176 с. ISBN 978-617-12-4657-7.
2. Що таке HomeKit? Все що потрібно знати про «розумний будинок» [Електронний ресурс] – Режим доступу: <https://mac.org.ua/reviews/what-homekit> (дата звернення 30.03.2024).
3. «Розумний офіс Microsoft» [Електронний ресурс] – Режим доступу: <https://www.microsoft.com/uk-ua/microsoft-365/microsoft-office> (дата звернення 17.03.2024).
4. «Місто високих технологій microsoft» [Електронний ресурс] – Режим доступу: <https://uk.wikipedia.org/wiki> (дата звернення 29.03.2024).
5. Каско страхування [Електронний ресурс] – Режим доступу: <https://uniqa.ua/private/auto/kasko/> (дата звернення 12.04.2024).
6. Shamonina, V.H. et al. 2019. Using the Proteus virtual environment to train future IT professionals. Educational Dimension. 1, (Dec. 2019), 181–198. DOI:<https://doi.org/10.31812/educdim.v53i1.3842>.
7. Ready for Sky — додаток R4S Gateway [Електронний ресурс] – Режим доступу: https://f.ua/statik/files/products/redmond/rtp-m810s_1078.pdf (дата звернення 22.04.2024).
8. Модуль ESP8266 [Електронний ресурс] – Режим доступу: <https://uk.wikipedia.org/wiki/ESP32> (дата звернення 28.04.2024).
9. Технічні характеристики ESP8266 [Електронний ресурс] – Режим доступу: <https://beegreen.com.ua/ru-ru/wi-fi-modul-esp8266-esp-12e-arduino-avr-pic-10627> (дата звернення 30.03.2024).
10. Мікроконтролер ESP8266 [Електронний ресурс] – Режим доступу: <https://uk.wikipedia.org/wiki/ESP8266> (дата звернення 02.04.2024).
11. Технологія пам'яті. Прошивка. [Електронний ресурс] – Режим доступу: <https://uk.wikipedia.org/wiki> (дата звернення 18.04.2024).

12. Мікроконтролер ESP8266 використання. [Електронний ресурс] – Режим доступу: <https://uk.wikipedia.org/wiki/ESP32> (дата звернення 18.04.2024).
13. Андрій Денисенко. Що таке MQTT і для чого він потрібний [Електронний ресурс] – Режим доступу: <https://highload.today/uk/shho-take-mqtt-i-dlya-chogo-vin-potribnij/> (дата звернення 2.04.2024).
14. Що таке SDK? Повний посібник для розробників [Електронний ресурс] – Режим доступу: <https://polaridad.es/uk/> (дата звернення 13.04.2024).
15. LUA - файл мови програмування [Електронний ресурс] – Режим доступу: <https://docs.fileformat.com/uk/programming/lua/> (дата звернення 15.04.2024).
16. Датчик вологості та температури DHT11 [Електронний ресурс] – Режим доступу: <https://arduino.ua/prod185-datchik-vlajnosti-i-temperatyri-dht11> (дата звернення 10.04.2024).

ДОДАТОК А

МОДУЛЬ ІНІЦІАЛІЗАЦІЇ

Лістинг А.1 – код модуля ініціалізації

```
-- file : init.lua
app = require("application")
config = require("config")
setup = require("setup")
webserver = require("webserver")
wifi_connect = require("wifi_connect")
io_module = require("io_module")
dht11 = require("dht11")
setup.start()

print("Hi! NODEMCU START!")
```

ДОДАТОК Б

МОДУЛЬ КОНФІГУРАЦІЇ КОНТРОЛЕРА

Лістинг Б.1 – Конфігурація контролера

```
-- file : config.lua
local module = {}

-- название и пароль Wi-Fi сети
module.SSID = {} module.SSID["ACORP"] = "UXGA4R6P"

-- ip-адрес и порт MQTT-брокера
module.HOST = "192.168.1.3"
module.PORT = 1884

-- Ветки MQTT куда посылваем и с которых считываем информацию
module.ID = node.chipid()
module.MQTT_PUBLISH = "message/"
module.ENDPOINT = "nodemcu/"

-- пины контроллера NODEMCU
module.IO0 = 0
module.IO1 = 1
module.IO2 = 2
module.IO3 = 3
module.IO4 = 4
module.IO5 = 5
-- сверху реле, внизу датчики dht11
module.IO6 = 6
module.IO7 = 7
module.IO8 = 8
return module
```

```
-- пины ESP2866
-- GPIO16
-- GPIO5
-- GPIO4
-- GPIO0
-- GPIO2
-- GPIO14
-- GPIO12
-- GPIO13
-- GPIO15
```

ДОДАТОК В

НАСТРОЙКА МОДУЛЯ ВИВОДІВ

Лістинг В.1 – Код модуля виводів

```
-- file: setup.lua

local module = {}

function module.start()
    print("Config PINS ...")

    -- настраиваем ПИНЫ
    gpio.mode(config.IO0,gpio.OUTPUT)
    gpio.mode(config.IO1,gpio.OUTPUT)
    gpio.mode(config.IO2,gpio.OUTPUT)
    gpio.mode(config.IO3,gpio.OUTPUT)
    gpio.mode(config.IO4,gpio.OUTPUT)
    gpio.mode(config.IO5,gpio.OUTPUT)
    gpio.mode(config.IO6,gpio.OUTPUT)
    gpio.mode(config.IO7,gpio.OUTPUT)
    gpio.mode(config.IO8,gpio.OUTPUT)

    gpio.write(config.IO0,gpio.HIGH)
    gpio.write(config.IO1,gpio.HIGH)
    gpio.write(config.IO2,gpio.HIGH)
    gpio.write(config.IO3,gpio.HIGH)
    gpio.write(config.IO4,gpio.HIGH)
    gpio.write(config.IO5,gpio.HIGH)
    gpio.write(config.IO6,gpio.HIGH)
    gpio.write(config.IO7,gpio.HIGH)
    gpio.write(config.IO8,gpio.HIGH)
```

```
-- конфигурируем wifi
print("Configuring WiFi ...")
-- as WiFi access point
-- wifi_accesspoint.start()

-- connect to wifi network
wifi_connect.start()

end

return module
```

ДОДАТОК Г

Модуль підключення до мережі wifi

Лістинг Г – Код підключення до мережі

```
-- file : wifi_connnect.lua
local module = {}

local function wifi_wait_ip()
    if wifi.sta.getip() == nil then
        print("IP unavailable, Waiting...")
    else
        tmr.stop(1)
        print("\n=====")
        print("ESP8266 mode is: " .. wifi.getmode())
        print("MAC address is: " .. wifi.ap.getmac())
        print("IP is " .. wifi.sta.getip())
        print("=====")

        -- -- Законнектились к wi-fi получили IP-адрес. Запускаем наше
MQTT- приложение
        app.start()
        -- webserver.start()
    end
end

local function wifi_start(list_aps)
    if list_aps then
        for key, value in pairs(list_aps) do
            if config.SSID and config.SSID[key] then
                wifi.setmode(wifi.STATION);
            end
        end
    end
end
```

```

        wifi.sta.config(key,config.SSID[key])
        wifi.sta.connect()
        print("Connecting to " .. key .. " ...")
        --config.SSID = nil -- can save memory
            -- Законнектились к wi-fi. ждем IP-адресс
        tmr.alarm(1, 2500, 1, wifi_wait_ip)
    end
end
else
    print("Error getting AP list") end
end

function module.start()

-- пытаемся законнектиться к wi-fi
    print("Trying connect to WiFi")
    wifi.setmode(wifi.STATION);
    wifi.sta.getap(wifi_start)
end

return module

```

ДОДАТОК Д

Модуль створення web-серверу

Лістинг Д – Код для створення Web-серверу

```
-- file : webserver.lua
local module = {}
srv = nil
function module.start()
print("Start webserver")
srv=net.createServer(net.TCP)
srv:listen(80,function(conn)
    conn:on("receive", function(client,request)
        local buf = "";
        local _, _, method, path, vars = string.find(request, "([A-Z]+)
(\\.+)?(\\.+ HTTP");
        if(method == nil)then
            _, _, method, path = string.find(request, "([A-Z]+) (\\.+
HTTP");
        end
        local _GET = {}
        if (vars ~= nil)then
            for k, v in string.gmatch(vars, "(%w+)=(%w+)&*") do
                _GET[k] = v
            end
        end
        buf = buf.."<h1> ESP8266 Web Server</h1>";
buf = buf.."<p>GPIO1 <a
href=\\\"?pin=1ON\\\"><button>ON</button></a>&nbsp;<a
href=\\\"?pin=1OFF\\\"><button>OFF</button></a></p>";
        buf = buf.."<p>GPIO2 <a
href=\\\"?pin=2ON\\\"><button>ON</button></a>&nbsp;<a
href=\\\"?pin=2OFF\\\"><button>OFF</button></a></p>";
        buf = buf.."<p>GPIO3 <a
```

```

href="\?pin=3ON\"><button>ON</button></a>&nbsp;<a
href="\?pin=3OFF\"><button>OFF</button></a></p>";
    buf = buf.."<p>GPIO4 <a
href="\?pin=4ON\"><button>ON</button></a>&nbsp;<a
href="\?pin=4OFF\"><button>OFF</button></a></p>";
    buf = buf.."<p>GPIO5 <a
href="\?pin=5ON\"><button>ON</button></a>&nbsp;<a
href="\?pin=5OFF\"><button>OFF</button></a></p>";
    buf = buf.."<p>GPIO6 <a
href="\?pin=6ON\"><button>ON</button></a>&nbsp;<a
href="\?pin=6OFF\"><button>OFF</button></a></p>";
    buf = buf.."<p>GPIO7 <a
href="\?pin=7ON\"><button>ON</button></a>&nbsp;<a
href="\?pin=7OFF\"><button>OFF</button></a></p>";
    buf = buf.."<p>GPIO8 <a
href="\?pin=8ON\"><button>ON</button></a>&nbsp;<a
href="\?pin=8OFF\"><button>OFF</button></a></p>";
    local _on, _off = "", ""
    local set_pin = _GET.pin;
    io_module.set_io(set_pin)
    client:send(buf);
    client:close();
    collectgarbage();
end)
end)
end
return module

```

ДОДАТОК Е

МОДУЛЬ КЕРУВАННЯ ВХОДАМИ

Лістинг Е – Код керування входами

```
-- file: io.lua
local module = {}

function module.set_io(set_pin)

    if(set_pin == "0ON")then
        gpio.write(config.IO0, gpio.HIGH);
    elseif(set_pin == "0OFF")then
        gpio.write(config.IO0, gpio.LOW);

    elseif(set_pin == "1ON")then
        gpio.write(config.IO1, gpio.HIGH);
    elseif(set_pin == "1OFF")then
        gpio.write(config.IO1, gpio.LOW);

    elseif(set_pin == "2ON")then
        gpio.write(config.IO2, gpio.HIGH);
    elseif(set_pin == "2OFF")then
        gpio.write(config.IO2, gpio.LOW);

    elseif(set_pin == "3ON")then
        gpio.write(config.IO3, gpio.HIGH);
    elseif(set_pin == "3OFF")then
        gpio.write(config.IO3, gpio.LOW);

    elseif(set_pin == "4ON")then
        gpio.write(config.IO4, gpio.HIGH);
```

```

elseif(set_pin == "4OFF")then
    gpio.write(config.IO4, gpio.LOW);

    elseif(set_pin == "5ON")then
        gpio.write(config.IO5, gpio.HIGH);
    elseif(set_pin == "5OFF")then
        gpio.write(config.IO5, gpio.LOW);

        elseif(set_pin == "6ON")then
            gpio.write(config.IO6, gpio.HIGH);
        elseif(set_pin == "6OFF")then
            gpio.write(config.IO6, gpio.LOW);

            elseif(set_pin == "7ON")then
                gpio.write(config.IO7, gpio.HIGH);
            elseif(set_pin == "7OFF")then
                gpio.write(config.IO7, gpio.LOW);

                elseif(set_pin == "8ON")then
                    gpio.write(config.IO8, gpio.HIGH);
                elseif(set_pin == "8OFF")then
                    gpio.write(config.IO8, gpio.LOW);

            end

        end

    end

function module.start()
end
return module

```

ДОДАТОК Ж

ОСНОВНИЙ ЛОГІЧНИЙ МОДУЛЬ ПРОГРАМИ

Лістинг Ж – Код логічного модуля програми

```
-- file : application.lua
local module = {}
m = nil

-- Отсылаем информацию брокеру каждые 2 сек.
-- функция ping'a
local function send_ping()
    -- Читаем состояние пинов контроллера
    local state_byte = 0;
    local state_str = "" .. gpio.read(config.IO0);

    state_str = state_str .. "|" .. gpio.read(config.IO1);
    state_str = state_str .. "|" .. gpio.read(config.IO2);
    state_str = state_str .. "|" .. gpio.read(config.IO3);
    state_str = state_str .. "|" .. gpio.read(config.IO4);
    state_str = state_str .. "|" .. gpio.read(config.IO5);

    -- Читаем температуру

    dht11.read(config.IO6)
    state_str = state_str .. "|" .. dht11.getTemperature()
    state_str = state_str .. "," .. dht11.getHumidity()

    dht11.read(config.IO7)
    state_str = state_str .. "|" .. dht11.getTemperature()
    state_str = state_str .. "," .. dht11.getHumidity()
```

```

        m:publish(config.ENDPOINT .. "ping","{\"id\" : \" .. config.ID ..
        \",\"state\" : \"\"..
        state_str ..\"}\"",0,0)

collectgarbage();
end

-- посылаем брокеру MQTT свой ID для регистрации
local function register_myself()
    m:subscribe(config.ENDPOINT .. config.MQTT_PUBLISH ..
config.ID,0,function(conn)
        --m:subscribe(config.ENDPOINT .. config.ID,0,function(conn)
        print("Successfully subscribed to data endpoint")
    end)
end

local function mqtt_start()
    m = mqtt.Client(config.ID, 120)
    -- обработки получения сообщения в данной MQTT-ветке
    m:on("message", function(conn, topic, data)
        if data ~= nil then
            print(topic .. ": " .. data)

-- Приняли сообщение в свою MQTT-ветку. Обработываем его.

io_module.set_io(data)

        end
    end)
-- соединяемся с брокером
m:connect(config.HOST, config.PORT, 0, 1, function(con)
    register_myself()
-- Посылаем инфорамфию о своем присутствии каждые 4 секунды
    tmr.stop(6)
    tmr.alarm(6, 2000, 1, send_ping)
end)

```

```
end
-- функция начала работы модуля
function module.start()
    mqtt_start()
end

return module
```

ДОДАТОК К

Модуль роботи з датчиком dht11

Лістинг К – Код датчика DHT11

```
local module = {}

local humidity
local temperature
local checksum
local checksumTest
local humiditTenth
local temperatureTenth

function module.read(pin)
    humidity = 0
    temperature = 0
    checksum = 0
    humiditTenth=0
    temperatureTenth=0

    gpio_read = gpio.read
    gpio_write = gpio.write

    bitStream = {}
    for j = 1, 40, 1 do
        bitStream[j] = 0
    end
    bitlength = 0

    -- Шаг 1: посылаем начальный сигнал в DHT11
```

```

gpio.mode(pin, gpio.OUTPUT)
gpio.write(pin, gpio.LOW)
tmr.delay(20000)

```

```

gpio_read=gpio.read
gpio_write=gpio.write

```

```

gpio.mode(pin, gpio.INPUT)

```

```

-- Шаг 2: DHT11 датчик посылает начальный ответный сигнал
-- Ждем когда датчик поднимет и опустит шину данных (изменит
уровень напряжения на ней с высокого низкого на низкий и обратно )

```

```

c=0

```

```

while (gpio_read(pin) == 0 and c < 800) do c = c + 1 end

```

```

c=0

```

```

while (gpio_read(pin) == 1 and c < 100) do c = c + 1 end

```

```

-- bus will always let up eventually, don't bother with timeout

```

```

c=0

```

```

while (gpio_read(pin) == 0 and c < 800) do c = c + 1 end

```

```

c=0

```

```

while (gpio_read(pin) == 1 and c < 100) do c = c + 1 end

```

```

-- Шаг 3: DHT11 посылает данные. 40 бит - то есть 5 байт. 2 байта
температуры - целую и дробную часть. 2 байта влажность. и байт
контрольной суммы

```

```

for j = 1, 40, 1 do

```

```

while (gpio_read(pin) == 1 and bitlength < 10 ) do

```

```

bitlength = bitlength + 1

```

```

end

```

```

bitStream[j] = bitlength

```

```

bitlength = 0

```

```

c=0
while (gpio_read(pin) == 0 and c < 800) do c = c + 1 end
end

--Обработка данных.
for i = 1, 8, 1 do
    if(bitStream[i+0]>2)then
        humidity=humidity+2^(8-i)
    end
    if(bitStream[i+8]>2)then
        humiditTenth=humiditTenth+2^(8-i)
    end
    if(bitStream[i+16]>2)then
        temperature=temperature+2^(8-i)
    end
    if(bitStream[i+24]>2)then
        temperatureTenth=temperatureTenth+2^(8-i)
    end
    if (bitStream[i+32]>2)then
        checksum=checksum+2^(8-i) end
    end

    checksumTest=(humidity+humiditTenth+temperature+temperatureTenth)%2
56
    if checksum ~= checksumTest then
        humidity = -1
        -- temperature = -1
    End
    collectgarbage(); end

-- Функция получить температру
function module.getTemperature()
    return temperature
end

```

```
-- Функция получить влажность  
function module.getHumidity()  
    return humidity  
end  
  
return module
```

Схема підключення пристроїв без брокера

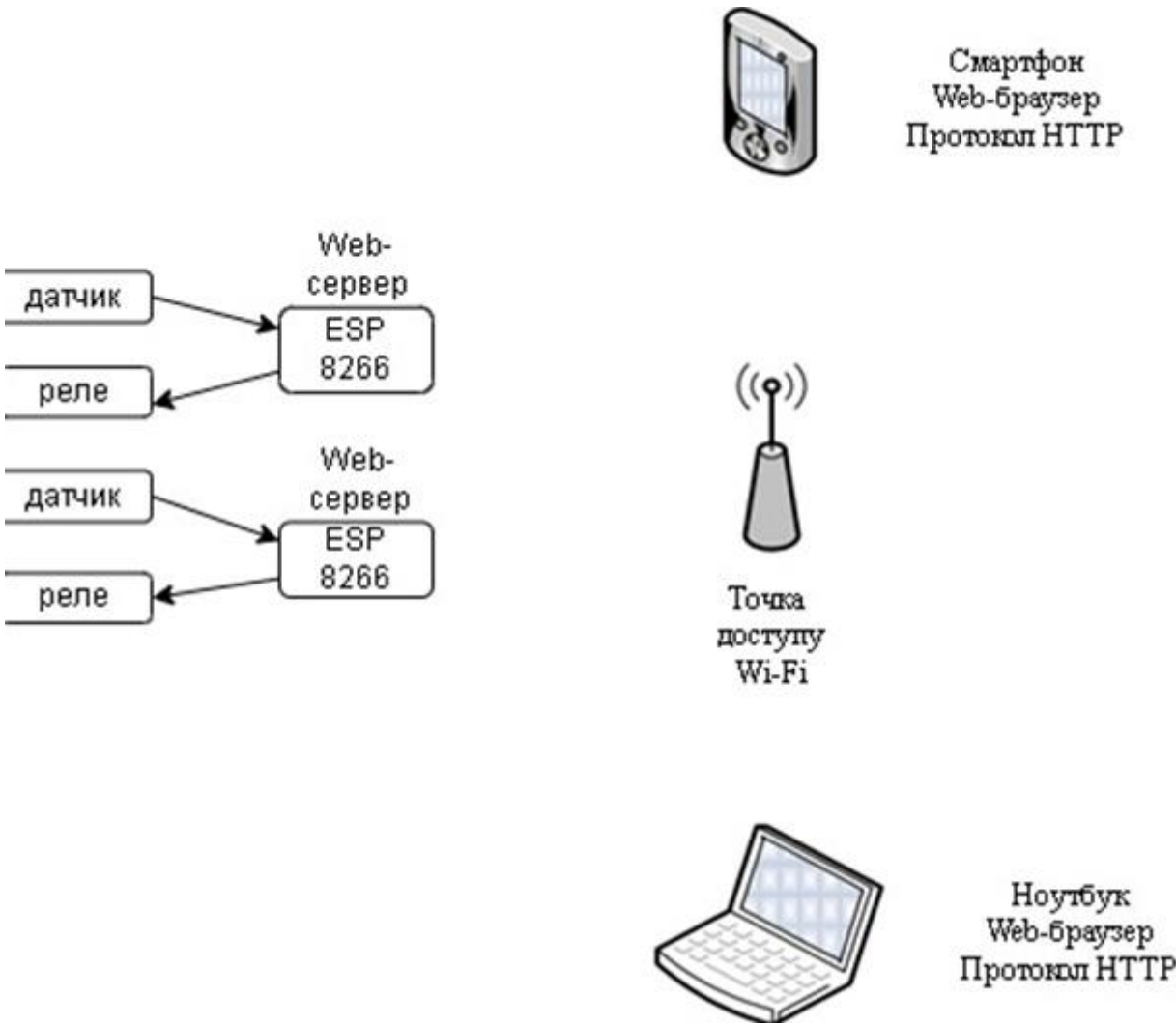
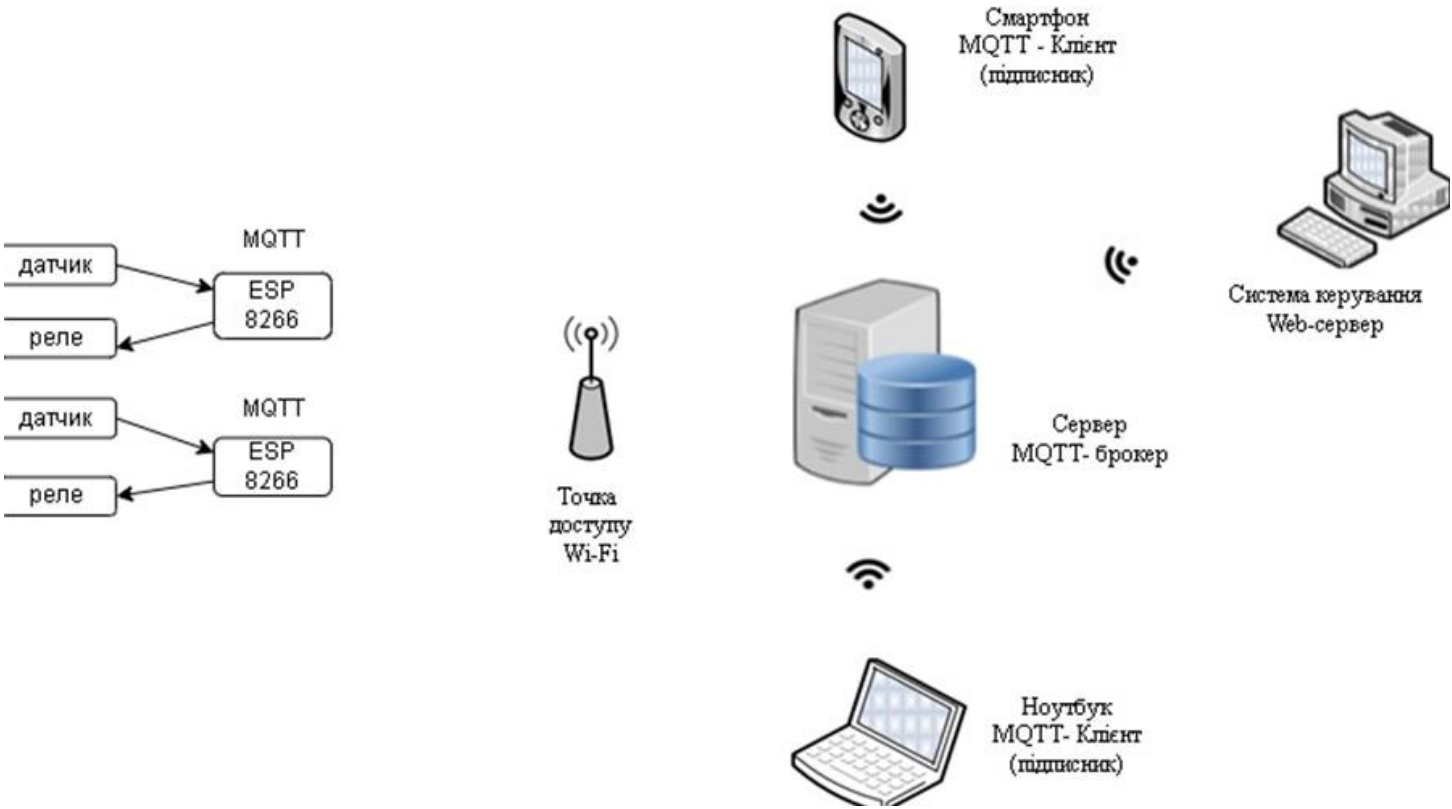
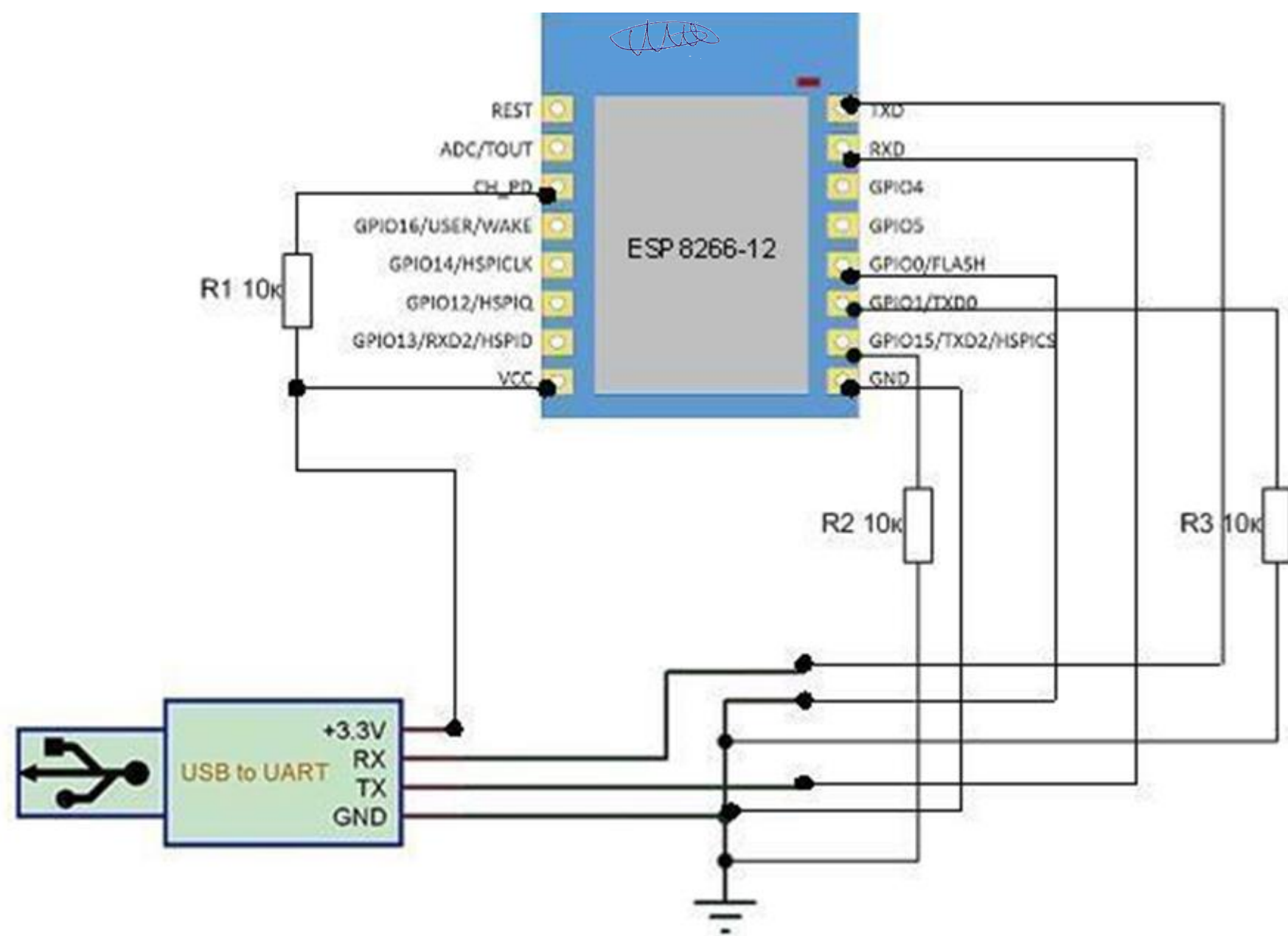


Схема підключення пристроїв



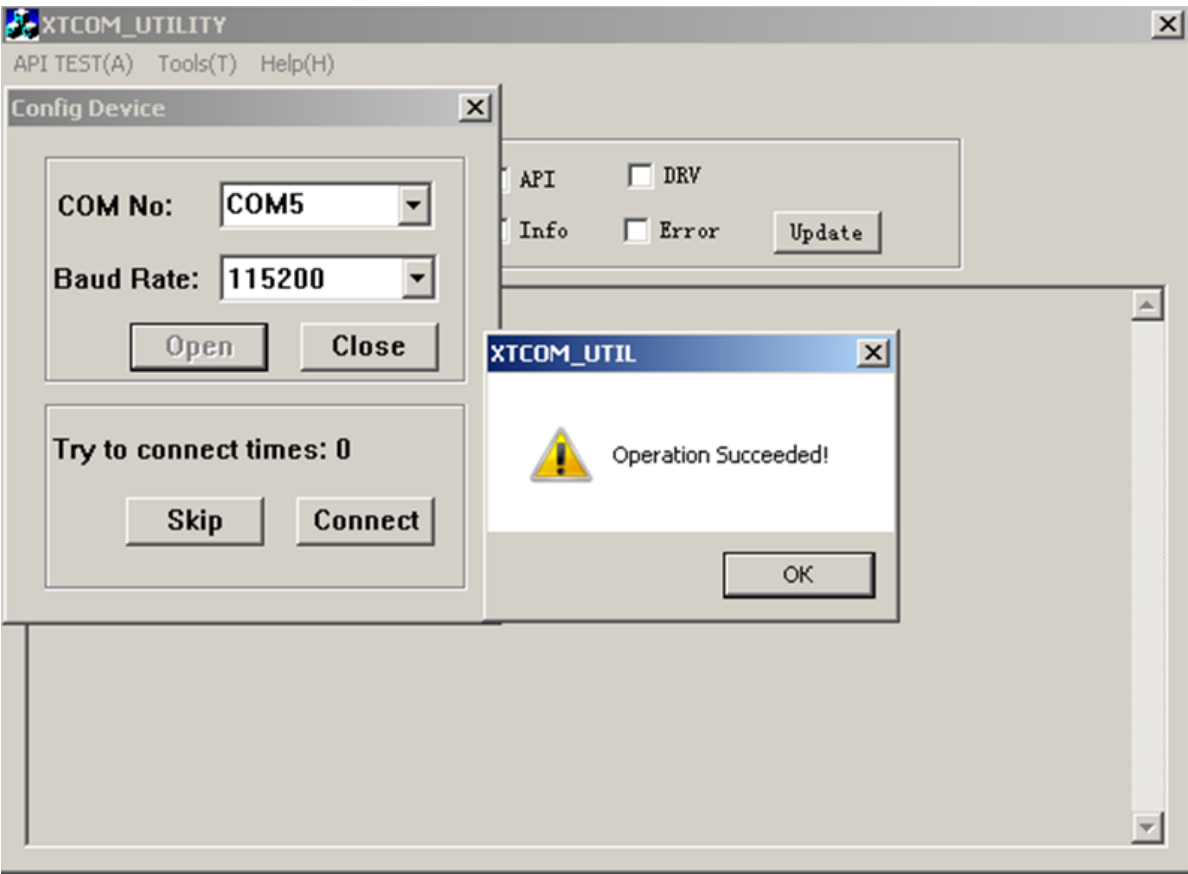
					13.02070849.00045 ПЛ1						
					Розробка програмного забезпечення мікроконтролері на основі ESP8266 з підтримкою бездротової мережі Wi-Fi	Літ.		Маса		Масштаб	
Зм.	Арк.	№ документа	Підпис	Дата		Аркуш 1		Аркушів 1			
Розробив		Мусіченко О. Є.									
Перевірів		Ільяшенко М.Б.									
Т.контр.											
Н.контр.		Польська О.В.			Схема підключення пристроїв	НУ «Запорізька політехніка», гр. КНТ-520					
Затвердив		Кудерметов Р.									

Схема підключення модуля до USB to UART під час прошивання

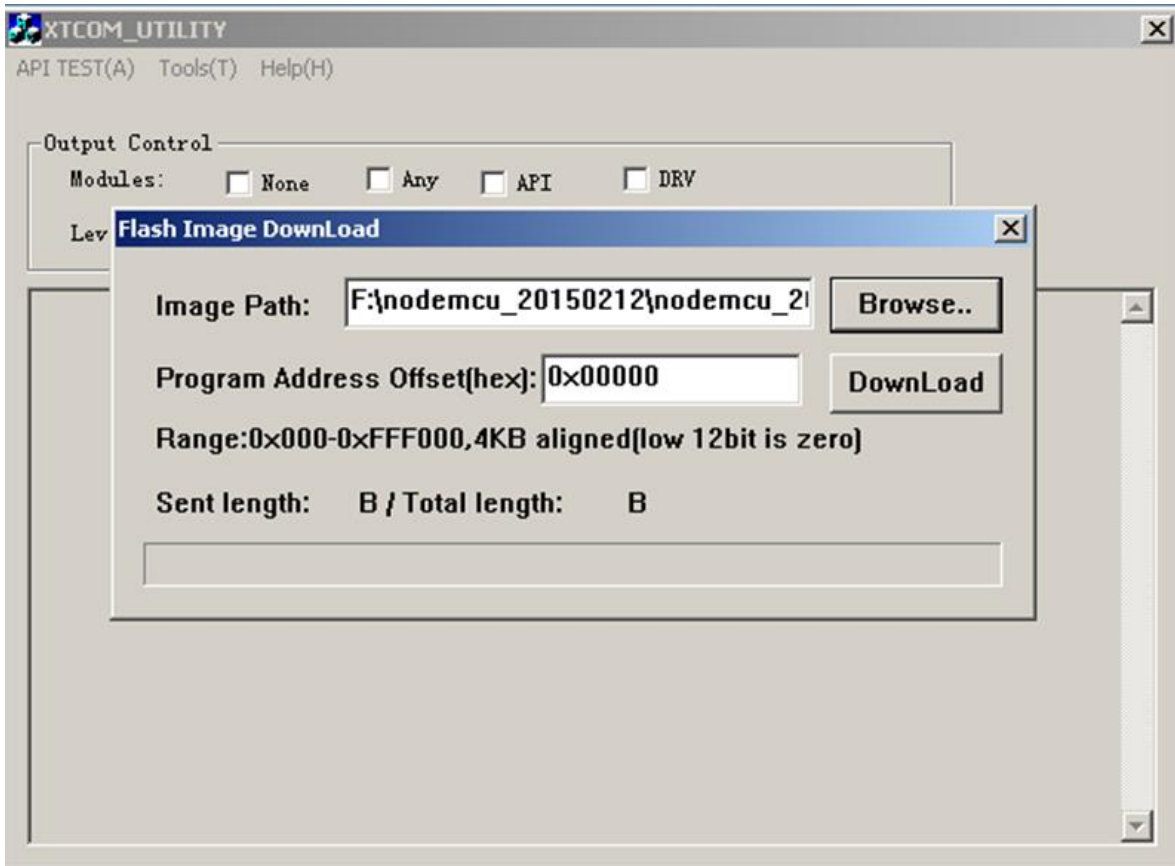


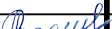
					13.02070849.00045 ПЛ2			
					Розробка програмного забезпечення мікроконтролері на основі ESP8266 з підтримкою бездротової мережі Wi-Fi	Літ.	Маса	Масштаб
Зм.	Арк.	№ документа	Підпис	Дата				
Розробив		Мусяченко О. С.						
Перевірив		Ілляшенко М.Б.						
Т.контр.						Аркуш 1	Аркушів 1	
Н.контр.		Польська О.В.			Схема підключення модуля до USB to UART під час прошивання	НУ «Запорізька політехніка», гр. КНТ-520		
Затвердив		Кудерметов Р.К.						

З'єднання з контролером

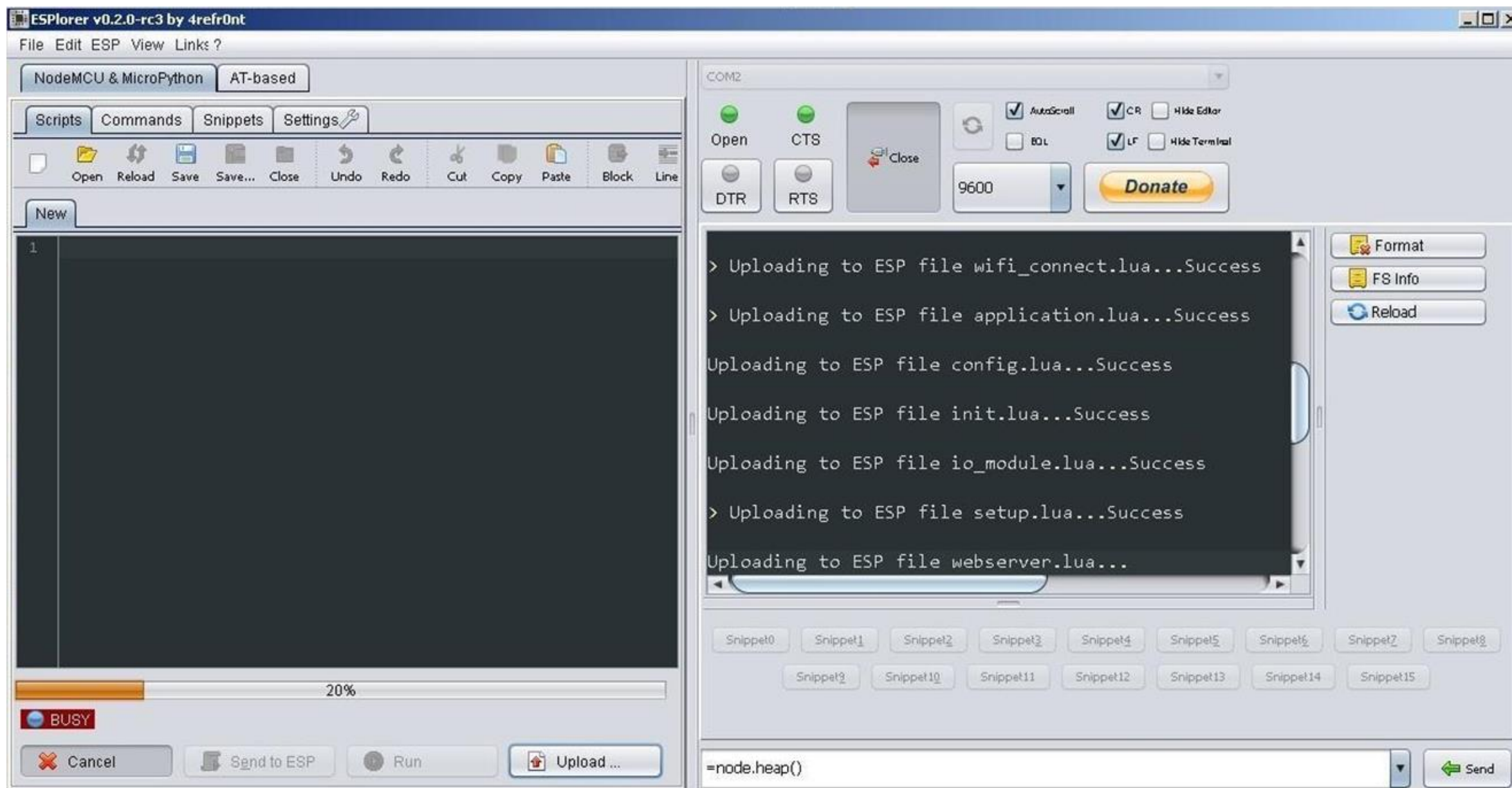


Початок прошивки



					13.02070849.00045 ПЛЗ							
					Розробка програмного забезпечення мікроконтролерів на основі ESP8266 з підтримкою бездротової мережі Wi-Fi	Лім.			Маса		Масштаб	
Зм.	Арк.	№ документа	Підпис	Дата	Розробка програмного забезпечення мікроконтролерів на основі ESP8266 з підтримкою бездротової мережі Wi-Fi	Лім.			Маса		Масштаб	
Розробив		Мусіченко О. Є.										
Перевірів		Ільяшенко М.Б.										
Т.контр.												
		—			Прошивка плати	Аркуш 1			Аркушів 1			
Н.контр.		Польська О.В.										
Затвердив		Кудерметов Р.А.										

Завантаження скриптів у пам'ять мікроконтролера за допомогою програми ESPlorer



					13.02070849.00045 ПЛ4						
					Розробка програмного забезпечення мікроконтролері на основі ESP8266 з підтримкою бездротової мережі Wi-Fi	Лім.		Маса		Масштаб	
Зм.	Арк.	№ документа	Підпис	Дата	Завантаження скриптів у пам'ять мікроконтролера за допомогою програми ESPlorer	Аркуш 1		Аркушів 1		НУ «Запорізька політехніка», гр. КНТ-520	
Розробив		Мусіченко О. Є.									
Перевірів		Льяшенко М.Б.									
Т.контр.											
Н.контр.		Польська О.В.									
Затвердив		Кудерметов Р.К.									