Міністерство освіти і науки України

Львівський національний університет імені Іванка Франка

Факультет електроніки та комп’ютерних технологій Кафедра системного проектування

Курсова робота

Віддалене керування пристроєм ІоТ через Arduino Cloud

Виконав: студент групи ФеІ – 21

спеціальності 122 – Комп’ютерні науки

Крамар А.О.

Науковий керівник:

доц. Кушнір О.О.

« » 2024 р.

Львів 2024

**ЗМІСТ**

**АНОТАЦІЯ(шо то таке?!)**.......................................................................................

**ВСТУП**........................................................................................................................

**РОЗДІЛ 1. ТЕОРЕТИЧНІ ВІДОМОСТІ ПРО IOT ТА СУЧАСНІ ПІДХОДИ ДО ПРОЕКТУВАННЯ СИСТЕМ НА ЙОГО ОСНОВІ**................................

1.1 Теоретичні відомості.......................................................................................

1.1.1 Знайомство з IoT.....................................................................................

1.1.2 Концепція технології та підходи до її проектування..........................

1.2 Постановка завдання.......................................................................................

1.2.1 Що планується отримати в результаті виконання проекту.................

1.2.2 Вибір засобів та технологій для виконання проекту...........................

1.2.3 Аналіз аналогічних, чи близьких рішень.............................................

**РОЗДІЛ 2. РЕАЛІЗАЦІЯ ПРОЕКТУ**....................................................................

2.1 Реалізація апаратної частини проекту............................................................

2.1.1 Розгляд модуля ESP-01 для вирішення задач проекту........................

2.1.2 Побудова пристрою на макетній платі.................................................

2.1.3 Підключення до ПК та перевірка роботи модуля ESP-01..................

2.2 Підключення до Arduino Cloud......................................................................

2.2.1 Підготовка до роботи з платформою та додавання пристрою...........

2.2.2 Підключення пристрою та завантаження його виконавчого коду.....

2.3 Реалізація бібліотеки Python для керування пристроєм...............................

**РОЗДІЛ 3. ТЕСТУВАННЯ ТА ДЕМОНСТРАЦІЯ ФУНКЦІОНАЛЬНИХ МОЖЛИВОСТЕЙ ПРОЕКТУ**...........................................................................

3.1 Тестування проекту..........................................................................................

3.1.1 Методики тестування проекту...............................................................

3.1.2 Тестування проекту................................................................................

3.2 Демонстрація функціональних можливостей проекту.................................

**ВИСНОВКИ**...............................................................................................................

Основні результати проекту..................................................................................

Шляхи покращення...............................................................................................

Перспективи використання..................................................................................

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**..............................................................

**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

забезпезпечують 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 функціонал бібліотек для мов

програмування Python та JavaScript, що дозволяє встановлювати зв’язок з

Arduino Cloud навіть у тих проектах, де можливість такої взаємодії може

здатися неочевидною.

**РОЗДІЛ 1. ТЕОРЕТИЧНІ ВІДОМОСТІ ПРО IOT ТА СУЧАСНІ ПІДХОДИ ДО ПРОЕКТУВАННЯ СИСТЕМ НА ЙОГО ОСНОВІ**

1.1 Теоретичні відомості

1.1.1 Знайомство з IoT

Розглядаючи загальні основи даної технології та припускаючи, що дана

робота є ознайомчою, вважатимемо доцільним виокремити основи даної концепції.

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», яке вдало репрезентує суть концерції, адже засобами для взаємодії з мережею в даному випадку може бути буть-який, навіть буденний предмет, і те, які саме дані він буде передавати / одержувати, може обмежуватись лише поточними потребами та цілями проекту.

Зауважимо, що взаємодія між пристроями та мережею найчастіше обмежена різними об’єктивними факторами, які неможливо наперед передбачити (проблеми з підключенням пристрою, негаразди зі живленням, необхідність зміни програмного забезпечення тощо), відповідно до цього в концепції IoT окремо розділяють пристрій, як фізичний об’єкт, що не завжди може обмінюватись даними, та *річ* (Thing) — абстрактне представлення даного об’єкту в мережі, що відображає його стан та дані, якими він обмінюється. Дане розмежування досить вдало відображене у Arduino Cloud, де можна у зручному форматі ознайомитись з цією концепцією.

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.
* *Актуатори (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 здійсення первинної обробки зібраних даних до передачі даних на сервер або хмару для подальшого аналізу.
* *Концентратори (хаби)* — це пристрої, що групують кілька IoT-пристроїв та забезпечують їх зручне підключення до основної мережі. Нерідко їх наявність може бути зумовлена відсутністю можливості у деяких IoT-пристроїв здійснити безпосереднє підключення до мережі.
* *Хмарні платформи (сервери)* — це системи обробки, зберігання та аналізу великих обсягів даних, які збираються IoT-пристроями. 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.

Одним з ключових аспектів, де з’являються відмінності, є те, як саме взаємодіють пристрої IoT з мережею. Найчастіше, для них не передбачається значних обчислень, їх роль в даній системі часто обмежується лише одержанням даних та виконання певних дій в першу чергу на апаратному рівні, а для побудови моделі взаємодії всіх підключених до мережі пристроїв використовується серверна або хмарна інфраструктура. Такий підхід можна охарактеризувати як хмарнийпідхід *(«Cloud Computing»*).

Даний підхід, хоча і є в міру загальноприйнятим, але подеколи не може забезпечити необхідну швидкість обробки інформації і коректну роботу системи загалом. В такому разі застосовують периферійний підхід («*Edge Computing»*), де левова частка обробки зібраних даних відбувається безпосередньо на пристроях («переферія» мережі).

Розглядаючи проектування таких систем загалом, можна згадати про рівневий (*«Layered Architecture»*) підхід, на який особливо слід звертати увагу при побудові масштабних багатопланових мереж пристроїв, адже при ньому

IoT-система поділяється на окремі рівні, кожен з яких виконує свою функцію в архітектурі. Зазвичай виділяють рівень сенсорів, рівень передачі даних, рівень обробки та рівень застосунків.

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»*), 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. Питання захисту даних при передачі в значній мірі залежить від використаного протоколу передачі даних та його безпекових можливостей.

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. Для дистанційного керування доступна панель

Dashboard, а для інтеграції з іншими проектами Arduino Cloud підтримує

бібліотеки для Python, JavaScript та Web-API.

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. Детальні

характеристики, особливості підключення та роботи даного контролера будуть

описані при власне побудові пристрою в розділі 2, 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.

### 2. ***Google Cloud IoT Core* — платформа Google для побудови IoT-рішень. Надає широкі можливості для аналізу даних та машинного навчання, зручність для великих корпоративних IoT-рішень, забезпечує інтеграцію з іншими сервісами Google Cloud (BigQuery, ML Engine).**

### 3. ***Amazon AWS IoT Core —* частина Amazon Web Services, що дозволяє підключати IoT-пристрої до хмари AWS для збору, обробки та аналізу даних. Вирізняється** високою безпекою завдяки AWS IoT Device Defender, надійністю, інтеграцією з іншими сервісами AWS (S3, Lambda), гнучкі інструменти для великих масштабів.

**4.** *Blynk* — платформа для розробників IoT, що надає можливість реалізовувати інтерфейси для створення мобільних інтерфейсів та їх роботи з широким спектром мікроконтролерів.

Описані платформи відрізняються функціоналом і можливостями, тож вибір платформи може залежати від конкретних потреб вашого 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.

**РОЗДІЛ 2. РЕАЛІЗАЦІЯ ПРОЕКТУ**

2.1 Реалізація апаратної частини проекту

2.1.1 Розгляд модуля ESP-01 для вирішення задач проекту

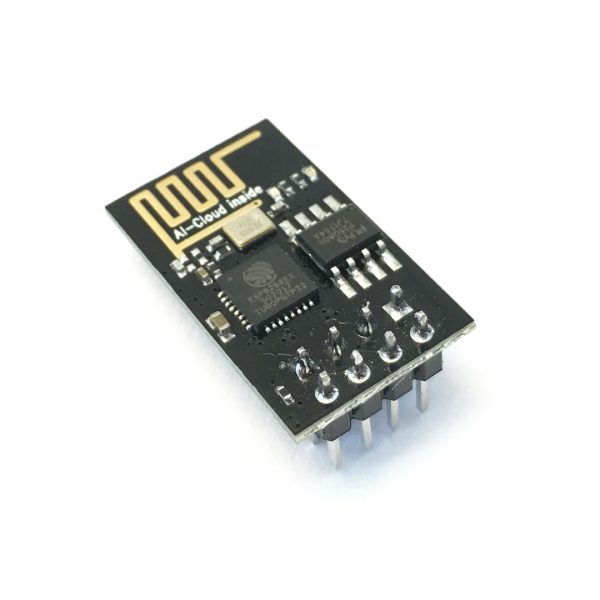


Рис. 2.1. Фактичне зображення модуля ESP-01

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

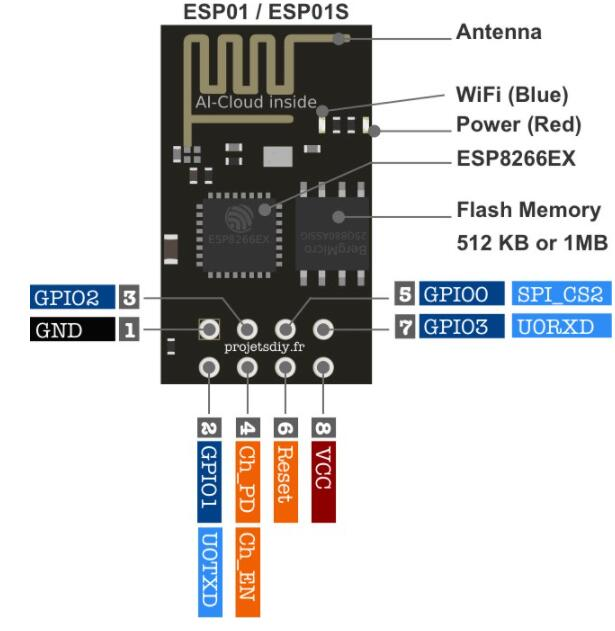


Рис. 2.2. Ілюстративне зображення виходів та зовнішніх елементів модуля ESP-01

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 унеможливлювати 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 витоку 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 контроллер 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 піни 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.

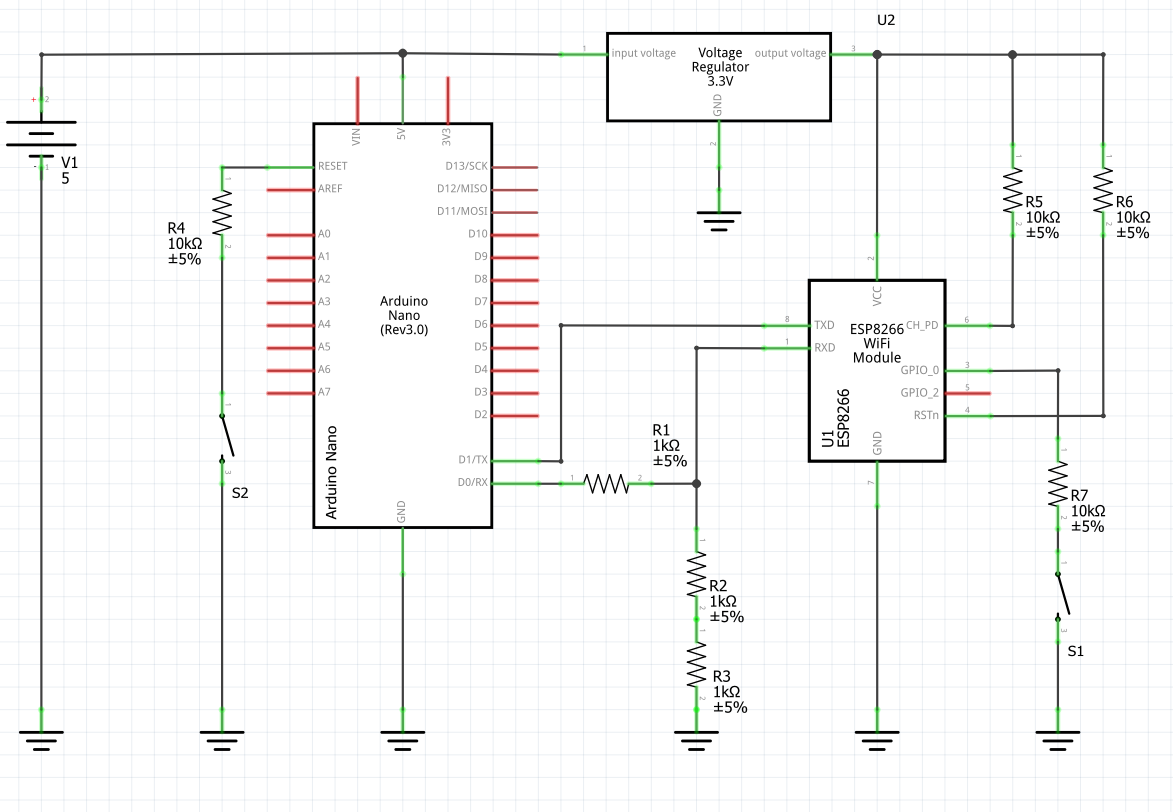
Усі вищеописані цифрові контакти, окрім своїх особливих функцій, разом зі GPIO2 дозволяють як і видавати потрібний логічний рівень назовні, так і визначити вхідний рівень, що сумарно дає можливість взаємодії з цифровими сенсорами та актуаторами. Для кожного з даних виходів доступне використання вбудованої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

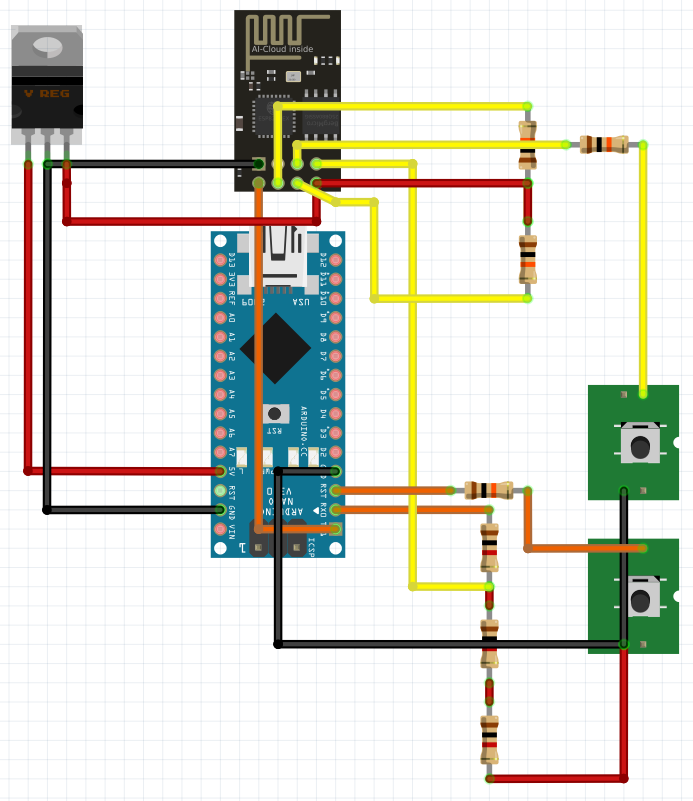


Рис. 2.4. Демонстраційна блок-схема пристрою

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 напряму. 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 сприйняє 3.3 В як високий логічний рівень, між контактами TXD додаткових елементів не потрібно.

Ключ S2 дозволяє перемикати 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, перемикати 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 приблизно 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 використаймо програму Arduino IDE.

Оберемо відповідний COM-порт, до якого під’єднаний пристрій:

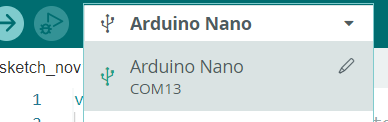


Рис. 2.5

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

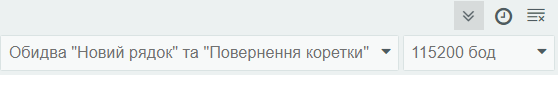


Рис 2.6

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

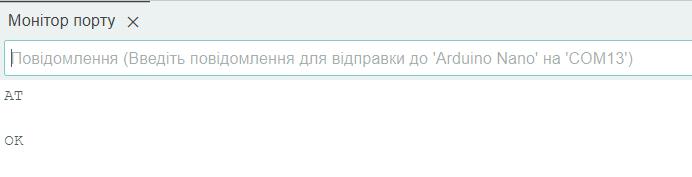


Рис 2.7

В разі необхідності, ми можемо використати й інші команди, проте поточний варіант 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/*](https://app.arduino.cc/)», використавши корпоративну скриньку.

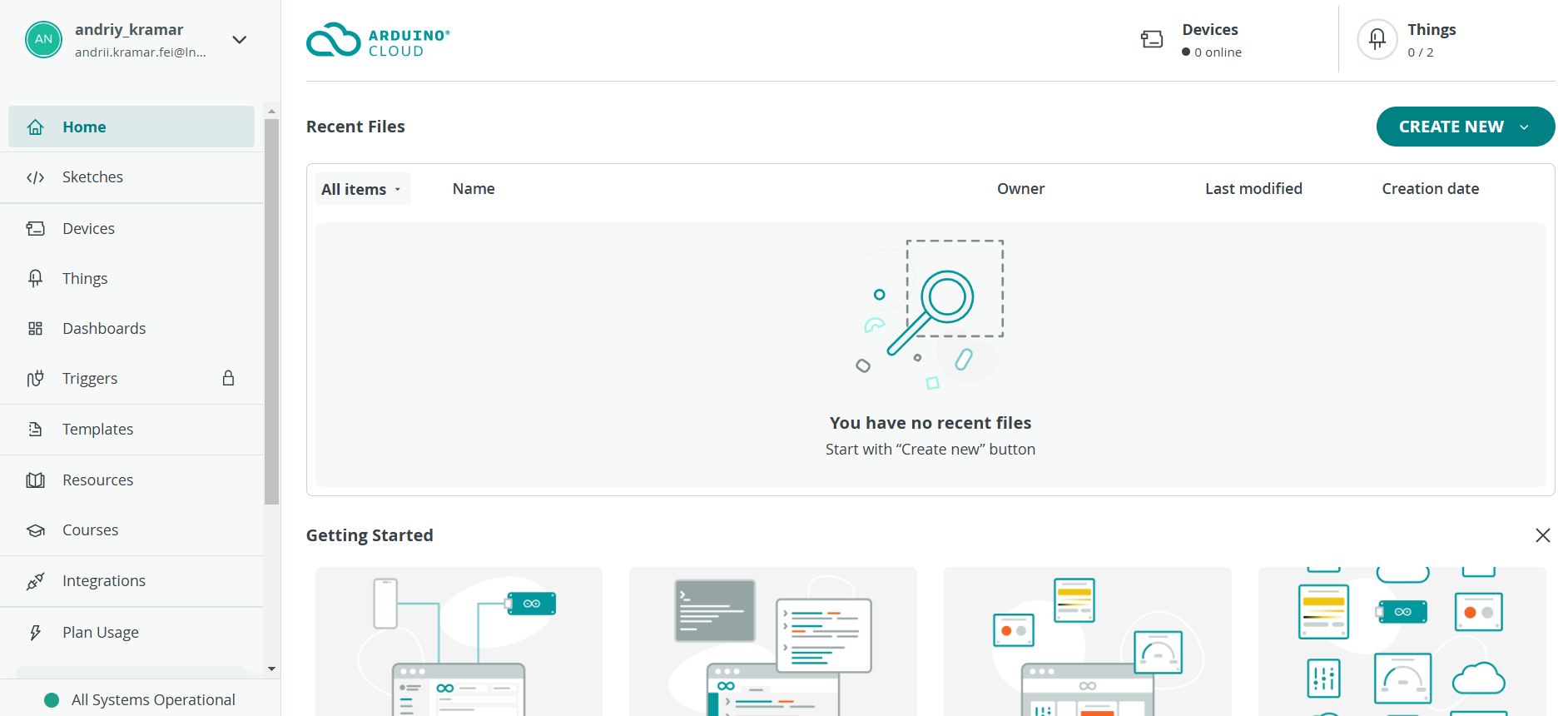


Рис 2.8. Загальний вигляд головної сторінки платформи

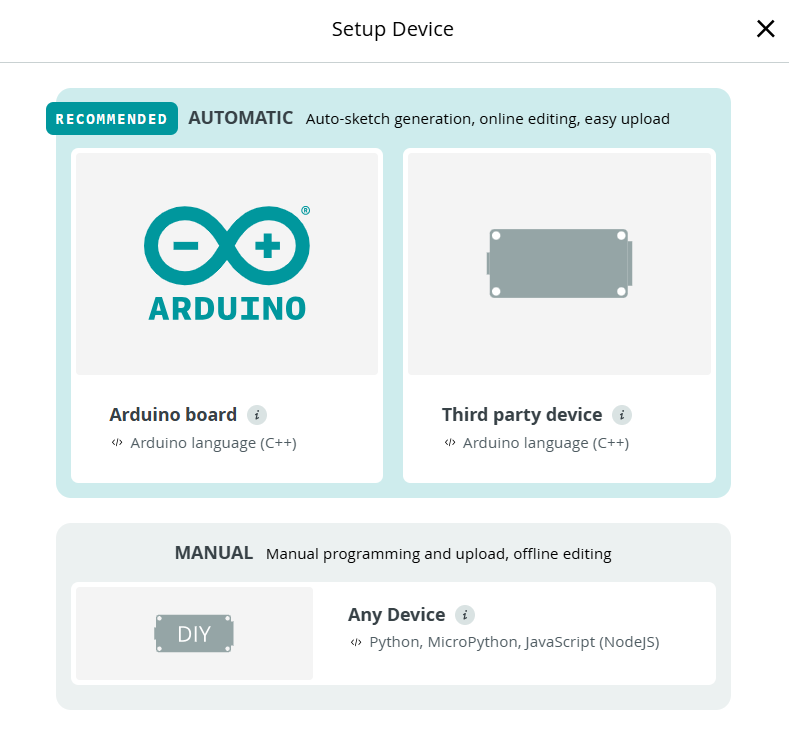


Рис. 2.9. Різновиди можливих пристроїв для роботи у Arduino Cloud

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*»). Третій варіант вартий окремої уваги: при ньому, доданий на платформі пристрій, фактично, не прив’язується до реального контролера напряму. 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. <aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa>

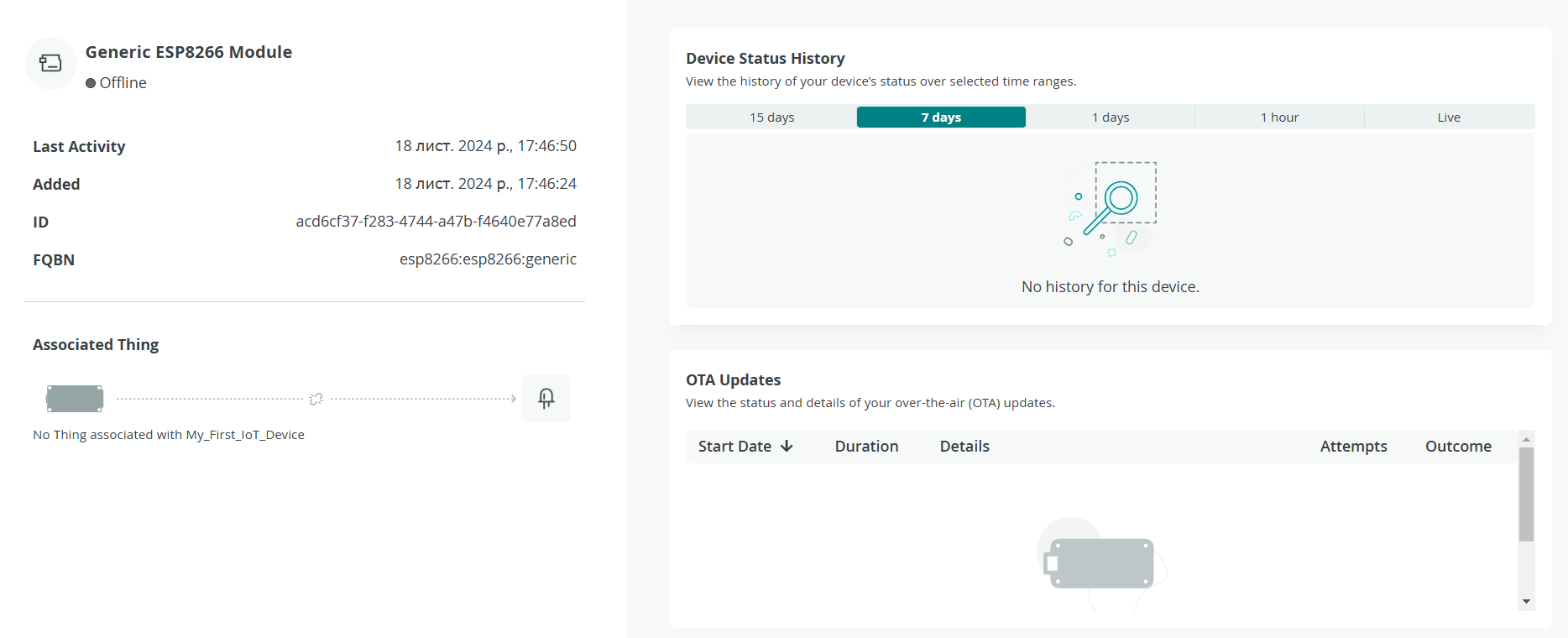


Рис. 2.10. Панель управління пристроєм у Arduino Cloud

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).

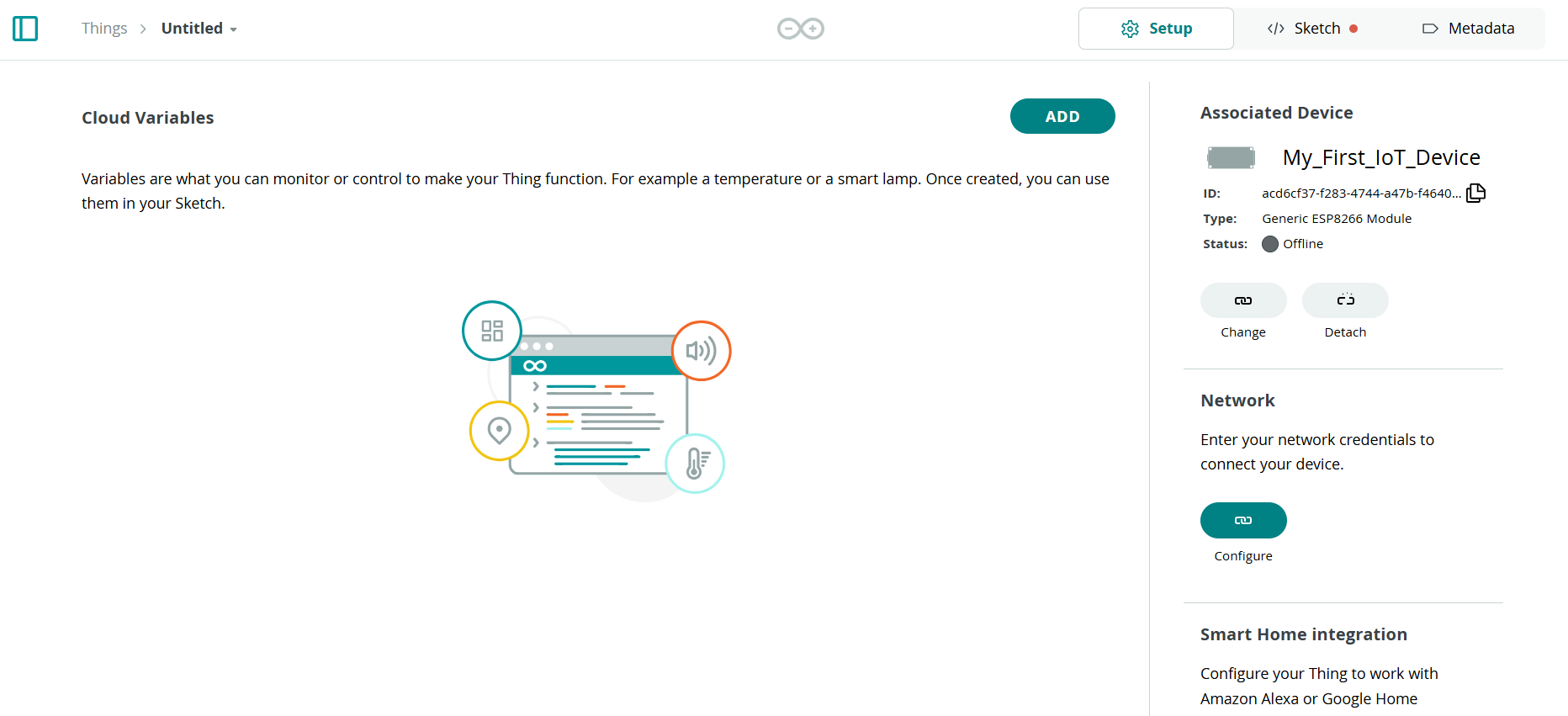
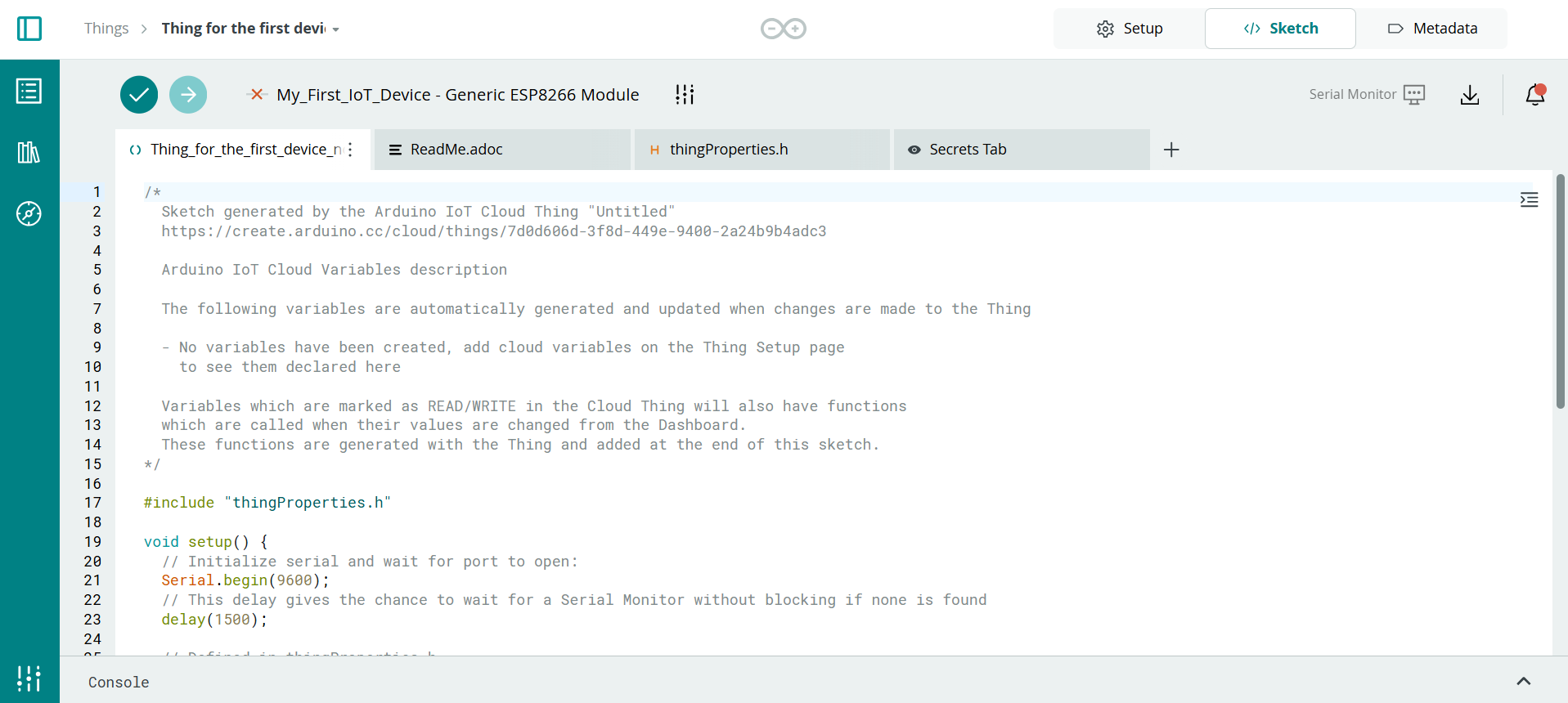


Рис. 2.11. Панель налаштування речі у Arduino Cloud

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/*](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 платформою 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 синхронізуватимуться 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 зміних 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).

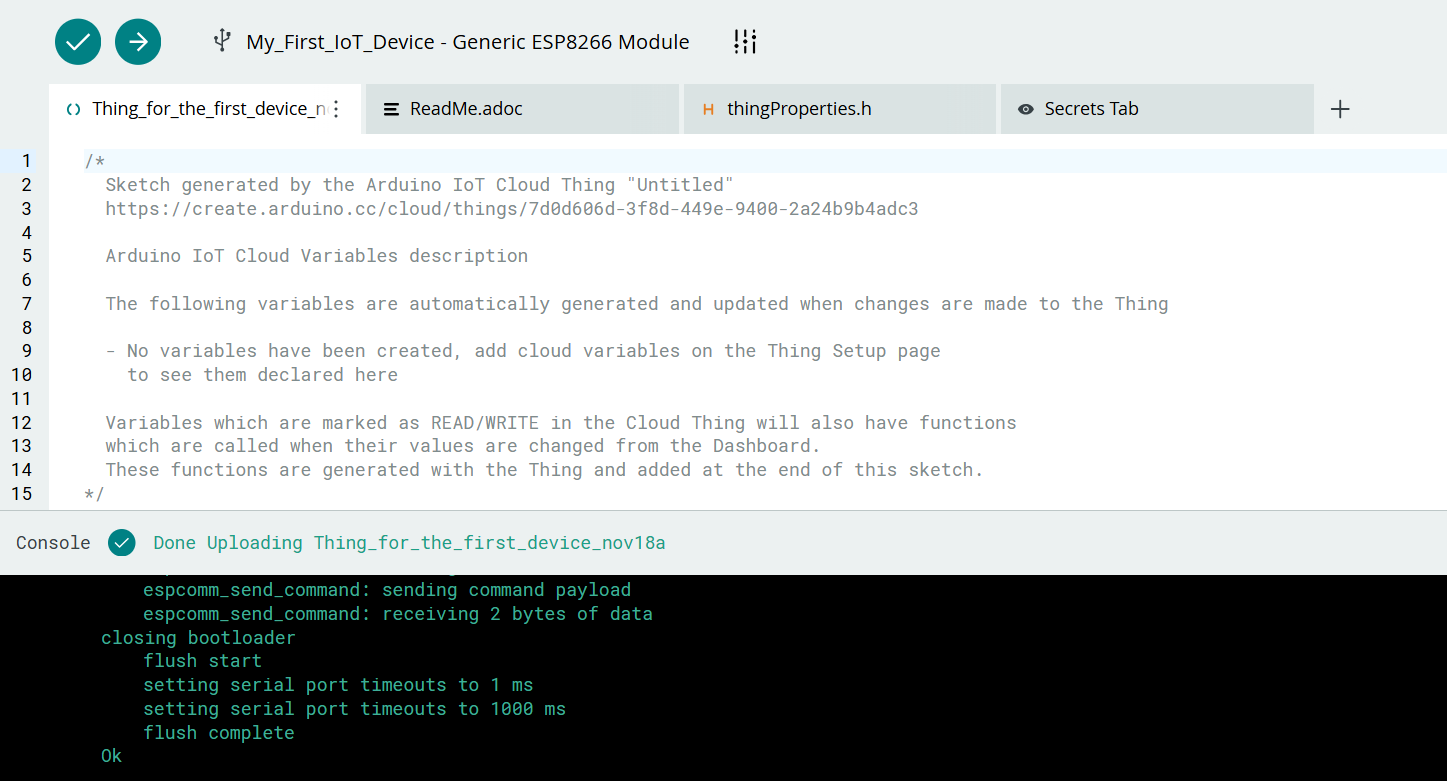


Рис. 2.13. Приклад успішного результату завантаження в консолі

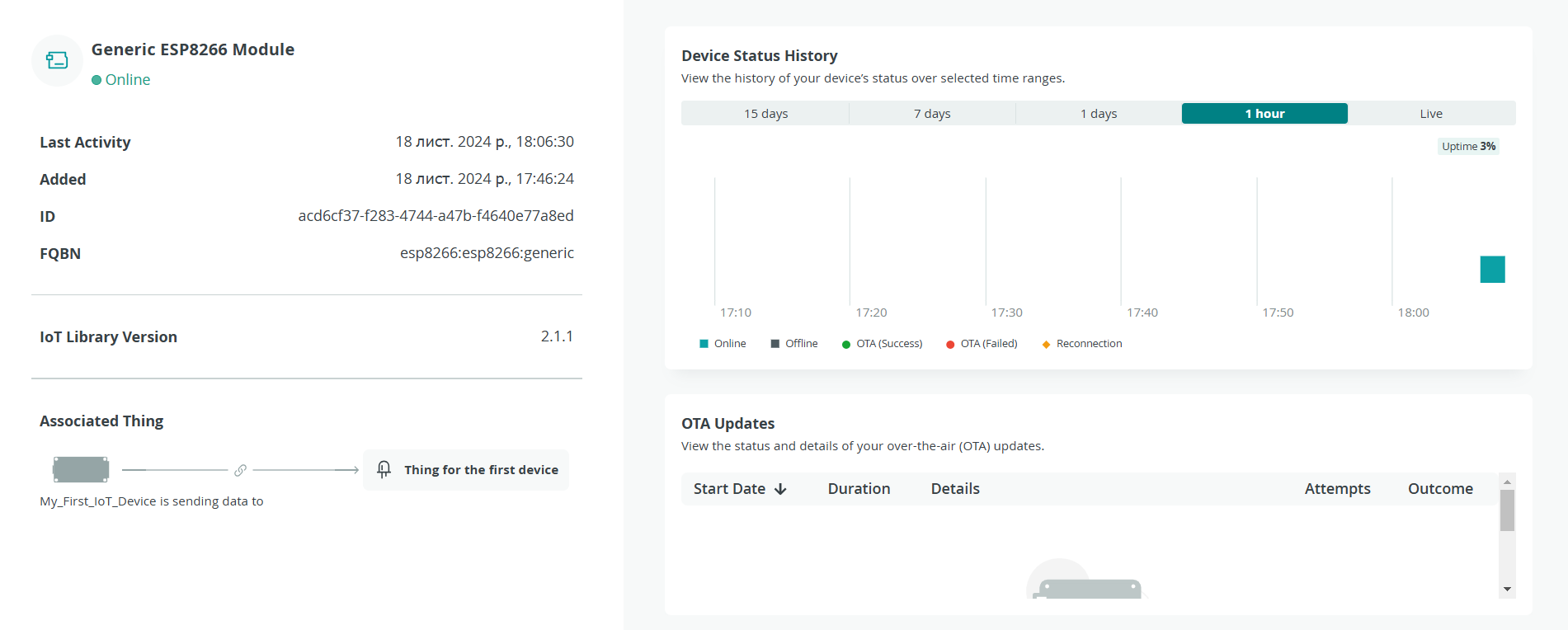


Рис. 2.14. Відображення активності пристрою після завантаження ПЗ

**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 певні компоненти з великого

/\*

**перевірити згадку RST Nano**

3.1 Тестування проекту..........................................................................................

3.1.1 Методики тестування проекту...............................................................

3.1.2 Тестування проекту................................................................................

3.2 Демонстрація функціональних можливостей проекту.................................

**ВИСНОВКИ**...............................................................................................................

Основні результати проекту..................................................................................

Шляхи покращення...............................................................................................

Перспективи використання..................................................................................

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**..............................................

\*/

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

|  |  |
| --- | --- |
| Електронні ресурси | Internet of Things, IoT. URL: https://www.it.ua/knowledge-base/technology-innovation/internet-veschej-internet-of-things-iot |
|  | IoT Platform Design Methodology: Top 5 Approaches,URL:https://minnovation.com.au/iot-platform/iot-platform-design-methodology-top-5-approaches/ |
|  | 10+ Best IoT Cloud Platforms in 2024, URL:https://euristiq.com/best-iot-cloud-platforms/ |
|  | Wi-Fi модуль ESP8266 версія ESP-01URL:https://arduino.ua/prod980-wifi-modyl-esp8266 |
|  | https://www.utmel.com/components/esp-01-wi-fi-module-esp-01-pinout-programming-and-esp-01-vs-esp8266-faq?id=990 |