

Міністерство освіти і науки України
Львівський національний університет імені Іванка Франка
Факультет електроніки та комп'ютерних технологій
Кафедра радіофізики та комп'ютерних технологій

Курсова робота

Віддалене керування пристроєм IoT через Arduino Cloud

Виконав: студент
групи FeI – 21
спеціальності 122 – Комп'ютерні науки

_____ Крамар А.О.

Науковий керівник:
_____ доц. Кушнір О.О.

«____» _____ 2024 р.

Львів 2024

АНОТАЦІЯ

У даній курсовій роботі мною було досліджений весь процес побудови та приєднання простого пристрою IoT до хмарної платформи Arduino Cloud, поетапно розглядаючи усі можливі аспекти, починаючи з розгляду принципів вдалого використання обраного WiFi-модуля, і закінчуючи описом можливостей взаємодії пристроїв на платформі між собою, зокрема з інтеграцією логіки їх роботи на стороні окремого коду на сторонніх мовах програмування.

ПЕРЕКЛАД АНОТАЦІЇ

In this coursework, I explored the entire process of building and connecting a simple IoT device to the Arduino Cloud cloud platform, gradually considering all possible aspects, starting with considering the principles of successful use of the selected WiFi module, and ending with a description of the possibilities of interaction of devices on the platform with each other, in particular with the integration of their operating logic on the side of separate code in third-party programming languages.

ЗМІСТ

| | |
|---|-----------|
| АНОТАЦІЯ..... | 2 |
| ВСТУП..... | 5 |
| РОЗДІЛ 1. СУЧАСНІ ПІДХОДИ ДО ПРОЕКТУВАННЯ СИСТЕМ НА ОСНОВІ ІОТ..... | 6 |
| 1.1 Теоретичні відомості..... | 6 |
| 1.1.1 Знайомство з IoT..... | 6 |
| 1.1.2 Концепція технології та підходи до її проектування..... | 7 |
| 1.2 Постановка завдання..... | 10 |
| 1.2.1 Що планується отримати в результаті виконання проекту..... | 10 |
| 1.2.2 Вибір засобів та технологій для виконання проекту..... | 10 |
| 1.2.3 Аналіз аналогічних, чи близьких рішень..... | 11 |
| РОЗДІЛ 2. РЕАЛІЗАЦІЯ ПРОЕКТУ..... | 13 |
| 2.1 Реалізація апаратної частини проекту..... | 13 |
| 2.1.1 Розгляд модуля ESP-01 для вирішення задач проекту..... | 13 |
| 2.1.2 Побудова пристрою на макетній платі..... | 15 |
| 2.1.3 Підключення до ПК та перевірка роботи модуля ESP-01..... | 17 |
| 2.2 Підключення до Arduino Cloud..... | 19 |
| 2.2.1 Підготовка до роботи з платформою та додавання пристрою..... | 19 |
| 2.2.2 Підключення пристрою та завантаження його виконавчого коду..... | 21 |
| 2.3 Робота з пристроями з ручним управлінням..... | 24 |
| 2.3.1 Суть концепції..... | 24 |
| 2.3.2 Приклад окремого демонстраційного пристрою з ручним управлінням та опису його логіки на Python..... | 24 |
| РОЗДІЛ 3. ТЕСТУВАННЯ ТА ДЕМОНСТРАЦІЯ ФУНКЦІОНАЛЬНИХ МОЖЛИВОСТЕЙ ПРОЕКТУ..... | 26 |
| 3.1 Тестування проекту..... | 26 |
| 3.1.1 Методики тестування проекту..... | 26 |

| | |
|---|-----------|
| 3.1.2 Тестування роботи змінних та дистанційного присвоєння значень для них..... | 26 |
| 3.2 Демонстрація функціональних можливостей проекту..... | 30 |
| ВИСНОВКИ..... | 32 |
| Основні результати проекту..... | 32 |
| Шляхи покращення..... | 33 |
| СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ..... | 34 |
| ДОДАТОК А..... | 36 |

ВСТУП

Розглядаючи сучасні тенденції у розвитку концепцій взаємодії мережевих пристроїв, їх взаємної комунікації та практичного застосування, не складно помітити актуальності розгляду концепції інтернету речей (IoT) та активного розвитку як і підходів до проектування подібних систем і спектру технологій, що при ньому використовуються, так й еволюції протоколів зв'язку, що забезпечують зв'язок між пристроями. Відповідно до цього, метою даної роботи є всеосяжне знайомство з даною концепцією, зокрема, на прикладі самостійної побудови простого пристрою IoT та підключення його до платформи Arduino Cloud, зрозуміти технологію поетапного проектування схожих систем, акцентуючи увагу як і на апаратній частині проекту, так і на етапі завантаження програмного забезпечення на пристрій та його підключенні до мережі і дистанційної взаємодії. Робота з даною хмарною технологією є доречною ще і з тих причин, що вона надає широкий спектр інструментів для встановлення зв'язку та віддаленого керування мережею пристроїв через створення клієнтів з використанням мов програмування Python та JavaScript [7], що дозволяє встановлювати зв'язок з Arduino Cloud навіть у тих проектах, де можливість такої взаємодії може здатися неочевидною.

РОЗДІЛ 1. СУЧАСНІ ПІДХОДИ ДО ПРОЕКТУВАННЯ СИСТЕМ НА ОСНОВІ ІОТ

1.1 Теоретичні відомості

1.1.1 Знайомство з IoT

За узагальнюючим визначенням, запропонованим компанією Gartner, «IoT (Internet of Things) — це мережа фізичних об'єктів, які мають вбудовані технології, що дозволяють здійснювати взаємодію з зовнішнім середовищем, передавати відомості про свій стан і приймати дані ззовні» [1], яке вдало репрезентує суть концепції, адже засобами для взаємодії з мережею в даному випадку може бути будь-який, навіть буденний предмет, і те, які саме дані він буде передавати / одержувати, може обмежуватись лише поточними потребами та цілями проекту.

Так як IoT-пристрої не завжди підключені до мережі, в контексті даних систем вони репрезентуються, як *річ* (Thing) — абстрактне представлення даного об'єкту в мережі, що відображає його стан та дані, якими він обмінюється, що відображене як і у у Arduino Cloud [7], так і у інших хмарах, як-от у Amazon AWS [4, розд. 2.4.3] та інших.

Також доповнимо зазначену інформацію рядом термінів, що часто використовуються у даному контексті та описують ключові елементи типової моделі інтернету речей:

- *Сенсори (датчики, сенсорні пристрої)* — це пристрої, які призначені для збору інформації з навколишнього середовища або про стан об'єкта через вимірювання фізичних величин, як-от температура, вологість тощо. Сенсори є джерелом даних для IoT-систем [10].
- *Актuatorи (виконавчі пристрої)* — це пристрої, які виконують певні фізичні дії на основі отриманих команд із системи управління. Вони перетворюють сигнали

від сенсорів або контролерів на фізичні дії, наприклад, включення світла, рух механізмів, запуск двигунів тощо.

- *Контролери (пристрої управління)* — це пристрої, які забезпечують управління сенсорами та актуаторами, беручи на собі задачу здійснення первинної обробки зібраних даних до передачі даних на сервер або хмару для подальшого аналізу [11].
- *Концентратори (хаби)* — це пристрої, що групують кілька IoT-пристроїв та забезпечують їх зручне підключення до основної мережі, не рідко об'єднуючи їх у контексті певної екосистеми [12].
- *Хмарні платформи (сервери)* — це системи обробки, зберігання та аналізу великих обсягів даних, які збираються IoT-пристроями [3]. Хмарні платформи задають бізнес-модель системи, дозволяють працювати безлічі пристроїв, як одне ціле.

1.1.2 Концепція технології та підходи до її проектування

Аналізуючи створення відповідних систем IoT, стає очевидним, що, залежно від потреб конкретного проекту, підхід до її проектування може бути зовсім різний. З точки зору створення схожих систем, ключовою поточною проблемою у сфері технологій для сприяння розгортанню систем IoT є визначення еталонної архітектури, що нерідко зумовлено прагненням кожного постачальника подібних послуг пропонувати свою платформу для подібних застосувань [4, розд. 2]. Так чи інакше, коротко розглянемо окремі архітектури IoT.

Перш за все, охарактеризуємо хмарну («*Cloud Computing*») архітектуру, адже більшість платформ для реалізації IoT-систем у тій, чи іншій мірі її реалізують [4, розд. 2.4.3]. У цьому типі архітектури велика кількість даних, створених об'єктами, зберігається, обробляється та представляється користувачеві через служби, доступні з хмари. Також нам розуміти, що даний підхід до реалізації існує в контексті серверної («*Server-Based Architecture*») архітектури, чиєю характерною особливістю

можна назвати відсутністю у пристроїв прямого доступу до мережі, а важливу роль у даному випадку виконує шлюз [4, розд. 2.4.2]. Таким чином, можливими обмеженнями, що можуть бути пов'язані з дефіцитом обчислювальних ресурсів і їх неоднорідністю, необхідно керувати лише на найнижчому рівні, де реалізується зв'язок між пристроями та самим шлюзом.

Хмарний підхід, хоча і є в міру загальноприйнятим, але можливі проблеми, пов'язані із затримками передачі даних і безпекою в управлінні даними, призвели до появи дещо іншого підходу — архітектури на основі периферійних обчислень («*Edge Computing-Based Architectures*») [4, розд. 2.4.4]. Граничні обчислення, визначені як ІТ-архітектура, в якій обчислювальні додатки, сховища даних і послуги повністю або частково розміщені біля кінцевого користувача, можуть значно покращити обговорювану парадигму IoT. Побудова архітектури на основі Edge Computing може мати ряд переваг:

- Мінімізує затримку; багато дій виконуються дуже близько до місця дії;
- Дозволяє економити смугу пропускання;
- Вирішує деякі проблеми безпеки, оскільки багато рішень приймаються в підмережі та не піддаються ризикам, що виникають із зовнішньої мережі.

Описані вище варіанти реалізації IoT-систем у тій, чи іншій мірі реалізують рівневий («*Layered Architecture*») підхід [2, розд. 2], адже кожна з описаних архітектур передбачає розподіл системи на певні умовні рівні, кожен з яких виконує свою функцію. Хоча, залежно від обраного підходу, модель взаємодії між рівнями може різнитись, при узаальненій для всього IoT трирівневній архітектурі («*Three-Level Architecture*») [4, розд. 2.1] розділяють:

- рівень сприйняття, що представляє фізичний рівень об'єктів і взаємодіє з навколишнім середовищем шляхом збору та обробки інформації, який включає у себе об'єкти, які, будучи здатними взаємодіяти із зовнішнім світом і будучи оснащеними обчислювальними можливостями, стають у певному сенсі «інтелектуальними» або «розумними»;

- мережевий рівень, що має завдання транспортувати дані, які надає рівень сприйняття, на прикладний рівень, та включає в себе всі технології та протоколи, які роблять це з'єднання можливим;
- прикладний рівень, що включає все програмне забезпечення, необхідне для надання конкретної послуги. На прикладному рівні дані з попередніх рівнів зберігаються, агрегуються, фільтруються та обробляються, а також використовуються бази даних, програмне забезпечення для аналізу тощо, і саме на цьому етапі з'являється поділ на хмарні обчислення, де такі послуги, як зберігання або обробка даних, надаються з набору вже існуючих ресурсів, та граничні обчислення, де обробка даних частково розподіляється на периферійних вузлах мережі для підвищення продуктивності систем IoT.

На фоні розглянутих аспектів проектування наявне ще одне питання, що не було порушено, та проте, яке має вирішальне значення у IoT-проектах, які виконують критичні задачі - фактор безпеки. Безпековий підхід («*Security-First*») [2, розд. 5], при якому ключовими є використання криптографічних методів для захисту даних при передачі; підхід, за якого на кожному шарі IoT-системи використовуються окремі механізми захисту (на апаратному, програмному та мережевому рівнях). Якщо не надати достатньої уваги даному аспекту, наслідки від несанкціонованого зовнішнього втручання у роботу системи можуть бути невідворотними [13, розд. 4]. Питання захисту даних актуальне саме на мережову рівні, і в значній мірі залежить від використаного протоколу передачі даних та його безпекових можливостей [4, розд. 3].

Розуміючи суть вищеописаних підходів та комбінуючи їх в тих, чи інших IoT-проектах, ми зможемо побудувати мережу пристроїв згідно з поставленими цілями та наданими ресурсами.

1.2 Постановка завдання

1.2.1 Що планується отримати в результаті виконання проекту

Як вже згадувалось раніше, метою даної роботи є самостійна побудова простого IoT-пристрою та підключення його до платформи Arduino Cloud. Під пристроєм у цьому випадку розуміємо підключений до мережі та повністю готовий до роботи мікроконтролер, до якого в демонстраційному варіанті буде проведене підключення та дистанційне керування певного виконавця.

1.2.2 Вибір засобів та технологій для виконання проекту

Як платформа, на якій будуватиметься наш проект, обрано Arduino Cloud з причин, що вже були частково розкриті в розділі «ВСТУП» та пункті 1.1.1 розділу №1. Як згадувалось раніше, Arduino Cloud було обрано як платформу для проекту завдяки її зручному веб-інтерфейсу, що спрощує створення IoT- рішень. Платформа автоматично реалізує низку складних питань, таких як фактор безпеки, зв'язок за допомогою протоколу MQTT тощо. Користувачам залишається лише написання виконавчого коду для пристроїв (за прикладом Arduino IDE) і налаштування їх синхронізації за допомогою змінних, що синхронізуються з мережею [7]. Для дистанційного керування доступна панель Dashboard, а для інтеграції з іншими проектами Arduino Cloud підтримує бібліотеки для Python, JavaScript та Web-API [14].

В ролі виконавчого контролера було взято модуль ESP-01 на базі мікропроцесора ESP-8266. Окрім його підтримки у Arduino Cloud, даний вибір для поточного проекту зумовлений низкою його переваг: при низькій самовартості, цей модуль здатен одночасно виконувати роль окремого діючого контролера та здійснювати підключення до Wi-Fi, при цьому реалізуючи можливість здійснювати

з'єднання з мережею через безліч бездротових протоколів, зокрема HTTPS для використання модуля в ролі як клієнта або сервера для HTTP-запитів та згаданий раніше MQTT [5]. Детальні характеристики, особливості підключення та роботи даного контролера будуть описані при власне побудові пристрою в пункті 2.1.1.

1.2.3 Аналіз аналогічних, чи близьких рішень

Попри згадані переваги Arduino Cloud, зокрема для знайомства з технологією, для великих проектів часто застосовують інші платформи, що можуть забезпечити як докладніший контроль на програмному та апаратному рівнях за пристроями, так і кращу масштабованість та більший об'єм інтеграції сторонніх служб та інструментів. Відповідно до цього, розглянемо також інші схожі платформи в даному контексті:

1. *Azure IoT Hub* — це платформа для створення, керування та моніторингу IoT-пристроїв, яка входить до екосистеми Microsoft Azure. Підтримує двосторонній обмін даними, надійні засоби безпеки, вбудовані аналітичні інструменти, підтримку MQTT, AMQP і HTTPS протоколів [3, розд. 3].
2. *Amazon AWS IoT Core* — частина Amazon Web Services, що дозволяє підключати IoT-пристрої до хмари AWS для збору, обробки та аналізу даних. Virізняється високою безпекою завдяки AWS IoT Device Defender, надійністю, інтеграцією з іншими сервісами AWS (S3, Lambda), гнучкі інструменти для великих масштабів [3, розд. 1].
3. *Blynk* — платформа для розробників IoT, що надає можливість реалізовувати інтерфейси для створення мобільних інтерфейсів та їх роботи з широким спектром мікроконтролерів [3, розд. 11].
4. *Oracle IoT Intelligent Applications* — екосистема для реалізації IoT-рішень від Oracle, що є найкращим рішенням для розумного виробництва та прогнозованого технічного обслуговування, пропонує такий ряд послуг, як

підключення до Oracle IoT Cloud Service, моніторингу активів та аналізу й інтеграції даних [3, розд. 4].

Описані платформи відрізняються функціоналом і можливостями, тож вибір платформи може залежати від конкретних потреб проекту: масштабу, вимог до безпеки, аналітики та інтеграції з іншими сервісами.

Стосовно вибору мікроконтролера, спектр можливих альтернатив обмежується доступними для підключення пристроями у власне самій Arduino Cloud, де відзначимо можливість використання як і офіційних плат Arduino (Nano 33 IoT, MKR WiFi 1010 тощо), так й інші пристрої на базі використаного ESP8266, зокрема ESP32 та NodeMCU [14].

РОЗДІЛ 2. РЕАЛІЗАЦІЯ ПРОЕКТУ

2.1 Реалізація апаратної частини проекту

2.1.1 Розгляд модуля ESP-01 для вирішення задач проекту

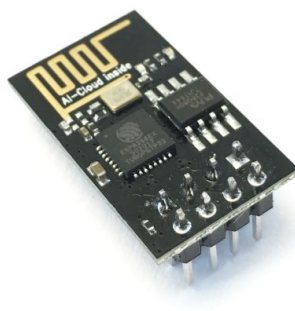


Рис. 2.1. Фактичне зображення модуля ESP-01

На даному етапі нам слід dokonано зрозуміти правильний порядок використання обраної плати для поточного проекту, і найкращим рішенням для цього буде поступовий розгляд кожного з контактів схеми.

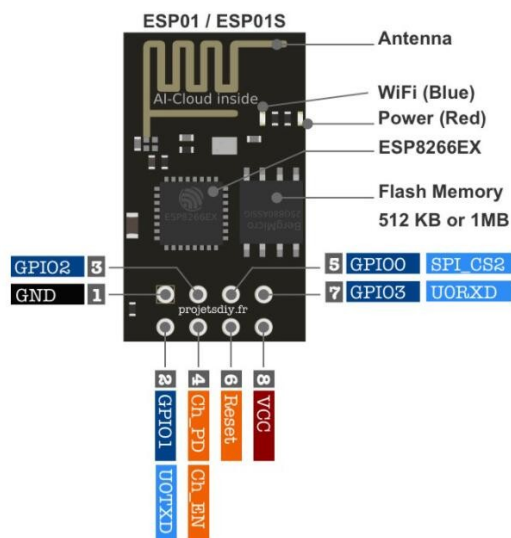


Рис. 2.2. Ілюстративне зображення виходів та зовнішніх елементів модуля ESP-01

1. *GND* — загальний
2. *TXD* (GPIO1) — передача даних
3. *GPIO2* — виведення загального призначення 2
4. *CH_PD* — вимкнення модуля (низький рівень активний, для включення модуля слід подати *Vcc*)
5. *GPIO0* — виведення загального призначення 0
6. *RST* — скидання модуля (низький рівень — активний)
7. *RXD* (GPIO3) — прийом даних
8. *Vcc* — живлення, +3,3 В (максимально 3,6 В)

Як бачимо, підключення зовнішнього живлення відбувається за виходами 1 *GND* та 8 *Vcc*, і неабияк важливим тут фактором є саме високий логічний рівень у 3.3 В, і відповідно такі ж напруги логічної «1» при роботі з цифровими виходами GPIO0-GPIO3. В умовах нашого проекту, де повсякчас використовуватиметься 5 В, слід завжди унеможлилювати можливість подачі вищої напруги на даний модуль, адже напруги >3,6 В можуть вивести його з ладу. Окремо також слід зважати на споживання пристрою: хоча у вимкненому режимі заявлений струм витoku не перевищує 10 мкА, за його увімкнення і активної роботи струм, що споживається, може в періоди пікової активності зростати до 200 мА, тому варто завжди для нього забезпечувати окреме живлення.

Виходи 6 *RST* (reset) та 4 *CH_PD* схожі за своєю суттю та дозволяють відповідно перемкнути контролер у стан повного перезавантаження (6) та у стан сну (4). Дані контакти є інверсними, тобто активним рівнем для них є логічний «0».

Особливо загостримо увагу на цифрових виходах GPIO0-GPIO3. Контакти GPIO1 та GPIO3 використовуються для обміну даними за протоколом UART, де ці контакти виступають як *TXD* (transmit, для передачі даних) і *RXD* (receive, для одержання даних) відповідно. За допомогою них даний модуль як і надає інтерфейс для роботи з ним у разі, якщо використовується вбудоване програмне забезпечення,

так і дозволяє завантажити свій програмний код, що й буде виконано в ході цієї роботи.

Неабияк варто згадати також про особливе призначення виходу GPIO0: у разі подання на нього низького логічного рівня у момент увімкнення контролера, до його наступного повторного увімкнення або перезавантаження він перейде у режим завантаження / оновлення програмного забезпечення.

Усі вищеописані цифрові контакти, окрім своїх особливих функцій, разом зі GPIO2 дозволяють як і видавати потрібний логічний рівень назовні, так і визначити вхідний рівень, що сумарно дає можливість взаємодії з цифровими сенсорами та актуаторами. Для кожного з даних виходів доступне використання вбудованої 10-бітної ШІМ, що також надає безліч можливостей.

2.1.2 Побудова пристрою на макетній платі

Згідно з описаними вище особливостями модуля ESP-01, можна дійти до висновку, що задля завантаження у нього програмного коду і, як результат, можливості підключення до Arduino Cloud, необхідно використати сторонні інструменти, адже оновлення програмного забезпечення у даному випадку можливе лише через контакти RXD та TXD. Для цього можливі два варіанти: використання адаптера UART-USB для безпосереднього підключення до ПК, або більш витончений спосіб, що ми й застосуємо — використати для цієї мети інший контролер, в поточному випадку — Arduino Nano. Такий варіант простий у реалізації, адже нам просто достатньо з'єднати обидва пристрої через виходи UART та під'єднати контакт «RESET» Arduino Nano до GND. В результаті таких дій, при спробі завантажити виконавчий код через USB, змін у Arduino Nano не відбудеться через його, фактично, вимкнений стан, а так як оновлення ПЗ у ньому відбувається також через UART, всі дані у незмінному вигляді «попадатимуть» на виходи RXD і TXD та, відповідно, на ESP-01. Описаний варіант підключення вдалий з тих причин, що, опісля необхідних дій з

ESP-8266, відключивши вихід «RESET» Arduino Nano від GND, ми матимемо одразу готову до використання об'єднану пару контролерів, що здатні незалежно обмінюватись даними і використовувати переваги одні одного, як-от великий спектр доступний аналогових та цифрових контактів Arduino Nano тощо.

Підсумовуючи описану інформацію, складемо принципову схему необхідного пристрою та її ілюстративний відповідник у додатку *Fritzing*.

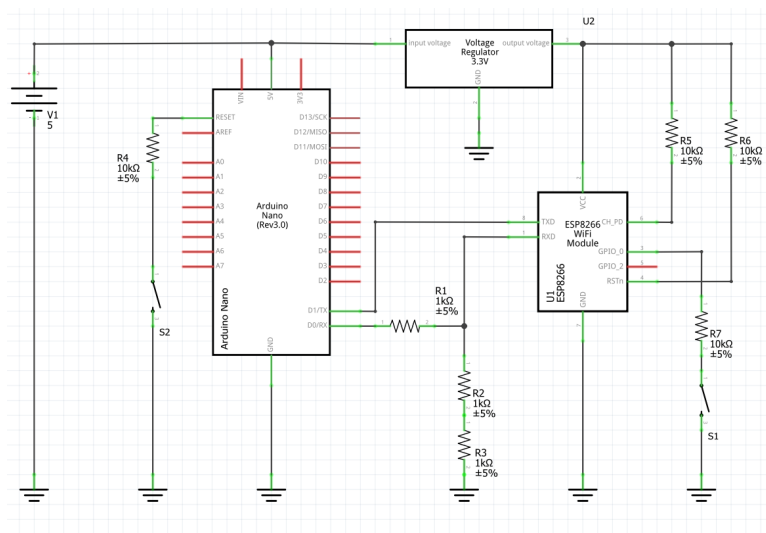


Рис. 2.3. Принципова схема пристрою

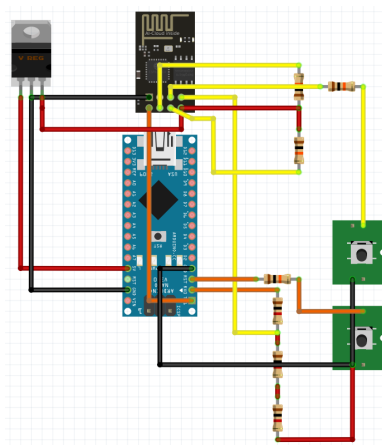


Рис. 2.4. Демонстраційна блок-схема пристрою

На зображеній принциповій схемі одразу помітно використання перетворювача напруги U2 для живлення контролера ESP-01 (зі згаданих причин, ми не можемо використати для цієї мети контакт «3v3» Arduino Nano з міркувань споживання струму).

При підключенні контролерів між собою за протоколом UART, здебільшого з'єднання контактів відбувається між TX (*transmit*, передає дані) та RX (*receive*, одержує дані), але так як в Arduino Nano виходи RXD та TXD вже розміщені в перехресному порядку, підключення здійснюємо напряду. При цьому, важливу роль грає подільник напруги, складений з опорів R1, R2 та R3, що перетворює сигнал, що передається від Arduino Nano, з 5 В до 3.3 В. Зі зворотнього боку, так як Arduino Nano успішно сприймає 3.3 В як високий логічний рівень, між контактами TXD додаткових елементів не потрібно.

Ключ S2 дозволяє перемикає Arduino Nano в стан «RESET», що, на практиці, буде відбуватись майже постійно задля безпосередньої комутації

ESP-8266 та ПК.

Замикання ключа S1 дозволяє, при увімкненні, перемикає ESP-01 у стан очікування та завантаження нового програмного коду.

Контакти RSTn та CH_PD через резистори R5 та R6 підтягуються до напруги живлення модуля (3.3 В) задля його коректної роботи.

Додатково наголосимо, що при роботі з будь-якими контактами ESP-8266, а особливо всіма цифровими виходами GPIO, максимальний струм, який вони можуть забезпечити, рівний 6 мА, і з цих міркувань в зображеній схемі використані опори R5, R6, та R7 задля використання мінімально можливого струму. Це й ж фактор слід пам'ятати при підключенні зовнішнього навантаження до даних виходів.

2.1.3 Підключення до ПК та перевірка роботи модуля ESP-01

Окрім як використання ESP-8266 у ролі окремого виконавчого контролера, наявний також варіант його використання лише як WiFi-модуль, взаємодіючи з його вбудованим ПЗ за допомогою спеціальних AT-команд. Хоча даний спосіб неможливо використати для підключення саме до обраної нами платформи, але з допомогою окремих відповідних команд може з легкістю перевірити факт коректної роботи модуля у побудованій нами схемі.

Для перевірки роботи пристрою, з урахуванням побудови схеми за рис. 2.3, замкнемо ключ S2 для переведення Arduino Nano в режим апаратного скидання, подамо живлення на схему (V1) та підключимо USB Type-C вихід Arduino Nano до ПК.

Для взаємодії з послідовним портом використаємо програму Arduino IDE. Оберемо відповідний COM-порт, до якого під'єднаний пристрій:

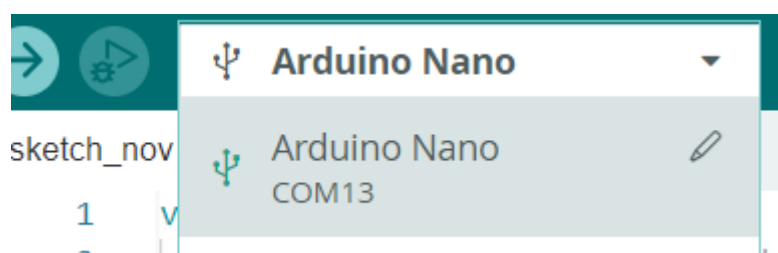


Рис. 2.5. Вибір порта

Відкривши монітор порту, оберемо необхідну для роботи з даним ПЗ швидкість передачі даних та їх розмежувувач.

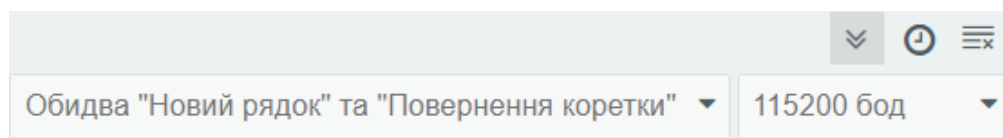


Рис 2.6. Налаштування швидкості порта та розмежувувач даних

Надалі ми можемо взаємодіяти з пристроєм, використовуючи AT-команди. Найпростіша з них — «AT», дозволяє перевірити саму можливість взаємодіяти з контролером.

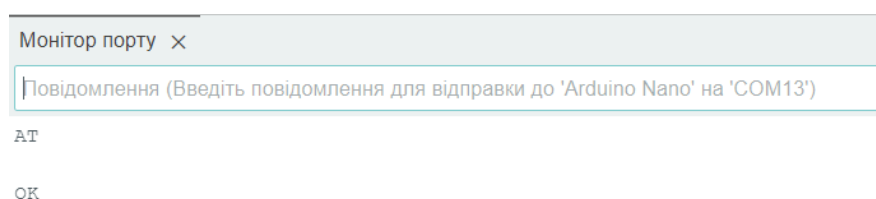


Рис 2.7. Результат виконання команди «AT»

В разі необхідності, ми можемо використати й інші команди, проте поточний результат вже демонструє коректну роботу модуля та можливість реалізовувати вже подальшу взаємодію з Arduino Cloud.

2.2 Підключення до Arduino Cloud

2.2.1 Підготовка до роботи з платформою та додавання пристрою

Для початку роботи, створимо обліковий запис на офіційному ресурсі «<https://app.arduino.cc/>», використавши корпоративну скриньку.

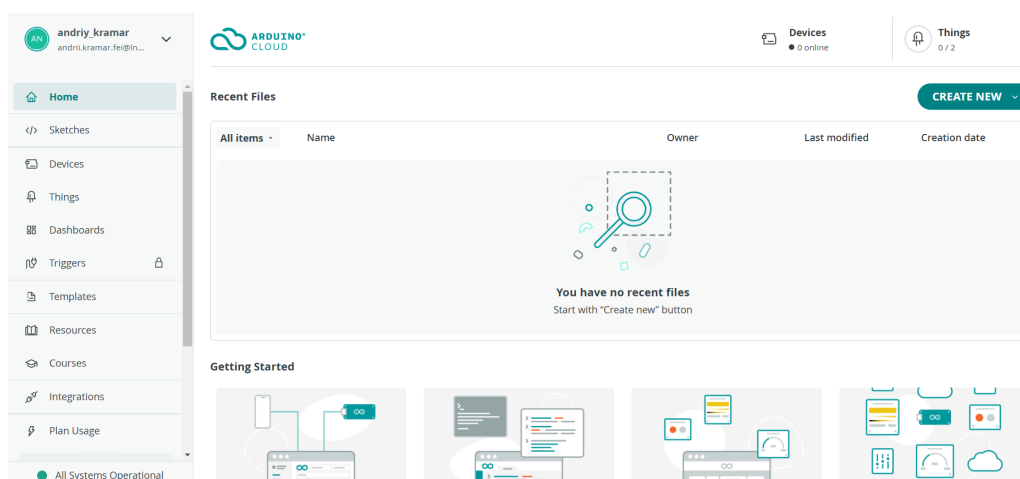


Рис 2.8. Загальний вигляд головної сторінки платформи

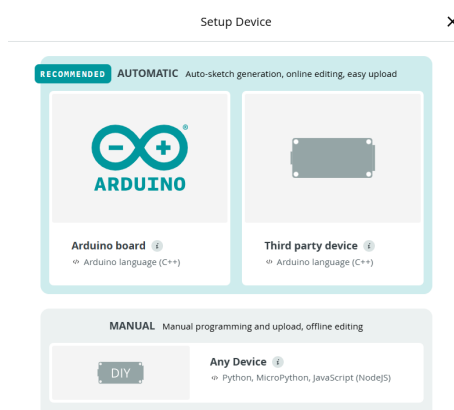


Рис. 2.9. Різновиди можливих пристроїв для роботи у Arduino Cloud

Для додавання пристрою, перейдемо на сторінку «*Devices*» (Рис. 2.8). Важливо відмітити вибір власне контролера при додаванні: можливі варіанти умовно розділені на офіційні плати Arduino («*Arduino Board*») з вбудованою можливістю підключення саме до цієї платформи, сторонні девайси («*Third party device*»), до яких відноситься і обраний нами модуль ESP-01, та ручні пристрої («*Any Device*»), що будуть описані у підрозділі 2.3.

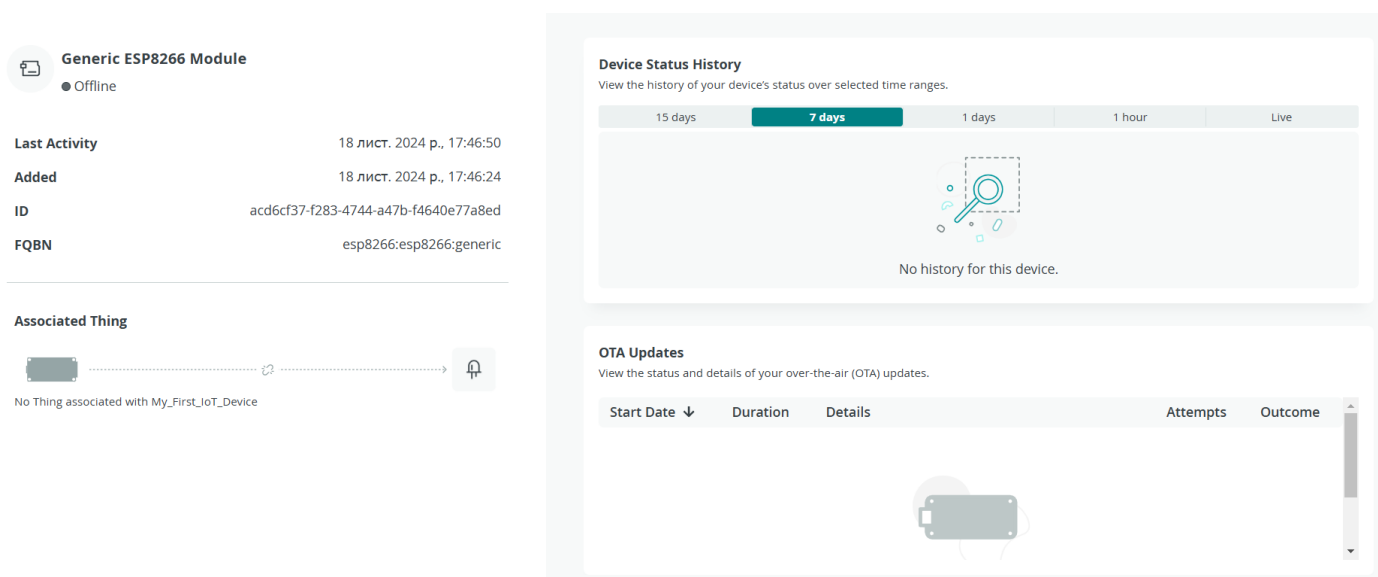


Рис. 2.10. Панель управління пристроєм у Arduino Cloud

Додавши пристрій «*Generic ESP8266 Module*» у розділі «*Third party device*», одержимо панель управління, що зображена на Рис. 2.10, де можна знайти історію активності пристрою, нещодавні бездротові оновлення (доступне лише для окремих контролерів, зокрема з розділу «*Arduino Board*», Рис. 2.9) та, що найголовніше, можливість створення та синхронізації з пристроєм IoT «речі», що також можна зробити на вкладці «*Things*» головної сторінки платформи (Рис. 2.8).

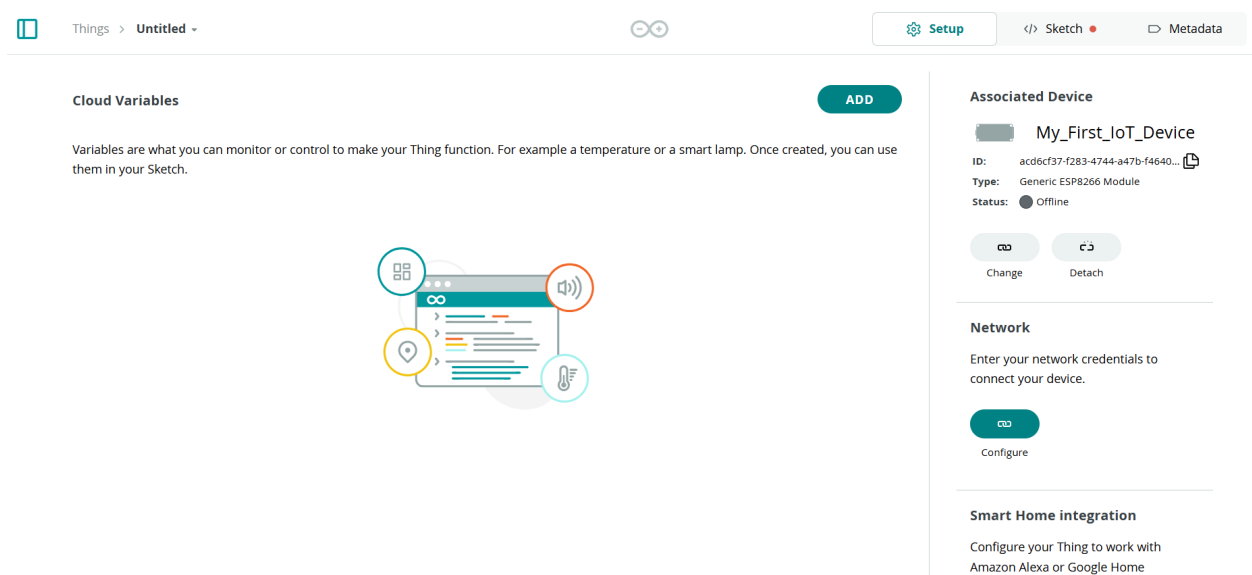


Рис. 2.11. Панель налаштування речі у Arduino Cloud

2.2.2 Підключення пристрою та завантаження його виконавчого коду

Перш за все, задля можливості підключити пристрій, нам слід вказати дані WiFi-мережі (SSID та пароль), що застосовуватиметься пристроєм для підключення до хмари. Робити це слід у вкладці «*Setup*» в пункті «*Network*». Окрім цього, при додаванні необхідно вказати секретний ключ відповідного пристрою, що автоматично генерується та надається нам при його створенні на платформі (пункт 2.2.1).

Перед подальшими діями, варто переконатись, що на відповідному ПК встановлений Arduino Cloud Agent, який власне й розпізнає підключені до послідовних портів пристрої та забезпечує їх взаємодію зі платформою. Задля встановлення даної утиліти слід можна перейти за посиланням «<https://cloud.arduino.cc/download-agent/>».

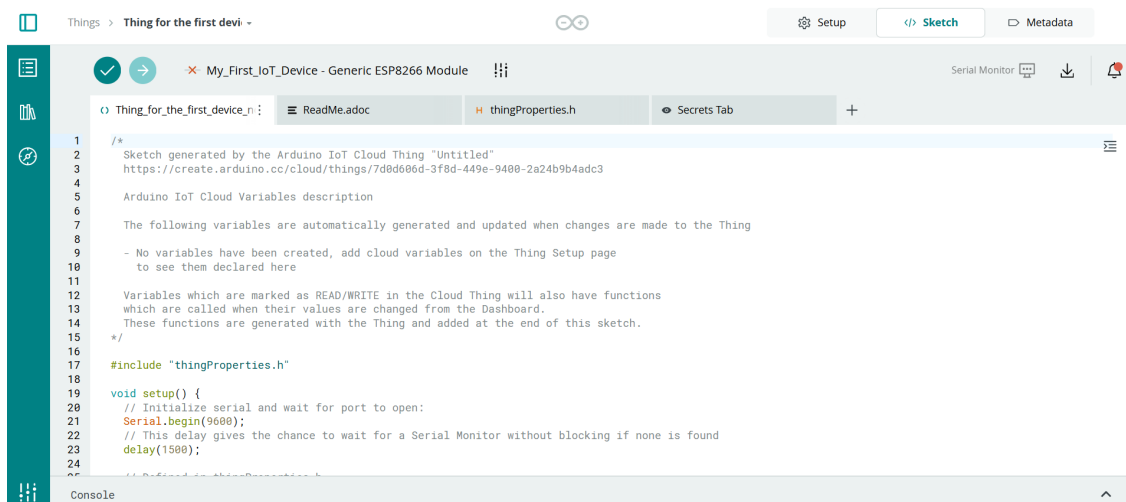
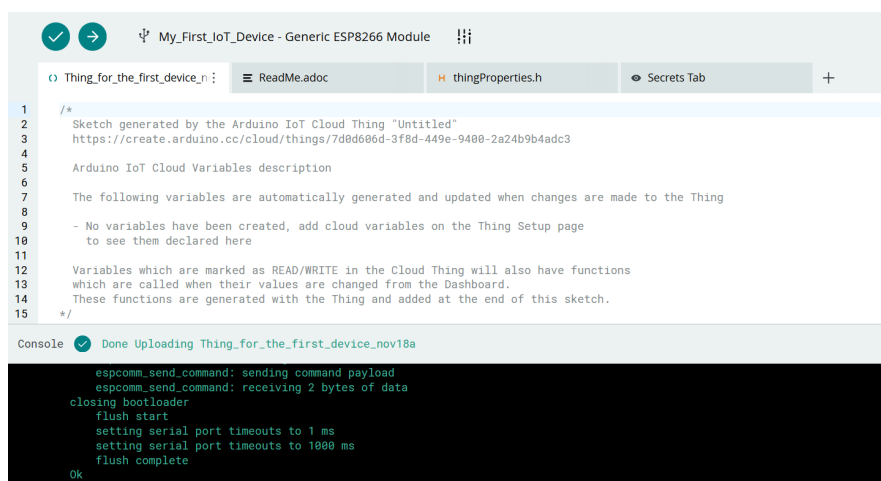


Рис. 2.12. Сторінка з виконавчим кодом відповідної речі в Arduino Cloud

Відкривши вкладку «Sketch», попри інші допоміжні файли, побачимо автоматично згенерований платформною скетч, що завантажуватиметься на пристрій, та файл «*thingProperties.h*», що імпортується у скетч. Дане розділення зумовлене тим, що ключові для підключення дії виконується саме у імпортованому файлі, де й генерується основна частина нового коду при різноманітних діях з пристроям у хмарі, як-от додавання нових змінних, що синхронізуюватимуться з мережею. Відповідно до цього, будь-який код, що ми самостійно прописуватимемо, слід розміщувати у скетчі, при цьому ігноруючи «*thingProperties.h*», використовуючи його хіба що для одержання назв згенерованих змінних та оголошених функцій для обробки подій.

Згенерований скетч уже самодостатній для простого підключення до мережі, тому можемо тестово його провести. Для цього, відповідно до схеми на рис. 2.3, замикаємо ключ S1 та перезапускаємо модуль ESP-01 (для цього можна або на короткий інтервал часу підключити вихід RSTn замкнути на GND, або ж переподати живлення на схему). Зауважимо, що бажано уникати подачу напруги лише через USB й забезпечувати окреме джерело задля уникнення проходження великого струму через Arduino Nano.

Провівши всі попередні дії, маючи вдало запущений Arduino Agent та підключену схему з переведеним ESP-01 в режим програмування, можемо запускати компіляцію коду та його завантаження на сторінці «*Sketch*». Відмітимо, що як і компіляція, так і завантаження даних на модуль потребують певного часу. Ідентифікувати вдалий процес передачі програмного коду, окрім як на консолі в самій платформі, можна за світлодіодом на самій платі, що об'єднаний з виходом TXD (GPIO1).



```

1  /*
2  Sketch generated by the Arduino IoT Cloud Thing "Untitled"
3  https://create.arduino.cc/cloud/things/7d0d606d-3f8d-449e-9400-2a24b9b4adc3
4
5  Arduino IoT Cloud Variables description
6
7  The following variables are automatically generated and updated when changes are made to the Thing
8
9  - No variables have been created, add cloud variables on the Thing Setup page
10 to see them declared here
11
12 Variables which are marked as READ/WRITE in the Cloud Thing will also have functions
13 which are called when their values are changed from the Dashboard.
14 These functions are generated with the Thing and added at the end of this sketch.
15 */

```

```

Console Done Uploading Thing_for_the_first_device_nov18a
espcomm_send_command: sending command payload
espcomm_send_command: receiving 2 bytes of data
closing bootloader
flush start
setting serial port timeouts to 1 ms
setting serial port timeouts to 1000 ms
flush complete
Ok

```

Рис. 2.13. Приклад успішного результату завантаження в консолі

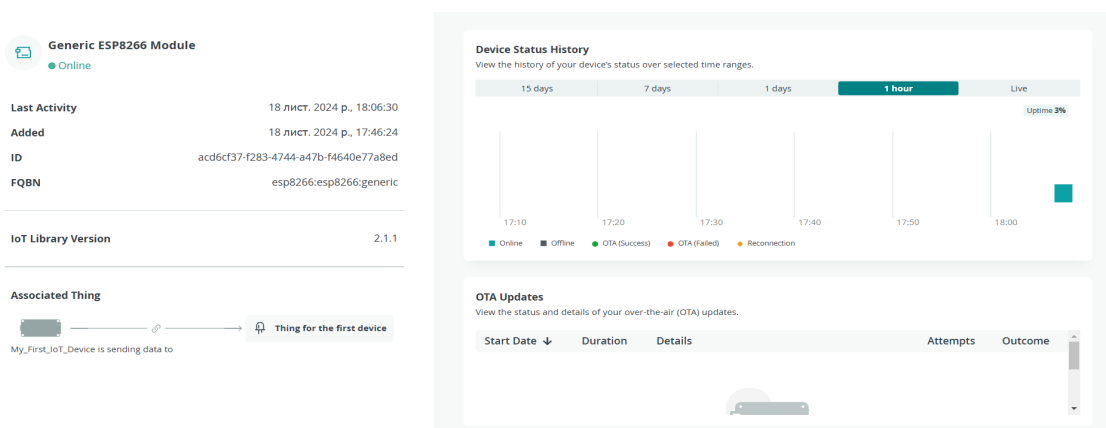


Рис. 2.14. Відображення активності пристрою після завантаження ПЗ

2.3 Робота з пристроями з ручним управлінням

2.3.1 Суть концепції

Розглядаючи різновиди можливих для додавання у Arduino Cloud пристроїв у пункті 2.2.1, питання ручних пристроїв («*Any Device*») було окремо винесено для більш детального обговорення.

Обираючи даний варіант при створенні пристрою, він, фактично, не прив'язується до реального контролера, як такого. Даний варіант не передбачає використання вбудованих в саму платформу інструментів для завантаження програмного коду, при ньому весь контроль над «пристроєм» виконується зі сторони окремих програм на Python, MicroPython чи NodeJS, а контекст їхнього розміщення, чи умов застосування, стає неважливим. Разом з можливістю синхронізувати змінні між різними IoT-Thing між собою, концепція створення винятково програмного пристрою дозволяє не просто забезпечувати зручну взаємодію між контролерами, а й інтегрувати їх у більш складні та багатоваріантні системи.

2.3.2 Приклад окремого демонстраційного пристрою з ручним управлінням та опису його логіки на Python

Задля розуміння описаних принципів, розглянемо приклад простого пристрій даного типу з описом логіки на мові програмування Python.

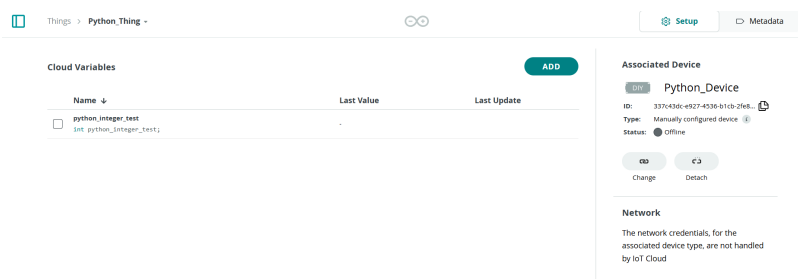


Рис. 2.15. Вікно керування речі для ручного пристрою

Для цього, повторюючи всі попередні кроки, створимо новий пристрій, обравши тип «*Any Device*» при створенні, і додамо тестову змінну для взаємодії.

Після цього, встановивши попередньо бібліотеку «*arduino-iot-cloud*», створюємо об'єкт класу «*ArduinoCloudClient*», вказавши у конструкторі дані пристрою, що були одержані при його створенні, та режим роботи програми.

```

1  from secrets import DEVICE_ID
2  from secrets import SECRET_KEY
3  from arduino_iot_cloud import ArduinoCloudClient
4  import time
5
6  client = ArduinoCloudClient(device_id=DEVICE_ID, username=DEVICE_ID, password=SECRET_KEY, sync_mode=True)
7  client.register(aiotobj: "python_integer_test", value=1)
8  client.start()
9  time_update = time.time()
10
11 while True:
12     client.update()
13     time.sleep(0.01)
14     if time.time() - time_update >= 1:
15         time_update = time.time()
16         client["python_integer_test"] += 1
17

```

Рис. 2.16. Код, що реалізує поведінку пристрою з ручним управлінням

Зображений на рис. 2.14 цикл, працюючи у синхронному режимі, у найпростішому в реалізації способі щосекунди інкрементує хмарну змінну «*python_integer_test*». В разі необхідності, у рядку №7 можна також вказати функцію зворотнього виклику для обробки події запису нового значення у дану змінну.

РОЗДІЛ 3. ТЕСТУВАННЯ ТА ДЕМОНСТРАЦІЯ ФУНКЦІОНАЛЬНИХ МОЖЛИВОСТЕЙ ПРОЕКТУ

3.1 Тестування проекту

3.1.1 Методики тестування проекту

Виконавши всі попередні кроки, тобто здійснивши побудову та підключення пристрою до Arduino Cloud, ми можемо будувати на його основі певні IoT рішення. Головний спосіб комунікації модуля та хмари — обмін даними через синхронізовані змінні («*Cloud Variables*»), тому побудова моделі взаємодії буде зводитись до певних дій на стороні пристрою (робота з іншими модулями, зокрема сенсорами та виконавцями тощо), які, як результат, здійснюватимуть запис у дані змінні, або ж навпаки здійснювати певну роботу внаслідок оновлення їх значення, що ми й виконаємо у декількох варіантах в контексті тестування та демонстрації роботи результатів даного проекту.

Зауважимо, що окрім як з використанням пристроїв, змінювати значення описаних змінних можливо й з допомогою вбудованої панелі управління — «*Dashboard*», де, створивши певну панель, можна додати візуальних компонентів для керування, чиє призначення так, чи інакше, буде зводитись до зміни значень створених нами змінних.

3.1.2 Тестування роботи змінних та дистанційного присвоєння значень для них

Задля тестування вдалої передачі даних через хмарні змінні, створімо просту змінну булевого типу, значення якої на самому контролері буде напяму передаватись на один з цифрових виходів GPIO. Також зауважимо, що при додаванні нових змінних наявна можливість задання умови її синхронізації: при зміні значення більше, ніж на певний поріг («*Threshold*»), чи з певною сталою періодичністю.

Add variable

Boolean eg. true

Declaration

```
bool boolean_test;
```

Variable Permission

☒ Read & Write ☐ Read Only

Variable Update Policy

☒ On change ☐ Periodically

CANCEL ADD VARIABLE

Рис. 3.1. Фрагмент вікна для додавання змінних у Arduino Cloud

Створимо нову Dashboard та додамо на неї компоненту для зміни значення нової змінної за прикладом на рис. 3.2.

Switch

Widget Settings

Name

Switch

Hide widget frame

Linked Variable

boolean_test

from Thing for the first device

Change Detach

Show Thing name on widget

Switch Labels

DONE

Рис. 3.2. Вікно для додавання компонент на Dashboard у Arduino Cloud

Додавши нову тестову змінну булевого типу, можемо перейти до опису взаємодії з нею в самому скетчі відповідної речі.



Рис. 3.3. Фрагмент файлу «*thingProperties.h*»

Як згадувалось раніше, при змінах, як-от додавання нової хмарної змінної, її оголошення та приєднання до мережі автоматично додається у файл «*thingProperties.h*». Для нас тут корисними тут є назви власне самої змінної та функції «*void onBooleanTestChange()*», що виступає обробником події оновлення значення даної змінної.

Додамо у опис даної функції подання значення функції на 2 цифровий вихід, попередньо налаштувавши GPIO2, як «*OUTPUT*» у функції «*void setup()*» командою «*pinMode(2, OUTPUT)*».

```

void onBooleanTestChange() {
    digitalWrite(2, boolean_test);
}

```

Рис. 3.4. Опис функції «*void onBooleanTestChange()*»

Підключімо до GPIO2 певне навантаження для розуміння коректної роботи пристрою. Найбільш вдале рішення для цього — світлодіод, а так як ми обмежені напругою живлення у 3.3 В, оберемо варіант з порівняно невеликим спадом напруги, для прикладу — червоний колір випромінювання з приблизним спадом напруги у 2.0 В. Обчислимо номінал резистора для обмеження струму для даного світлодіода з

урахуванням максимального струму, що можуть безпроблемно забезпечити виходи GPIO:

$$R_d = \frac{U_{cc} - U_d}{I_{max}} = \frac{3,3V - 2,0V}{6mA} \approx 216,66 \Omega$$

Оберемо найближчий за значенням можливий для нас номінал у 220Ω.

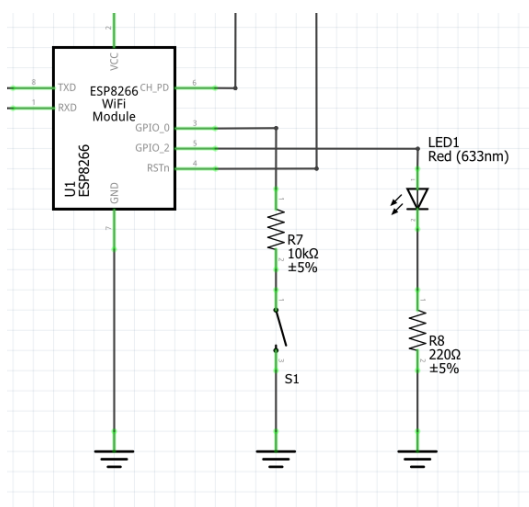


Рис. 3.5. Фрагмент принципової схеми з рис. 2.3. з доданим тестовим навантаженням

За розглянутим алгоритмом у пункті 2.2.2, завантажимо новий варіант скетча на пристрій. Попри те, що він автоматично після завантаження коду переходить у звичайний режим роботи, не забуваймо розімкнути ключ S1 для уникнення переходу у режим завантаження нового ПЗ після перезавантаження.

В результаті описаних дій, одержимо можливість дистанційного керування навантаженням з допомогою компоненти, додавання якої було зображено на рис. 3.2.

3.2 Демонстрація функціональних можливостей проекту

Задля демонстрації можливостей побудованого проекту, скористаймося описаними раніше перевагами складеної схеми, а саме можливістю легкої інтеграції у логіку проекту модуля Arduino Nano, що згадувалось у пункті 2.1.2. Тут слід зауважити, що для завантаження скетчу на ESP-01 використовувався, перш за все, апаратний UART, і хоча він також може використовуватись для обміну даних між даним модулем та Arduino Nano, реалізуючи демонстраційний варіант даного проекту, використаємо програмний спосіб використання даного протоколу задля уникнення ряду проблем, пов'язаних з доступом до апаратного послідовного порту.

Реалізуємо наступну логіку: при зміні значення певної змінної, WiFi-модуль надсилатиме по програмному UART значення цієї змінної на цифрові виходи Arduino Nano, який у свою чергу оброблятиме його для більш складного навантаження, до прикладу, 7-сегментного індикатора під керуванням 8-бітного зсувного регістра.

Для цього, додамо до IoT-речі нову цілочисельну змінну, та в обробнику її зміни використаємо команду для надсилання її значення по Serial (зауважимо, що виклик функції для підготовки до роботи з портом «*Serial.begin(9600)*» вже був прописаний у функції `setup()` автоматично).

```
void onIntegerTestChange() {  
    // Add your code here to act upon IntegerTest change  
    Serial.print(integer_test);  
}
```

Рис. 3.6. Тіло функції обробки зміни значення цілочисельної змінної

Завантаживши програмний код на ESP-01 та переконавшись, що даний модуль не в режимі програмування, роз'єднуємо ключ S2 задля подальшої роботи вже зі самою Arduino Nano.

Приклад скетчу для даного контролера, який одержуватимиме ціле число по програмному UART за заданими цифровими виходами та виводитимиме його на модуль зі 7-сегментним індикатором, описаний у додатку А. Його завантаження на пристрій можна виконати з допомогою вже використаної раніше Arduino IDE. Наголосимо на необхідності перемкнути виходи TX та RX на вказані цифрові виходи, адже передача даних відбувається саме за ними з допомогою бібліотеки «*SoftwareSerial.h*».

Хорошим рішенням у даній демонстраційній версії проекту також є випробування синхронізації хмарних змінних між створеними раніше пристроями, щоб дозволить здійснювати керування відображуваним числом не тільки з допомогою Dashboard, а і опису пристрою з ручним керуванням на Python, у випадку з поточною версією коду — значення даних змінних буде щосекунди інкрементуватись.

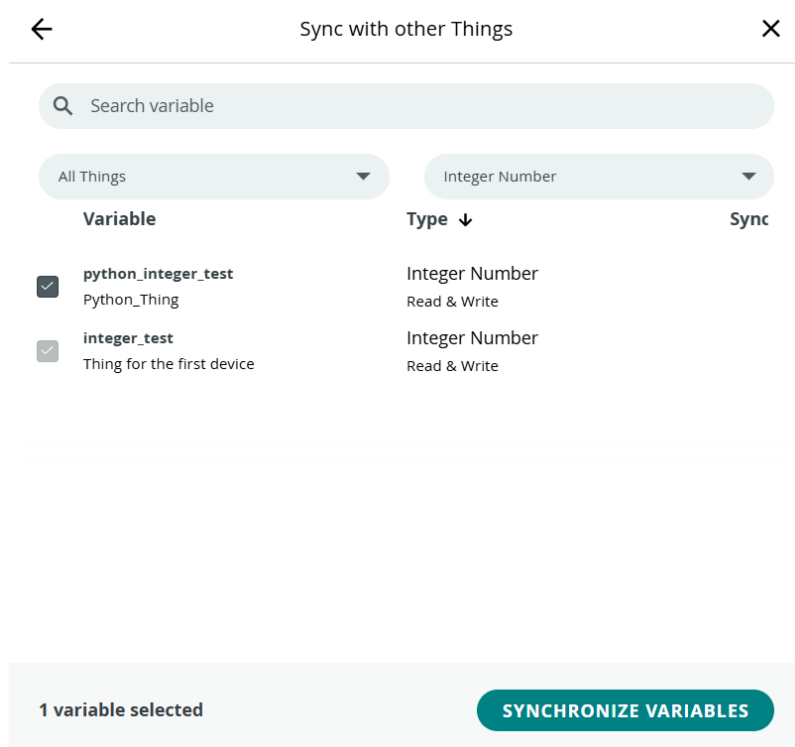


Рис. 3.7. Вікно для синхронізації хмарних змінних у Arduino Cloud

ВИСНОВКИ

Основні результати проекту

В результаті виконання даного проекту було складено простий IoT-пристрій на базі мікроконтролера ESP-8266 з використанням ряду простих радіокомпонентів та контролера Arduino Nano для роботи з ПК задля вивчення можливостей платформи Arduino Cloud. В результаті виконання ряду тестів, спостерігались наступні результати:

- успішне завантаження виконавчого коду на мікроконтролер з його подальшим перемиканням у звичайний режим роботи та автоматичним підключенням до хмари;
- вдала обробка зміни булевої хмарної змінної на стороні пристрою, а саме подання відповідного логічного рівня на цифровий вихід і, як результат, на підключене навантаження (світлодіод);
- вдале створення та підключення клієнта на Python, модифікація на його стороні певних хмарних змінних та відображення результату цих дій у хмарі;
- відсутність вдалих результатів при спробах передати дані між ESP-01 та Arduino Nano по апаратному UART, та, навпаки, вдала передача даних по цифровому UART.

З оглядом на відповідні результати, підтверджено наступні властивості побудованого IoT-рішення:

- змога пристрою здійснювати автоматичне підключення до хмарної платформи з можливістю моніторингу відповідних процесів на самому веб-ресурсі;
- можливість пристрою вдало обмінюватись даними з платформою, обробляти їх, передавати іншим пристроям або ж здійснювати певну роботу на їх основі;
- можливість використання вбудованих у платформу інструментів для безпосередньої взаємодії з пристроями;

- можливість взаємодії між пристроями на платформі, зокрема використання сторонніх клієнтів для взаємодії з платформою;
- факт використання обраної хмари зумовлює безпеку при передачі та зберіганні даних;
- можливість масштабування складеного IoT-рішення у контексті обраної платформи.

Окремо згадаємо також відсутність у складеного пристрою будь-яких використаних чи-то апаратних, чи-то програмних рішень, що забезпечували б його енергоефективність та автономність.

Шляхи покращення

Резюмуючи можливість покращення, вдосконалення даного проекту зводиться або до його масштабування (збільшення кількості підключених пристроїв, ускладнення їх внутрішньої логіки та способів використання хмарних змінних, зокрема шляхів запису у них даних), або ж до вибору інших платформ для роботи та / або інших пристроїв власне для підключення. Наголосимо, що технічно даний проект є перш за все вивченням, та, як результат, демонстрацією можливостей платформи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Internet of Things, IoT [Електронний ресурс] // IT-Enterprise – Режим доступу до ресурсу: <https://www.it.ua/knowledge-base/technology-innovation/internet-veschej-internet-of-things-iot>.
2. IoT Platform Design Methodology: Top 5 Approaches [Електронний ресурс] // Minnovation Technologies. – 2024. – Режим доступу до ресурсу: <https://minnovation.com.au/iot-platform/iot-platform-design-methodology-top-5-approaches/>.
3. Muts I. 10+ Best IoT Cloud Platforms in 2024 [Електронний ресурс] / Ivan Muts // euristiq. – 2024. – Режим доступу до ресурсу: <https://euristiq.com/best-iot-cloud-platforms/>.
4. Lombardi M. Internet of Things: A General Overview between Architectures, Protocols and Applications [Електронний ресурс] / M. Lombardi, F. Pascale, D. Santaniello // MDPI. – 2021. – Режим доступу до ресурсу: <https://www.mdpi.com/2078-2489/12/2/87>.
5. ESP-01 Wi-Fi Module: ESP-01 Pinout, Programming and ESP-01 VS ESP8266 [FAQ] [Електронний ресурс] // Utmel Electronics. – 2021. – Режим доступу до ресурсу: <https://www.utmel.com/components/esp-01-wi-fi-module-esp-01-pinout-programming-and-esp-01-vs-esp8266-faq?id=990>.
6. ESP8266EX Datasheet [Електронний ресурс] // Espressif Systems IOT Team. – 2015. – Режим доступу до ресурсу: https://cdn-shop.adafruit.com/product-files/2471/0A-ESP8266_Datasheet_EN_v4.3.pdf.
7. Söderby K. Getting Started with Arduino Cloud [Електронний ресурс] / Karl Söderby // Arduino Documentation. – 2024. – Режим доступу до ресурсу: <https://docs.arduino.cc/arduino-cloud/guides/overview/>.
8. Python [Електронний ресурс] // Arduino Documentation. – 2024. – Режим доступу до ресурсу: <https://docs.arduino.cc/arduino-cloud/guides/python/>.
9. Arduino IoT Cloud Python client [Електронний ресурс] // The Python Package Index (PyPI). – 2024. – Режим доступу до ресурсу: <https://pypi.org/project/arduino-iot-cloud/>.
10. Everything You Wanted to Know About IoT Sensors [Електронний ресурс] // iottechtrends. – 2019. – Режим доступу до ресурсу: <https://www.iottechtrends.com/everything-about-iot-sensors/>.

11. What is an IoT Controller – List of Top IoT Controller Devices [Электронный ресурс] // The IoT Academy. – 2024. – Режим доступа до ресурсу: <https://www.theiotacademy.co/blog/iot-controller/>.
12. Smart home hub [Электронный ресурс] // Wikipedia. – 2024. – Режим доступа до ресурсу: https://en.wikipedia.org/wiki/Smart_home_hub.
13. Интернет речей [Электронный ресурс] // Wikipedia. – 2024. – Режим доступа до ресурсу: <https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82%D1%80%D0%B5%D1%87%D0%B5%D0%B9>.
14. Söderby K. Device Types [Электронный ресурс] / Karl Söderby // Arduino Documentation. – 2024. – Режим доступа до ресурсу: <https://docs.arduino.cc/arduino-cloud/hardware/devices/>.

ДОДАТОК А

```
#include <GyverSegment.h> // бібліотека для зручної роботи з 7-сегментними індикаторами
#include <SoftwareSerial.h> // бібліотека для роботи з програмним UART

#define sclk 5 // задання виходів для 7-сегментного індикатора під керуванням зсувного регістра 74HC595
#define rclk 6
#define dio 7
#define softwareRX 2 // задаються виходи для програмного UART
#define softwareTX 3

SoftwareSerial espSerial(softwareRX, softwareTX); // об'єкт для роботи з програмним UART
Disp595_4 disp(dio, sclk, rclk); // об'єкт для роботи з індикатором
uint8_t value;

/*
  Функція для ручного одержання цілих чисел з UART.
  Факт її використання зумовлений тривалим часом
  роботи методу parseInt().
*/
int readIntegerFromUART(SoftwareSerial &serial) {
  int number = 0;
  bool isNegative = false;
  while (serial.available()) {
    char c = serial.read();

    if (c == '-') {
      isNegative = true;
    } else if (c >= '0' && c <= '9') {
      number = number * 10 + (c - '0');
    } else if (c == '\n' || c == '\r') {
      break;
    }
  }
  return isNegative ? -number : number;
}

void setup() {
  espSerial.begin(9600); // ініціалізується використання порту зі заданою швидкістю
  disp.clear(); // очищення відображуваного вмісту на індикаторі
}

void loop() {
  if (espSerial.available()) {
    value = readIntegerFromUART(espSerial);
    if (value != 0) {
      disp.clear();
      disp.setCursor(0);
      disp.print(value);
      disp.update();
    }
  }
  disp.tick();
}
```