МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Л.М. Олещенко

ПРОЄКТУВАННЯ ТА РОЗРОБЛЕННЯ МЕРЕЖЕВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Лабораторний практикум

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для студентів, які навчаються
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення мультимедійних та
інформаційно-пошукових систем»)

Київ
КПІ ім. Ігоря Сікорського
2023

Рецензенти: Чумаченко Сергій Мколайович, д-р техн. наук, проф.
Мошенський Андрій Олександрович, канд. техн. наук, доц.

Відповідальний редактор Легеза Віктор Петрович, д-р техн. наук, проф.

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського
(протокол № 6 від 30.03.2023 р.)
за поданням Вченої ради факультету прикладної математики
(протокол № 7 від 27.02.2023 р.)

Електронне мережне навчальне видання

Олещенко Любов Михайлівна, канд. техн. наук, доцент

ПРОЄКТУВАННЯ ТА РОЗРОБЛЕННЯ МЕРЕЖЕВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ЛАБОРАТОРНИЙ ПРАКТИКУМ

Проєктування та розроблення мережевого програмного забезпечення: лабораторний практикум з дисципліни «Проєктування та розроблення мережевого програмного забезпечення» [Електронний ресурс] : навч. посіб. для студ. спеціальності 121 «Інженерія програмного забезпечення» (освітня програма «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»)/ Л.М. Олещенко; КПІ ім. Ігоря Сікорського. – Електронні текстові дані. – Київ: КПІ ім. Ігоря Сікорського, 2023. – 108 с.

Навчальний посібник розроблено для ознайомлення студентів з технологіями програмування JavaScript та Python для проектування мереж IoT та розроблення програмного забезпечення для інтелектуальних пристроїв, використання API для мережевих додатків, вимогами до оформлення звітів виконаних лабораторних робіт. Навчальний посібник призначений для студентів, які навчаються за спеціальністю 121 Інженерія програмного забезпечення факультету прикладної математики КПІ ім. Ігоря Сікорського.

ЗМІСТ

ВСТУП

З кожним роком зростає кількість інтелектуальних пристроїв, які підключаються до мережі Інтернет. На сьогодні використання інтелектуальних систем набуло поширення як у повсякденному житті, так і на промисловому рівні. Впровадження інтелектуальних пристроїв потребує розроблення спеціалізованого програмного забезпечення для налаштування їх роботи в мережі. Крім програмування так званих розумних пристроїв, необхідно також вміти забезпечувати їх безпеку та захист від несанкціонованого доступу. Розвиток технологій Інтернету речей (англ. Internet of Things, IoT) зумовлює потребу у фахівцях, які здатні конфігурувати та програмно налаштовувати різноманітні пристрої IoT для мереж різної складності.

Метою вивчення дисципліни «Проєктування та розроблення мережевого програмного забезпечення» є формування у студентів здатностей розробляти програмне забезпечення для централізованого адміністрування мереж IoT згідно визначених вимог; програмно налаштовувати пристрої IoT для їх функціонування у мережі заданої топології, використання API (Application Programming Interface) для мережевих додатків.

Метою лабораторного практикуму є отримання практичних навичок програмування та налаштування пристроїв мережі IoT. У лабораторному практикумі надаються теоретичні відомості з кожної теми, завдання на лабораторні роботи з цієї теми, методичні вказівки щодо виконання завдання, вимоги щодо оформлення звітів до лабораторних робіт та рекомендована література.

Навчальний посібник призначений для студентів, які навчаються за спеціальністю 121 «Інженерія програмного забезпечення» факультету прикладної математики КПІ ім. Ігоря Сікорського.

ЛАБОРАТОРНА РОБОТА 1.

ПІДКЛЮЧЕННЯ ПРИСТРОЇВ ІОТ ДО РОЗУМНОГО БУДИНКУ

Мета роботи*:* отримати практичні навички налаштування та підключення пристроїв IoT до мережі розумного будинку, у середовищі моделювання Packet Tracer спроєктувати мережу згідно інструкцій до лабораторної роботи, додати бездротові пристрої Wired IoT до Smart Home Network, налаштувати мережевий адаптер та перевірити стан пристроїв через IoT Server.

Теоретичні відомості

Інтернет речей (IoT) описує мережу фізичних об'єктів – «речей», у які вбудовано датчики, програмне забезпечення та інші технології для підключення та обміну даними з іншими пристроями та системами через мережу Інтернет. Ці пристрої варіюються від звичайних побутових предметів до складних промислових інструментів.

Для проєктування мереж та розроблення мережевого програмного забезпечення для IoT використовуються різні середовища моделювання, прикладомом є платформа Wokwi – онлайн-симулятор для проєктування, налаштування та роботи електроніки та пристроїв IoT, для симуляції Arduino, ESP32 та багатьох інших популярних плат, деталей та датчиків (рис. 1.1).

Можливості середовища моделювання Wokwi:

1. Симуляція Wi -Fi – підключення змодельованого проєкту до мережі Інтернет, використовуючи MQTT, HTTP, NTP та інших мережевих протоколів.
2. Віртуальний логічний аналізатор. Використання цифрових сигналів у симуляції (наприклад, UART, I2C, SPI) та їх аналіз на своєму комп'ютері.
3. Розширене налагодження за допомогою GDB – потужний налагоджувач Arduino та Raspberry Pi Pico для досвідчених користувачів.

4. Моделювання SD – карти, зберігання та витягування файлів та каталогів з програмного коду, можливість завантаження двійкових файлів.
5. API мікросхем – створення власних мікросхем та деталей, можливість поширення їх для учасників спільноти.
6. Інтеграція коду Visual Studio – моделювання проєктів безпосередньо з коду VS.

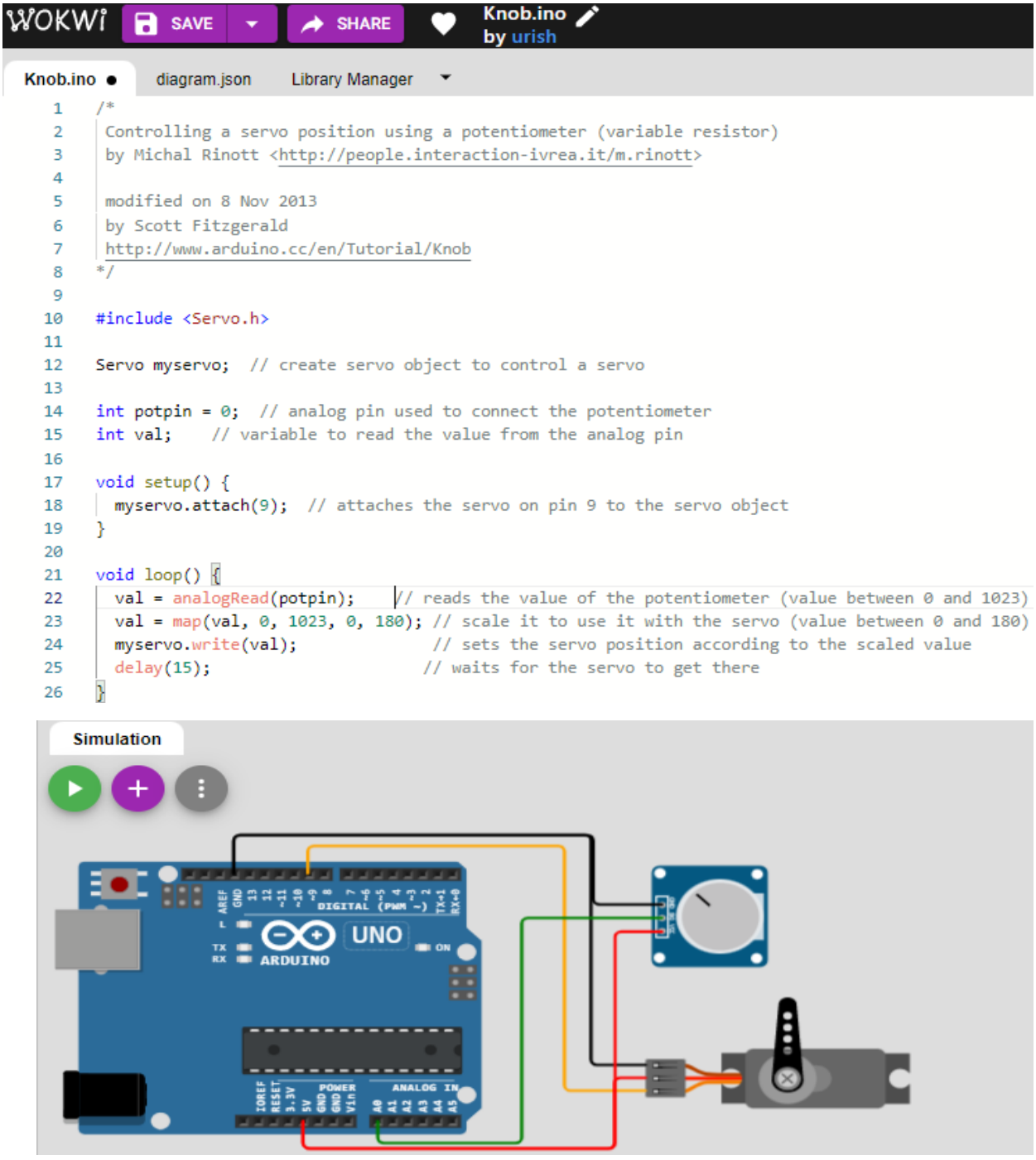


Рис. 1.1. Інтерактивне середовище програмування пристроїв Wokwi [1]

Середовище проектування комп’ютерних мереж Cisco Packet Tracer містить широкий набір інструментів для моделювання фізичних топологій мереж, налаштування та програмування розумних пристроїв для різного роду задач (рис.1.2).

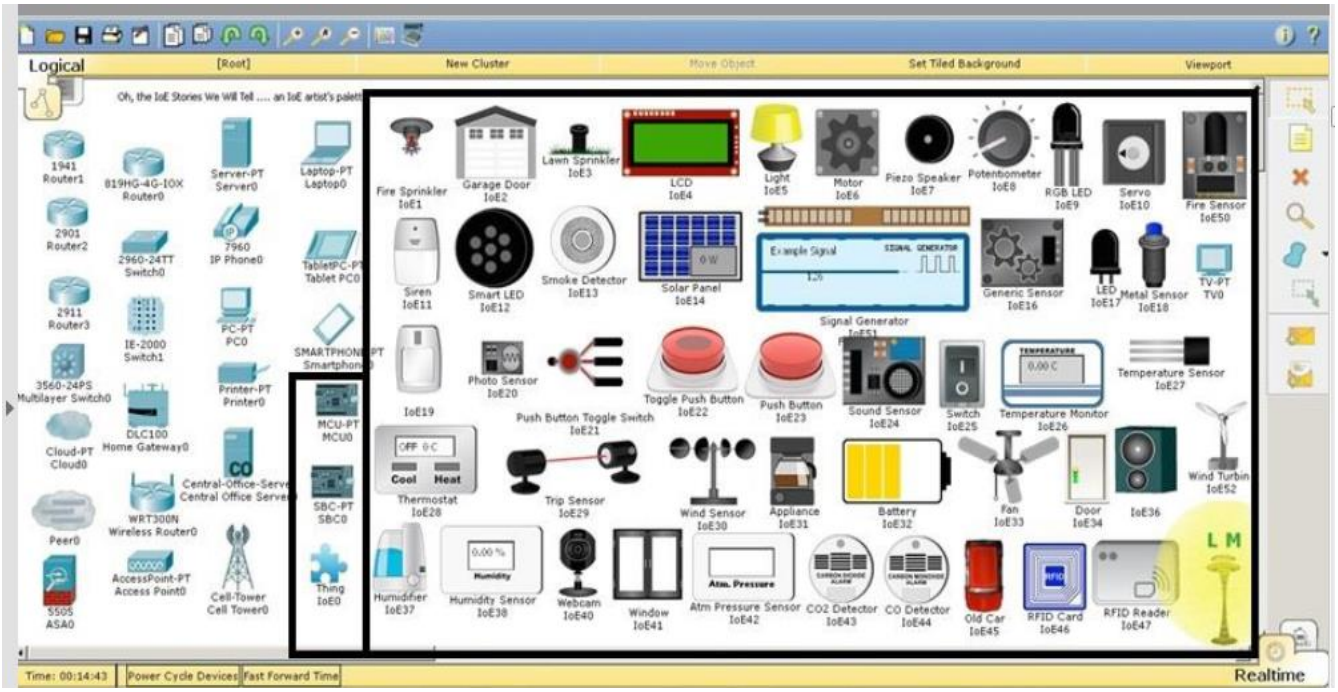


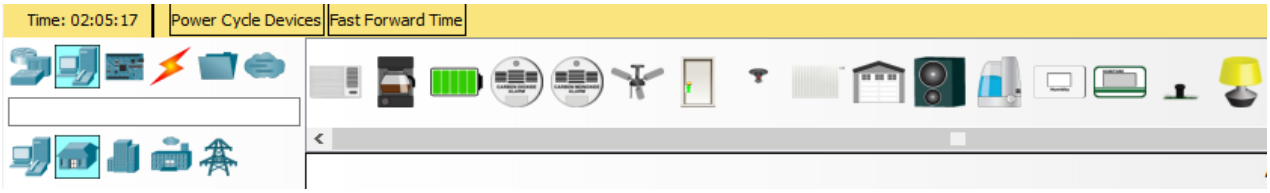
Рис.1.2. Можливості середовища моделювання Packet Tracer для підключення пристроїв IoT до мережі [2]

Завдання до лабораторної роботи 1

У даній лабораторній роботі потрібно відкрити файл Packet Tracer Smart_Home_Network.pkt з вже створеною інтелектуальною мережею, дослідити пристрої, які присутні в мережі та додати IoT компоненти, які можна під'єднати через кабель та бездротову мережу.

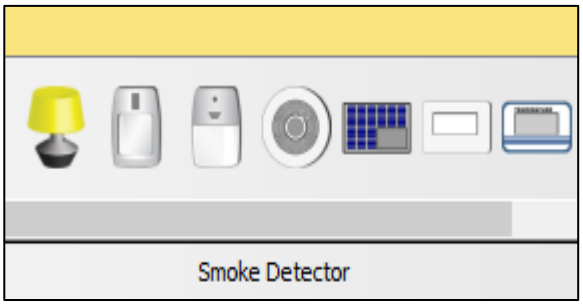
Відкрийте файл Smart_Home_Network.pkt. Збережіть файл на своєму комп'ютері. Дослідіть інтелектуальну домашню мережу. Дослідіть кінцеві пристрої IoT.

У лівому нижній частині середовища Packet Tracer знайдіть і натисніть значок [End Devices] у верхньому рядку, значок [Home] в нижньому рядку Вибір типу пристрою.

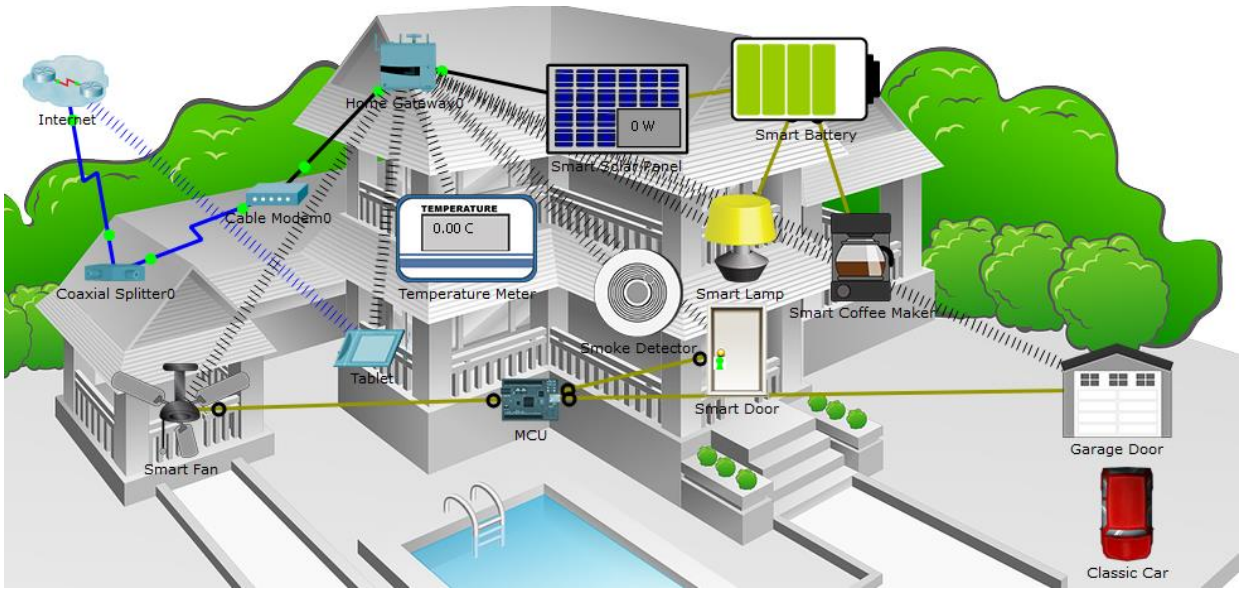


У нижній частині вікна Packet Tracer поле Device-specific Selection відображає доступні пристрої Smart Home IoT.

Перемістіть вказівник миші на кожний пристрій і зверніть увагу, що описове ім'я пристрою відображається в нижній частині вікна Device-specific Selection. Подивіться на кожний тип пристрою.



Дослідіть мережу Розумного будинка.

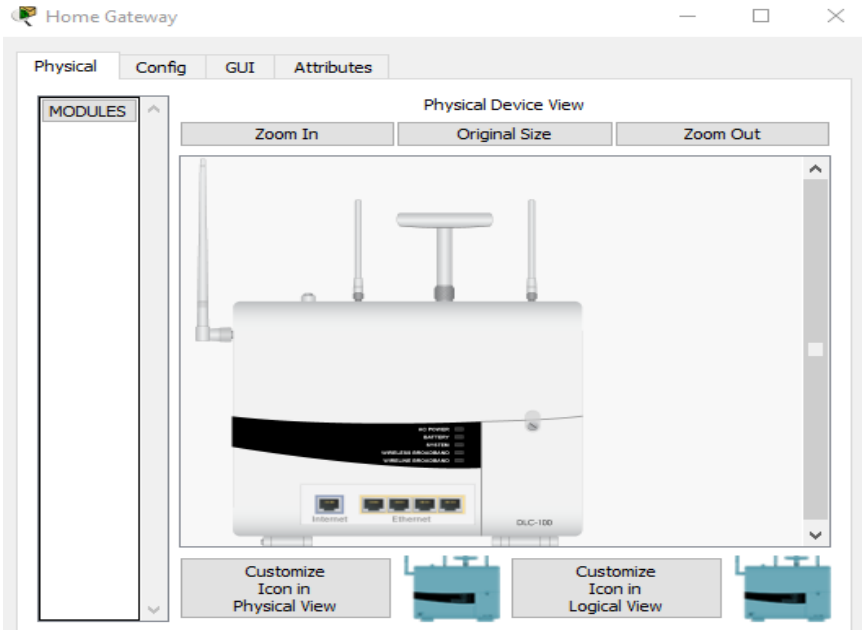


У робочому середовищі Packet Tracer є попередньо побудована розумна домашня мережа, яка складається з проводових та бездротових пристроїв IoT та пристроїв мережевої інфраструктури.

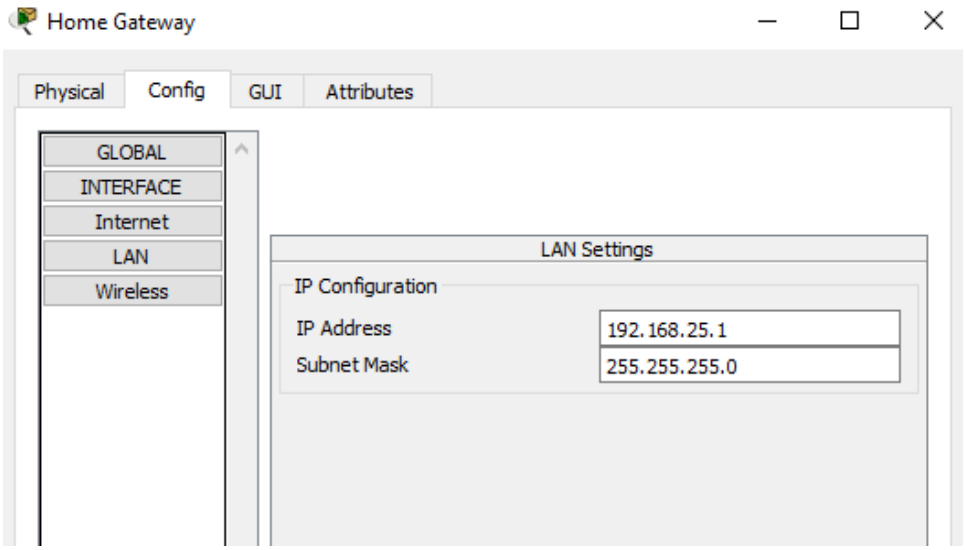
Коли ви наводите курсор на пристрій, наприклад, Smart Fan, відкривається інформаційне вікно, що містить основну інформацію про мережу на цьому пристрої.

Щоб увімкнути або активувати пристрій, натисніть Alt на клавіатурі, а потім за допомогою лівої кнопки миші оберіть пристрій. Спробуйте це на кожному із смарт-пристроїв, щоб спостерігати, що вони можуть виконувати. Інтелектуальна домашня мережа складається з інфраструктурних пристроїв, таких як домашній шлюз. Натисніть на Home Gateway, щоб відкрити вікно Home Gateway.

Вкладка Physical вибрана за замовчуванням та відображає зображення домашнього шлюзу.

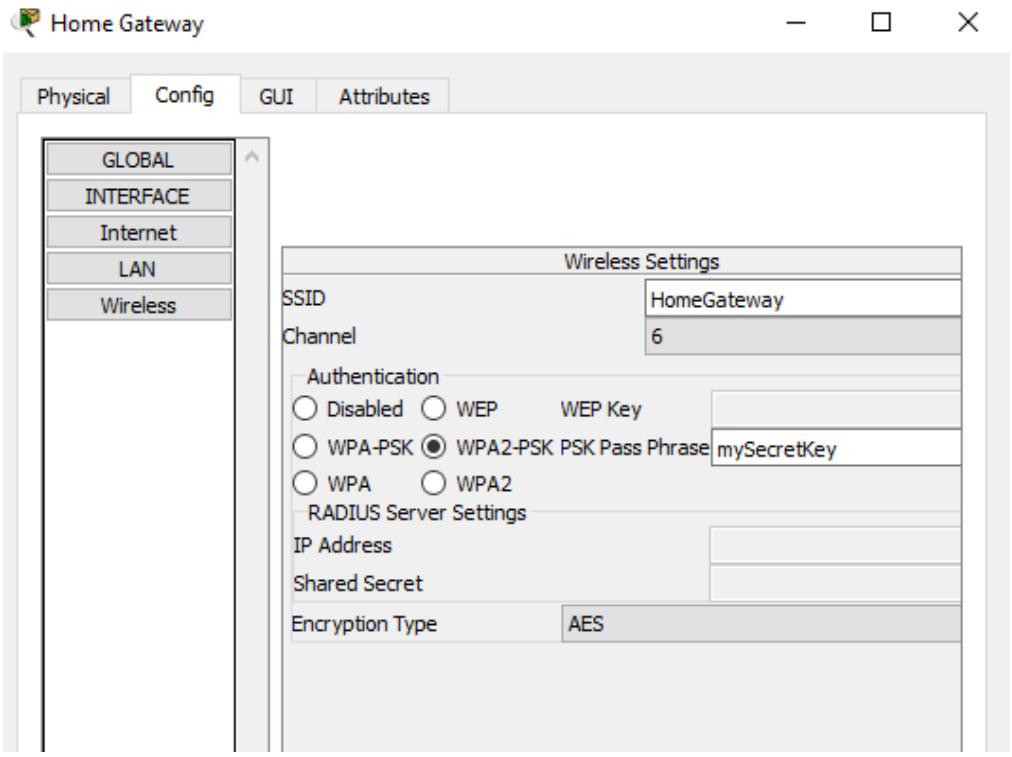


Далі клацніть вкладку Конфігурація, а потім на лівій панелі клацніть LAN, щоб переглянути параметри локальної мережі LAN домашнього шлюзу Home Gateway. Запишіть IP-адресу домашньої мережі для її подальшого використання.



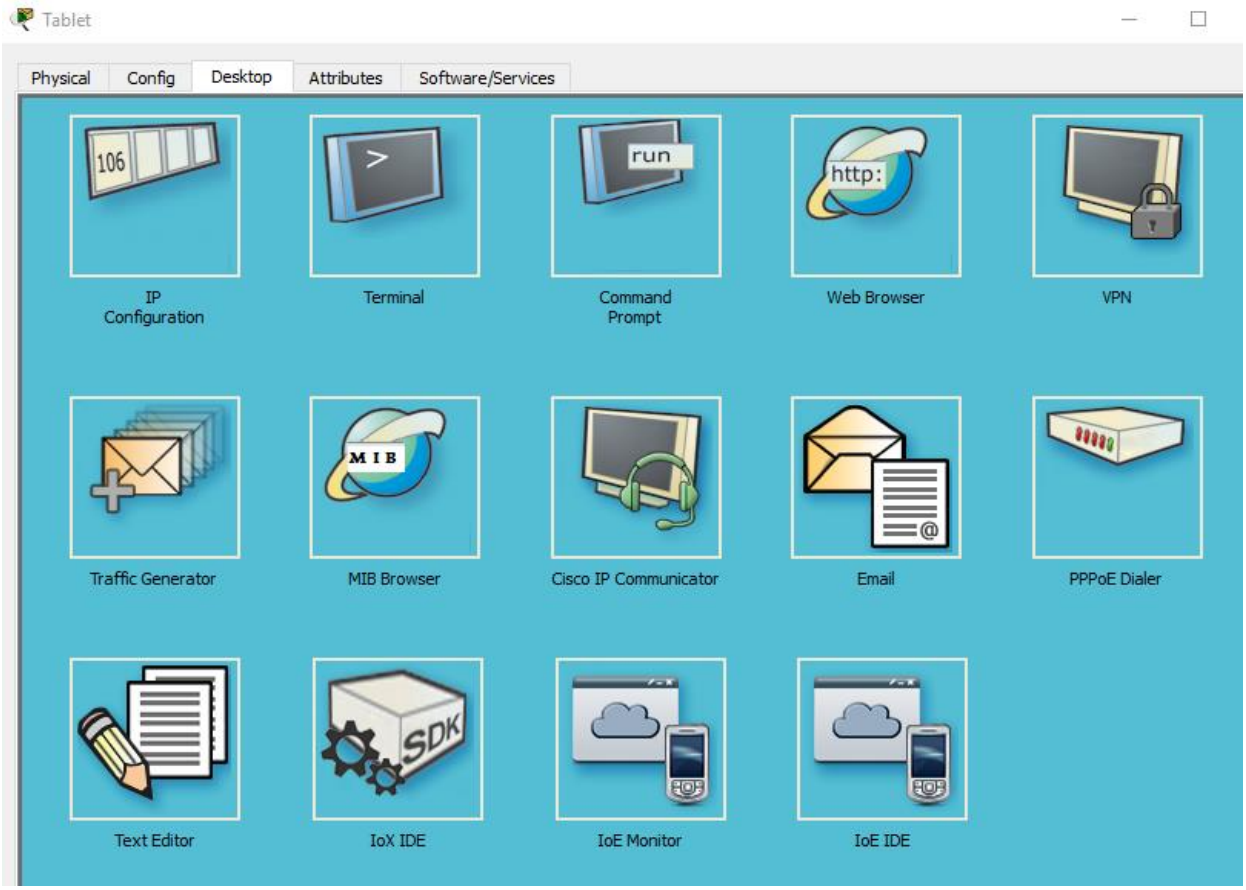
Натисніть Wireless на лівій панелі, щоб переглянути параметри бездротового зв'язку домашнього шлюзу.

Випишіть значення SSID даної мережі та WPA2-PSK Pass Phrase для їх подальшого використання у роботі.

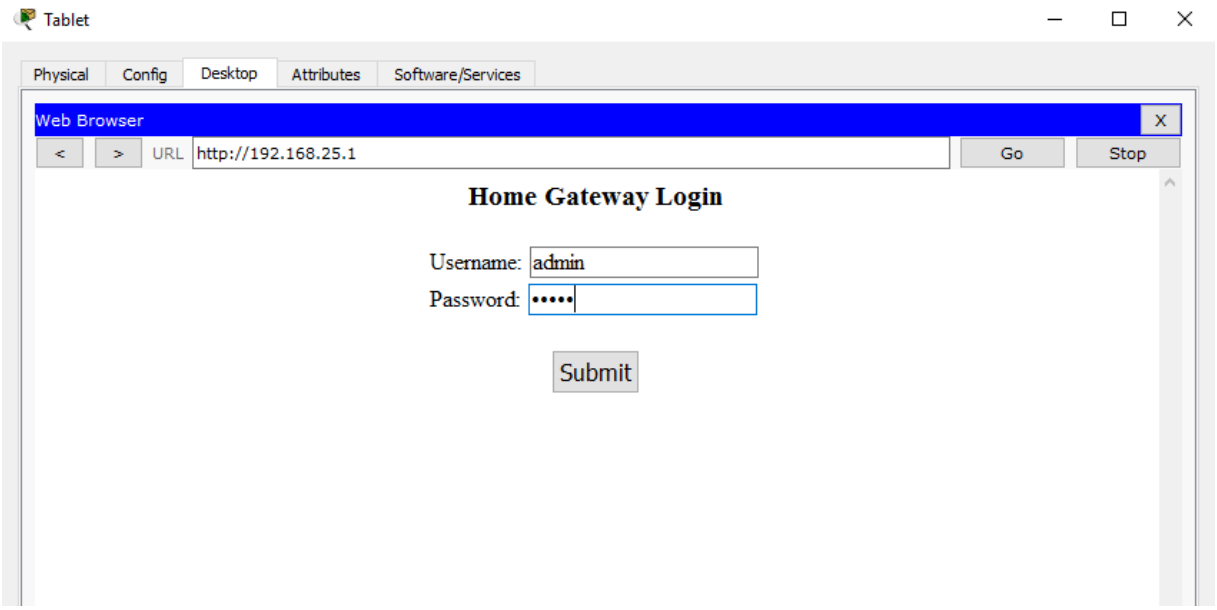


Закрийте вікно Home Gateway, натисніть значок пристрою Tablet, щоб відкрити вікно планшета.

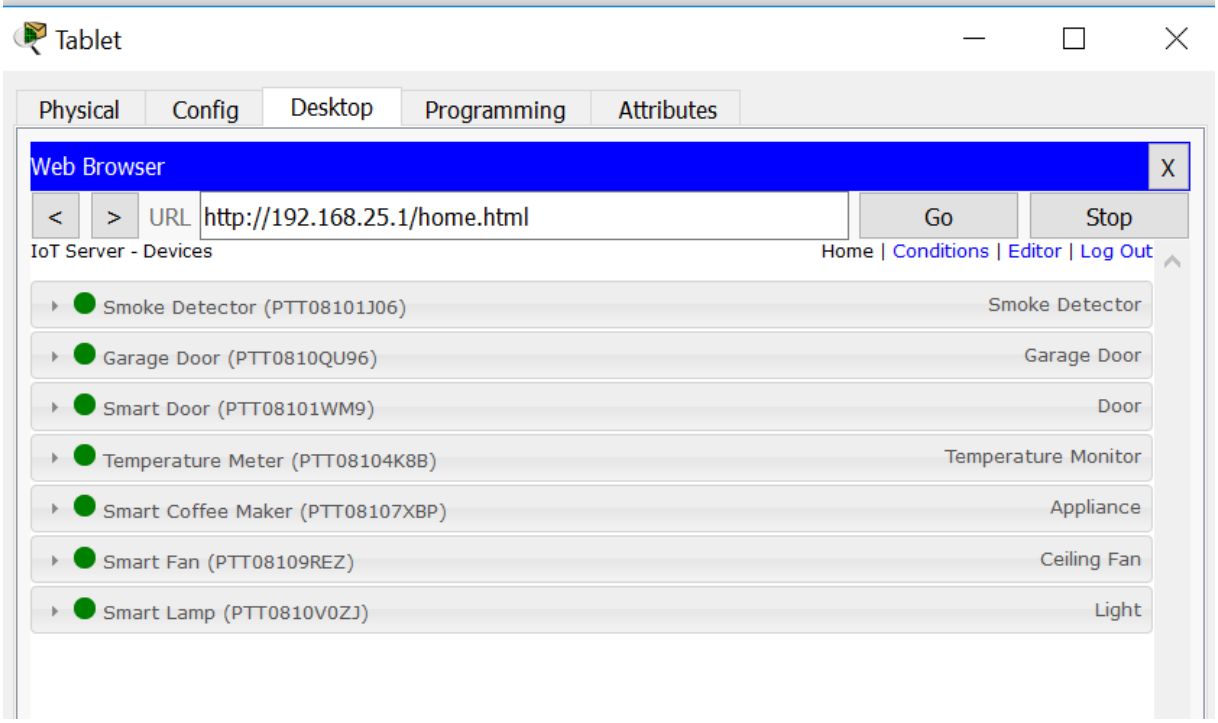
У вікні планшета виберіть вкладку Desktop та натисніть значок веббраузера.



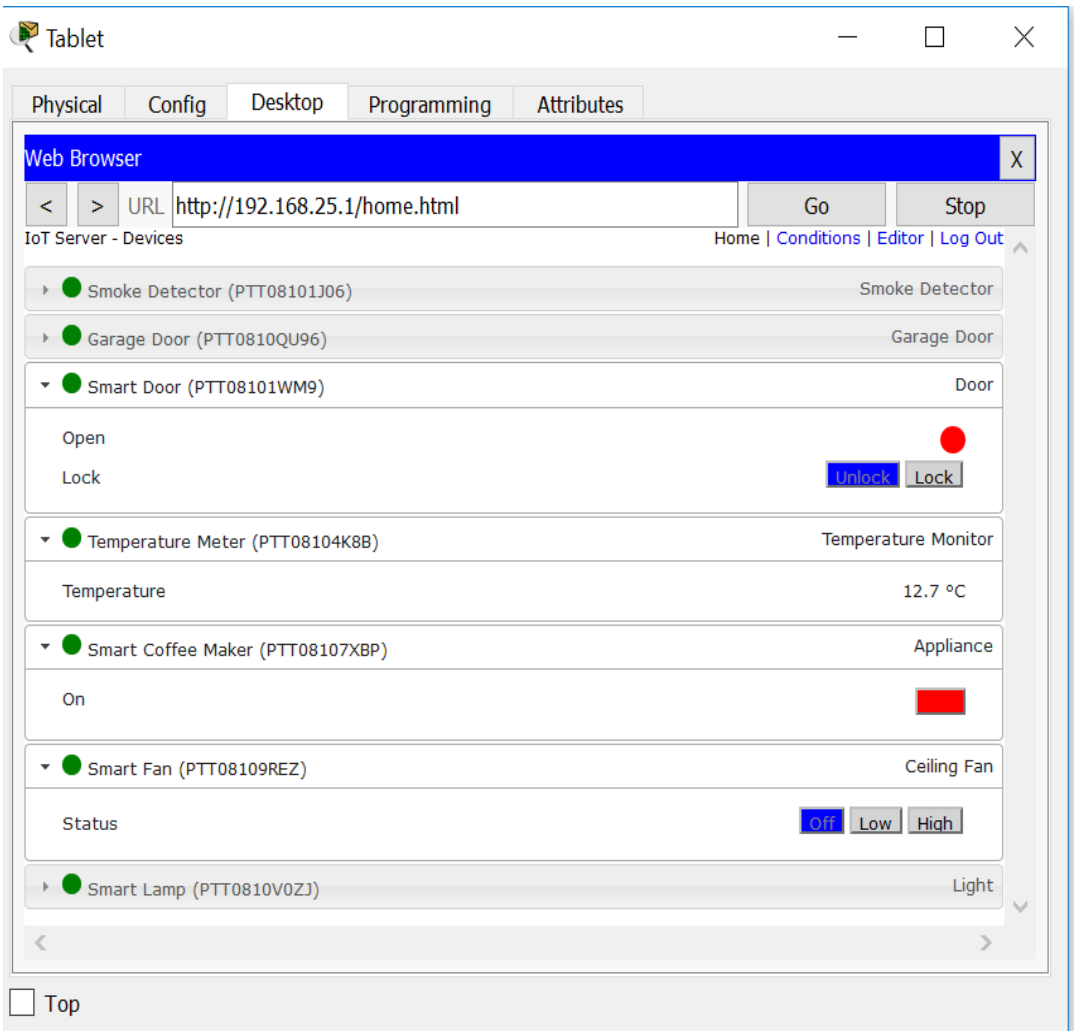
У вікні Web Browser введіть IP-адресу домашнього шлюзу 192.168.25.1 у поле URL-адреси та натисніть Go. На екрані входу в головний шлюз введіть ім’я та пароль користувача – admin, натисніть Надіслати.



Після підключення до вебінтерфейсу домашнього шлюзу з'явиться список усіх підключених пристроїв IoT.



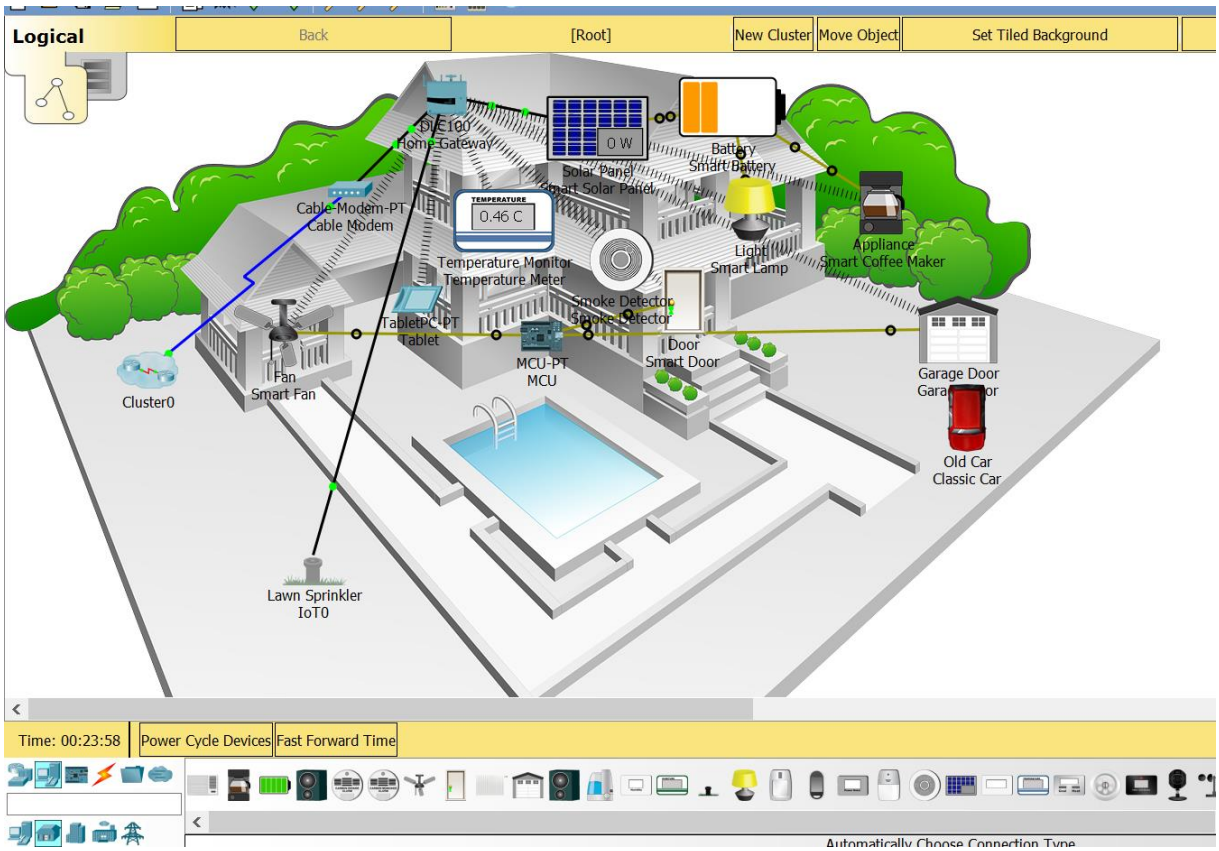
Коли ви натискаєте на пристрій у списку, відображається статус та налаштування цього пристрою.



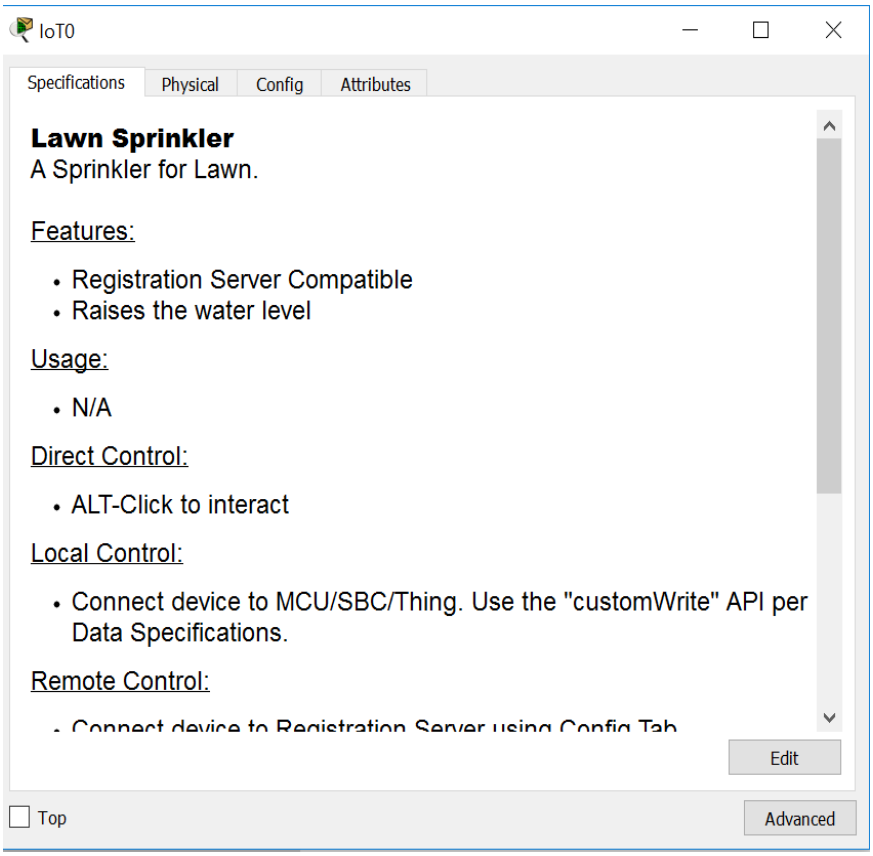
Закрийте вікно планшета. Додайте Wired IoT пристрої до Smart Home Network. Підключіть пристрій до мережі. У вікні Device-specific Selection натисніть значок розбризкувача води [Lawn Sprinkler], оберіть робочу область, для розташування Lawn Sprinkler. Підключіть Fire Sprinkler до Home Gateway. У вікні вибору типу пристрою (Device-Type Selection) клацніть піктограму з'єднання [Connections] (виглядає як блискавка). Натисніть на тип з'єднувача Copper Straight Through в полі Device-Specific Selection.

Потім натисніть на значок Sprinkler та підключіть один кінець кабелю до FastEthernet0 Sprinkler. Далі натисніть значок Home Gateway і підключіть інший кінець кабелю до доступного Ethernet.

Налаштуйте розбризгувач для підключення до мережі. Натисніть на Lawn Sprinkler у робочому середовищі, щоб відкрити вікно пристрою. Зверніть увагу, що назва розбризгувача є загальною IoT0.

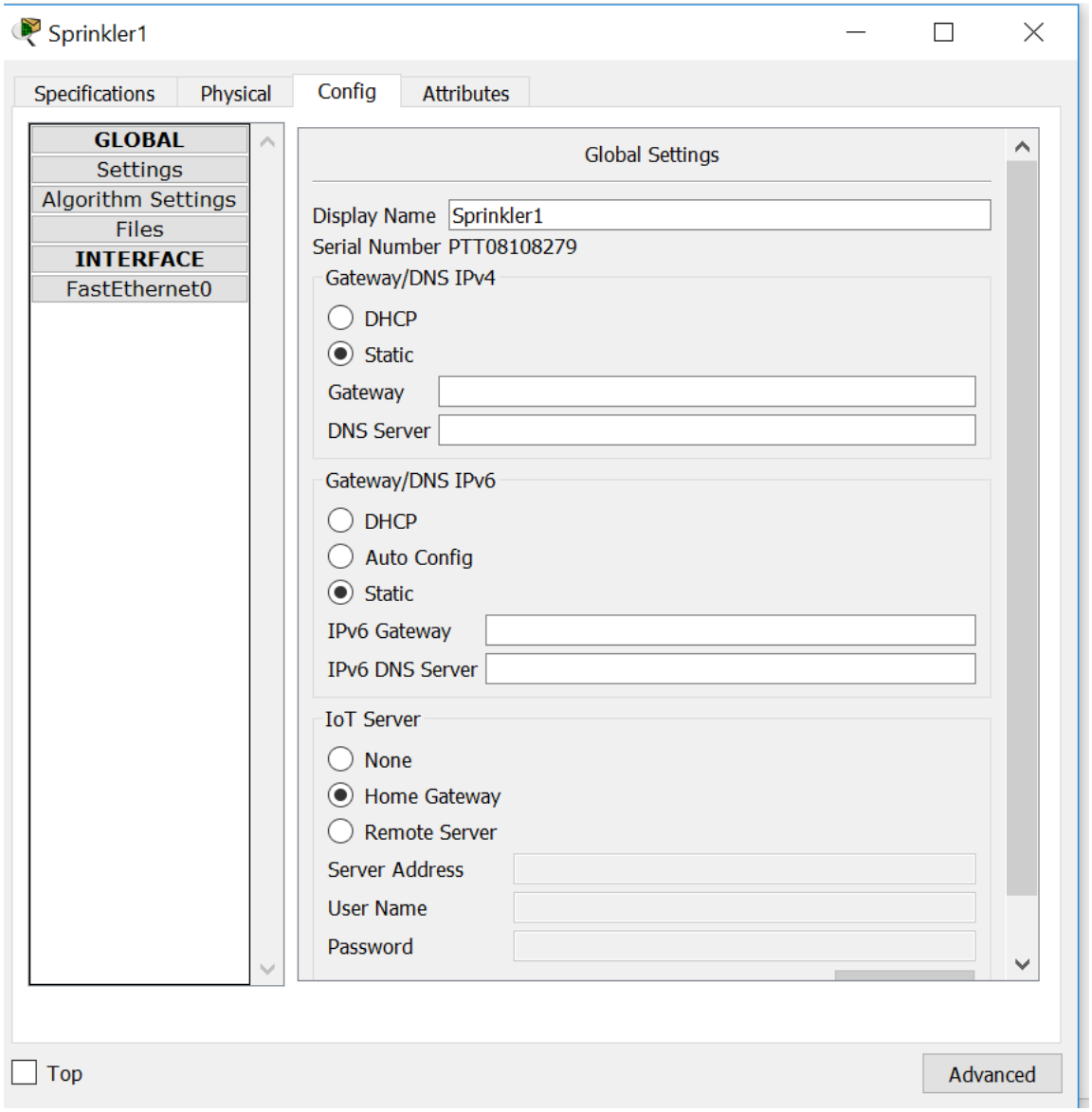


Вікно пристрою відкриється на вкладці Специфікація, в якій міститься інформація про пристрій, який можна редагувати.

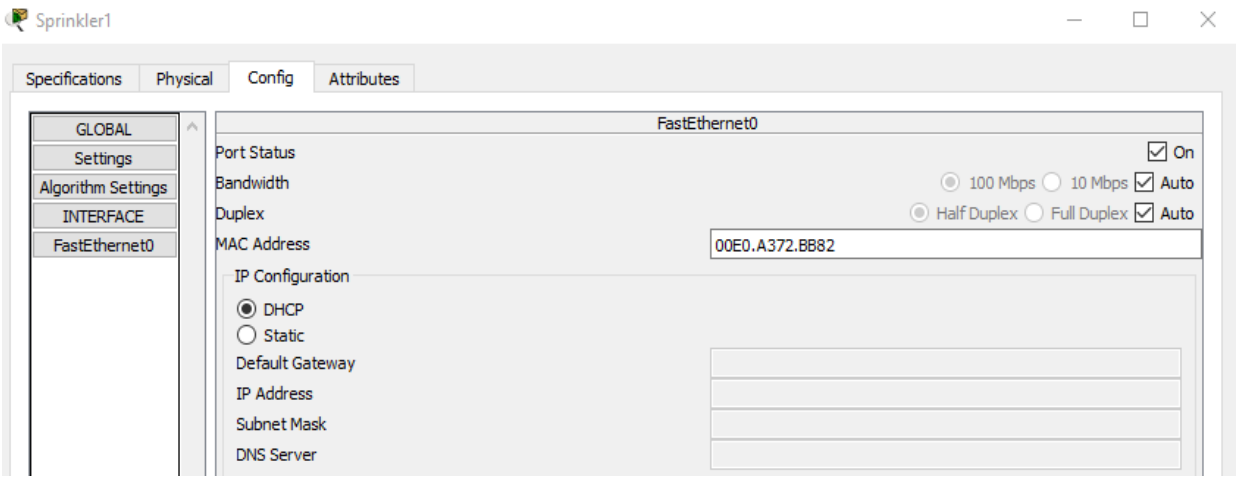


Натисніть вкладку Config, щоб змінити налаштування конфігурації пристрою. На вкладці Config внесіть такі зміни до налаштувань:

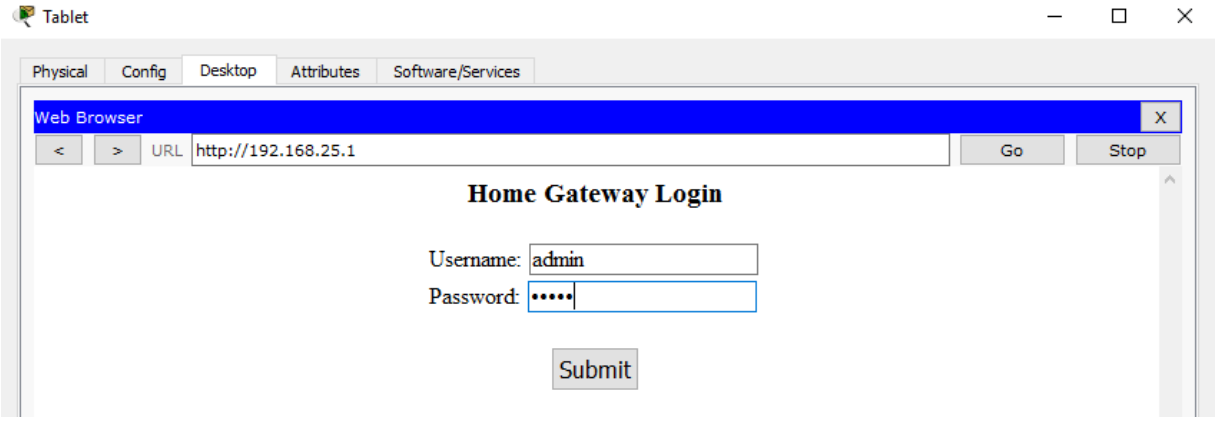
- Установіть Display Name до Sprinkler1 (ім'я вікна змінюється на Sprinkler1).
- Встановіть IoT-сервер на Home Gateway.



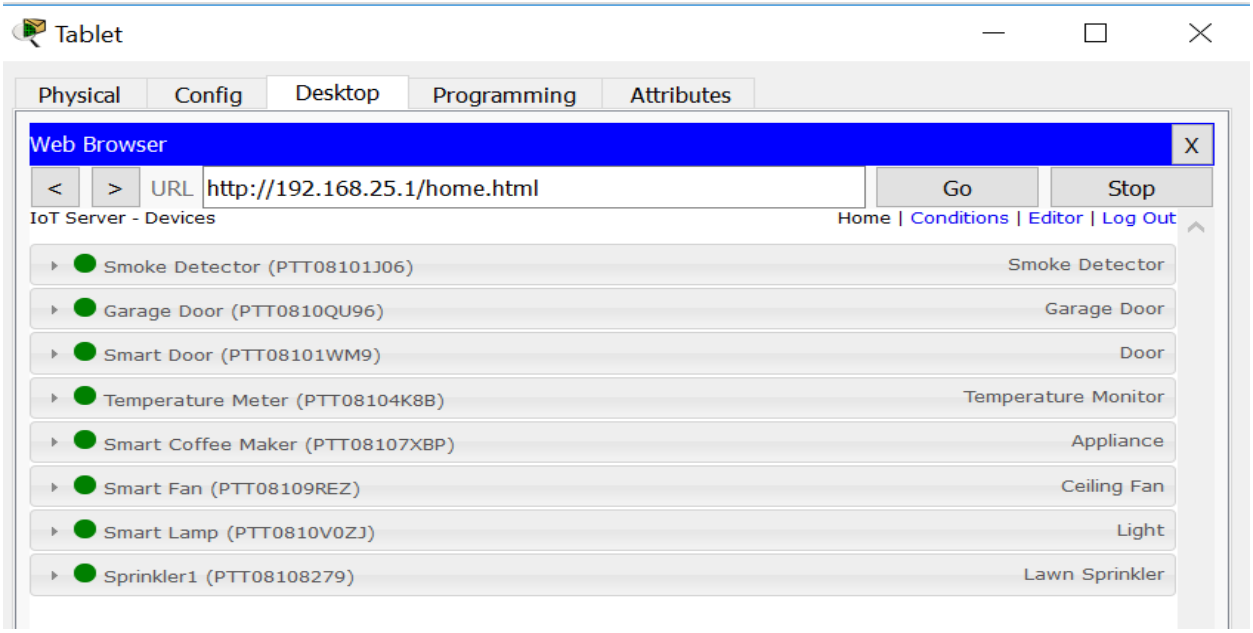
Натисніть FastEthernet0 та змініть IP Configuration на DHCP.



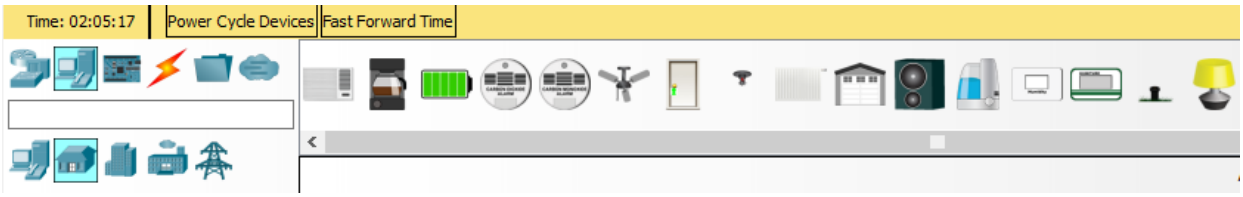
Закрийте вікно пристрою Sprinkler1. Перевірте, чи є розбризкувач у мережі. Увійдіть у домашній браузер з планшета.



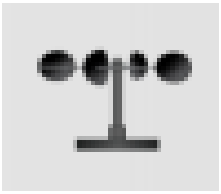
Пристрій Sprinkler 1 тепер повинен з'явитися в списку IoT Server - Devices.



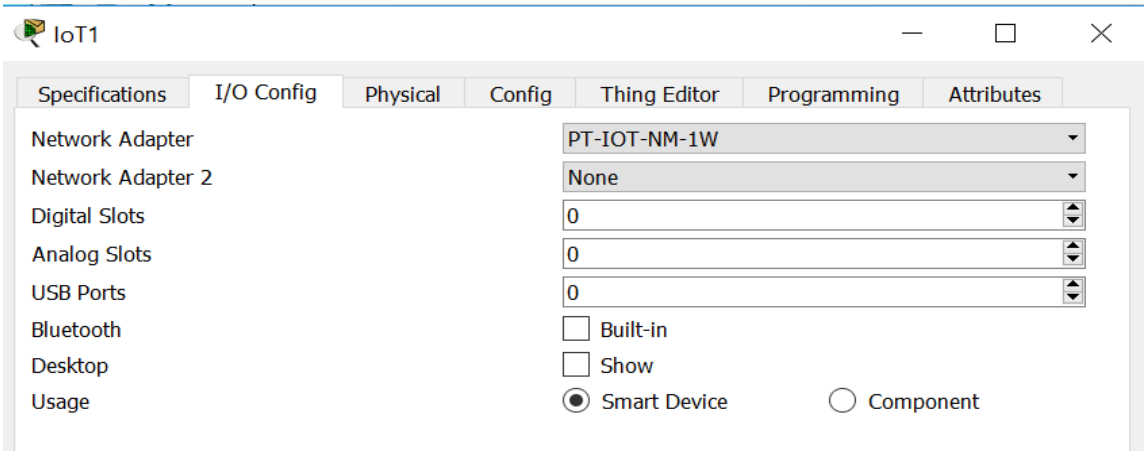
Закрийте вікно планшета. Додайте інші види пристроїв IoT до розумної домашньої мережі. Додайте бездротові пристрої IoT до Smart Home Network. У вікні Device-specific Selection натисніть значок Wind Detector, а потім натисніть на робочу область, де ви хочете розмістити детектор вітру Спеціальний вибір пристрою.



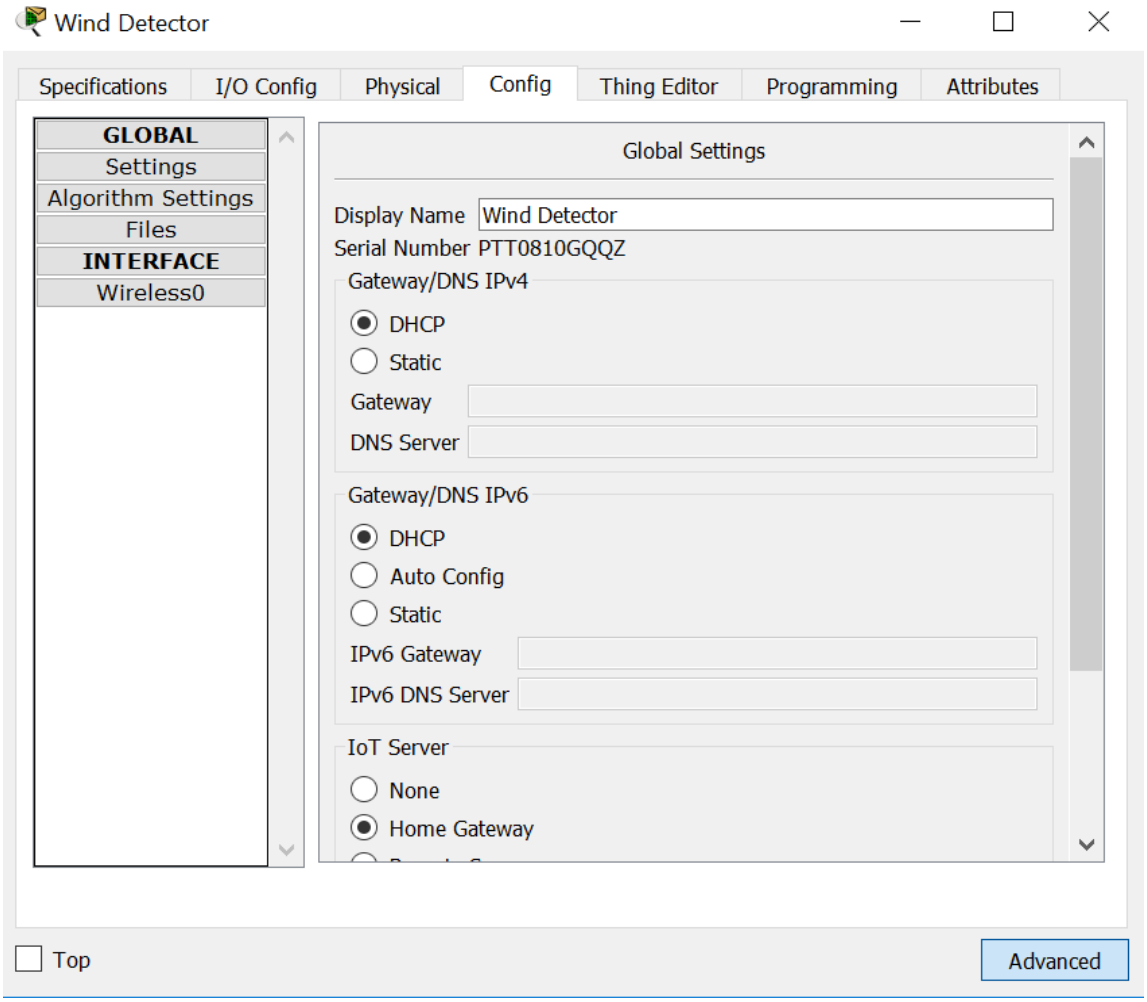
Додайте бездротовий модуль до детектора вітру. Натисніть на Wind Detector у робочому середовищі, щоб відкрити вікно пристрою IoT.



Знизу у правому куті властивостей пристрою IoT оберіть пункт меню Додатково. Натисніть вкладку введення / виведення Config.

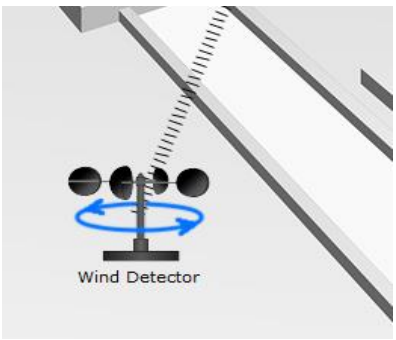


Змініть випадаючий список Мережевий адаптер на PT-IOT-NM-1W. Натисніть вкладку Config. Змініть відображуване ім’я на Wind_Detector і змініть сервер IoT на Home Gateway.



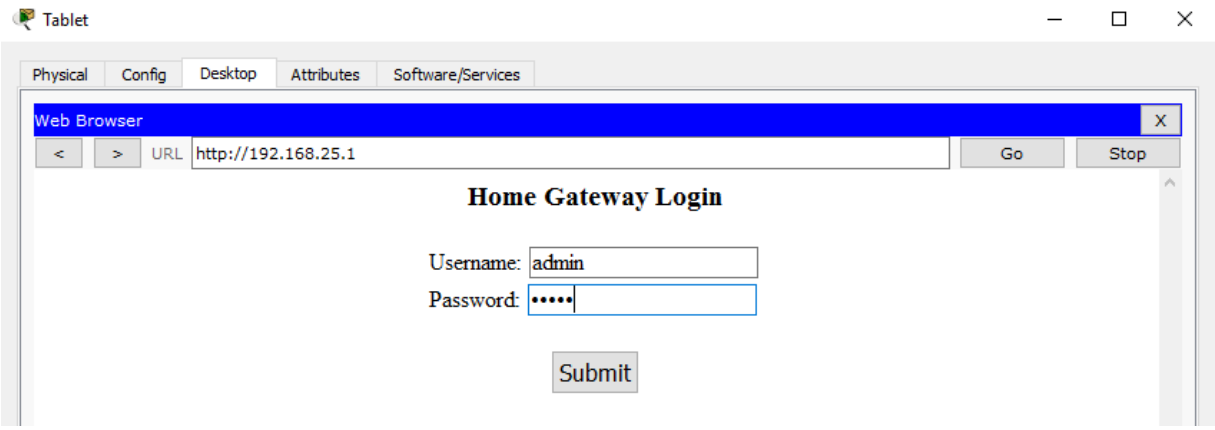
Оберіть Wireless0, встановіть тип аутентифікації WPA2-PSK і в поле PSC Pass Phrase mySecretKey для налаштування бездротової мережі домашнього шлюзу.

Бездротовий зв'язок повинен бути сформований між детектором вітру та домашнім шлюзом.

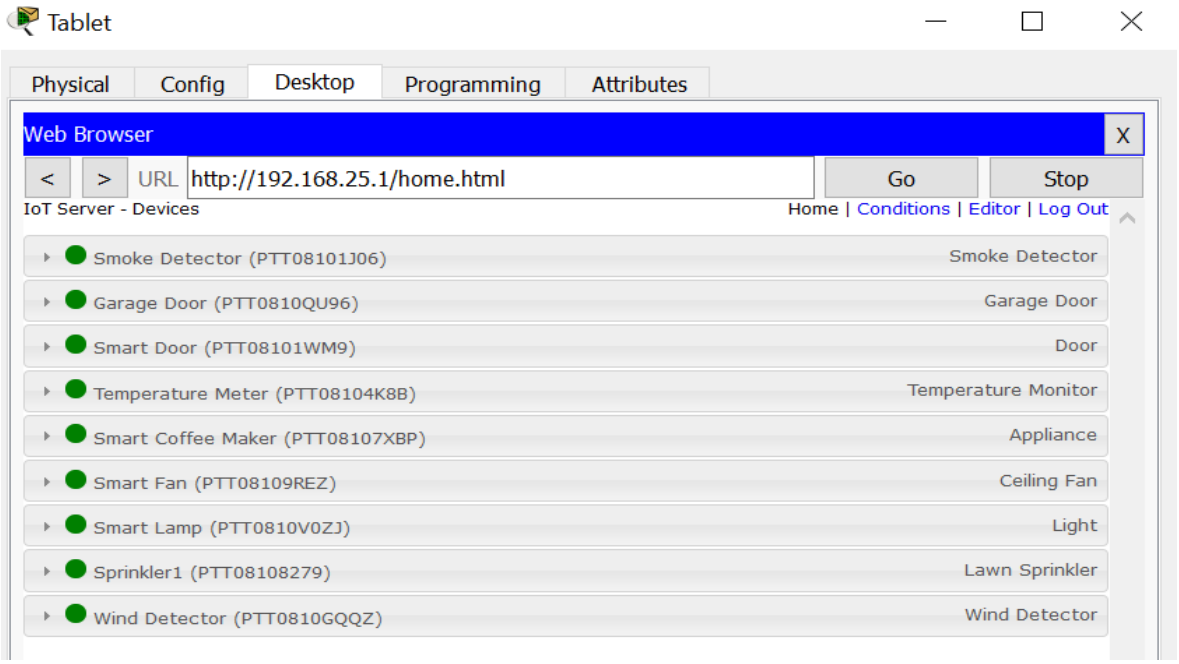


Перевірте детектор вітру в мережі.

Увійдіть у домашній браузер з планшета.



Детектор вітру тепер повинен з'явитися в списку IoT Server - Devices.



Закрийте вікно планшета. Додайте інші типи пристроїв IoT до розумної домашньої бездротової мережі.

Питання для самоперевірки

1. Що таке Інтернет речей та «розумні пристрої»?
2. Які можливості містить середовище моделювання Packet Tracer для проектування IoT?
3. Які основні принципи налаштування мережі IoT?
4. Як відбувається підключення пристроїв IoT до вебінтерфейсу домашнього шлюзу?
5. Як можна змінити конфігурацію пристроїв IoT в мережі?

ЛАБОРАТОРНА РОБОТА 2.
МОНІТОРИНГ ПРИСТРОЇВ ІОТ У МЕРЕЖІ

Мета роботи*:* під'єднати домашній шлюз до мережі, підключити пристрої IoT до бездротової мережі, додати кінцевий пристрій користувача до мережі та виконати моніторинг пристроїв IoT у мережі за допомогою Home Gateway.

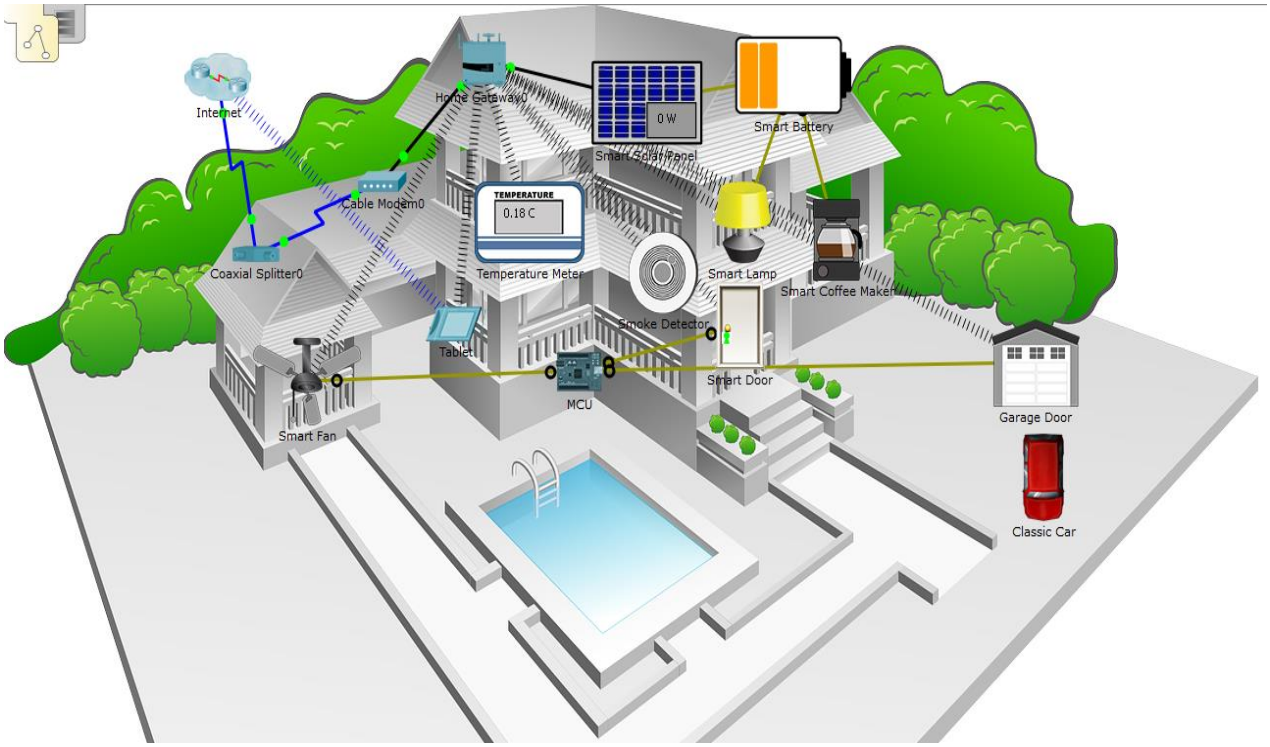
Теоретичні відомості

Інтелектуальні пристрої – це пристрої, які здатні підключатися до дротового або бездротового зв'язку мережі, налаштувати їх поведінки та взаємодії за допомогою попередньо завантаженої програми, написаної, наприклад, мовою JavaScript або Phyton. Датчики допомагають включати розумне освітлення, блоки змінного струму, сирени тривоги, зчитувачі RFID, датчики вологості, температури, рівня води тощо. Ці пристрої зазвичай є елементами типу «підключи та працюй», і їх просто потрібно підключити до локальної мережі LAN.

Після підключення до мережі смарт-пристроїв їх потрібно віддалено підключити до серверу IoT. Створення підключення IoT дає можливість користувачам перевірити стан та відстежувати список усіх підключених пристроїв IoT на домашній сторінці браузера IoT з домашнього планшета, також підключеного до локальної безпроводової мережі WLAN. При доступі до списку пристроїв також можна візуалізувати їх стан та віддалено взаємодіяти з пристроями. Наприклад, можна включити світло в саду або перевірити потужність акумулятору. Крім того, під час підключення до головної домашньої сторінки IoT також можна налаштувати основну логіку для створення та взаємодії між пристроями. Режим симуляції Packet Tracer дозволяє перевірити в мережі протоколи та комунікації для пристроїв, підключення, протоколи маршрутизації тощо. Цей режим допомагає фізично візуалізувати та усувати неполадки будь-якої мережі. Режим симуляції також відслідковує в режимі реального часу різні пакети, що транслюються в мережі, дозволяючи зануритися в корисне навантаження мережевих пакетів.

Завдання до лабораторної роботи 2

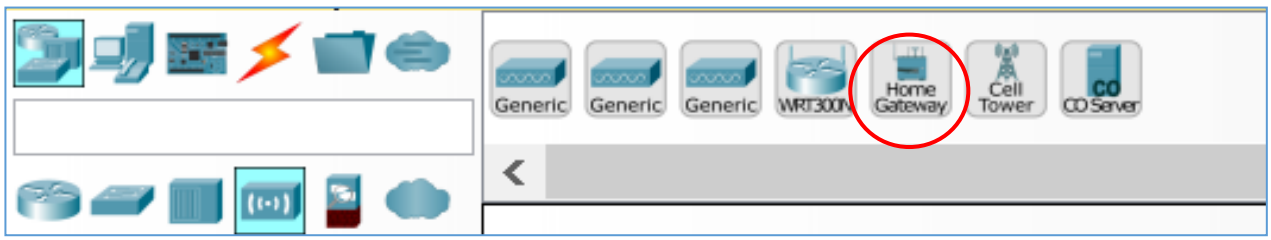
Необхідно виконати інструкції виконання лабораторної роботи з описом виконання відповідних кроків та на надати відповіді на поставлені питання.



1. Підключення домашнього шлюзу.

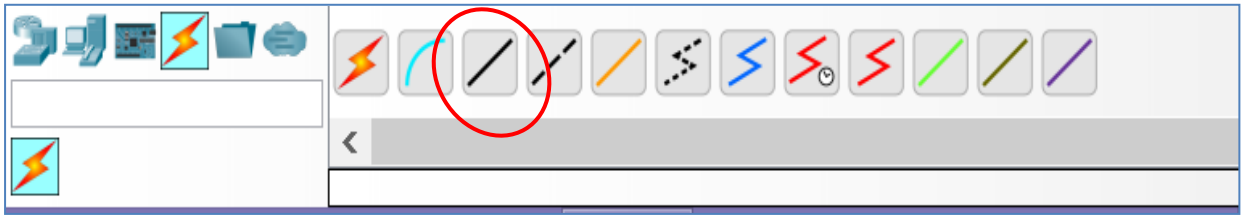
Виберіть пристрій для домашнього шлюзу.

Натисніть на іконку **Бездротові пристрої** у полі **Вибір типу пристрою**. Натисніть на іконку пристрою **Home Gateway**, потім натисніть на робочу область, щоб додати пристрій.

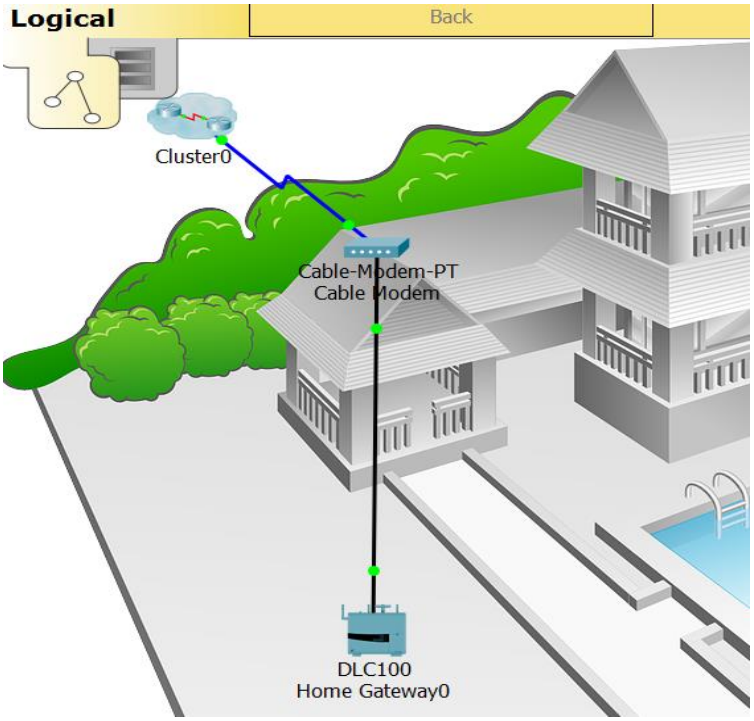


Підключіть домашній шлюз до кабельного модема.

Натисніть на іконку з'єднувача **Copper Straight-Through** у полі Тип вибору пристрою, а потім натисніть Home Gateway. Оберіть кабельний модем, щоб з'єднати інший кінець кабелю з портом **Інтернет**.

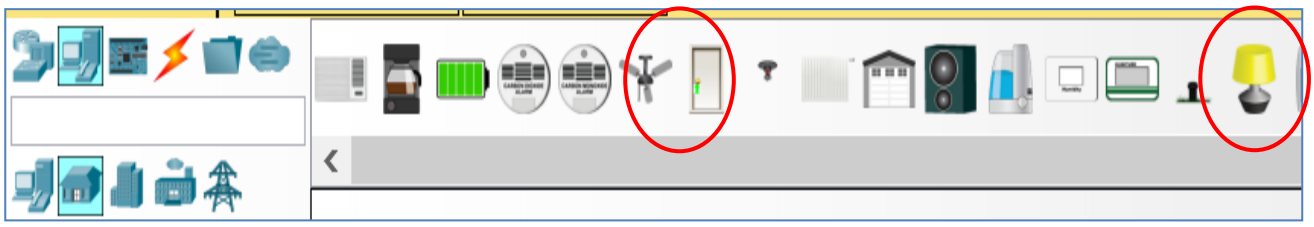


Кінці кабелю повинні замиготіти зеленим кольором, це вкаже на те, що зв'язок є.



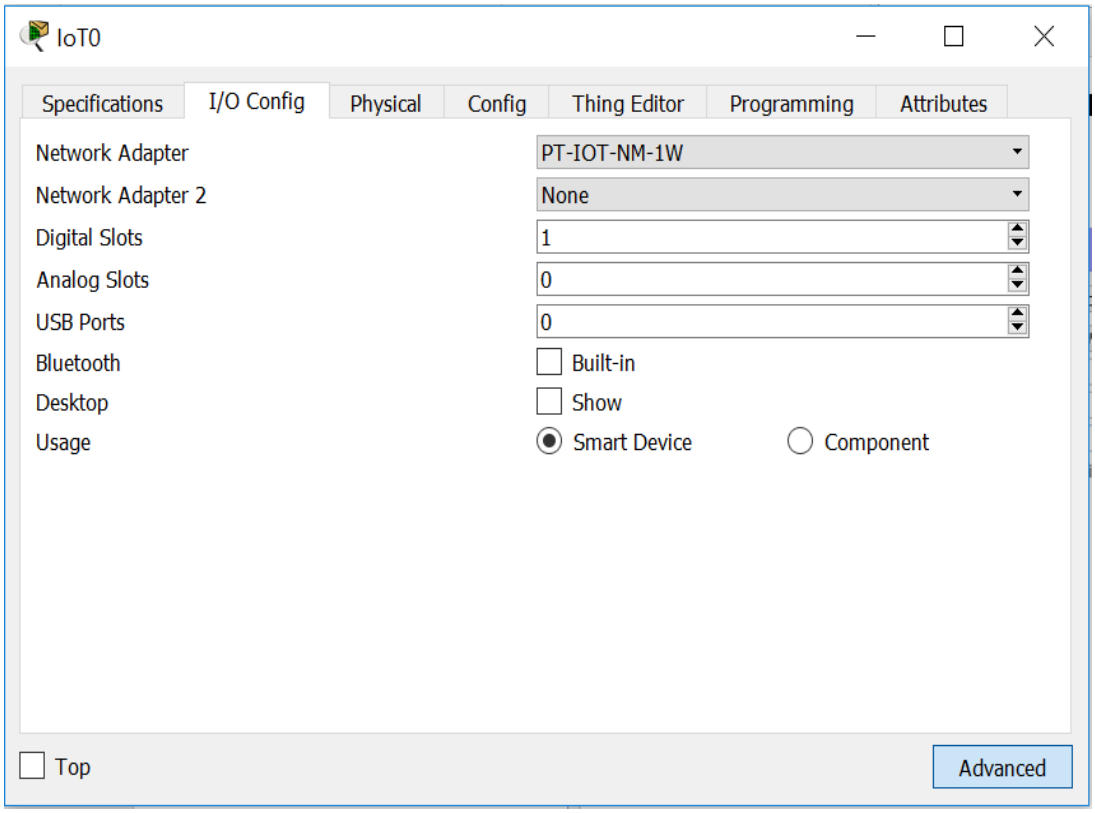
2. Підключення пристроїв IoT до бездротової мережі

Виберіть бездротові пристрої. Натисніть на іконку **Домашні пристрої** у вікні **Вибір пристрою** та додайте **вентилятор (Fan), двері (Door)** та **лампу (Lamp)** на робочу область.



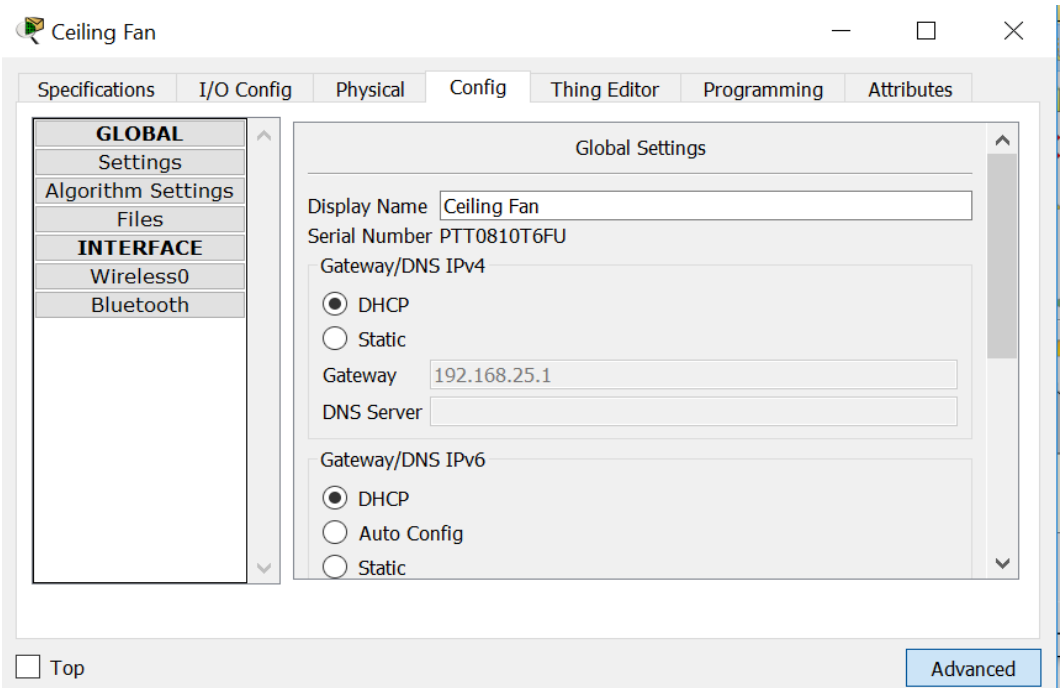
Підключіть пристрої до домашньої бездротової мережі. Додайте бездротовий адаптер до пристрою вентилятора.

Натисніть значок «Вентилятор» у робочій області, щоб відкрити вкладку «Конфігурація», а потім натисніть кнопку «Додатково». Натисніть вкладку **Вхід / вихід Config** та оберіть **PT-IOT-NM-1W**.



Змініть ім'я пристрою вентилятора **Fan**. Натисніть вкладку **Config**.

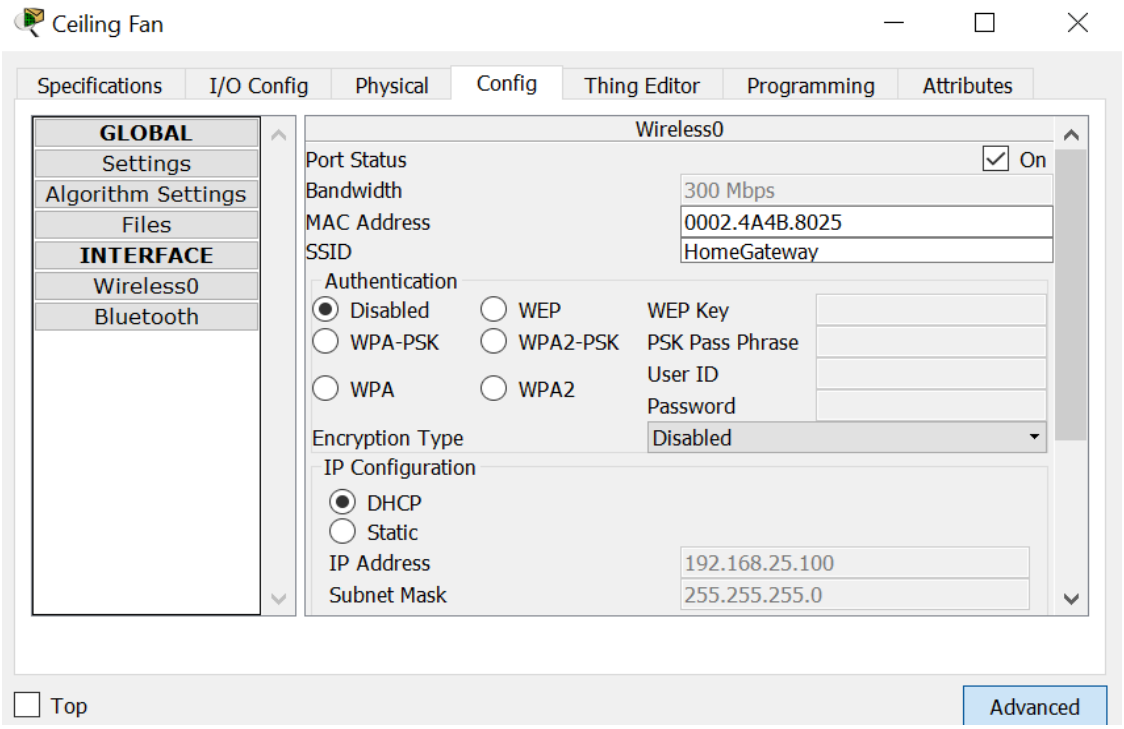
У полі імені введіть **Fan Ceiling**.



Переконайтеся, що вентилятор підключено до бездротової мережі.

Перебуваючи на вкладці **Config**, натисніть на **Wireless0** на лівій панелі.

У налаштуваннях мережа **HomeGateway** має бути вказана в полі SSID. Перевірте налаштування для підключення вентилятора до мережі, а саме, динамічно конфігурованої IP-адреси (192.168.25.100) та default gateway (192.168.25.1).

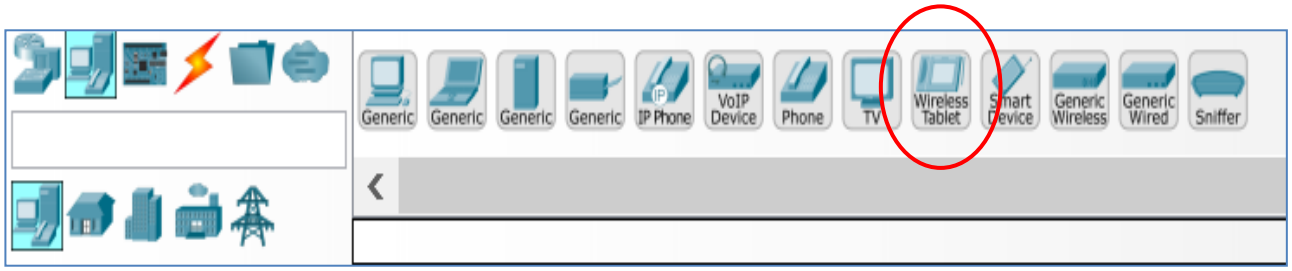


Закрийте вікно конфігурації вентилятора.

Підключіть двері та лампу до безпроводової мережі, виконуючи ті самі дії, які використовуються для вентилятора.

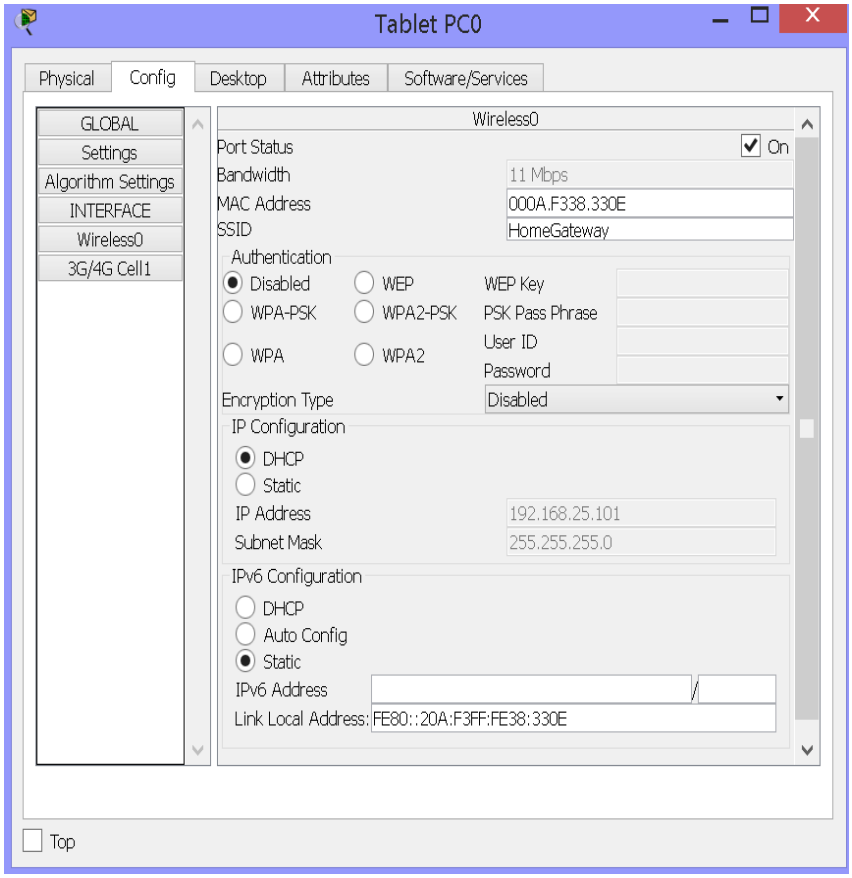
3. Підключення бездротового планшету до мережі.

Підключіть бездротовий планшет до робочої області. Натисніть на **End Devices** у вікні вибору пристрою та додайте **Wireless Tablet** до робочої області.



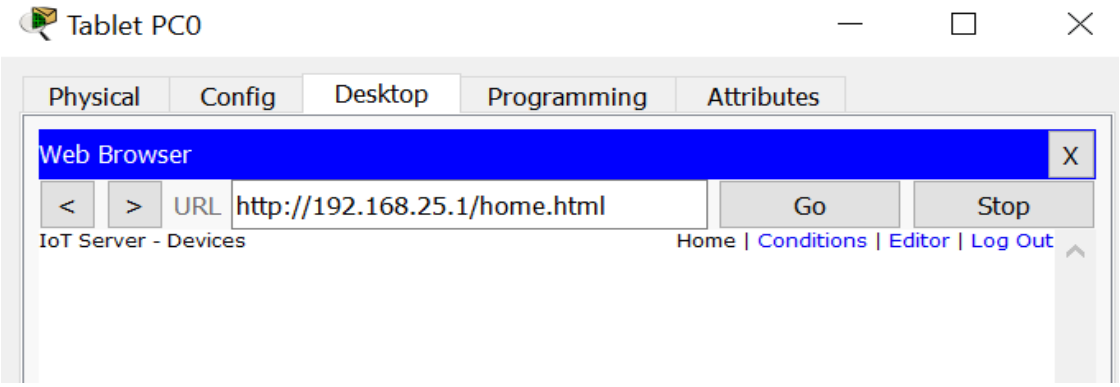
Підключіть бездротовий планшет до мережі **HomeGateway.** Змініть налаштування мережі бездротового планшета. Натисніть піктограму планшета (**Tablet**), щоб відкрити вікно конфігурації планшета.

Натисніть вкладку **Config**, потім натисніть на **Wireless0**. Змініть ідентифікатор SSID від **Default** на **HomeGateway**. Після змін мережевого SSID планшет повинен читати IP-адресу через DHCP протягом декількох секунд.



Отримайте доступ до сервера IoT домашнього шлюза з планшета.

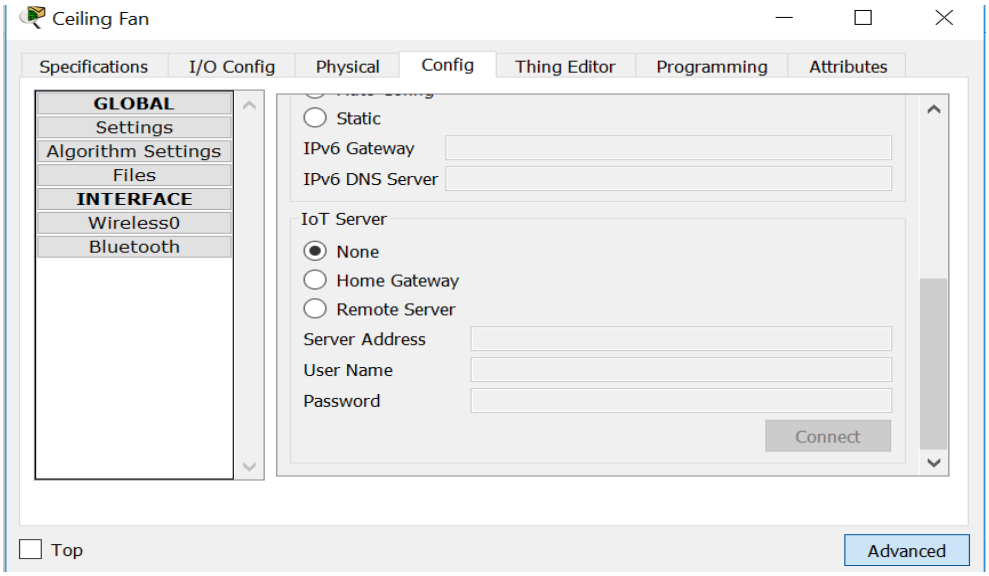
Оберіть в меню **Desktop**, а потім **Web Browser**, щоб відкрити веббраузер, 192.168.25.1 (default gateway) в полі URL-адреси, натисніть **Go**. У **Home Gateway Login** введіть **admin** як ім'я користувача та **admin** як пароль, оберіть **Надіслати**, щоб підключитися до сервера Home Gateway. Поки у Home Gateway IoT Server, Devices list не з'являються жодні пристрої.



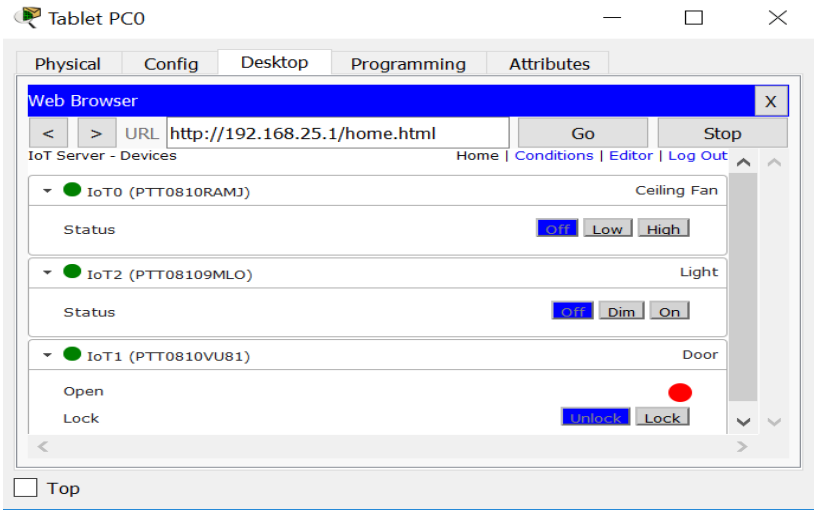
Закрийте вікно **Tablet**.

Налаштування пристроїв IoT для реєстрації на сервері домашнього шлюзу

Зареєструйте вентилятор до сервера домашнього шлюзу. Натисніть на **Fan** у робочому середовищі, потім на вкладку **Config**, і на **Налаштування** на лівій панелі. У списку параметрів **IoT Server** натисніть **Home Gateway**.



Закрийте вікно **Ceiling Fan**. Повторіть кроки розділу 3, щоб зареєструвати пристрої **Door** та **Lamp** до сервера. Переконайтеся, що пристрої зареєстровані на сервері домашнього шлюзу. Натисніть на **Tablet** у робочому середовищі та відкрийте **Web Browser**. Під'єднайтеся до домашнього шлюзу, наберіть **192.168.25.1** у полі URL-адреси, натисніть **Go**. Введіть ім'я користувача та пароль **admin**, натисніть **Submit**. Через кілька секунд всі три пристрої повинні бути перелічені в списку Home Gateway **IoT Server - Devices**. Закрийте **Tablet**.



Питання для самоперевірки

1. Як відбувається підключення домашнього шлюзу до мережі IoT?
2. Як відбувається підключення пристроїв IoT до бездротової мережі?
3. Як налаштовуються розумні пристрої для реєстрації на сервері Home Gateway?
4. Як відбувається моніторинг пристроїв IoT у мережі?

ЛАБОРАТОРНА РОБОТА 3.
АНАЛІЗ ДАНИХ ДАТЧИКІВ. ТУМАННІ ОБЧИСЛЕННЯ В РОЗУМНОМУ БУДИНКУ

Мета роботи: дослідити розумний будинок, використати туманні обчислення в розумному будинку для системи моніторингу впливу рівня диму, виявленого у будинку.

Теоретичні відомості

Хмарні обчислення (*Cloud computing, CC*) – це доставка IT-ресурсів на вимогу користувача через мережу Інтернет. Користувач може отримати доступ до технологічних послуг, таких як обчислювальна потужність, сховище та бази даних за потреби від хмарного постачальника, наприклад, Amazon Web Services (AWS).

Основними гравцями на ринку CC і супутніх продуктів є IBM, HP, Intel (Intel IoT Solution Alliance), Microsoft (Azure, NET, Node.js, Java, PHP), Google (Google App Engine – Python, Java, Go), Amazon (Elastic Cloud Compute – EC2, AWS, Simple Storage Service – S3), Cisco, Kaa IoT, ThingWorx, ThingSpeak та багато інших завдяки наявності спеціалізованих фреймворків IoT для швидкого створення вебдодатків та їх розгортання в хмарних інфраструктурах. Використовуючи такі структури, розробник додатків IoT зможе використовувати потужність CC, не знаючи низькорівневих деталей створення надійних і масштабованих додатків.

Хмара використовується для резервного копіювання даних, аварійного відновлення, електронної пошти, розробки та тестування програмного забезпечення, аналітики великих даних, аналізу IoT даних.

Моніторинг охорони здоров'я є однією з функцій пристроїв IoT, що передбачає збір та оцінку даних про пацієнтів протягом певного періоду часу.

Моніторингові пристрої, які носить пацієнт, підключаються до Інтернету (мережу датчиків тіла BSN, body sensor network). Шлюз поєднує сигнали від датчиків і безпечно передає дані в хмару (рис. 3.1).

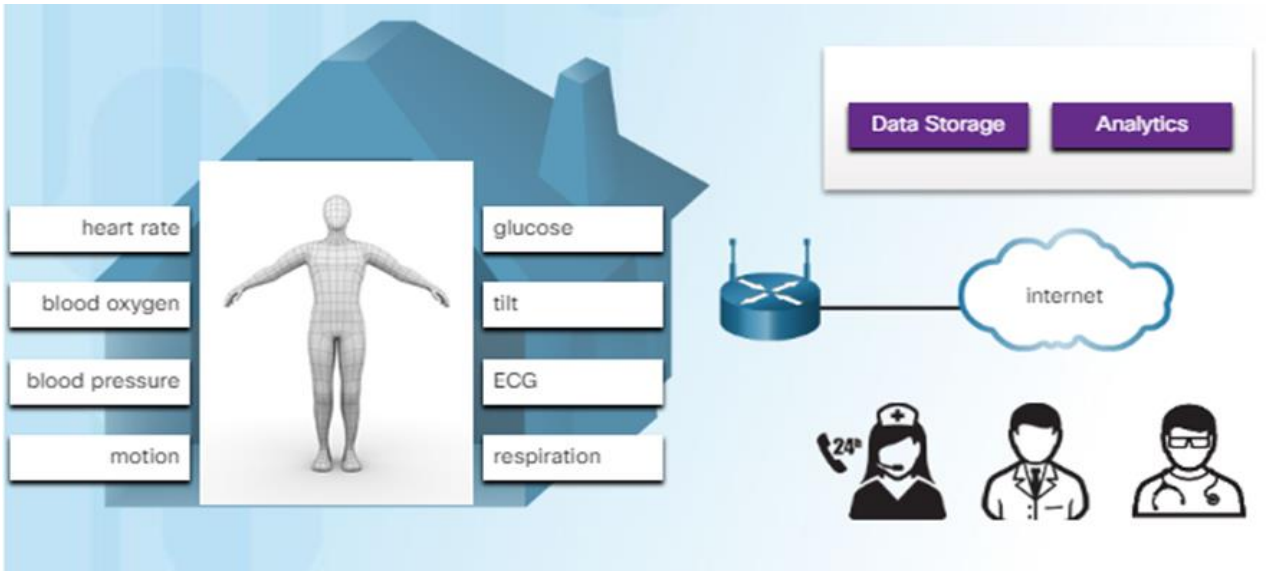
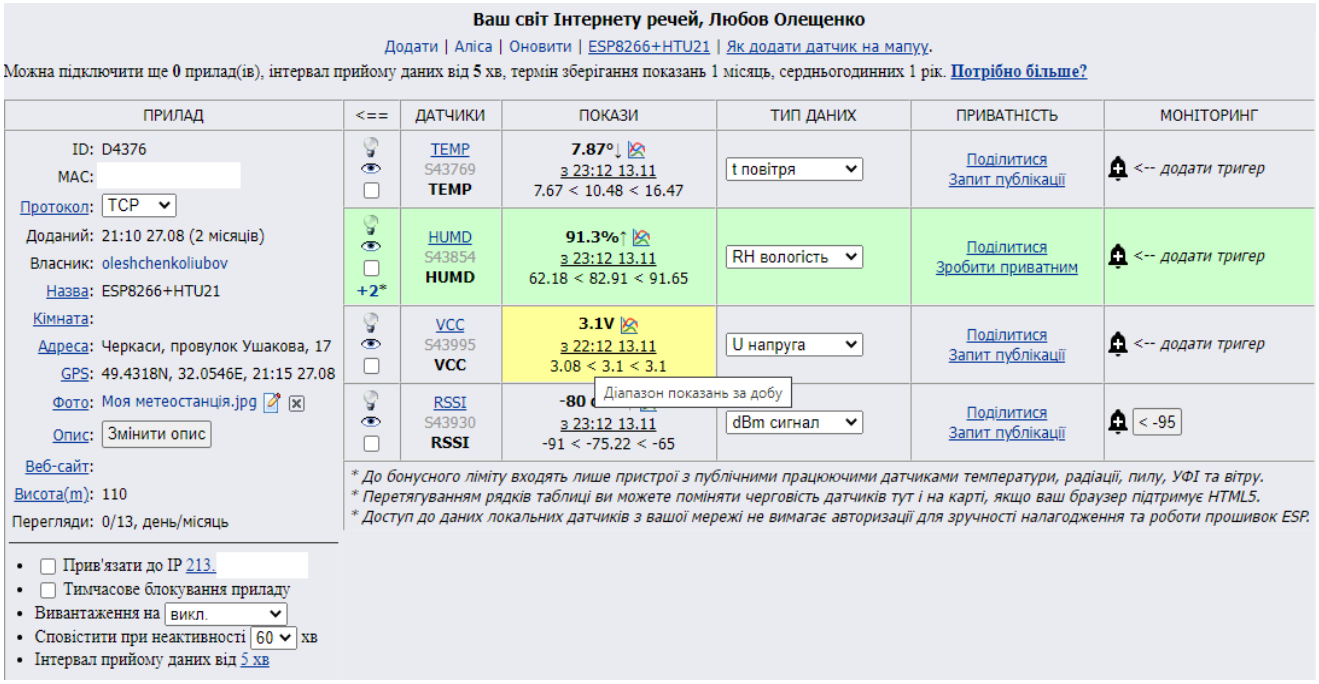


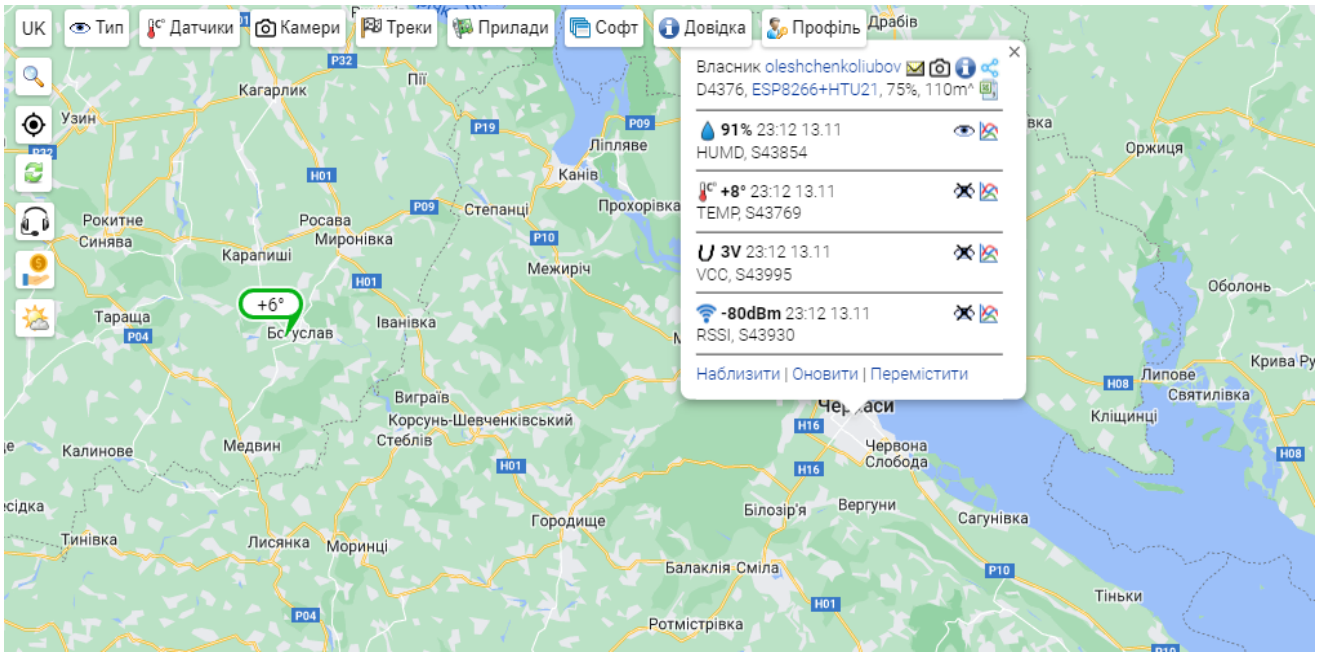
Рис. 3.1. Система RPM (Remote Patient Monitoring)

IoT надає різні функції підключеним медичним пристроям. Датчики можна використовувати для відстеження місцезнаходження пристроїв IoT і моніторингу роботи пристроїв, щоб виявити проблеми та запобігти збоям. У

медичних пристроях використовуються приводи, якими керує програмне забезпечення, щоб регулювати введення ліків, рідин і кисню.

Розглянемо приклад використання хмарних обчислень для аналізу даних датчика на базі ESP8266+HTU21, який був встановлений у серпні 2022 році у місті Черкаси (рис. 3.2). Даний датчик призначений для вимірювання температури та вологості повітря, крім того, він фіксує силу сигналу Wi-Fi мережі та рівень напруги елемента живлення [3].





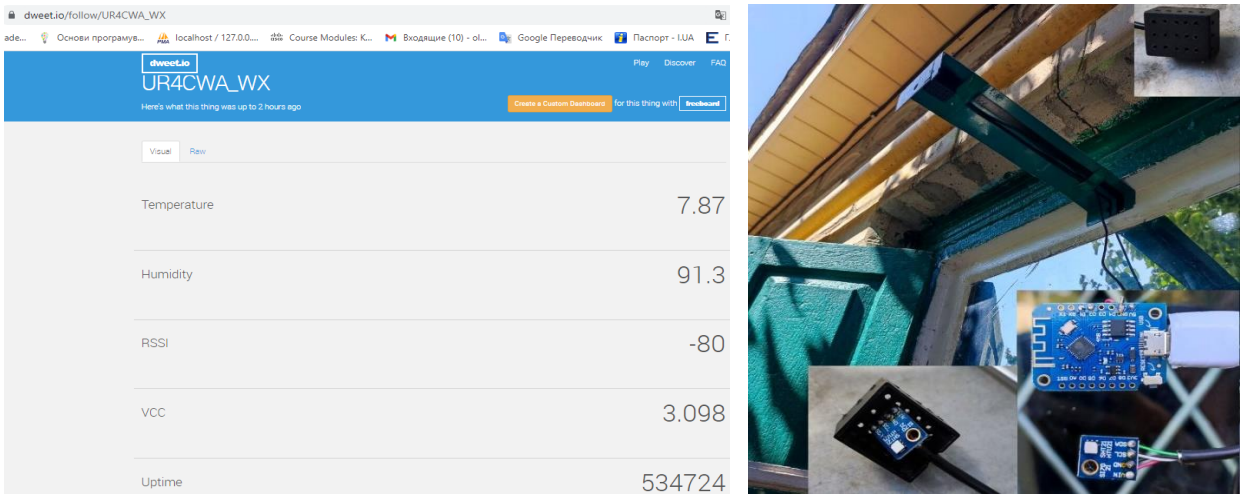


Рис. 3.2. Встановлення та підключення до мережі Інтернет датчика на базі ESP8266+HTU21 у м. Черкаси (серпень, 2022 рік)

Розглянемо технічні характеристики системи IoT на базі ESP8266+HTU21. Основою IoT пристрою є мікроконтролер ESP8266 (рис. 3.3).

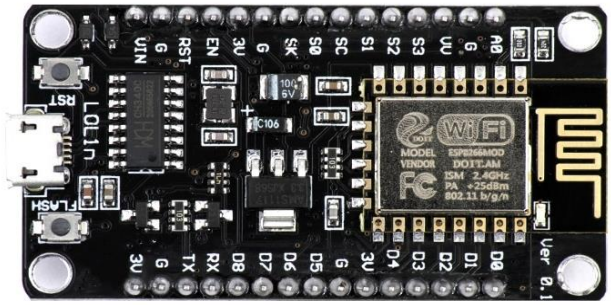


Рис. 3.3. Вигляд контролера ESP8266

Розпіновка мікроконтролера ESP8266 зображена на рис. 3.4 [3].

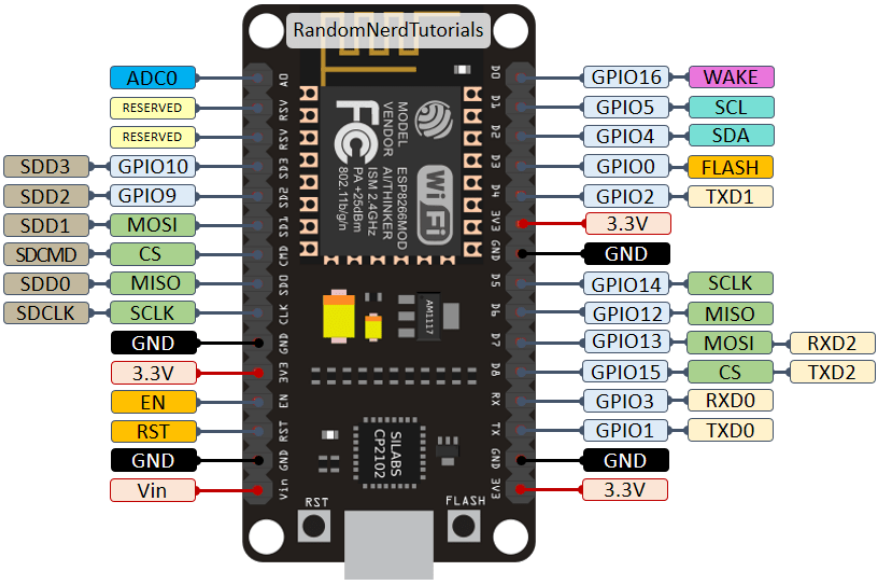


Рис. 3.4. Розпіновка контролера ESP8266

До мікроконтролера ESP8266 підключається датчик температури та вологості повітря HTU21 (рис. 3.5).

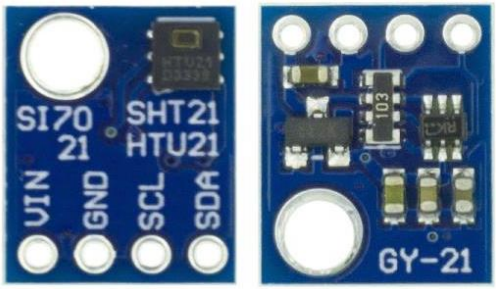


Рис. 3.5. Вигляд сенсора HTU21

Схема підключення сенсора HTU21 до мікроконтролера ESP8266 зображена на рис. 3.6.

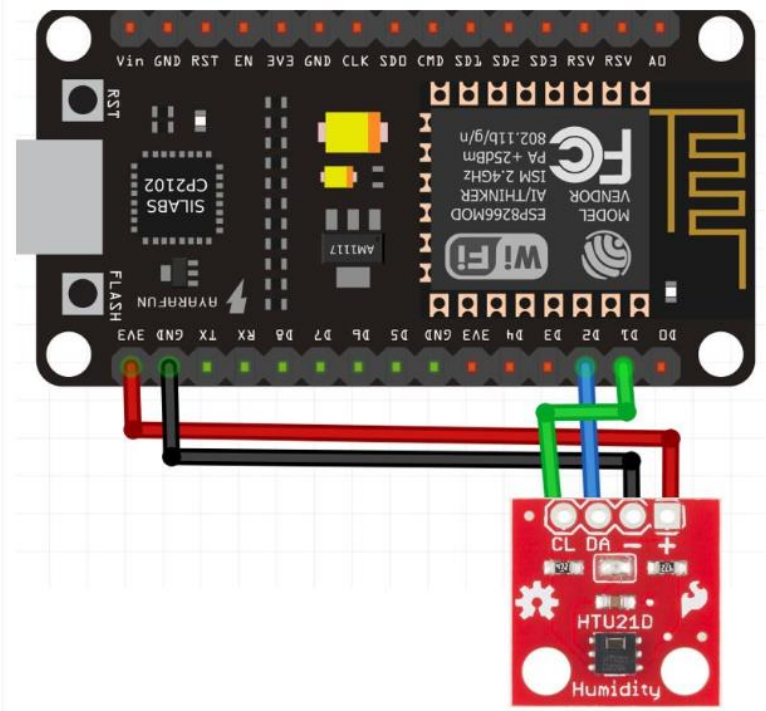


Рис. 3.6. Схема підключення ESP8266 з HTU21 [4]

Для проведення аналізу даних датчика використаємо датасет даних D4376-20220826-20221114-1H.csv, отриманих з даного датчика у період з серпня до середини листопада 2022 року. Розглянемо обсяг та основну статистичну інформацію про датасет (мінімум, максимум, середнє значення тощо), візуалізуємо дані, використовуючи можливості мови програмування Python. Для виконання цього завдання використаємо IPython ноутбук в середовищі для хмарних обчислень Google Colab.

Щоб використовувати Google Colab, розробнику програмного забезпечення не потрібно встановлювати та виконувати або оновлювати апаратне забезпечення свого комп'ютера, щоб відповідати вимогам Python до інтенсивного навантаження CPU/GPU.

Хмарний інструмент Google Colab надає безкоштовний доступ до обчислювальної інфраструктури, сховища, пам'яті, обчислювальної потужності, графічних процесорів (GPU) і тензорних процесорів (TPU). Компанія Google забезпечила програмованість цього хмарного інструменту для мови Python, враховуючи потреби програмістів з машинного навчання та аналітиків великих даних.

Користувачі Colab отримують безкоштовний доступ до GPU та TPU. GPU працює з процесором Intel Xeon @2,20 ГГц, 13 ГБ оперативної пам'яті, прискорювачем Tesla K80 і 12 ГБ відеопам'яті GDDR5. Середовище виконання TPU складається з процесора Intel Xeon @2,30 ГГц, 13 ГБ оперативної пам'яті та хмарного TPU з обчислювальною потужністю 180 терафлопс.

За допомогою Colab Pro або Pro+ можна використовувати більше процесорів, TPU та графічних процесорів на понад 12 годин. Colab містить NumPy, Pandas, Matplotlib, PyTorch, TensorFlow, Keras та інші бібліотеки машинного навчання, також дозволяє встановлювати бібліотеки, які не належать до Colaboratory (AWS S3, GCP, SQL, MySQL тощо).

Google Colab використовує квоту пам'яті на Диску Google для збереження файлів. Таким чином, можна відновити роботу з кодом з будь-якого комп'ютера, на якому є доступ до облікового запису Диска Google. Хмарне сховище також функціонує як резервна копія даних від будь-яких катастроф.

Перед тим, як завантажити наданий датасет в Google Colab, імпортуємо необхідні бібліотеки:

```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
%matplotlib inline
```

Завантажимо наданий датасет показників, які були зафіксовані датчиком вимірювання температури та вологості повітря. Оскільки файл було отримано у форматі CSV, то була використана відповідна функція для завантаження в датафрейм. Оскільки в наданому CSV файлі у якості розділювача замість коми використовувалась крапка з комою, то зазначимо в параметрах імпортування розділювач «;»:

```
pathToWeatherDataset = '/content/drive/MyDrive/data_colab/bd_course/kr/D4376-
20220826-20221114-1H.csv'
weatherSensorDf = pd.read_csv(pathToWeatherDataset, sep=';')
```

Перевіримо правильність імпортування, шляхом відображення перших п'яти записів в датафреймі:

```
weatherSensorDf.head()
```

	UNIXTIME	Дата	Час	TEMP	HUMD	RSSI	VCC
0	1661626800	27.08.2022	21:00:00	20.94	69.70	-41.00	3.1
1	1661630400	27.08.2022	22:00:00	21.51	62.06	-42.17	3.1
2	1661634000	27.08.2022	23:00:00	20.45	68.52	-43.42	3.1
3	1661637600	28.08.2022	00:00:00	20.15	69.38	-44.75	3.1
4	1661641200	28.08.2022	01:00:00	19.68	70.85	-46.45	3.1

Як бачимо, вміст CSV файлу був коректно завантажений в датафрейм. В таблиці наявні сім колонок: UNIXTIME, Дата, Час, TEMP, HUMD, RSSI, VCC. Колонки Дата та Час відповідають за дату та час взяття показника, і дублюють інформацію з колонки UNIXTIME, в якій дата та час подані в форматі UNIX timestamp.

Додамо нову колонку з назвою TIMESTAMP в датафрейм, в яку скопіюємо значення з колонки UNIXTIME та перетворимо їх на Python-тип datetime, застосувавши відповідну функцію:

```
weatherSensorDf['TIMESTAMP'] = pd.to_datetime(weatherSensorDf['UNIXTIME'], unit='s',
utc=True).dt.tz_convert('Europe/Kyiv')

weatherSensorDf
```

	UNIXTIME	Дата	Час	TEMP	HUMD	RSSI	VCC	TIMESTAMP
0	1661626800	27.08.2022	21:00:00	20.94	69.70	-41.00	3.1	2022-08-27 22:00:00+03:00
1	1661630400	27.08.2022	22:00:00	21.51	62.06	-42.17	3.1	2022-08-27 23:00:00+03:00
2	1661634000	27.08.2022	23:00:00	20.45	68.52	-43.42	3.1	2022-08-28 00:00:00+03:00
3	1661637600	28.08.2022	00:00:00	20.15	69.38	-44.75	3.1	2022-08-28 01:00:00+03:00
4	1661641200	28.08.2022	01:00:00	19.68	70.85	-46.45	3.1	2022-08-28 02:00:00+03:00
...	...	...	...	...	...	...	...	...
1387	1668358800	13.11.2022	19:00:00	8.38	88.75	-85.27	3.1	2022-11-13 19:00:00+02:00
1388	1668362400	13.11.2022	20:00:00	8.06	89.60	-80.50	3.1	2022-11-13 20:00:00+02:00
1389	1668366000	13.11.2022	21:00:00	8.05	90.21	-81.33	3.1	2022-11-13 21:00:00+02:00
1390	1668369600	13.11.2022	22:00:00	7.75	91.43	-79.42	3.1	2022-11-13 22:00:00+02:00
1391	1668373200	13.11.2022	23:00:00	7.91	91.40	-80.67	3.1	2022-11-13 23:00:00+02:00

1392 rows × 8 columns

Як бачимо, в новій колонці TIMESTAMP в датафреймі тепер міститься коректна інформація про дату і час фіксування показників сенсора в читабельному форматі.

Тепер колонки UNIXTIME, Дата, Час не потрібні, оскільки дублюють наявну інформацію в колонці TIMESTAMP. Тому видалимо ці колонки UNIXTIME, Дата, Час з датафрейму:

```
weatherSensorDf.drop(['UNIXTIME', 'Дата', 'Час'], axis=1, inplace=True)

weatherSensorDf.head()
```

	TEMP	HUMD	RSSI	VCC	TIMESTAMP
0	20.94	69.70	-41.00	3.1	2022-08-27 22:00:00+03:00
1	21.51	62.06	-42.17	3.1	2022-08-27 23:00:00+03:00
2	20.45	68.52	-43.42	3.1	2022-08-28 00:00:00+03:00
3	20.15	69.38	-44.75	3.1	2022-08-28 01:00:00+03:00
4	19.68	70.85	-46.45	3.1	2022-08-28 02:00:00+03:00

Отже, отримано п'ять колонок в датафреймі. Колонка TEMP позначає температуру на вулиці в градусах Цельсія, колонка HUMD відображає відносну вологість повітря у відсотках.

Колонка RSSI є показником рівня потужності Wi-Fi сигналу (через який і надсилаються зафіксовані дані), що надходить до антени датчика, колонка VCC відображає у Вольтах напругу батарейок, які живлять цей датчик, колонка TIMESTAMP відображає дату та час фіксування показників датчиком.

Дізнаємось, скільки всього записів вимірювань датчику наявно в датафреймі:

```
len(weatherSensorDf)
```

```
1392
```

Отже, в датасеті міститься 1392 записів. Тобто обсяг датасету становить 1392 рядків, а кількість колонок становить 5.

Перевіримо типи даних кожної колонки в датасеті та переконаємось, що немає пропущених значень в рядках:

```
weatherSensorDf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1392 entries, 0 to 1391
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   TEMP       1392 non-null   float64
 1   HUMD       1392 non-null   float64
 2   RSSI       1392 non-null   float64
 3   VCC        1392 non-null   float64
 4   TIMESTAMP  1392 non-null   datetime64[ns, Europe/Kyiv]
dtypes: datetime64[ns, Europe/Kyiv](1), float64(4)
memory usage: 54.5 KB
```

Отже, пропущених значень в таблиці немає. Тому не потрібно фільтрувати датафрейм для виключення порожніх рядків.

Як бачимо, тип даних для колонки TIMESTAMP становить datetime, тобто конвертування було здійснено правильно.

Також дізнаємось основну статистичну інформацію: мінімум, максимум, середнє значення тощо:

```
weatherSensorDf.describe()
```

	TEMP	HUMD	RSSI	VCC
count	1392.000000	1392.000000	1392.000000	1392.000000
mean	13.461602	75.639325	-72.403247	3.097399
std	5.641691	17.412619	5.452205	0.004884
min	1.410000	23.540000	-87.830000	3.070000
25%	9.607500	64.250000	-75.250000	3.100000
50%	12.960000	79.600000	-72.500000	3.100000
75%	16.572500	90.172500	-70.080000	3.100000
max	37.370000	101.180000	-41.000000	3.100000

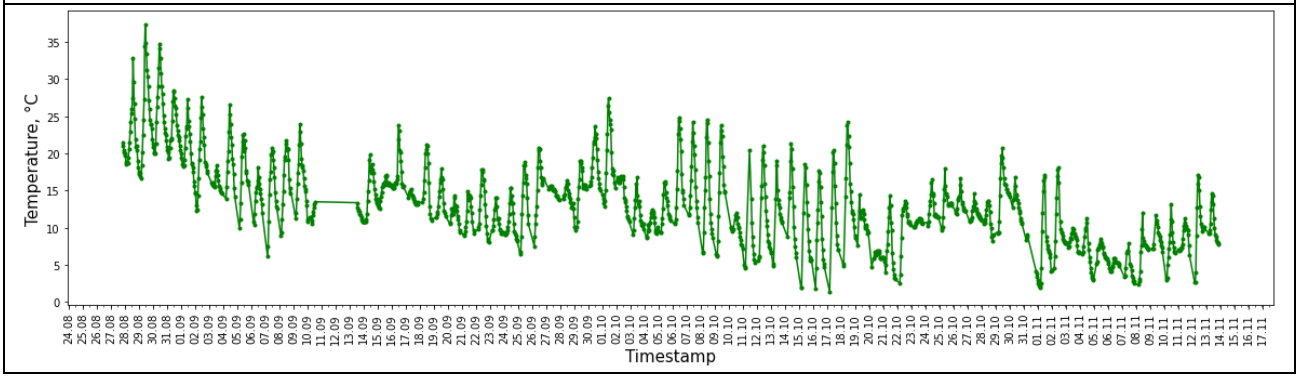
Отже бачимо, що максимальне значення температури становить 37.37 °C, а мінімальне – 1.41 °C. Середнє значення – 13.462 °C. Таку саму інформацію можна переглянути і про інші колонки. Окрім цього, у вищенаведеній статистичній інформації можна побачити значення середньоквадратичного відхилення, перцентилів рівня 25%, 50%, 75%.

Візуалізуємо дані сенсору, використовуючи можливості мови Python. Для цього побудуємо графік залежності всіх чотирьох вимірюваних величин (TEMP, HUMD, RSSI, VCC) від дати і часу їх фіксування. Для цього створимо окрему функцію, яка створюватиме графік з відповідними підписами:

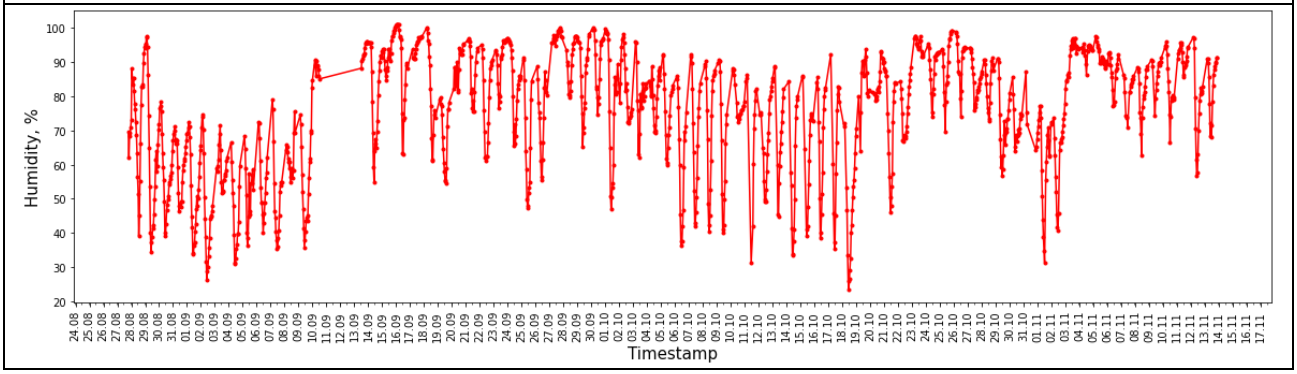
```
def plot_weather_sensor_data(column_name: str, column_label: str, color: str):
  fig, ax = plt.subplots(figsize=(20, 5))

  ax.plot(weatherSensorDf["TIMESTAMP"], weatherSensorDf[column_name], f'{color}o-',
markersize=3)
  plt.xticks(rotation=90)
  ax.xaxis.set_major_locator(mdates.DayLocator())
  ax.xaxis.set_minor_locator(mdates.HourLocator(interval=12))
  ax.xaxis.set_major_formatter(mdates.DateFormatter('%d.%m'))
  plt.ylabel(column_label, fontsize=15)
  plt.xlabel('Timestamp', fontsize=15)
  plt.show()
```

Маємо наступні чотири графіки, які візуалізують залежність температури (°C), відносної вологості (%), рівня потужності Wi-Fi сигналу (дБм), напруги акумулятора (В) від часу відповідно:

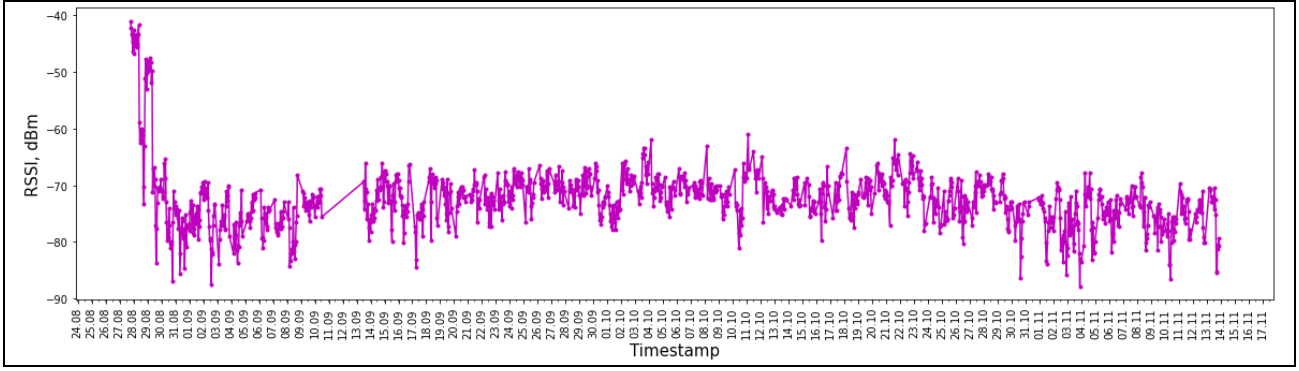
```
plot_weather_sensor_data("TEMP", "Temperature, °C", "g")
```



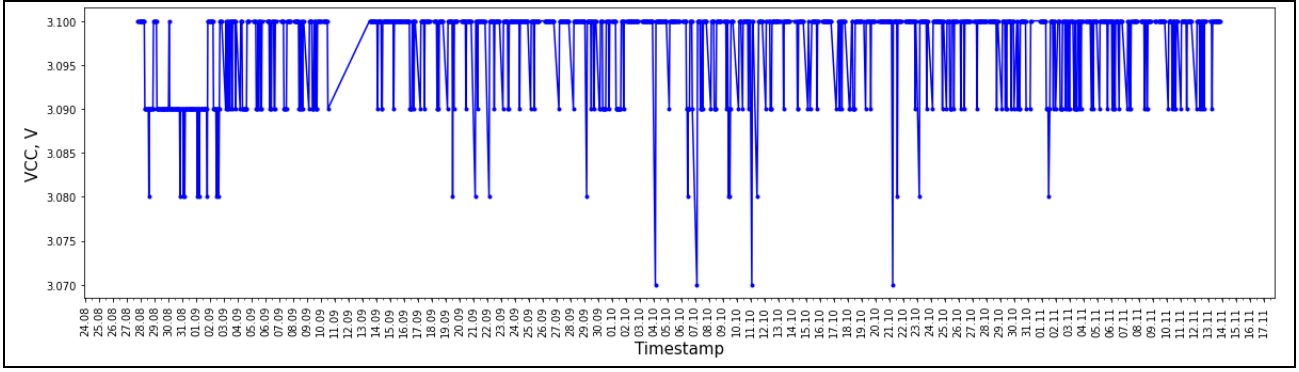

```
plot_weather_sensor_data("HUMD", "Humidity, %", "r")
```




```
plot_weather_sensor_data("RSSI", "RSSI, dBm", "m")
```




```
plot_weather_sensor_data("VCC", "VCC, V", "b")
```



Як бачимо вище на графіках, був один великий період (приблизно з 10 по 13 вересня 2022 року), в який була відсутня електроенергія, внаслідок чого сенсор не зміг зафіксувати відповідні показники в цей час та відправити їх через мережу Wi-Fi.

Знайдемо кореляцію між параметрами температури та вологості. Для виконання цього завдання знову використаємо IPython ноутбук в середовищі Google Colab. Для побудови теплової карти імпортуємо відповідну бібліотеку:

```
import seaborn as sns
```

Оберемо проміжок часу від 27.08.22 до 13.11.22.

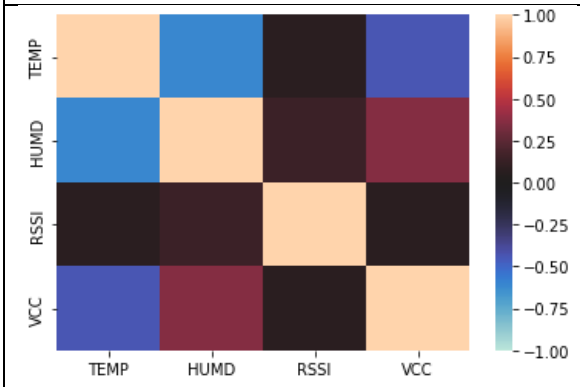
Побудуємо матрицю кореляції для параметрів: TEMP, HUMD, RSSI, VCC. Для визначення значення кореляції використаємо метод Пірсона:

```
weatherSensorDfCorr = weatherSensorDf.corr(method='pearson')
weatherSensorDfCorr
```

	TEMP	HUMD	RSSI	VCC
TEMP	1.000000	-0.601915	0.073954	-0.432898
HUMD	-0.601915	1.000000	0.144917	0.364643
RSSI	0.073954	0.144917	1.000000	0.075421
VCC	-0.432898	0.364643	0.075421	1.000000

Отримавши матрицю кореляції, побудуємо теплову карту для неї, щоб наочніше побачити, між якими змінними є кореляція. Чим світліше колір - тим більше дві змінні корелюють, а чим темніший колір – відповідно, навпаки, тобто менше корелюють або зовсім не корелюють:

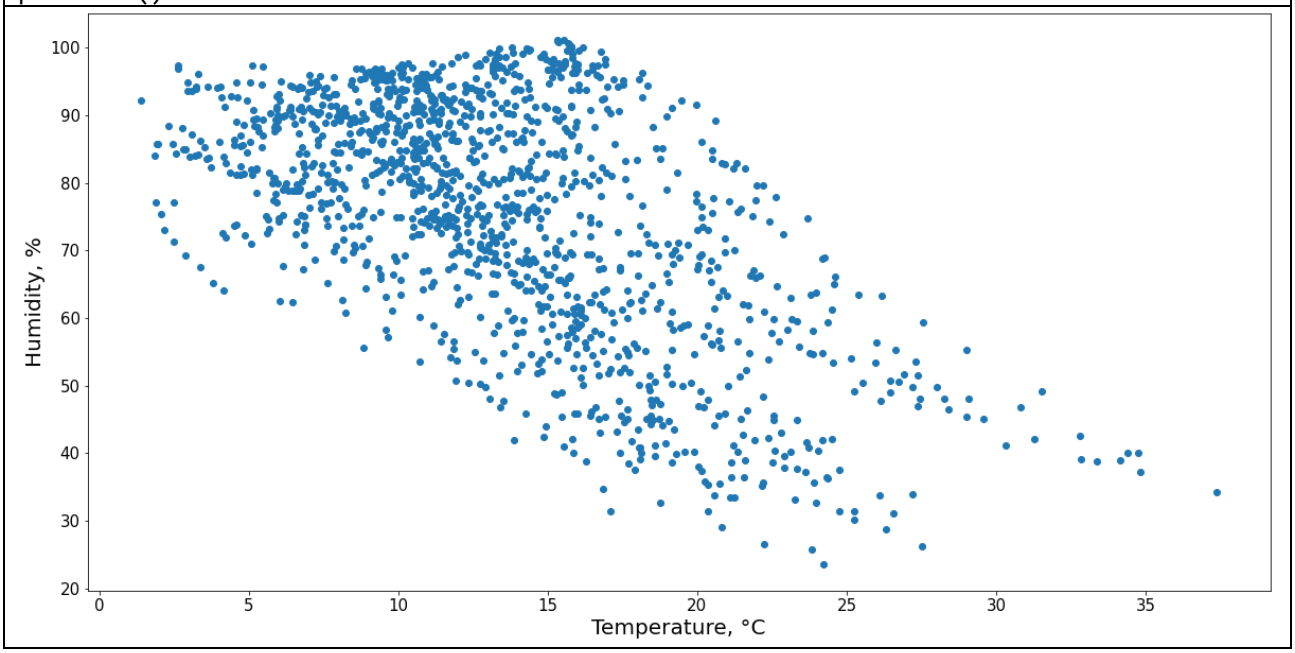
```
weatherSensorDfCorr = weatherSensorDf.corr(method='pearson')
weatherSensorDfCorr
```



Дослідимо кореляцію між параметрами температури та вологості, тому розглядатимемо саме ці два параметри. Можна побачити, що параметри температури та вологості мають кореляцію зі значенням –0.602, тобто це є помірна негативна (від'ємна) кореляція. З цього виходить, що зі збільшенням температури іноді може зменшуватись відносна вологість повітря. В цьому є певна логіка, адже зі збільшенням температури вода активніше випаровується, тому повітря є більш сухішим. Також отримана помірна кореляція може пояснюватись тим, що восени, коли температура зменшилась, було багато дощів, внаслідок чого вологість була вищою.

Побудуємо точкову діаграму залежності відносної вологості повітря від температури повітря:

```
y = weatherSensorDf["HUMD"]
x = weatherSensorDf["TEMP"]

plt.figure(figsize=(20,10))

plt.scatter(x, y)

plt.ylabel('Humidity, %', fontsize=20)
plt.xlabel('Temperature, °C', fontsize=20)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)

plt.show()
```



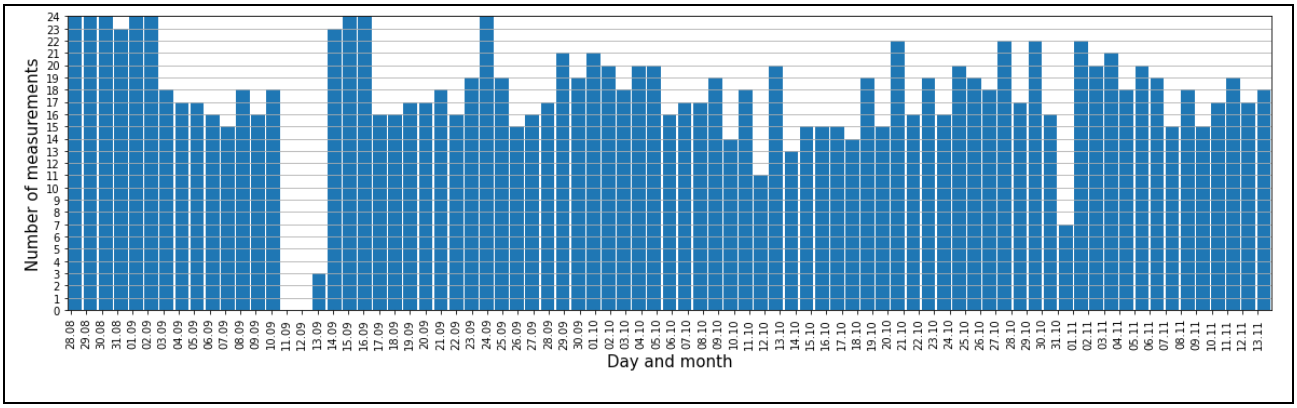
З точкової діаграми залежності відносної вологості повітря від температури повітря можна побачити, що дійсно при вищій температурі повітря вологість повітря частіше всього має менше значення.

Знайдемо, коли відбувались аварійні відключення датчика. Для виконання цього завдання знову використаємо IPython ноутбук в середовищі Google Colab.

Як можна було побачити вище, датчик в нормальному режимі відправляє показники раз в годину. Тому для того, щоб визначити аварійні відключення, необхідно знайти найбільші проміжки в часі між фіксуваннями показників датчика. Так як зафіксовані показники датчика збережені в хронологічному порядку, то достатньо буде порівявати час фіксування в кожній парі збережених показників в датафреймі.

Для наочного представлення побудуємо гістограму, де для кожного дня відобразимо, скільки всього було зафіксовано показників. Якщо в один день було зафіксовано 24 показники, це означає, що датчик працював цілодобово і аварійних відключень не було:

```
startTimestamp = weatherSensorDf["TIMESTAMP"].min()
endTimestamp = weatherSensorDf["TIMESTAMP"].max()
countDays = (endTimestamp - startTimestamp).days + 1

fig, ax = plt.subplots(figsize=(20, 5))

n, bins, patches = ax.hist(weatherSensorDf["TIMESTAMP"], countDays, rwidth=0.9)

plt.xticks(rotation=90)
ax.yaxis.set_major_locator(ticker.MultipleLocator(1))
ax.xaxis.set_major_locator(mdates.DayLocator())
ax.xaxis.set_major_formatter(mdates.DateFormatter('%d.%m'))
ax.yaxis.grid()
plt.ylabel('Number of measurements', fontsize=15)
plt.xlabel('Day and month', fontsize=15)
plt.ylim(0, 24)
plt.xlim(startTimestamp, endTimestamp)

plt.show()
```



Як бачимо з гістограми вище, тільки напочатку вимірювань, в серпні, датчик працював абсолютно цілодобово. Починаючи з вересня, можна побачити, що в середньому по 4-5 годин була відсутня електроенергія. Разом з тим можна побачити досить великі блекаути, навіть є один розміром в декілька днів. Напишемо функцію, яка перебиратиме всі зафіксовані значення в датафреймі, та шукатиме перерви у вимірюваннях більше ніж в одну годину. Далі ці перерви у вимірюваннях відсортуємо за величиною проміжку в годинах. Вважитимемо, що аварійні відключення – це ті відключення, які тривають 10 годин або більше:

```
last_timestamp = None
gaps = []
for index, measure in weatherSensorDf.iterrows():
  current_timestamp = measure["TIMESTAMP"]

  if last_timestamp == None:
    last_timestamp = current_timestamp
    continue

  hours_diff = int((current_timestamp - last_timestamp).total_seconds())//3600
  if hours_diff > 1:
    gaps.append((hours_diff, last_timestamp, current_timestamp))
  last_timestamp = current_timestamp

gaps.sort(key=lambda x: x[0], reverse=True)

print("Electricity outages (>= 10 hours):")

for row in gaps:
  if row[0] < 10:
    break
  print(f"Duration: {row[0]}\tFrom: {row[1]:%d.%m %H:%M}\tTill: {row[2]:%d.%m
%H:%M}")
```

```
Electricity outages (>= 10 hours):
Duration: 72 From: 10.09 15:00  Till: 13.09 15:00
Duration: 15 From: 31.10 08:00  Till: 31.10 23:00
Duration: 10 From: 25.09 20:00  Till: 26.09 06:00
Duration: 10 From: 09.10 20:00  Till: 10.10 06:00
```

Отже, всього було чотири аварійних відключення, які тривали 10 або більше годин. А саме маємо:

- 72 години без світла: з 10.09 15:00 до 13.09 15:00
- 15 годин без світла: з 31.10 08:00 до 31.10 23:00
- 10 годин без світла: з 25.09 20:00 до 26.09 06:00
- 10 годин без світла: з 09.10 20:00 до 10.10 06:00

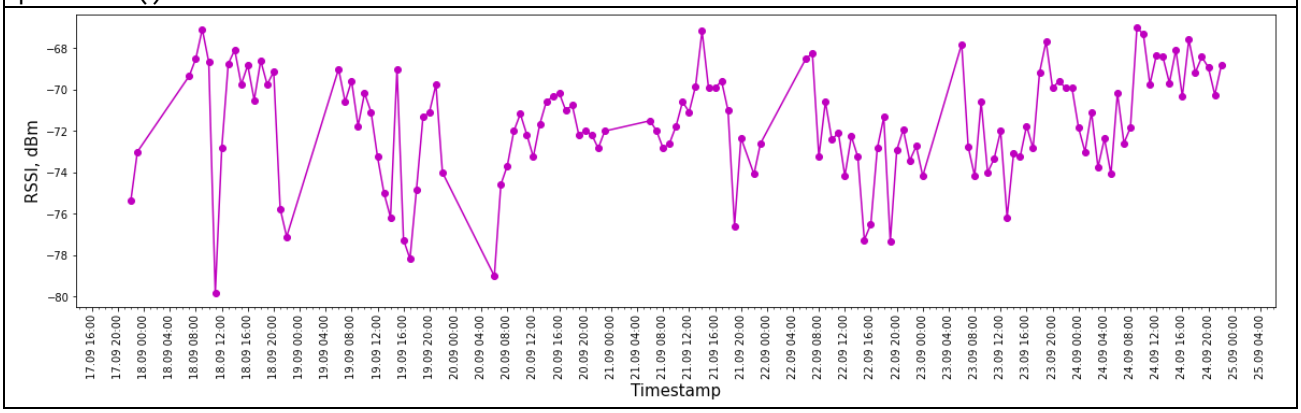
Знайдемо, коли відбувались зниження рівня сигналу за рахунок присутності людей в приміщенні за заданий проміжок часу. Для виконання цього завдання знову використаємо IPython ноутбук в середовищі Google Colab. Встановимо проміжок часу, в якому треба провести дослідження. Відфільтруємо датасет, щоб отримати дані, наприклад, за четвертий тиждень роботи датчика:

```
firstTimestamp = weatherSensorDf["TIMESTAMP"].min()
week = 4
startTimestamp = firstTimestamp + timedelta(weeks=(week - 1))
endTimestamp = firstTimestamp + timedelta(weeks=week)

weatherSensorDfFiltered = weatherSensorDf[
    (weatherSensorDf["TIMESTAMP"] >= startTimestamp) &
    (weatherSensorDf["TIMESTAMP"] <= endTimestamp)]

weatherSensorDfFiltered
```

	TEMP	HUMD	RSSI	VCC	TIMESTAMP
373	13.17	97.59	-75.33	3.1	2022-09-17 22:00:00+03:00
374	13.39	97.36	-73.00	3.1	2022-09-17 23:00:00+03:00
375	13.39	99.94	-69.33	3.1	2022-09-18 07:00:00+03:00
376	13.81	100.06	-68.50	3.1	2022-09-18 08:00:00+03:00
377	14.74	98.64	-67.08	3.1	2022-09-18 09:00:00+03:00
...	...	...	...	...	...
498	11.95	73.39	-69.17	3.1	2022-09-24 18:00:00+03:00
499	10.49	78.66	-68.42	3.1	2022-09-24 19:00:00+03:00
500	9.51	82.33	-68.92	3.1	2022-09-24 20:00:00+03:00
501	9.26	83.34	-70.25	3.1	2022-09-24 21:00:00+03:00
502	8.98	84.61	-68.83	3.1	2022-09-24 22:00:00+03:00

Отже, отримано 130 вимірювань за четвертий тиждень роботи датчика. Візуалізуємо відфільтровані дані з датафрейму, а саме побудуємо графік залежності рівня потужності Wi-Fi сигналу від часу в заданий проміжок часу:

```
fig, ax = plt.subplots(figsize=(20, 5))

ax.plot(weatherSensorDfFiltered["TIMESTAMP"], weatherSensorDfFiltered["RSSI"], 'mo-')

plt.xticks(rotation=90)
ax.xaxis.set_major_locator(mdates.HourLocator(interval=4))
ax.xaxis.set_minor_locator(mdates.HourLocator())
ax.xaxis.set_major_formatter(mdates.DateFormatter('%d.%m %H:%M',
tz=pytz.timezone('Europe/Kyiv')))
plt.ylabel('RSSI, dBm', fontsize=15)
plt.xlabel('Timestamp', fontsize=15)

plt.show()
```



Напишемо функцію, яка перебиратиме всі зафіксовані значення в датафреймі, та шукатиме ті значення рівня потужності Wi-Fi сигналу, які будуть менше або дорівнювати –75 дБм. Вважитимемо, що за рахунок людей в приміщенні сигнал Wi-Fi має знизитись до рівня – 75 дБм або нижче:

```
low_signals = []

for index, measure in weatherSensorDfFiltered.iterrows():
  current_rssi = measure["RSSI"]

  if current_rssi <= -75:
    current_timestamp = measure["TIMESTAMP"]
    next_timestamp = current_timestamp + timedelta(hours=1)
    current_row = [[current_rssi], current_timestamp, next_timestamp]
    len_low_signals = len(low_signals)

    if len_low_signals > 0 and low_signals[len_low_signals - 1][2] ==
current_timestamp:
      low_signals[len(low_signals) - 1][0].append(current_rssi)
      low_signals[len(low_signals) - 1][2] = next_timestamp
    else:
      low_signals.append(current_row)

print("Lowering the Wi-Fi signal strength (RSSI <= -75):")
```

```
for row in low_signals:
  rssi_mean = pd.DataFrame(row[0]).mean()[0]
  print(f"RSSI: {rssi_mean:.2f}\tFrom: {row[1]:%d.%m %H:%M}\tTill: {row[2]:%d.%m
%H:%M}")
```

```
Lowering the Wi-Fi signal strength (RSSI <= -75):
RSSI: -75.33 From: 17.09 22:00  Till: 17.09 23:00
RSSI: -79.83 From: 18.09 11:00  Till: 18.09 12:00
RSSI: -76.42 From: 18.09 21:00  Till: 18.09 23:00
RSSI: -75.59 From: 19.09 13:00  Till: 19.09 15:00
RSSI: -77.71 From: 19.09 16:00  Till: 19.09 18:00
RSSI: -79.00 From: 20.09 06:00  Till: 20.09 07:00
RSSI: -76.58 From: 21.09 19:00  Till: 21.09 20:00
RSSI: -76.88 From: 22.09 15:00  Till: 22.09 17:00
RSSI: -77.33 From: 22.09 19:00  Till: 22.09 20:00
RSSI: -76.17 From: 23.09 13:00  Till: 23.09 14:00
```

Об'єднаємо проміжки, у яких перерва у часі становить одну або дві години, бо вважитимемо причиною розділення таких проміжків похибкою вимірювань.

Отже, всього вісім разів були присутні люди в приміщенні на 4-ий тиждень роботи пристрою (з 17.09 22:00 по 24.09 22:00), адже рівень потужності Wi-Fi сигналу падав до -75 дБм або нижче. А саме маємо:

- З: 17.09 22:00 по: 17.09 23:00; середнє значення RSSI: -75.33 дБм
- З: 18.09 11:00 по: 18.09 12:00; середнє значення RSSI: -79.83 дБм
- З: 18.09 21:00 по: 18.09 23:00; середнє значення RSSI: -76.42 дБм
- З: 19.09 13:00 по: 19.09 18:00; середнє значення RSSI: -76.65 дБм
- З: 20.09 06:00 по: 20.09 07:00; середнє значення RSSI: -79.00 дБм
- З: 21.09 19:00 по: 21.09 20:00; середнє значення RSSI: -76.58 дБм
- З: 22.09 15:00 по: 22.09 20:00; середнє значення RSSI: -77.11 дБм
- З: 23.09 13:00 по: 23.09 14:00; середнє значення RSSI: -76.17 дБм

Туманні обчислення (*Fog Computing*) – це обчислювальний шар між хмарою та межею мережі. Граничні обчислення можуть надсилати великі потоки даних безпосередньо в хмару. З іншого боку, туманні обчислення можуть отримувати дані з крайового шару до того, як вони досягнуть хмари. Потім у хмарі зберігаються лише відповідні дані. У той же час нерелевантні дані можна видалити або проаналізувати в шарі Fog для віддаленого доступу або для інформування локалізованих моделей навчання.

Прикладом обчислень Fog може бути вбудована програма на виробничій лінії, де датчик температури, підключений до периферійного шлюзу, вимірює температуру кожну секунду. Потім ці дані надсилаються у хмару для моніторингу стрибків температури. З шаром туману, крайовий шлюз спочатку надсилає дані на рівень Fog через локальну мережу. На основі певних параметрів тут вирішується, чи надсилати дані в хмару і які саме дані надсилати. Однією з переваг такого підходує ефективність трафіку даних і зменшення затримки. Завдяки реалізації рівня Fog зменшується кількість даних, які хмара отримує для свого конкретного мережевого додатка. Це дозволяє безпосередньо реагувати на дані з шару туману. Інші переваги включають менше місця для зберігання, необхідного для хмарної програми та швидшу передачу даних завдяки зменшеному обсягу даних.

Інтернет речей сприяє інтенсивному обігу даних, включаючи будь-що, від розумних електричних мереж до фітнес-трекерів. Хмарні обчислення та штучний інтелект дозволяють динамічно обробляти та зберігати великі обсяги даних. Ці дані дозволяють організаціям приймати обґрунтовані рішення та захищати себе від вразливостей як на бізнес-, так і на технологічному рівнях.

Однак цей вибух даних змусив організації засумніватися в якості та кількості даних, які вони зберігають у хмарі. Відомо, що витрати на хмару швидко зростають, а перегляд петабайтів даних ускладнює реагування в реальному часі. Наприклад, розглянемо дані, які надсилає датчик температури на заводській лінії. Запис температури можна щосекунди надсилати в хмару за допомогою служби, яка перевіряє наявність коливань. Але більш розумним способом збереження цієї інформації є перевірка того, чи відбулися зміни температури за останні кілька секунд. Коли помічається зміна температури, дані надсилаються в хмару для зберігання, щоб перевірити належну роботу виробничої лінії. Температура може займати мало місця, але такий сценарій також поширений для таких пристроїв, як камери відеоспостереження, які створюють великі набори відео- та аудіоданих.

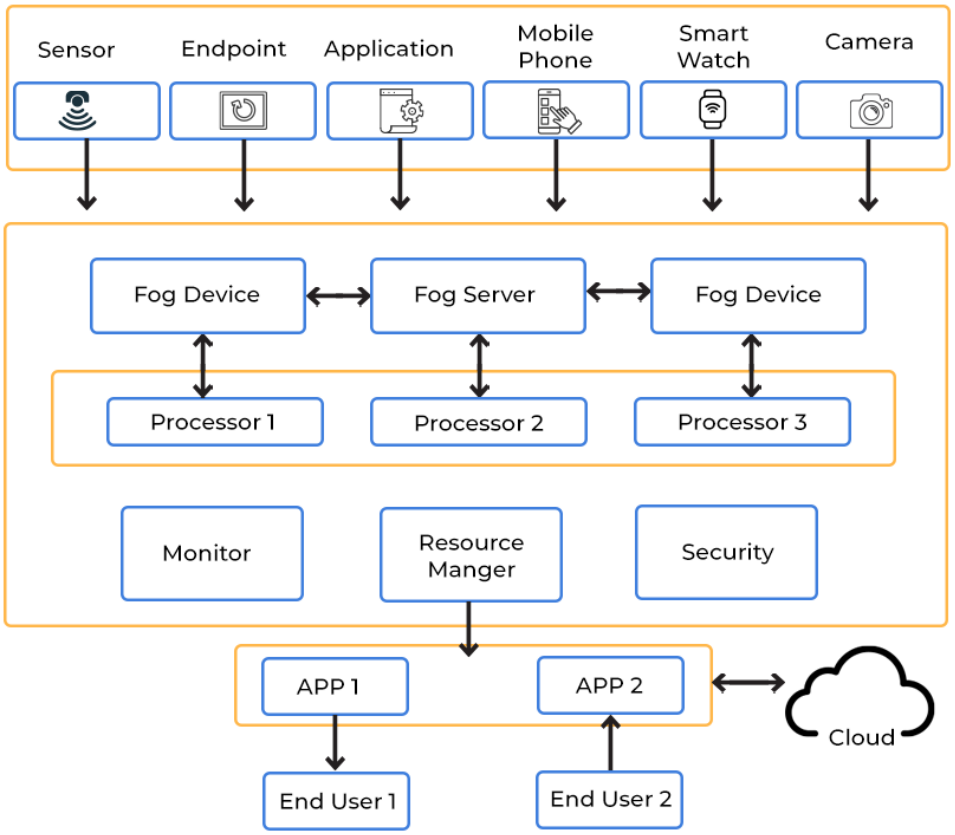


Рис. 3.7. Архітектура Fog Computing [5]

Граничні обчислення – це підмножина туманних обчислень, яка передбачає обробку даних безпосередньо в точці створення. Граничні пристрої включають маршрутизатори, камери, комутатори, вбудовані сервери, датчики та контролери. У периферійних обчисленнях дані, створені цими пристроями, зберігаються та обчислюються на самому пристрої, і система не дивиться на обмін цими даними з хмарою.

Завдання до лабораторної роботи 3

Для виконання лабораторної роботи необхідно виконати послідовність дій згідно інструкцій, додавши скріншоти виконання роботи на надати відповіді на поставлені питання.

Потрібно дослідити проєкт розумного будинку та використати туманні обчислення в розумному будинку для мережі пристроїв IoT заданої топології (рис. 3.8).

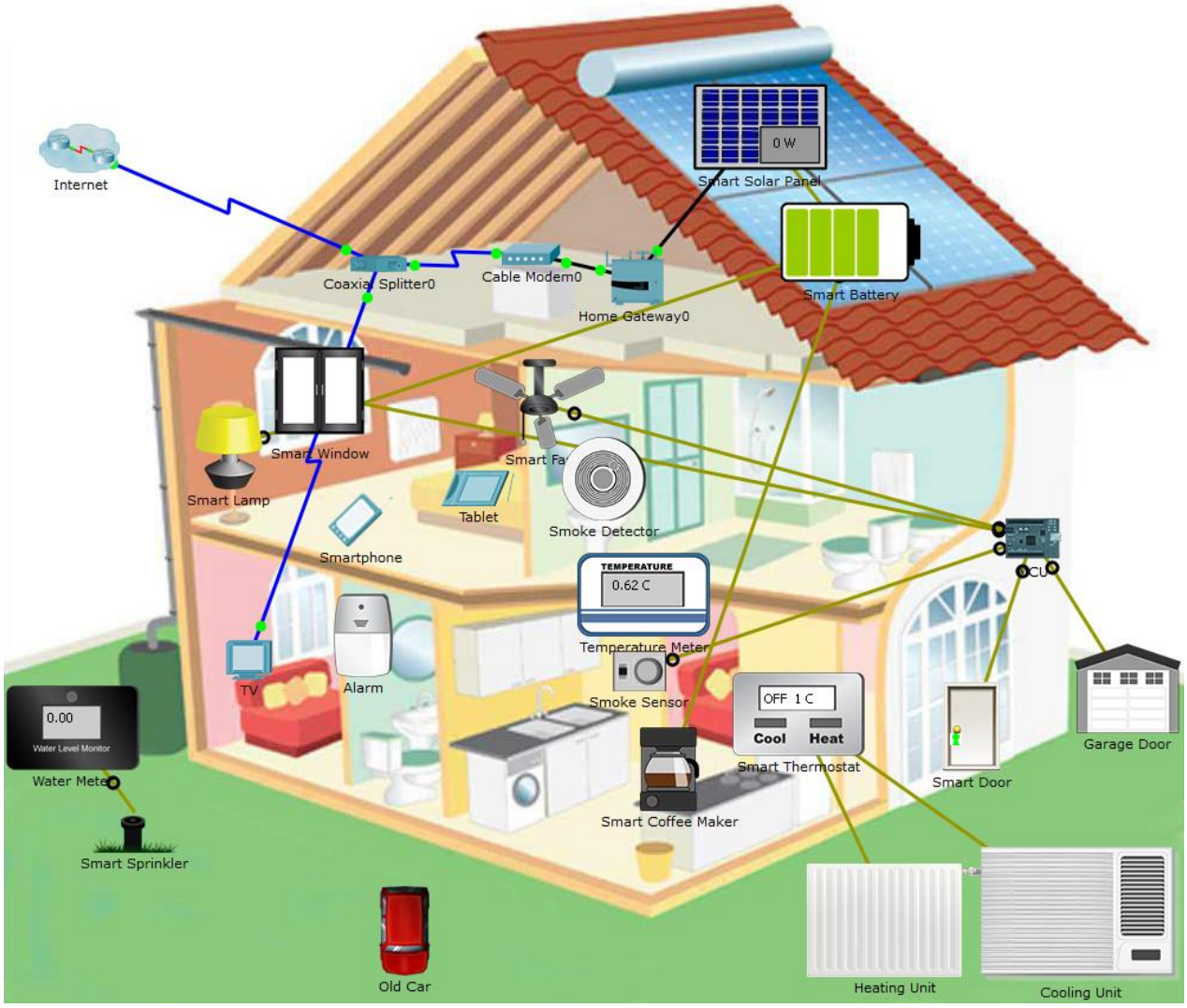


Рис. 3.8. Проєкт розумного будинку

Частина 1. Аналіз туманної обчислювальної системи для моніторингу та впливу на рівні диму, виявленого у розумному будинку

Інтернет-провайдери часто передають дані та відео через єдиний коаксіальний кабель. Починаючи з мансарди, коаксіальний спліттер використовується для відокремлення відеосигналу від сигналу даних.

Два коаксіальні кабелі залишають коаксіальний спліттер у відображеній топології. Які пристрої підключаються до коаксіального кабелю?

Кабельний модем – це інтерфейс між мережею Інтернет-провайдера та домашньою мережею. На які пристрої підключається кабельний модем?

Home Gateway – це концентратор і маршрутизатор для всіх внутрішніх пристроїв. Він також надає вебінтерфейс, який дозволяє користувачам

здійснювати моніторинг та керування різними розумними домашніми пристроями. Зверніть увагу, що домашні пристрої можуть підключатися до домашнього шлюзу через бездротовий та дротовий зв'язок.

Середовище Packet Tracer використовує пунктирні лінії для представлення бездротових з'єднань, але це може бути важким для читання топології мережі, коли присутні забагато пристроїв. Щоб увімкнути це, перейдіть до Options > Preferences > Hide Tab > uncheck Hide Wireless/Cellular Connection. Перерахуйте всі домашні пристрої, підключені до домашнього шлюзу.

Частина 2. Взаємодія з розумним будинком

Пристрої в розумному домі можна відслідковувати та управляти дистанційно через будь-який домашній комп'ютер. Оскільки всі розумні пристрої підключаються до домашнього шлюзу, на якому розміщено вебінтерфейс, планшети, смартфони, ноутбуки або настільні комп'ютери можуть використовуватися для взаємодії із розумними пристроями.

Натисніть Tablet , розташований на ліжку в головній спальні.

Перейдіть до Desktop > Web Browser.

В адресному рядку введіть 192.168.25.1. Це IP-адреса домашнього шлюзу.

Використайте admin/admin в якості імені користувача та паролю, для входу до Home Gateway.

Що відображається?

В даний час розумні двері відчинені (зелене світло на ручці дверей), але їх можна заблокувати віддалено. Натисніть на розумні двері в браузері.

Натисніть Lock, щоб зачинити двері.

Чи зачинено двері? Звідки ви це знаєте?

Натисніть Unlock, щоб відчинити двері.

Клацніть детектор диму в браузері. Який рівень диму забезпечується детектором диму?

Чи можна контролювати датчик диму? Як це можна робити програмно?

Розумними пристроями також можна керувати безпосередньо, представляючи фізичну взаємодію.

Утримуючи клавішу ALT, натисніть на розумну кавоварку, щоб увімкнути або вимкнути її.

Частина 3. Туманні обчислення в розумному будинку.

Блок мікроконтролера (MCU, microcontroller unit) – це невеликий самостійний комп'ютер, розміщений на одній інтегральній схемі або мікрочіпі. MCU відрізня'ться від настільного комп'ютера тим, що зазвичай використовується для однієї функції та найчастіше вбудовується в інші пристрої (наприклад, мобільні телефони, побутову електроніку). MCU, доданий в розумний дім, використовується для контролю за рівнем диму, зчитаним датчиком диму, і вирішує, чи потрібно ввімкнути вентиляцію. Якщо рівень монооксиду вуглецю вищий за 10,3 одиниці, MCU запрограмований на автоматичне відкриття вікна, передніх дверей, гаражних дверей та запуску вентилятора на високій швидкості. Ця дія лише повертається (зачиняйте двері та вікна та зупиніть вентилятор), коли рівень окису вуглецю знизиться нижче 1.

Створіть наступний проєкт запуску класичного автомобіля.

Власник тримає класичний автомобіль в гаражі і його потрібно запускати. Класичний автомобіль виробляє монооксид вуглецю, який підвищує рівень в приміщенні. Натисніть Tablet, розташований на ліжку в головній спальні. Перейдіть до Desktop > Web Browser.

В адресному рядку введіть 192.168.25.1. Це IP-адреса домашнього шлюзу. Використайте admin/admin в якості імені користувача та паролю, для входу до Home Gateway.

Натисніть детектор диму; залиште це вікно видимим, щоб ви могли контролювати рівень диму. Запустіть двигун автомобіля, утримуючи клавішу Alt і натиснувши на класичний автомобіль.

Що відбувається з повітрям всередині будинку з автомобілем, що працює всередині гаража?

Що відбувається з повітрям усередині будинку після того, як MCU відкриває двері та вікно, і запускає вентилятор?

Чи закриває MCU двері та вікно і зупиняє вентилятор?

Продовжуючи стежити за рівнями, зупиніть двигун класичного автомобіля, утримуючи клавішу Alt і натиснувши на класичний автомобіль.

Що відбувається з якістю повітря всередині будинку після зупинки двигуна?

Що відбувається з дверима, вікном та вентилятором?

Рішення між хмарними і туманними обчисленнями залежить від програми.

У прикладі розумного будинку найкращим варіантом було виконання туманних обчислень.

У прикладі смарт-будинку дані, отримані датчиками диму, оброблялись та використовувались для прийняття рішень щодо якості повітря в будинку. У цьому сценарії не потрібно було надсилати дані датчиків у хмару для обробки.

Хмарні обчислення можуть призвести до уповільнення часу відгуку, що може призвести до небезпеки життя людей. Інша можлива проблема пов'язана з інтернет-посиланням; якщо з'єднання з Інтернетом було втрачено, вся система могла б створити небезпеку для життя.

Питання для самоперевірки

1. Що таке хмарні обчислення, яке їх призначення та застосування?
2. Які переваги використання хмарних обчислень для систем IoT?
3. Наведіть приклади використання хмарних обчислень в системах IoT.
4. Які технічні характеристики системи IoT на базі ESP8266+HTU21 ви знаєте?
3. Для чого використовується середовище Google Colab? Які його переваги для аналізу даних пристроїв IoT?
4. Для чого використовуються туманні обчислення, які їх переваги?
5. Наведіть приклади використання туманних обчислень.

ЛАБОРАТОРНА РОБОТА 4.
НАЛАШТУВАННЯ БЕЗПЕКИ ПРИСТРОЇВ IOT.

Мета роботи: створити домашню мережу за допомогою бездротового маршрутизатора з налаштуваннями безпеки пристроїв IoT.

Теоретичні відомості

SSID (Service Set Identifier) використовується для позначення бездротової точки доступу Wi-Fi та її ідентифікації серед інших точок доступу. SSID є рядком розміром до 32 байт, який передається широкомовно в ефір. Розташовані поруч із мережею пристрої приймають назву і, якщо їм дозволено приєднатися до точки доступу, то з'єднуються з нею. З точки зору безпеки мережеві адміністратори іноді забороняють точці доступу повідомляти ідентифікатор, тоді такий пристрій у списку видимих точок доступу не відображається. Щоб підключитися до такої мережі, необхідно ввести ідентифікатор у пристрої, що підключається вручну. Існують точки доступу, що дають змогу розділяти абонентів по сегментах, у таких випадках одна точка доступу може мати декілька ідентифікаторів SSID. Обмін даними у Wi-Fi мережах регламентується стандартом IEEE 802.11.

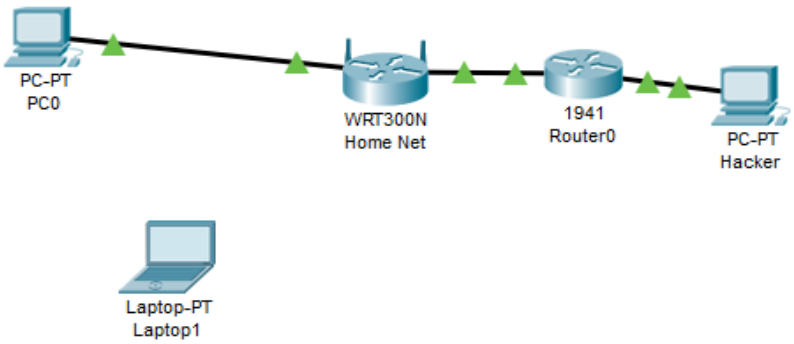
Метод приховування широкомовлення SSID не є безпечним, ідентифікатор все одно можна дізнатися, оскільки в деяких пакетах передачі даних між пристроєм і точкою доступу передається SSID у відкритому вигляді (не шифрується), користувач, який бажає отримати доступ до такої мережі, може прослухати мережу та виокремити з пакета передачі ідентифікатор SSID.

Щоб зробити мережу більш захищеною у стандарті 802.11, регламентується асоціювати SSID з однією або більше точками доступу. Іншими словами, за допомогою SSID можна ідентифікувати сегмент мережі, тоді навантаження на підтримку адміністратором мережі спрощується шляхом підтримки безпеки одного сегмента.

Завдання до лабораторної роботи 4

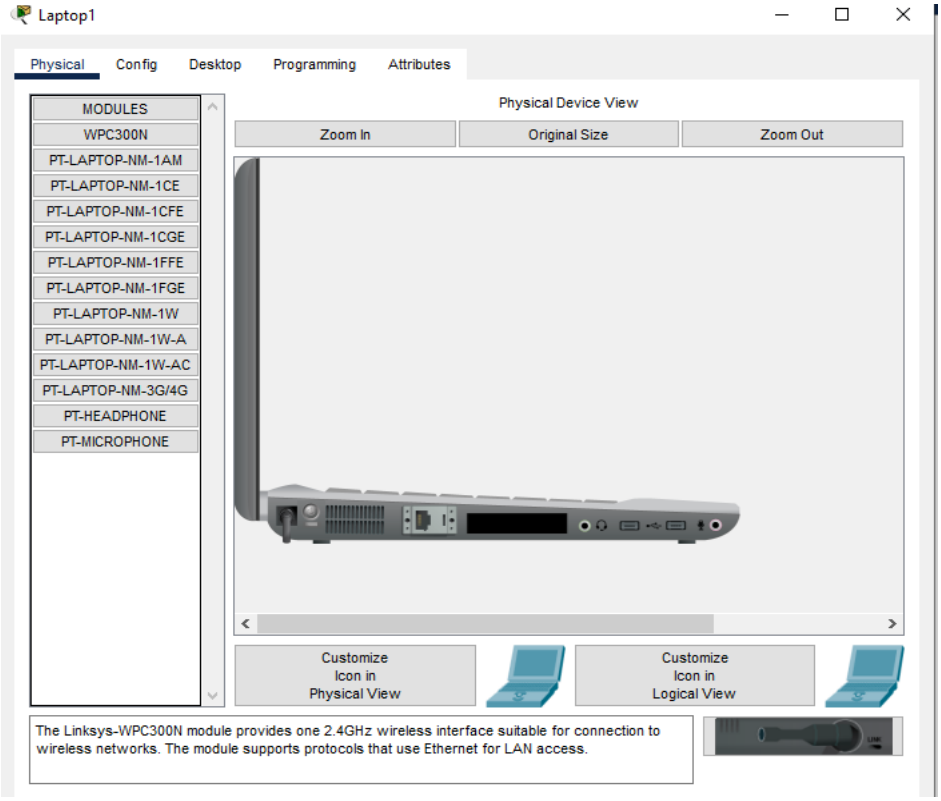
Необхідно виконати скріншоти виконання лабораторної роботи та надати відповіді на поставлені питання. У цій лабораторній роботі ви налаштуєте бездротовий маршрутизатор, щоб змінити пароль за замовчуванням, змінити та не транслювати стандартний SSID.

- Використовуйте WPA2 Personal як метод захисту.
- Покладайтеся на фільтрацію MAC для підвищення безпеки.
- Вимкніть віддалене керування.



Завантажте файл Packet Tracer – Configure Wireless Security.pkt.

Натисніть кнопку power на Laptop1, щоб вимкнути його.

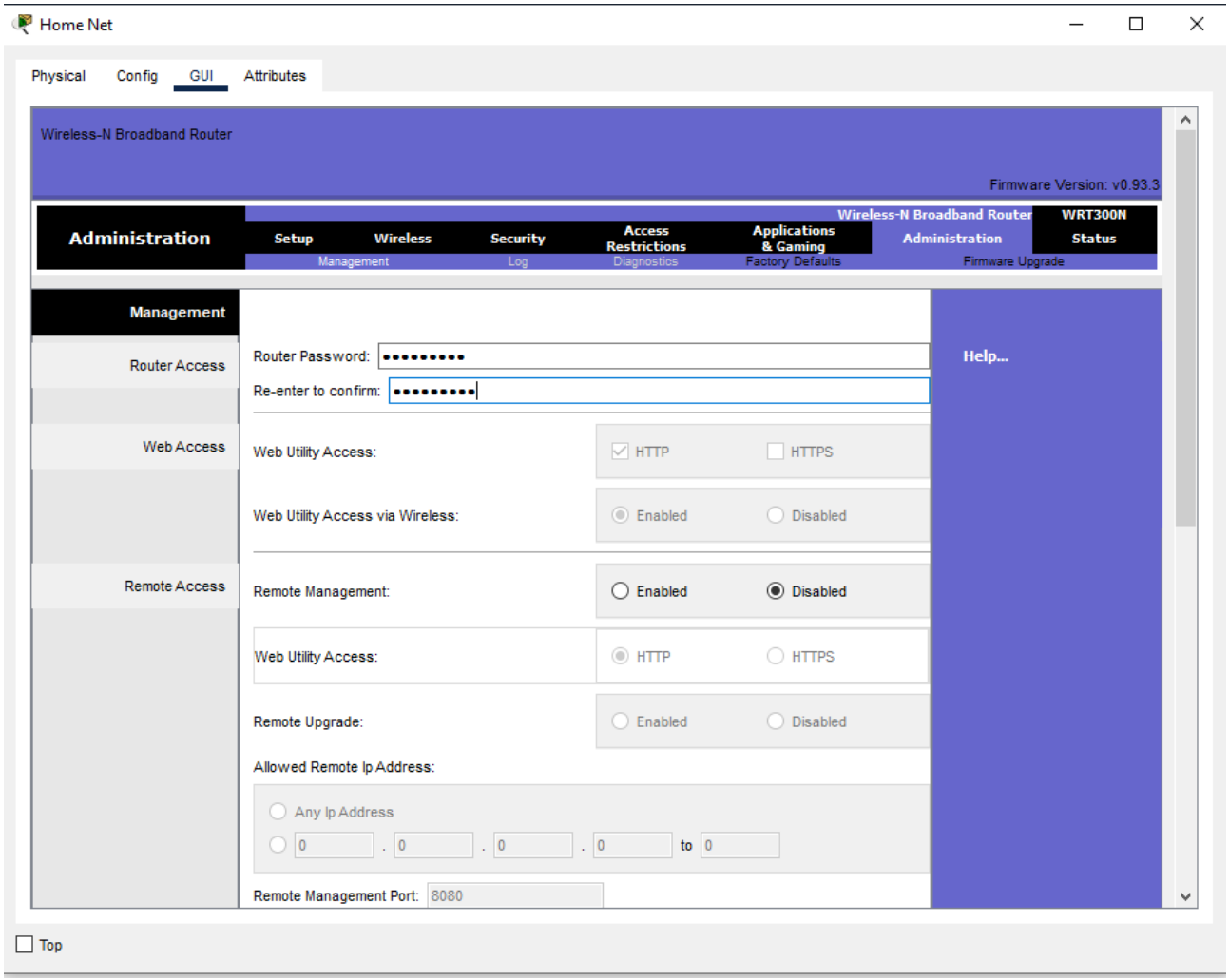


Перетягніть Ethernet порт до модулів, щоб видалити його.

Перетягніть модуль WPC300N у порожній слот на Laptop1 та натисніть кнопку power, щоб завантажити Laptop1.

Змініть пароль за замовчуванням.

Натисніть на бездротовий маршрутизатор і виберіть конфігурацію GUI.



Натисніть Administration > Management

Змініть пароль маршрутизатора на більш сильний, наприклад, aC0mpAny3. Зверніть увагу, що новий пароль має 8 знаків з цифрами верхнього та нижнього регістру, а деякі голосні змінені на цифри. Виберіть Save Settings внизу екрана.

Змініть ім'я SSID за замовчуванням та вимкніть функцію трансляції.

Натисніть Wireless та змініть ім'я SSID на Company.

Виберіть SSID Broadcast та натисніть Disabled. Натисніть Save Settings у нижній частині екрана.

Перевірте топологію мережі. Чи ноутбук втратив зв'язок з бездротовим маршрутизатором? Якщо так, то чому?

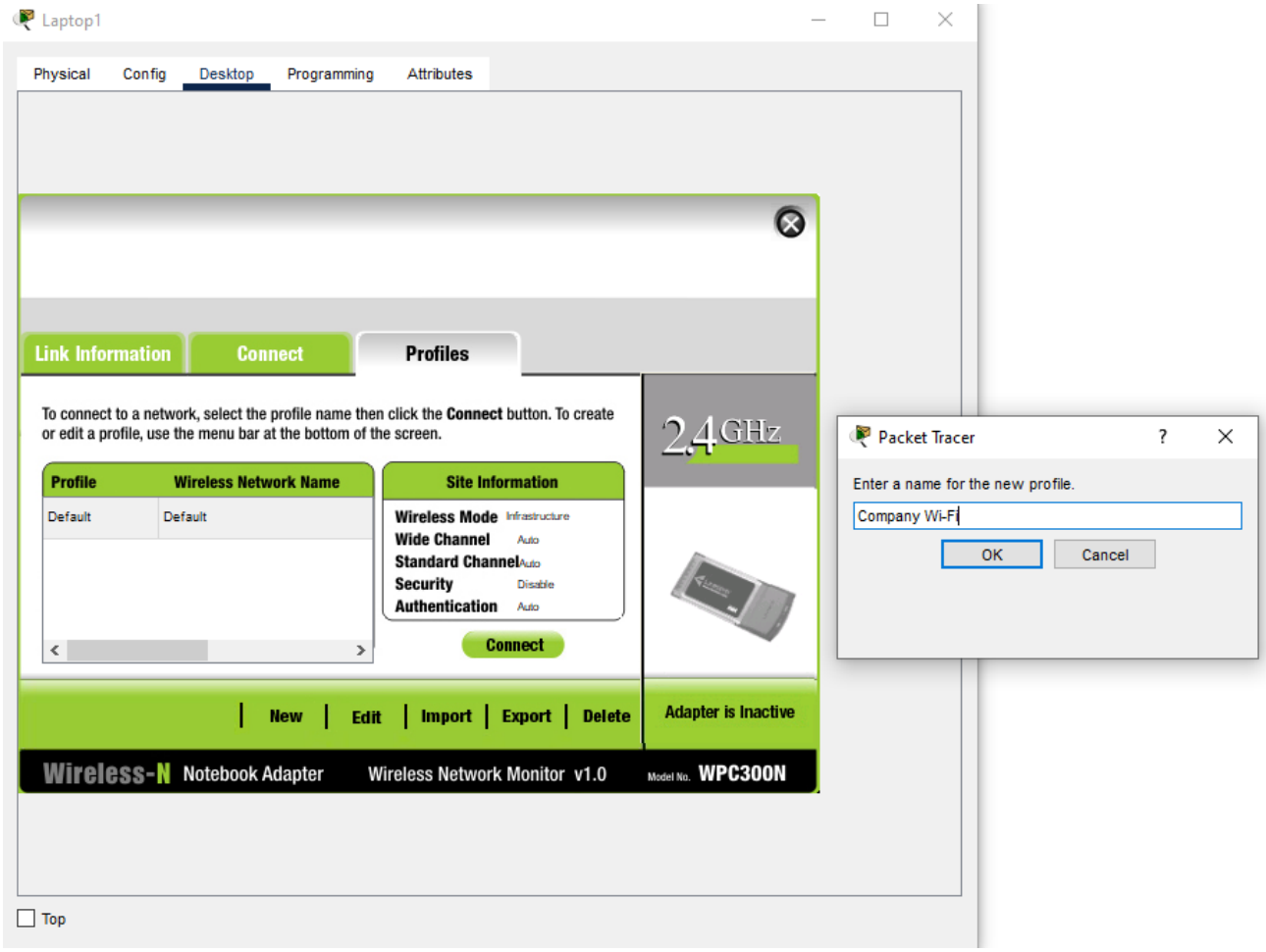
Налаштуйте безпеку WPA2 на бездротовому маршрутизаторі.

Поверніться на вкладку GUI бездротового маршрутизатора. Натисніть Wireless> Wireless Security. Змініть режим безпеки на WPA2 Personal. AES є надійним доступним протоколом шифрування. Залиште його вибраним.

Налаштуйте парольну фразу як CompWiFi. Прокрутіть донизу вікна та натисніть Save Settings.

Налаштуйте Laptop0 як бездротовий клієнт.

Налаштуйте Laptop0 для підключення до бездротової мережі за допомогою параметрів безпеки, налаштованих на бездротовому маршрутизаторі.



1) Виберіть Laptop0 > Desktop > PC Wireless.

2) Виберіть Profiles> New та додайте будь-яке ім'я для профілю.

3) Виберіть Advanced Setup у нижньому правому куті.

4) Виберіть Назва бездротової мережі та введіть нове ім'я SSID: Company.

5) Прийміть налаштування мережі за замовчуванням, вибравши Next.

6) Змініть Security за допомогою випадаючого списку на WPA2-Personal і виберіть Next.

7) Введіть попередньо загальний ключ: CompWiFi та виберіть Next.

8) Зберегти профіль.

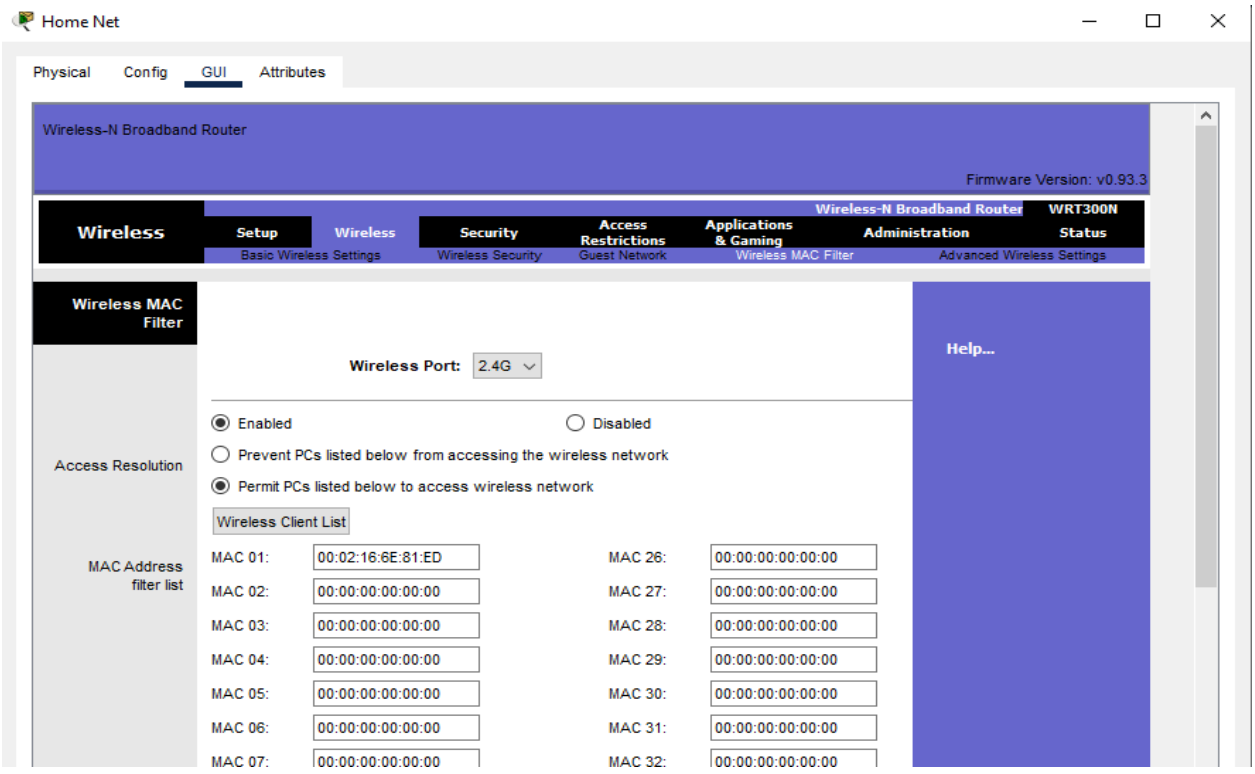
9) Виберіть Connect to Network.

10) Якщо ноутбук не з'єднаний успішно, поверніться до профілю та редагуйте його. Перевірте введення набору імені SSID та попередньо розділеного ключа.

Закрийте вікно бездротового зв’язку ПК і натисніть Command Prompt.

Введіть ipconfig /all і візьміть до уваги значення IP-адреси та MAC-адреси.

Налаштуйте WRS1 для підтримки фільтрації MAC.



Поверніться на сторінку конфігурації бездротового маршрутизатора.

Перейдіть до Wireless > Wireless MAC Filter.

Виберіть Enabled та Permit PCs listed below to access wireless network.

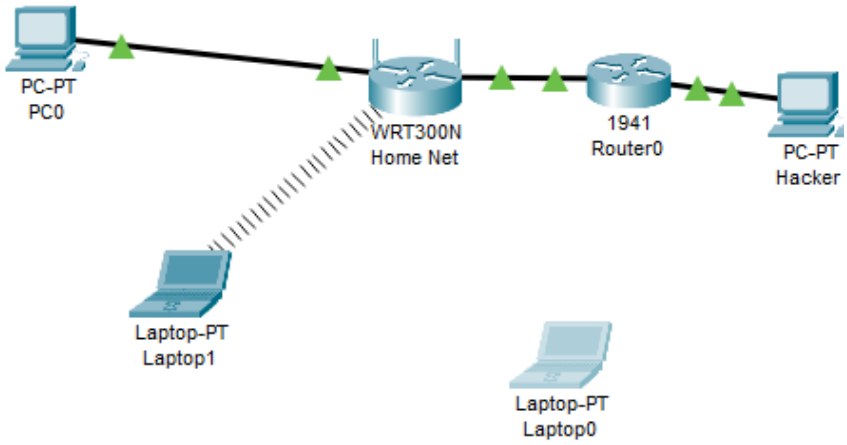
Введіть MAC-адресу для Laptop0 у полі MAC 01: . Зверніть увагу на те, що MAC-адреса має бути у форматі XX: XX: XX: XX: XX: XX.

Прокрутіть донизу вікна та натисніть Зберегти параметри.

Знову під'єднайте Laptop0 до мережі WRS1.

Перевірте MAC-фільтр WRS1.

Додати другий ноутбук до топології.



Натисніть кнопку power на новому ноутбуці, щоб вимкнути його та замінити картку Ethernet бездротовою.

Налаштуйте новий ноутбук із налаштуваннями безпеки, необхідними для підключення до бездротового маршрутизатора.

Чому ви не можете зв'язатись з точкою доступу?

Вимкніть віддалене керування на бездротовому маршрутизаторі.

Перевірте поточний стан дистанційного керування на бездротовому маршрутизаторі. Виберіть Administration > Remote Access. Якщо це налаштування вимкнено, увімкніть його.

Виберіть Hacker PC та виберіть Desktop > Web Browser. Введіть адресу 192.31.7.100 та натисніть Go. Вам слід подати запит на ідентифікатор користувача та пароль. Введіть дійсну інформацію, і ви повинні мати доступ до графічного інтерфейсу бездротового маршрутизатора.

Поверніться на сторінку конфігурації бездротового маршрутизатора.

Перейдіть назад до Administration > Management та прокрутіть униз до Remote Management. Виберіть Disabled та натисніть Save Settings у нижній частині цього екрана.

Поверніться до Hacker PC і виберіть Desktop > Web Browser. Введіть адресу 192.31.7.100. Ви більше не зможете підключатися через веббраузер.

Питання для самоперевірки

1. Для чого використовується ідентифікатор SSID?
2. Як зробити мережу більш захищеною у стандарті 802.11?
3. Які налаштування безпеки для мережі IoT ви знаєте?

ЛАБОРАТОРНА РОБОТА 5.
ПРОЄКТУВАННЯ МЕРЕЖІ ДЛЯ РОЗУМНОГО БУДИНКУ ТА ПРОГРАМУВАННЯ ПРИСТРОЇВ IOT

Мета роботи*:* створити модель розумного будинку згідно сценарію та налаштувати пристрої IoT для їх функціонування у мережі та взаємодії з мережевими пристроями. Налаштувати DHCP, WLAN та DNS-сервер, здійснити тестування налаштованих пристроїв IoT.

Теоретичні відомості

Середовище моделювання Packet Tracer містить кілька підкатегорій розумних речей: дім, розумне місто, промисловість і електромережа, системи керування на основі IoT для заводу. Система відстеження пакетів також складається з плат – мікроконтролерів (MCU-PT) і одноплатних комп'ютерів (SBC-PT), а також приводів та датчиків. Параметрами Інтернету в маршрутизаторі є ідентифікатор набору бездротових послуг (SSID) і пароль. На всіх бездротових пристроях використовуються однакові параметри SSID, пароля та протоколу динамічної конфігурації хоста (DHCP), тоді як локальний сервер використовує статичну IP-адресу. Статичні IP-адреси гарантують, що в

разі перезавантаження маршрутизатора WLAN IP-адреса сервера залишається незмінною, і не потрібно повторно налаштувати пристрої, які призначають нові IP-адреси сервера IoT. Сервер центрального офісу надає послуги DHCP, IoT і DNS. Ці функції забезпечують наявність внутрішнього інтелекту для симуляції IoT, а також сприяють розміщенню домашньої сторінки IoT, де кінцеві користувачі можуть підключатися та взаємодіяти з розумним будинком. Служба DNS допомагає перевести URL-адресу домашньої сторінки IoT в IP-адресу сервера IoT.

Смарт-пристрої віддалено підключаються до внутрішнього сервера IoT. Підключення IoT дозволяє користувачам перевіряти стан розумних пристроїв IoT на домашній сторінці браузера IoT. Домашня сторінка браузера IoT відображає список розумних пристроїв, дозволяє візуалізувати їх стан та віддалено взаємодіяти з пристроями. Логічну взаємодію між смарт-пристроями також можна налаштувати під час підключення до головної домашньої сторінки IoT. Режим реального часу середовища моделювання Packet Tracer надає можливість створювати мережу під мережею, підключати пристрої IoT і визначати логіку серверної частини IoT. Лише в режимі симуляції можна підтвердити, що мережевий рівень зв’язку дійсно відбувся між пристроями.

У режимі симуляції можна моделювати пакетний трафік між вузлами та пристроями, щоб перевірити підключення, протоколи маршрутизації та іншу мережеву логіку. Цей режим допомагає фізично візуалізувати та усунути неполадки будь-якої мережі, наприклад, налаштувати ping.

Розглянемо приклади проєктування мережі для розумного будинку та програмування пристроїв IoT.

У середовищі Cisco Packet Tracer створюємо новий проєкт та налаштовуємо задній фон: зображення будинку, на який потім будуть додаватись розумні пристрої. Після цього додаємо початкові елементи домашньої мережі, а саме: бездротовий маршрутизатор, два сервери – для DNS та IoT, планшет. Результат проєктування зображено на рис. 5.1.

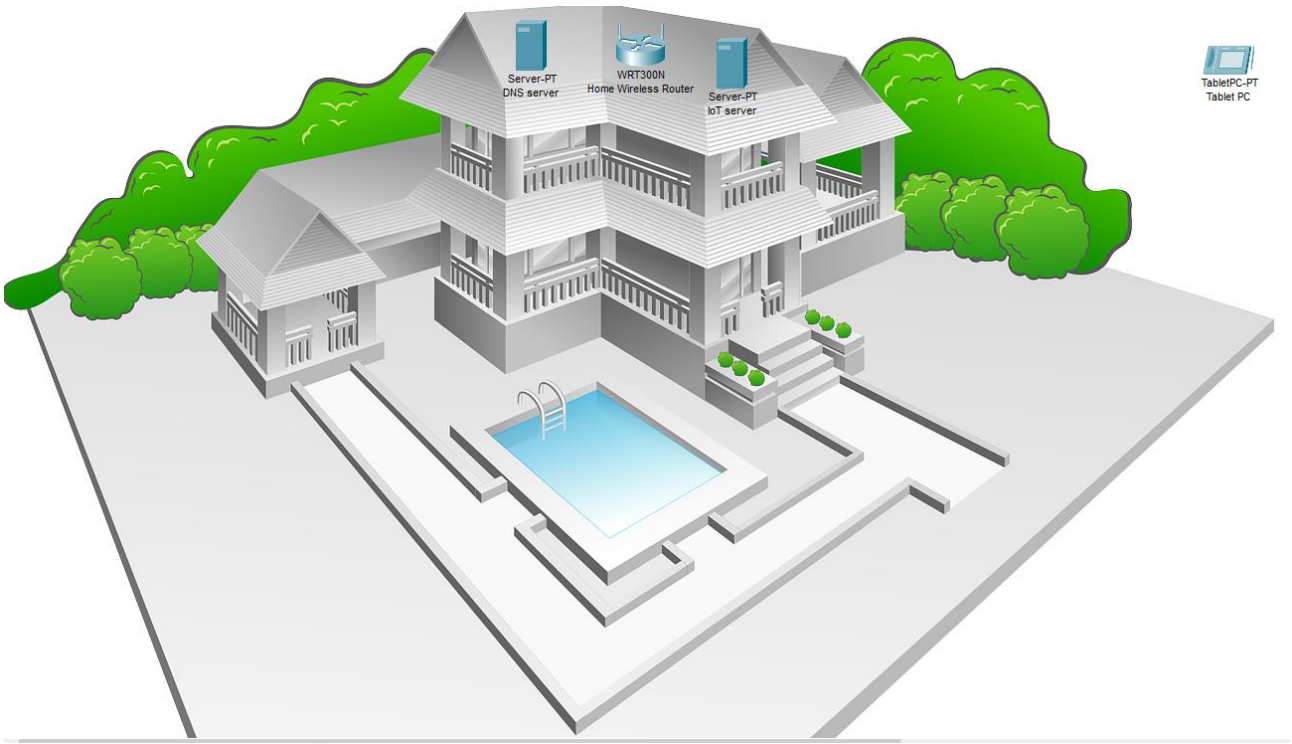


Рис. 5.1. Додавання бездротового маршрутизатора, планшету та серверів

Додаємо бездротовий маршрутизатор, для якого налаштовуємо WLAN відповідно до критеріїв безпеки: встановлюємо наступний ідентифікатор мережі SSID: Private_House_Network. Тип авторизації встановлено найбільш захищений – WPA2-PSK. Пароль для мережі встановлено наступний: SeCret_KeY_735$!&. Тип шифрування – AES. Результати встановлення налаштування зображено на рис. 5.2.

Відкриємо GUI бездротового маршрутизатора, тобто його графічний інтерфейс для подальшого налаштування. На рис. 5.3 видно, що сервіс DHCP увімкнений, а сам маршрутизатор має адресу 192.168.0.1 та маску підмережі 225.225.225.0. Адреси DHCP виділяються з 192.168.0.100 по 192.168.0.149.

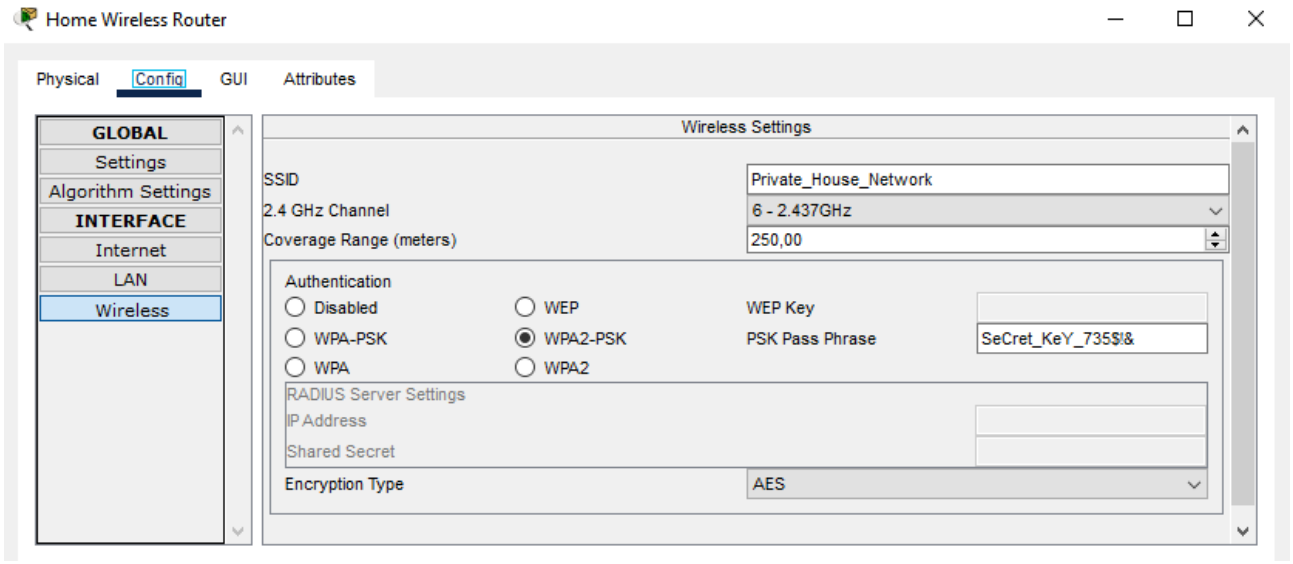


Рис. 5.2. Базове налаштування WLAN для бездротового маршрутизатора

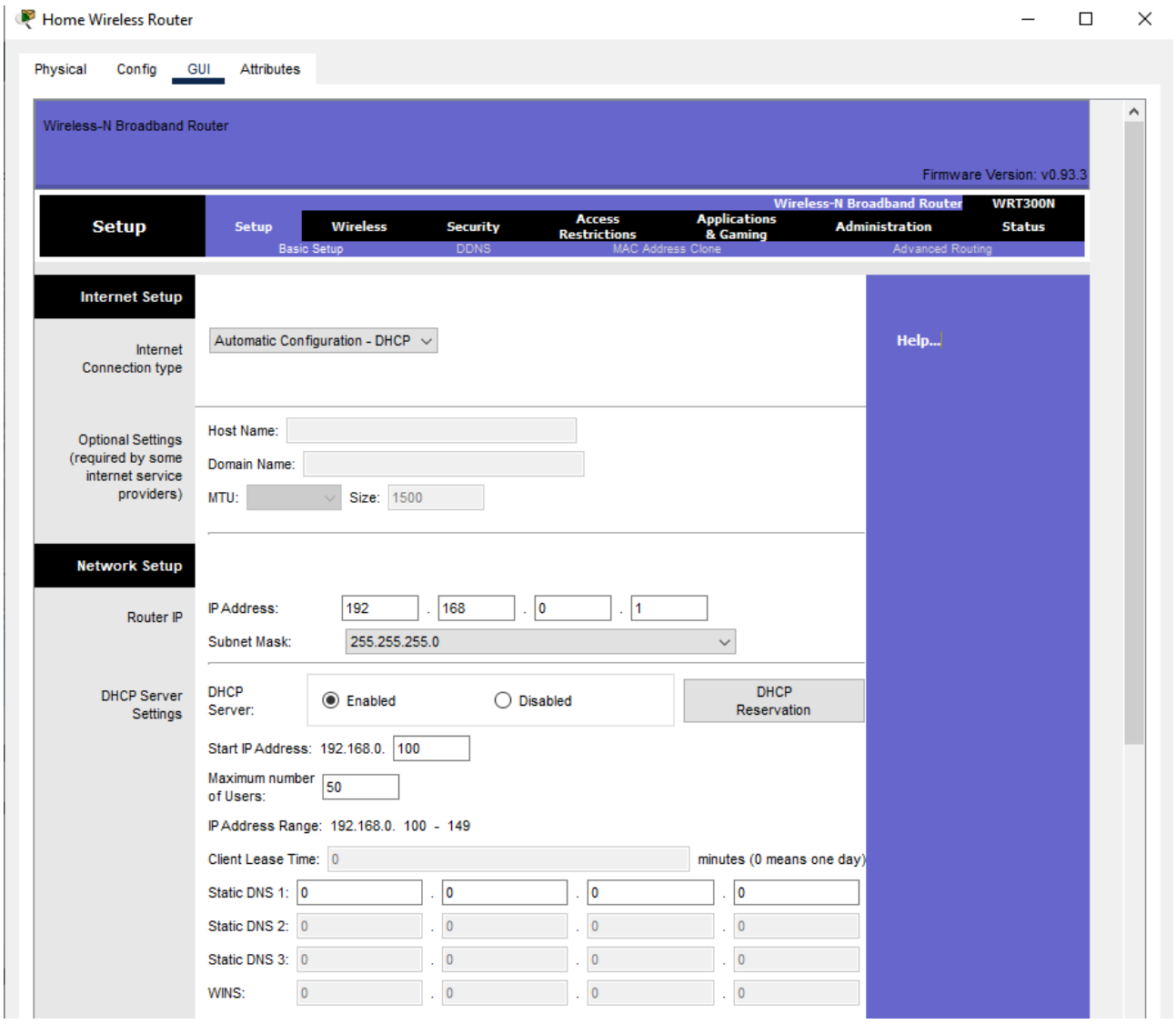


Рис. 5.3. Базове налаштування WLAN для бездротового маршрутизатора

Окрім цього, перевіримо, чи налаштування WLAN для домашнього маршрутизатора встановлені правильно (рис. 5.4). Також перевіримо, що налаштування безпеки для бездротової мережі, а саме пароль, тип шифрування – теж встановлені правильно (рис. 5.5). Як бачимо, все відповідає тому, що було встановлено попередньо через налаштування.

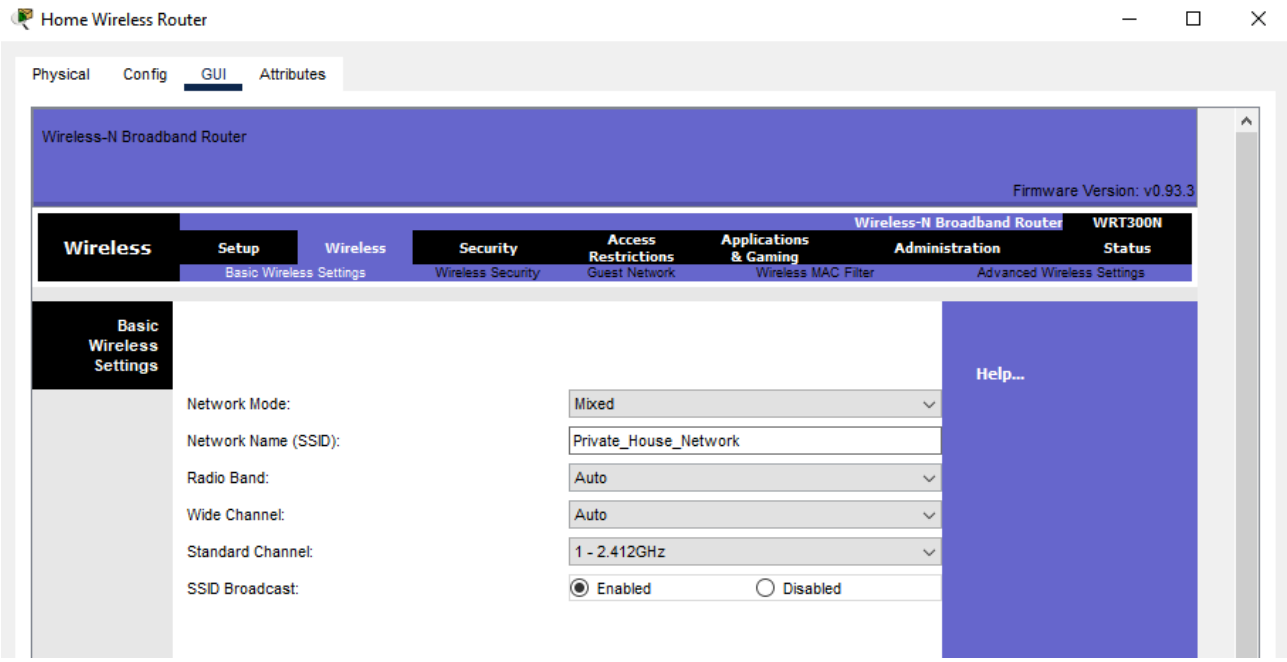


Рис. 5.4. Налаштування бездротової мережі маршрутизатора

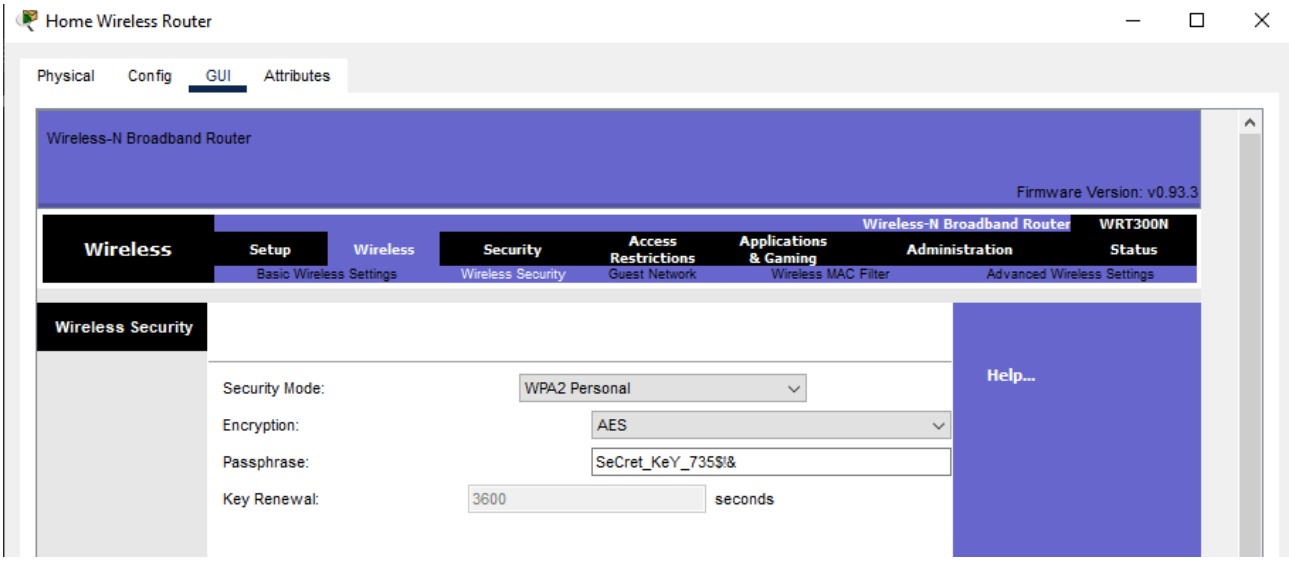


Рис. 5.5. Налаштування безпеки бездротової мережі маршрутизатора

Також перевіримо, чи має маршрутизатор встановлений пароль, та чи вимкнуто віддалене керування маршрутизатором (рис. 5.6). Як бачимо, всі налаштування безпеки встановлені правильно.

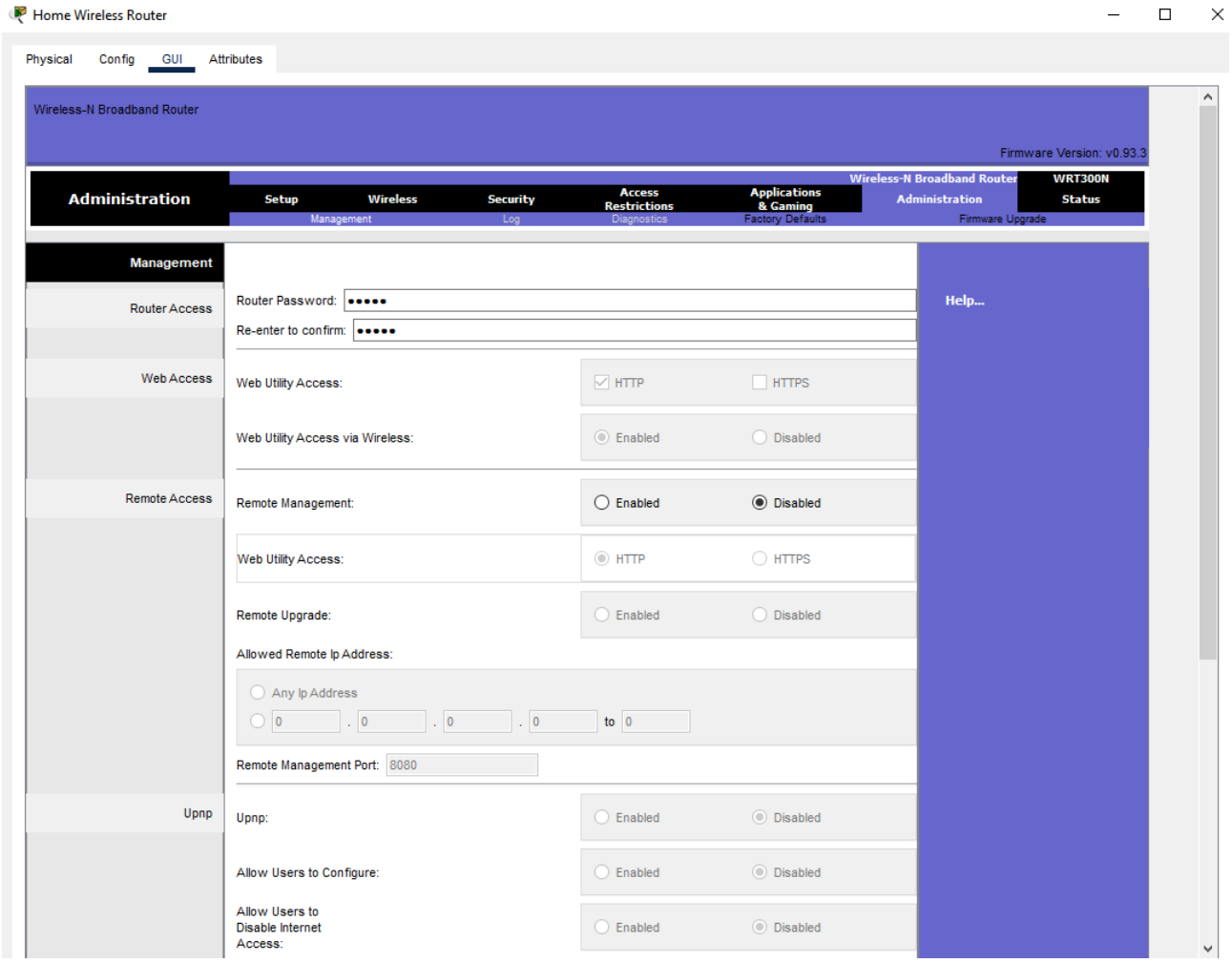


Рис. 5.6. Безпекові та адміністративні налаштування маршрутизатора

Після цього налаштовуємо два сервери, підписуємо їх як DNS та IoT сервери, оскільки вони виконуватимуть відповідні функції.

Потім їх підключаємо до домашнього маршрутизатора кабелем, результат цього підключення зображено на рис. 5.7.

Після цього для обох серверів налаштовуємо статичну IP-адресу. Для DNS сервера встановлюємо статичну IP-адресу 192.168.0.10 та маску підмережі 225.225.225.0 (рис. 5.8).

Для IoT сервера встановлюємо статичну IP-адресу 192.168.0.20 та маску підмережі 225.225.225.0 (рис. 5.9).

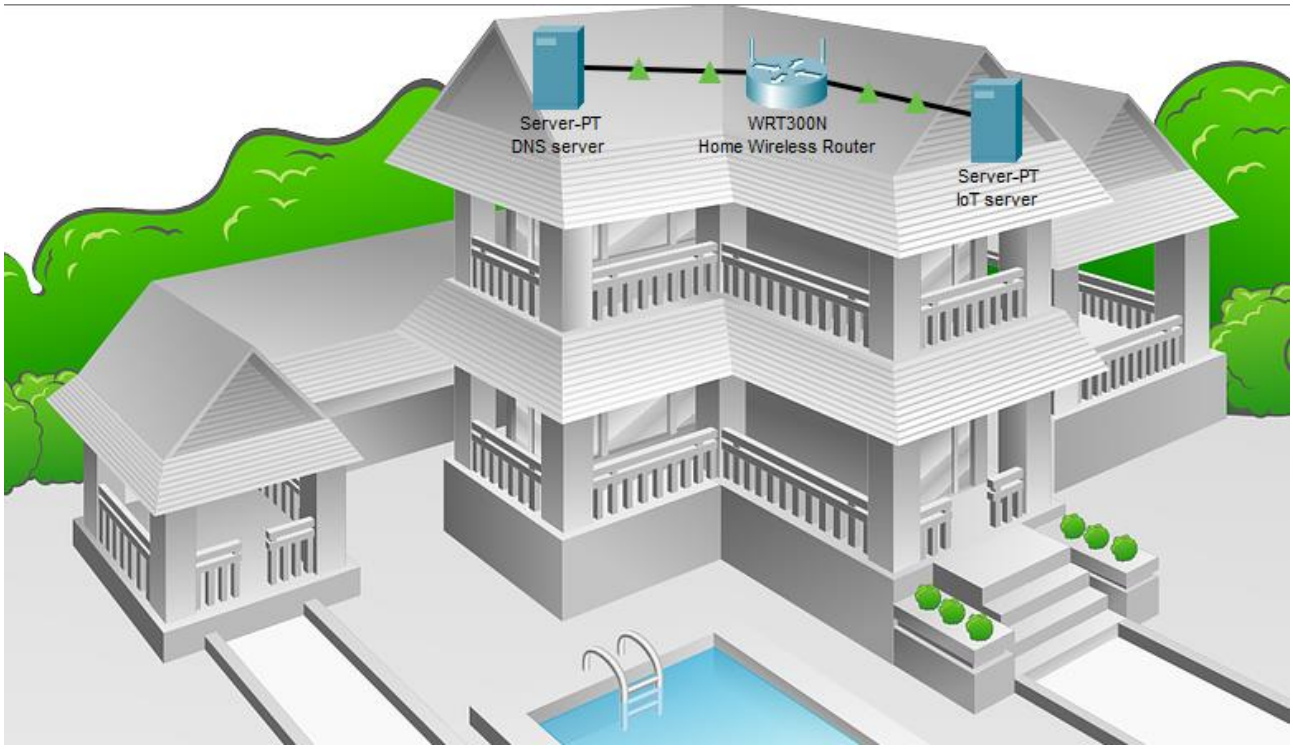


Рис. 5.7. Підключення серверів кабелем до домашнього маршрутизатора

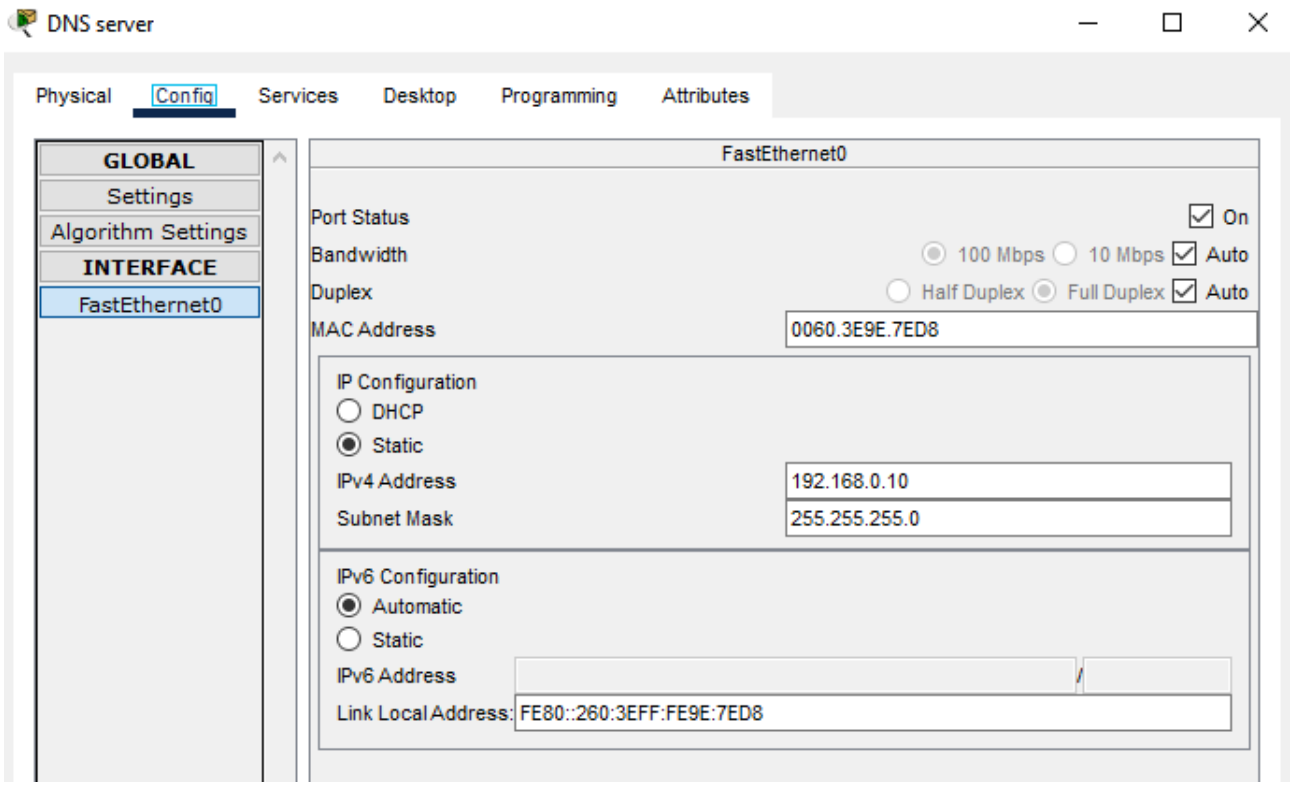


Рис. 5.8. Налаштування статичної IP-адреси для DNS сервера

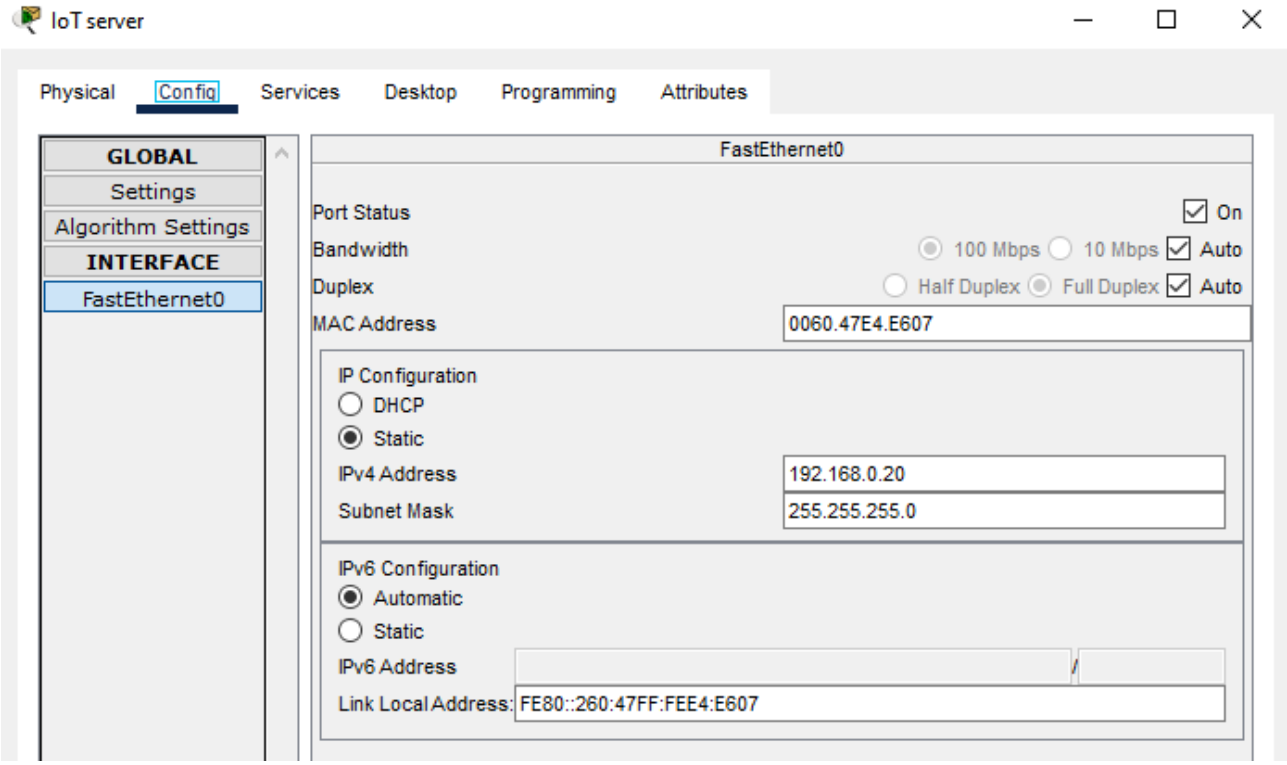


Рис. 5.9. Налаштування статичної IP-адреси для IoT сервера

Після того, як було задано статичні адреси для серверів, зокрема для DNS сервера, можемо налаштувати адресу DNS сервера на маршрутизаторі. Для цього відкриємо налаштування домашнього маршрутизатора, та в налаштуваннях DHCP встановимо адресу статичного DNS 192.168.0.10 і збережемо налаштування (рис. 5.10). В результаті ми налаштували адресу DNS сервера для домашнього маршрутизатора.

Для обох серверів, тобто для DNS та для IoT, налаштуємо адресу шлюзу за замовчуванням та DNS сервер. Адресу шлюзу за замовчуванням встановимо як 192.168.0.1 – бо це адреса домашнього маршрутизатора, а адресу DNS серверу встановимо як 192.168.0.10 – бо це є статичною адресою цього серверу. Вищесказане було налаштовано для обидвох серверів, результат цього налаштування зображено на рис. 5.11 та рис. 5.12.

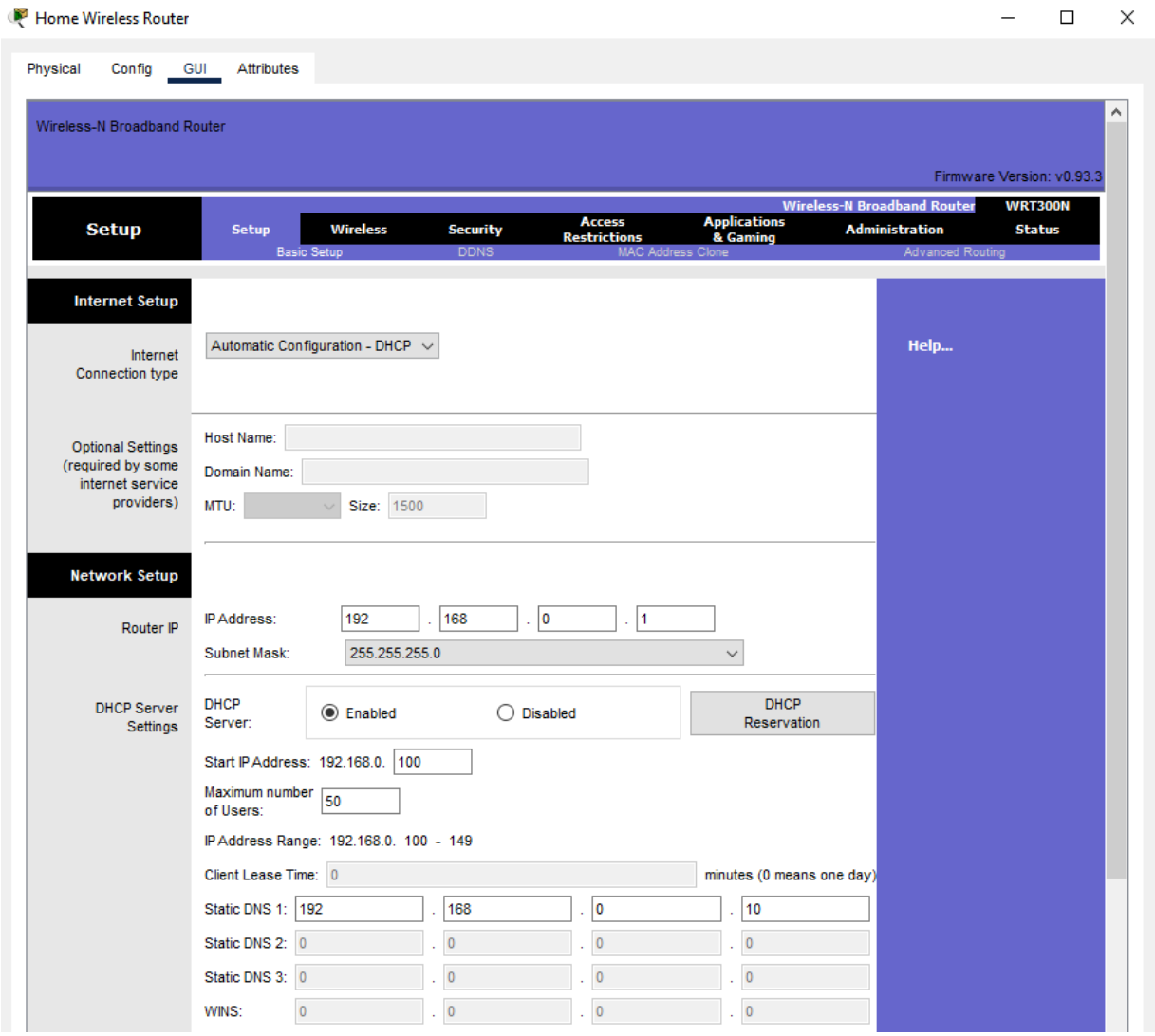


Рис. 5.10. Налаштування DNS для домашнього маршрутизатора

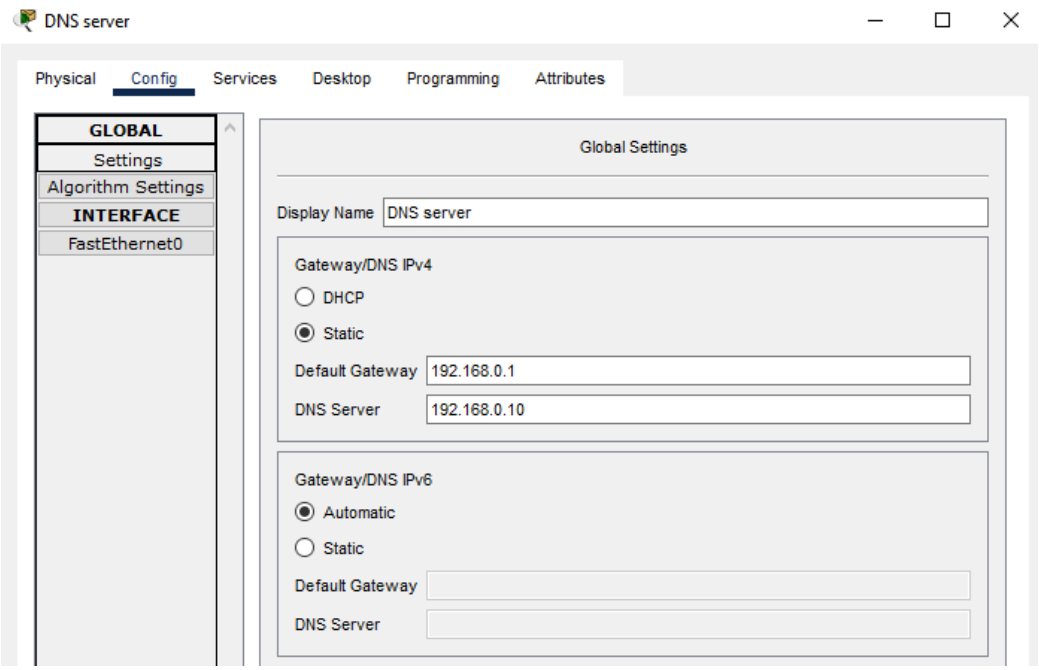


Рис. 5.11. Налаштування шлюзу за замовчуванням для DNS серверу

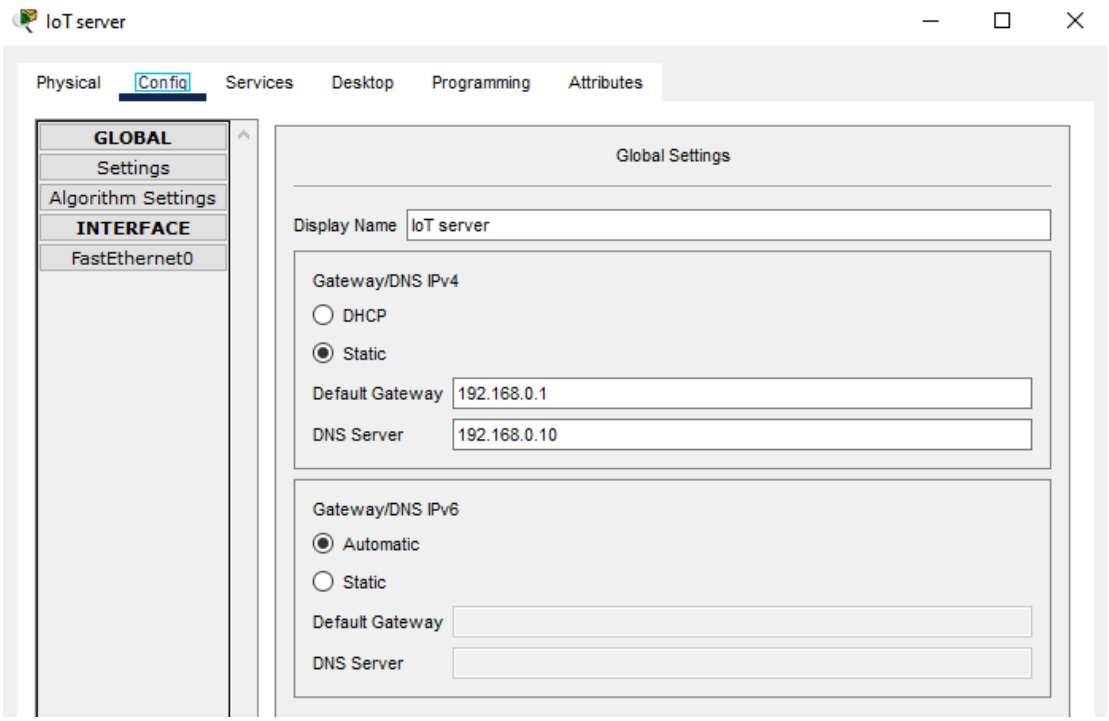


Рис. 5.12. Налаштування шлюзу за замовчуванням для IoT серверу

Відкриємо налаштування DNS серверу та перейдемо до сервісів. Відключимо усі сервіси, окрім DNS. DNS сервіс переведемо у ввімкнутий стан. Потім додаємо новий A Record з іменем my.home.net та адресою 192.168.0.20. Зберігаємо налаштування, результат зображено на рис. 5.13.

Таким чином, ми додали ім’я my.home.net, за яким буде відкриватись IP-адреса 192.168.0.20, тобто адреса IoT серверу.

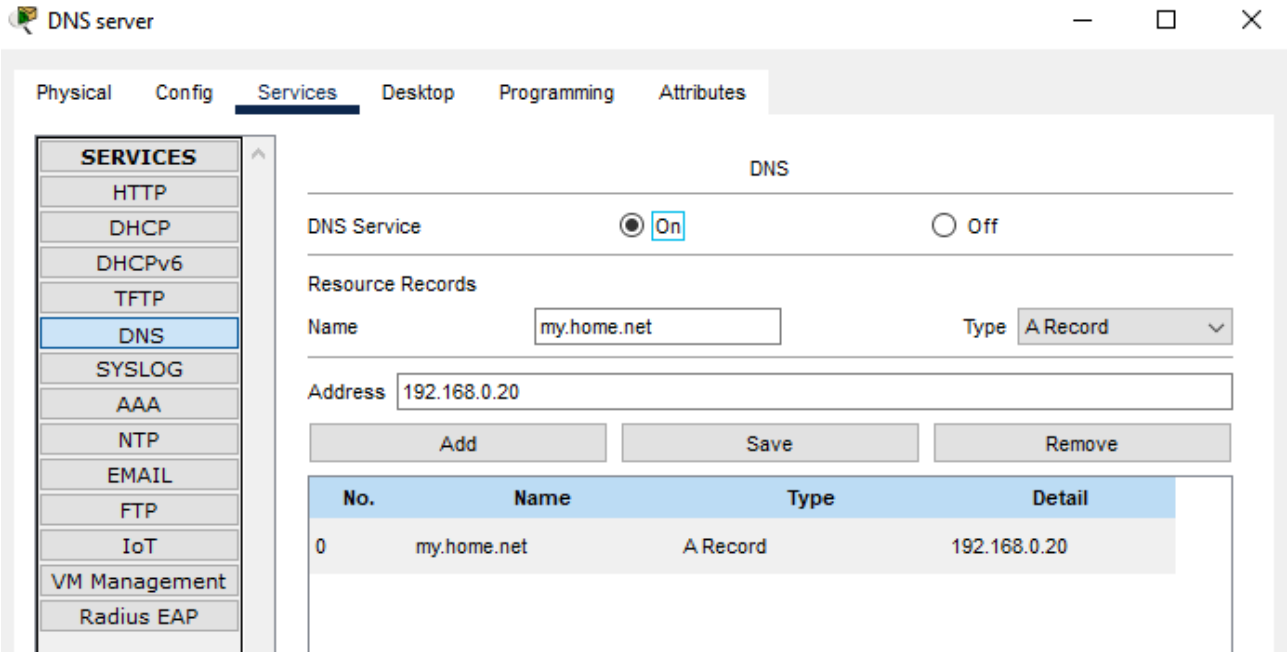


Рис. 5.13. Налаштування DNS та введення ім’я для IoT серверу

Тепер відкриємо налаштування IoT серверу, та перейдемо до сервісів. Відключимо усі сервіси, окрім IoT. IoT сервіс переведемо у ввімкнутий стан. Зберігаємо налаштування, результат зображено на рис. 5.14.

Таким чином, було включено сервіс IoT на цьому сервері, тепер до нього можна додавати нові IoT пристрої та переглядати їх стан або керувати ними за адресою my.home.net.

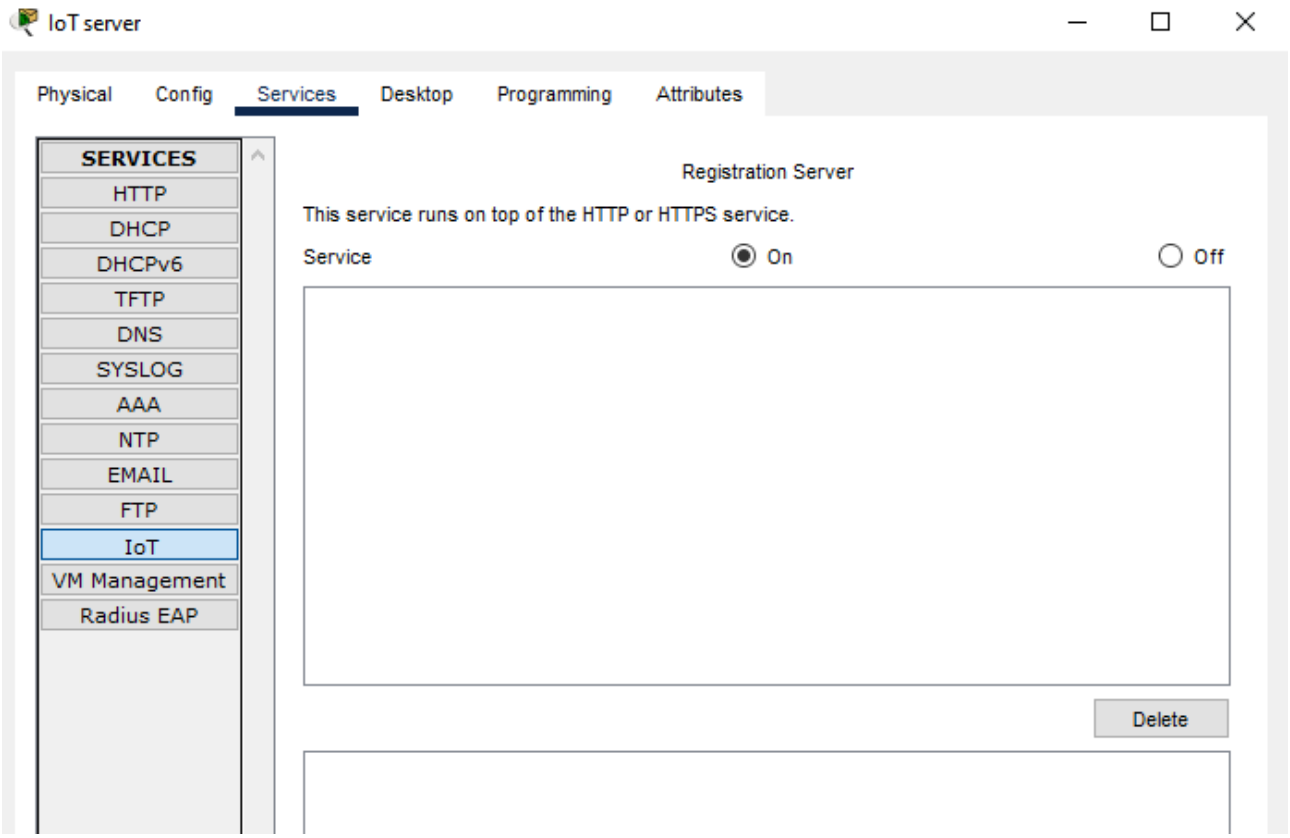


Рис. 5.14. Налаштування IoT сервісу

Підключимо планшет до домашньої мережі. Для цього введемо налаштування WLAN разом з SSID та паролем (рис. 5.15). Також встановимо налаштування DHCP (рис. 5.16).

Тепер можна спробувати підключитись за адресою my.home.net з планшету. Оскільки користувачів ще немає, то зареєструємо нового admin/admin, результат зображено на рис. 5.17. Після цього відкриється порожній список пристроїв IoT (рис. 5.18) – це означає, що все було виконано правильно. Тому можна перейти до наступних кроків – додавання пристроїв IoT.

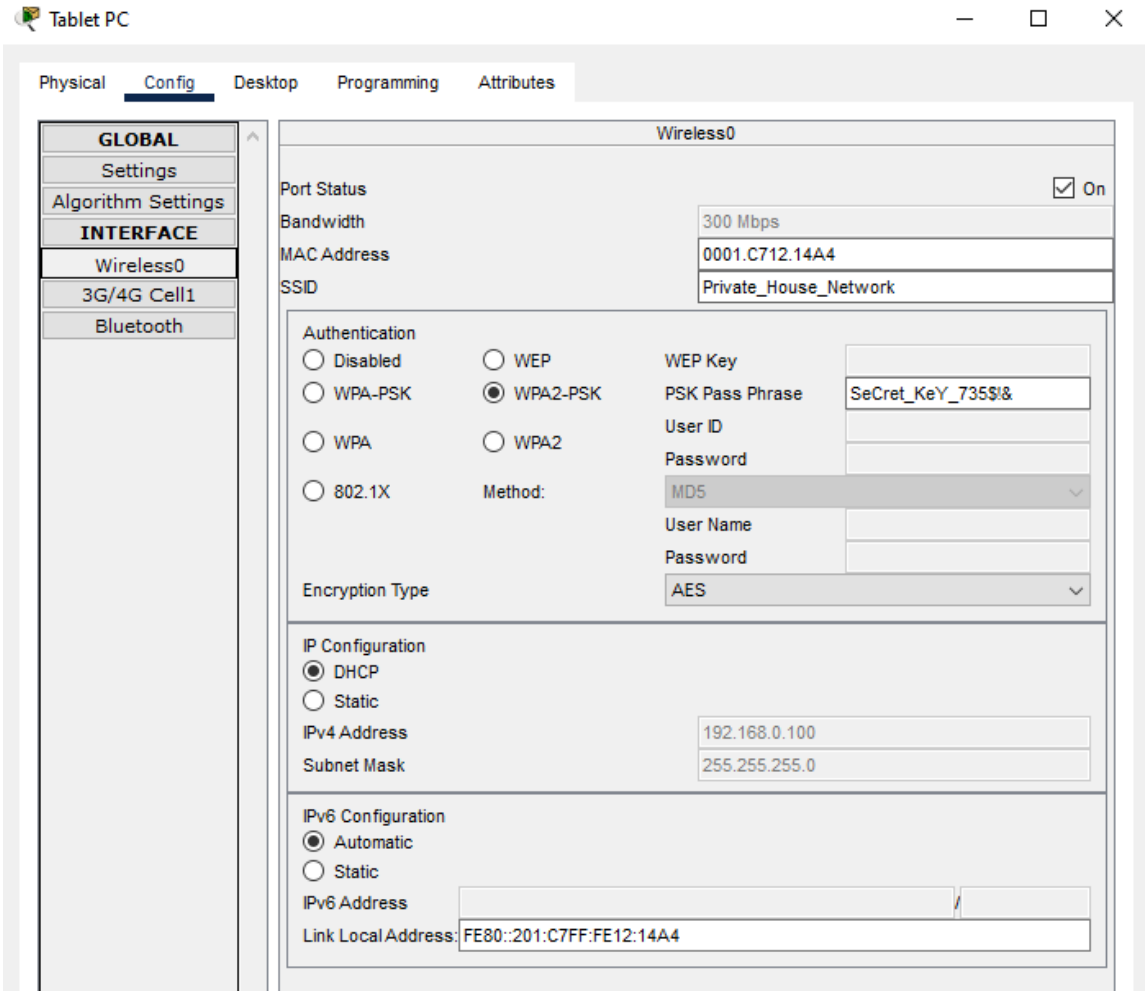


Рис. 5.15. Налаштування бездротової мережі для планшету

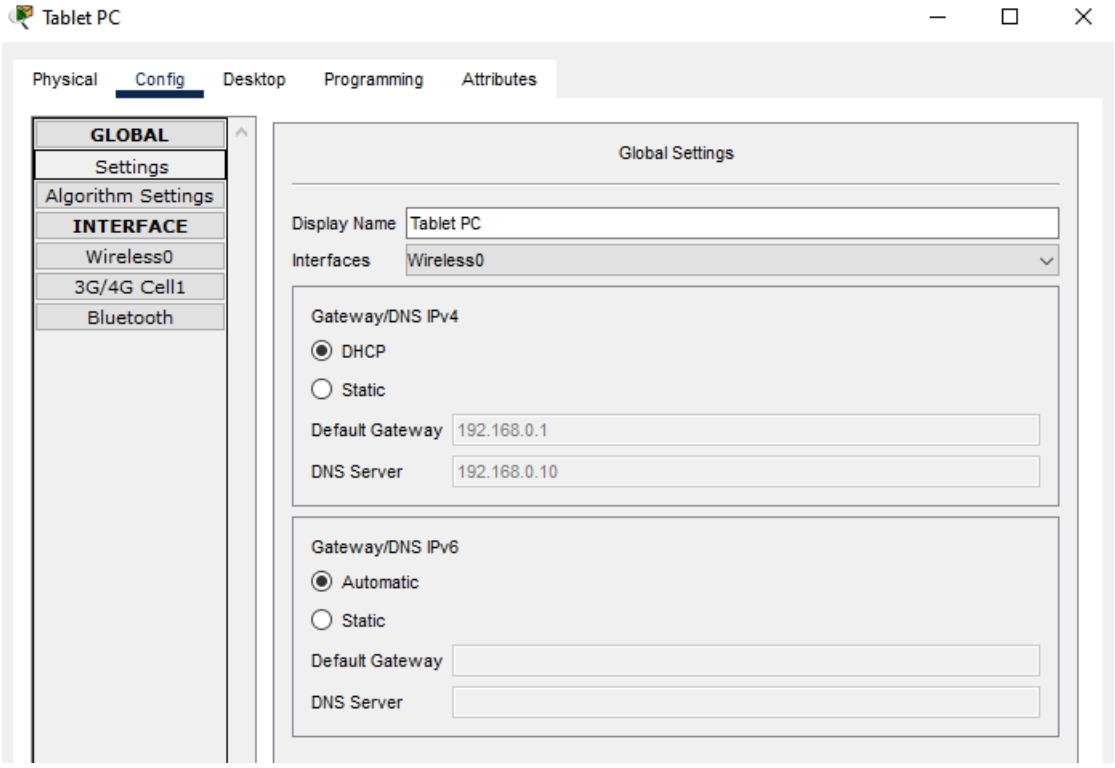


Рис. 5.16. Налаштування DHCP для планшету

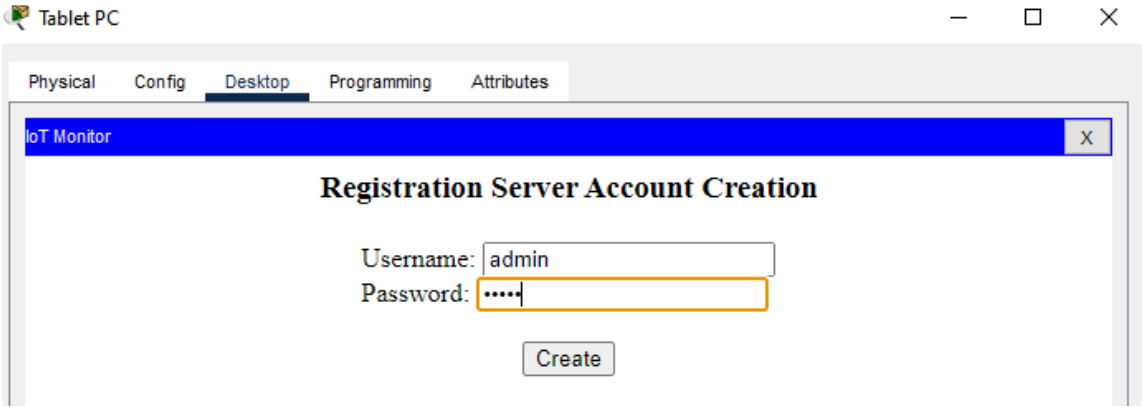


Рис. 5.17. Реєстрація в сервісі IoT з планшету

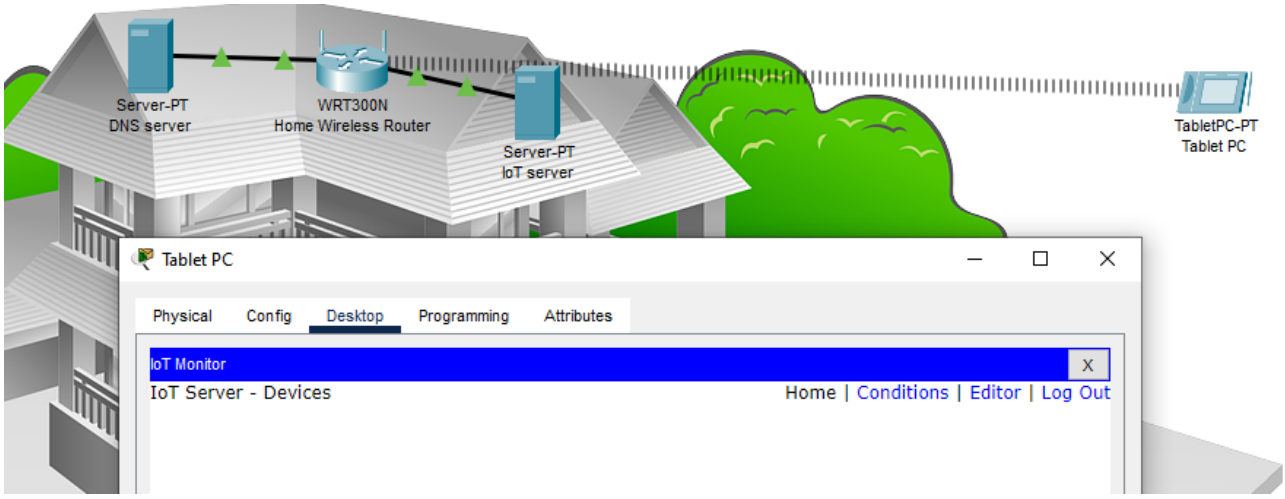


Рис. 5.18. Список пристроїв IoT

Приклад налаштування розумної системи спостереження за водою

Додаємо у середовище моделювання наступні пристрої: сирену, датчик рівня води, злив води, поливальник для газону та програмовану плату MCU (рис. 5.19). Налаштуємо для всіх доданих пристроїв, окрім MCU, налаштування бездротової мережі WLAN. Також налаштуємо для всіх IoT пристроїв сервер IoT, а саме встановимо адресу серверу my.home.net, ім’я користувача та пароль – admin та admin відповідно, і натиснемо кнопку підключення. Результат зображено на рис. 5.20. Також з’єднаємо сирену і злив води кабелем з платою MCU. Сирену під’єднаємо до цифрового піну з номером 0 на платі MCU, а злив води під’єднаємо до цифрового піну з номером 1 на платі MCU. Зчитувати рівень води будемо зі змінної середовища "Water Level". Результат з’єднання зображено на рис. 5.21.

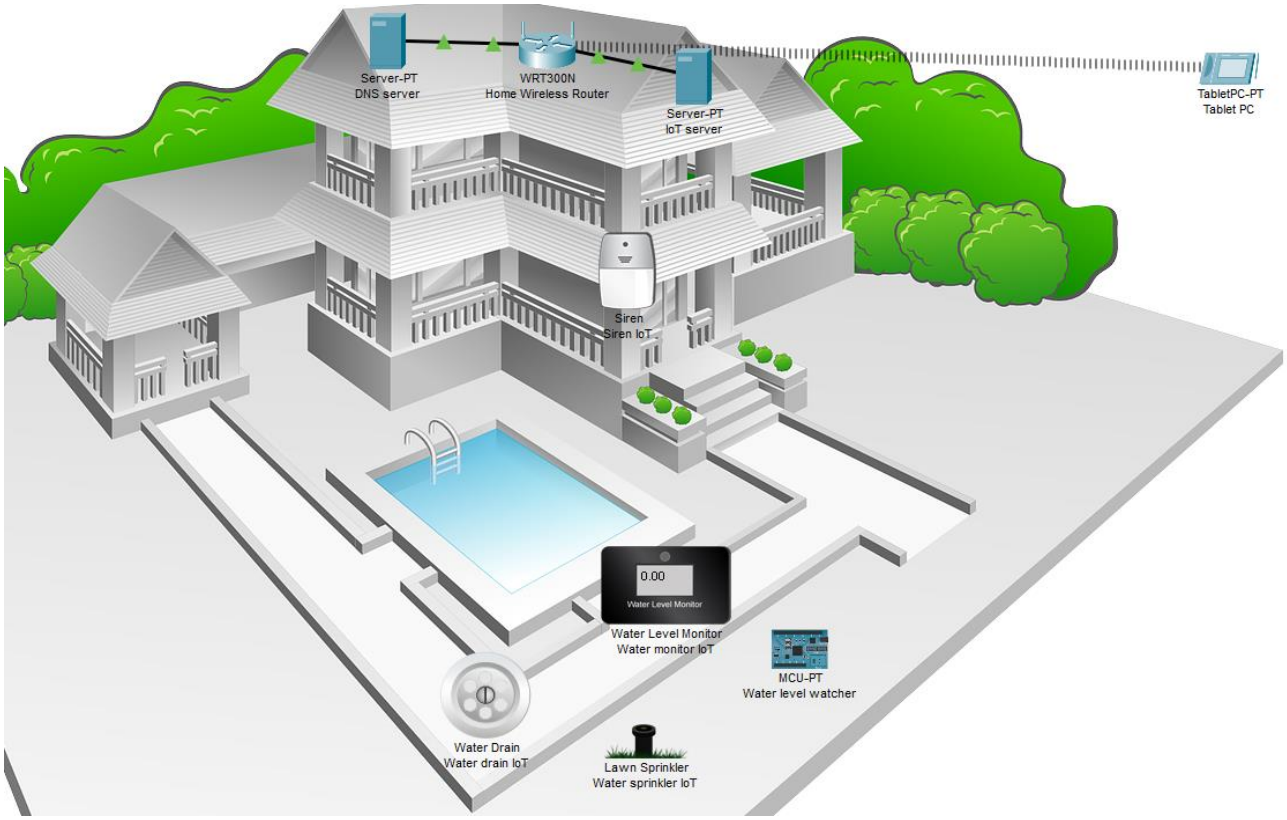


Рис. 5.19. Додання пристроїв для контролю рівня води

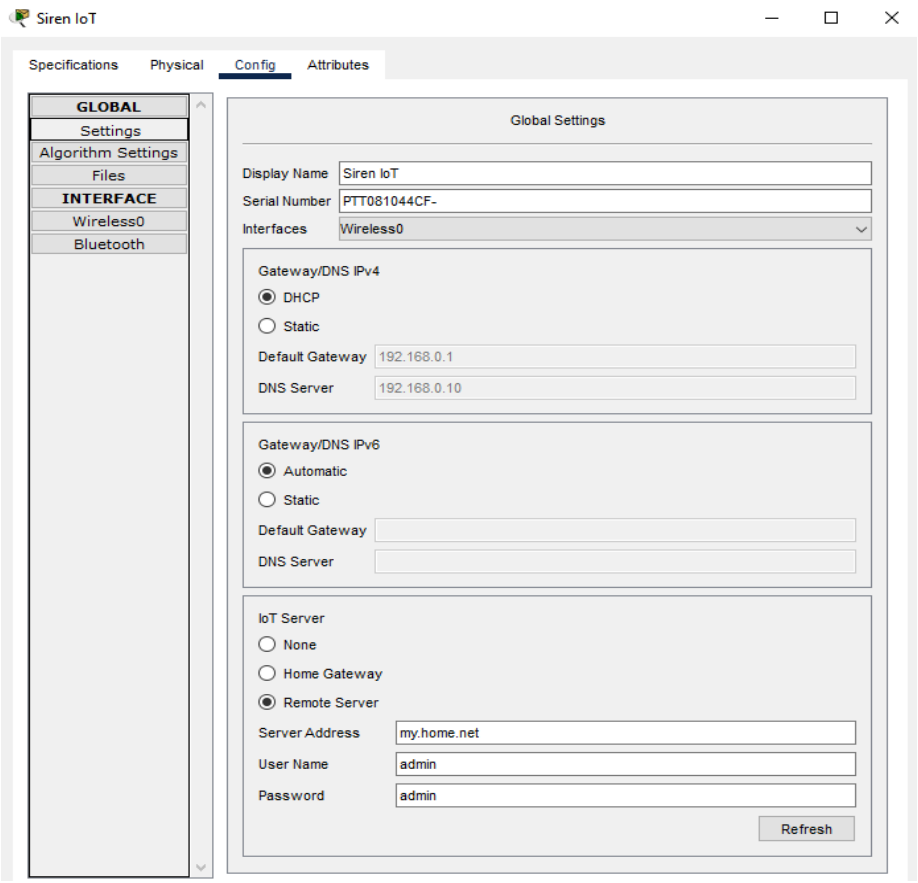


Рис. 5.20. Налаштування серверу IoT для пристроїв

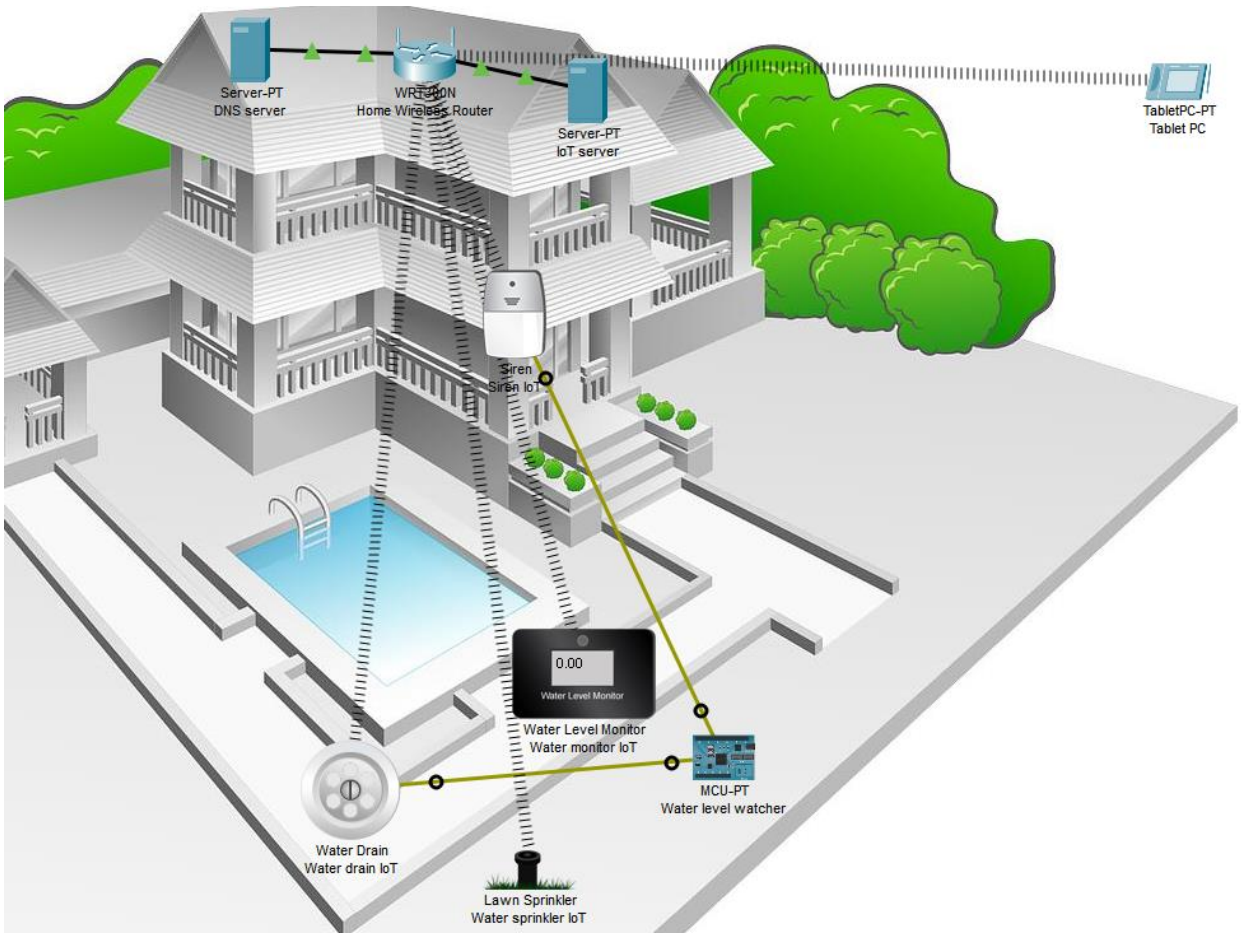


Рис. 5.21. З’єднання управляючих пристроїв IoT з платою MCU

Напишемо код логіки контролю рівня води для плати MCU на мові JavaScript.

При рівні більше 20 см, увімкнемо сирену та злив води. Як тільки рівень води впав нижче 0.01 см, то сирена вимикається і злив води теж.

Код програми зображено на рис. 5.22. Приклад роботи розумної системи контролю рівня води за допомогою плати MCU зображено на рис. 5.23.

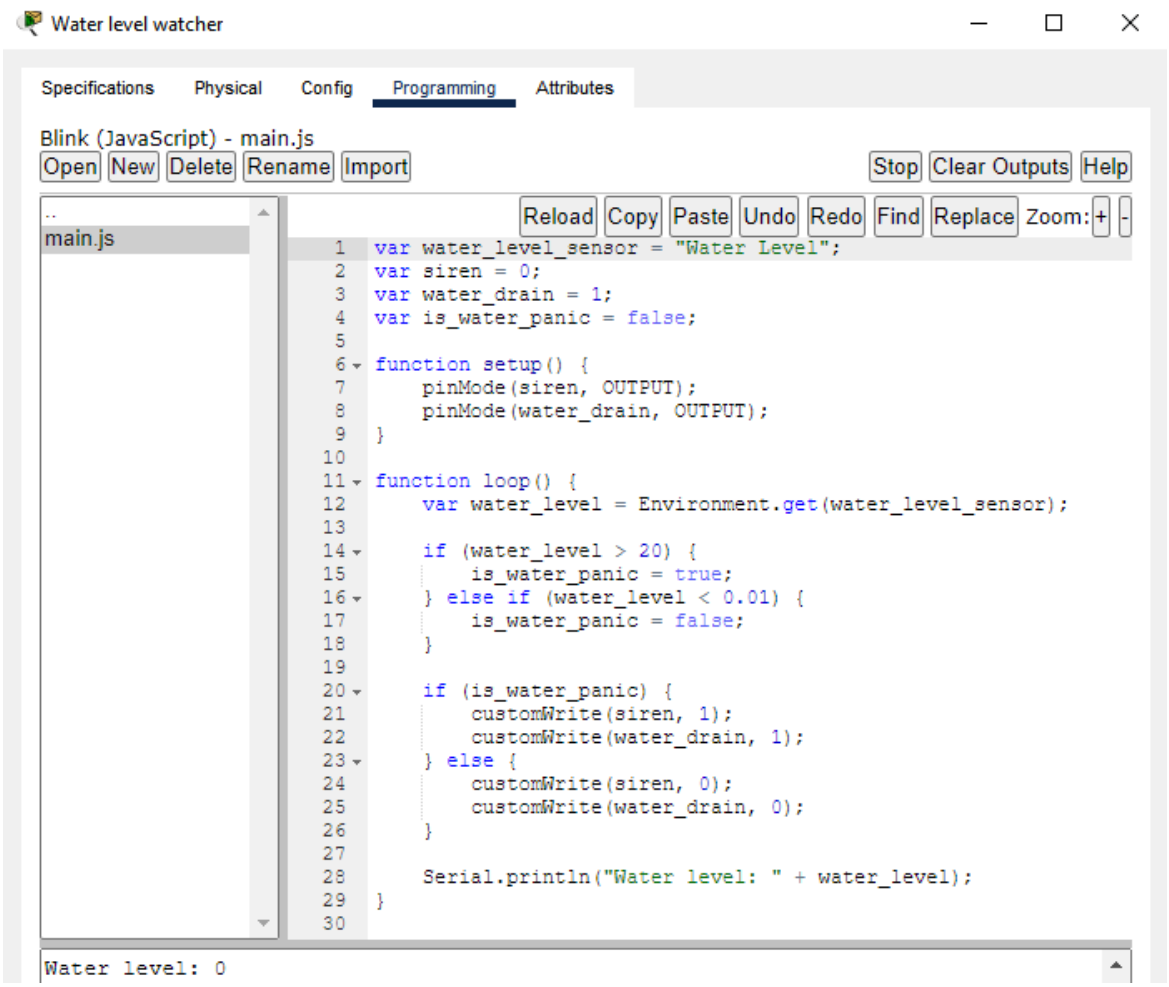


Рис. 5.22. Код контролювання рівня води для плати MCU

Рис. 5.23. Приклад роботи розумної системи контролю рівня води

Приклад налаштування розумної системи запобігання пожежі

Додамо наступні пристрої в середовище моделювання: датчик диму, датчик вогню, пожежний поливальник та дві програмовані плати MCU (рис. 5.24).

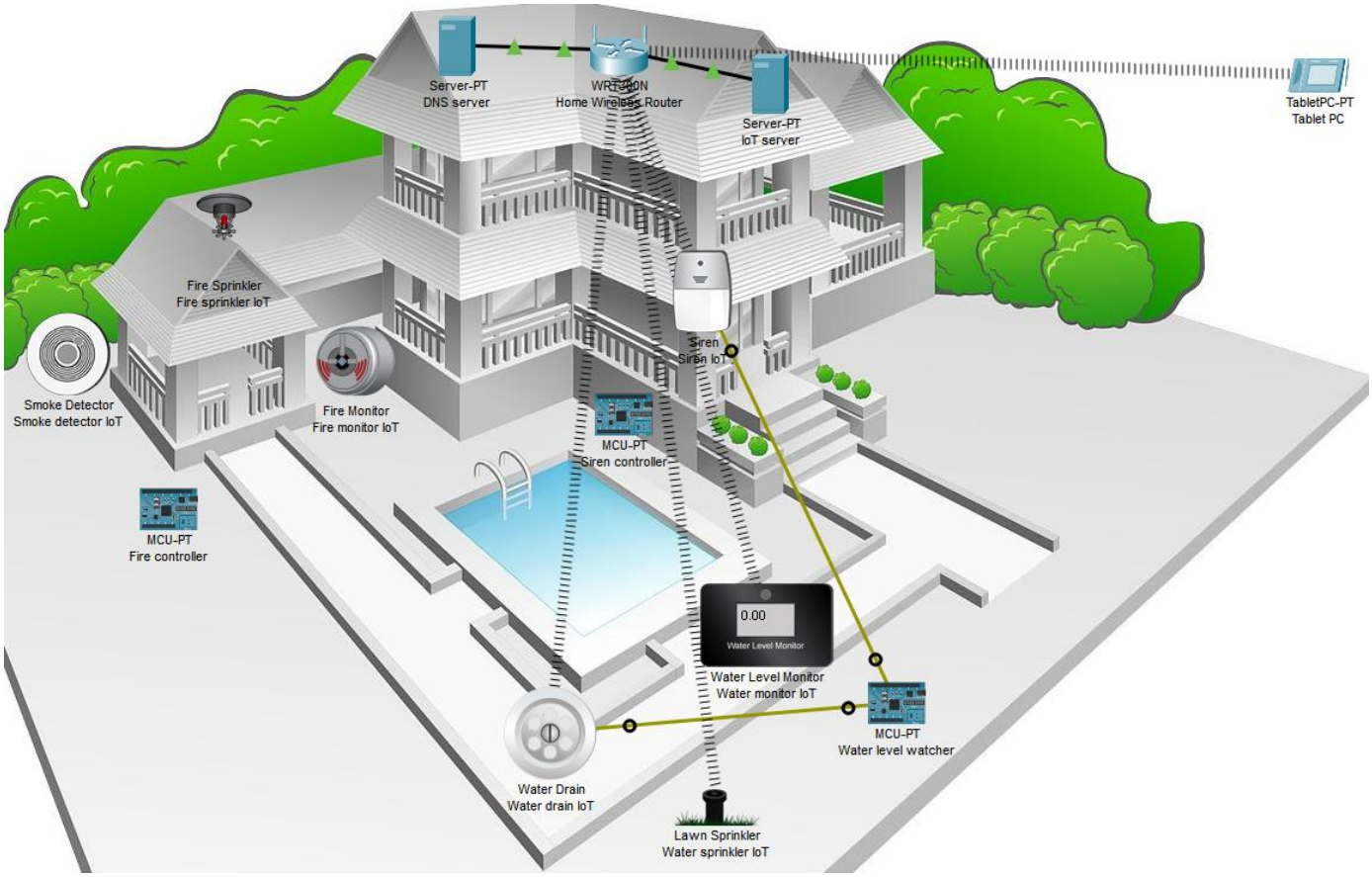


Рис. 5.24. Додання пристроїв для запобігання пожежі

Налаштуємо для всіх доданих пристроїв, окрім плат MCU, налаштування бездротової мережі WLAN. Також налаштуємо для всіх IoT пристроїв сервер IoT, а саме встановимо адресу серверу my.home.net, ім’я користувача та пароль – admin та admin відповідно, і натиснемо кнопку підключення.

Оскільки дві системи хочуть включати сирену, а сирена одна і порт в сирени лише один, то було додано ще одну плату, яка прийматиме два входи, і видаватиме один результуючий вихід за принципом АБО. Тобто при спрацюванні критичного рівня води або пожежі, сирена буде включатись.

З’єднаємо датчик вогню та пожежний поливальник кабелем з платою MCU.

Датчик вогню під’єднаємо до цифрового піну з номером 0 на платі MCU, а пожежний поливальник під’єднаємо до цифрового піну з номером 1 на платі MCU. Дану плату MCU цифровим піном 2 з’єднаємо з платою контролера сирени до цифрового піну 1.

Плату MCU контролю рівня води цифровим піном 0 з’єднаємо з платою контролера сирени до цифрового піну 2.

Плату контролера сирени цифровим піном 0 з’єднаємо з сиреною. Зчитувати рівень диму будемо зі змінної середовища "Smoke".

Результат з’єднання зображено на рис. 5.25.

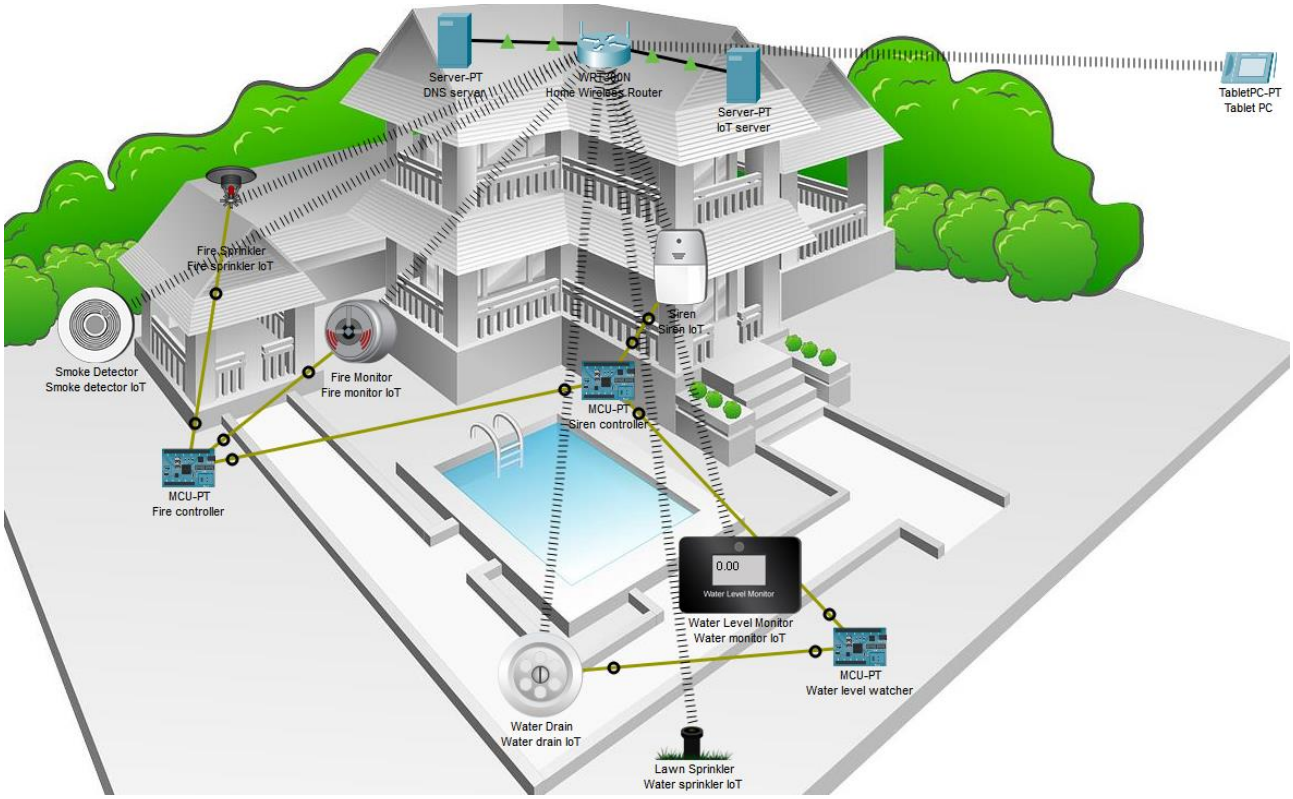


Рис. 5.25. З’єднання управляючих пристроїв IoT з платою MCU

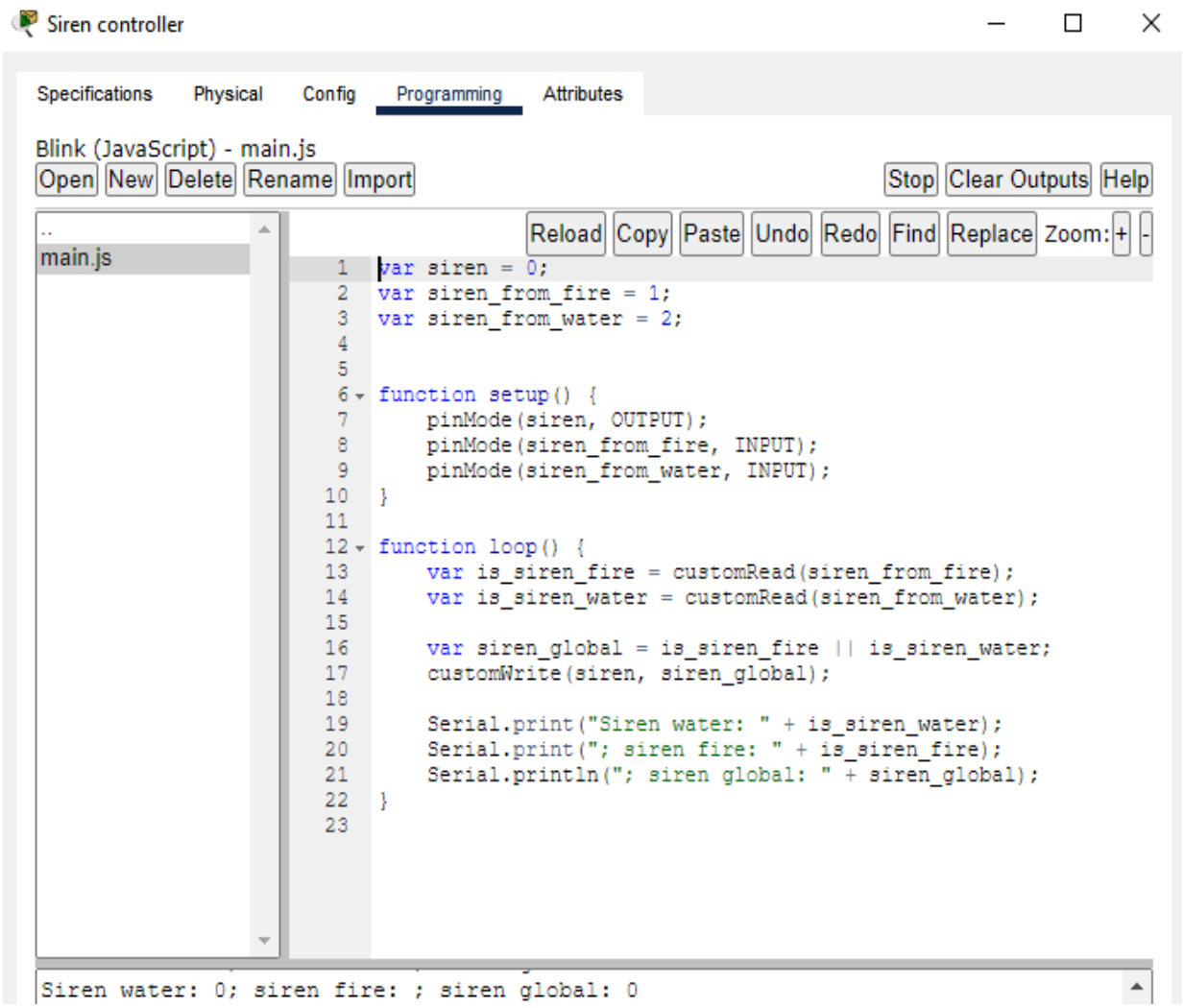


Рис. 5. 26. Код контролера сирени для плати MCU

Код логіки контролера сирени для плати MCU на мові JavaScript зображено на рис. 5.26.

Напишемо код логіки запобігання пожежі для плати MCU на мові JavaScript. При наявності вогню або якщо рівень диму більше 0.10, то вмикається сирена та пожежний поливальник.

Як тільки вогонь зник та рівень диму впав нижче 0.0001, то сирена вимикається і пожежний поливальник теж. Код для такого сценарію зображено на рис. 5.27.

Приклад роботи розумної системи запобігання пожежі за допомогою плати MCU зображено на рис. 5.28.

Fire controller

Specifications | Physical | Config | Programming | Attributes

Blink (JavaScript) - main.js

Open | New | Delete | Rename | Import | Stop | Clear Outputs | Help

Reload | Copy | Paste | Undo | Redo | Find | Replace | Zoom: + -

..
main.js

```
1  var smoke_detector_sensor = "Smoke";
2  var siren = 2;
3  var fire_monitor = 0;
4  var fire_sprinkler = 1;
5  var is_fire_panic = false;
6
7  function setup() {
8      pinMode(siren, OUTPUT);
9      pinMode(fire_sprinkler, OUTPUT);
10     pinMode(fire_monitor, INPUT);
11 }
12
13 function loop() {
14     var smoke_level = Environment.get(smoke_detector_sensor);
15     var fire_detected = digitalRead(fire_monitor);
16
17     if (fire_detected == HIGH || smoke_level > 0.10) {
18         is_fire_panic = true;
19     } else if (fire_detected == LOW && smoke_level < 0.0001) {
20         is_fire_panic = false;
21     }
22
23     if (is_fire_panic) {
24         customWrite(siren, 1);
25         customWrite(fire_sprinkler, 1);
26     } else {
27         customWrite(siren, 0);
28         customWrite(fire_sprinkler, 0);
29     }
30
31     Serial.print("Fire detected: " + fire_detected);
32     Serial.println("; smoke level: " + smoke_level);
33 }
34
```

```
Fire detected: 0; smoke level: 0.000002861019993360969 2
```

Рис. 5.27. Код запобігання пожежі для плати MCU

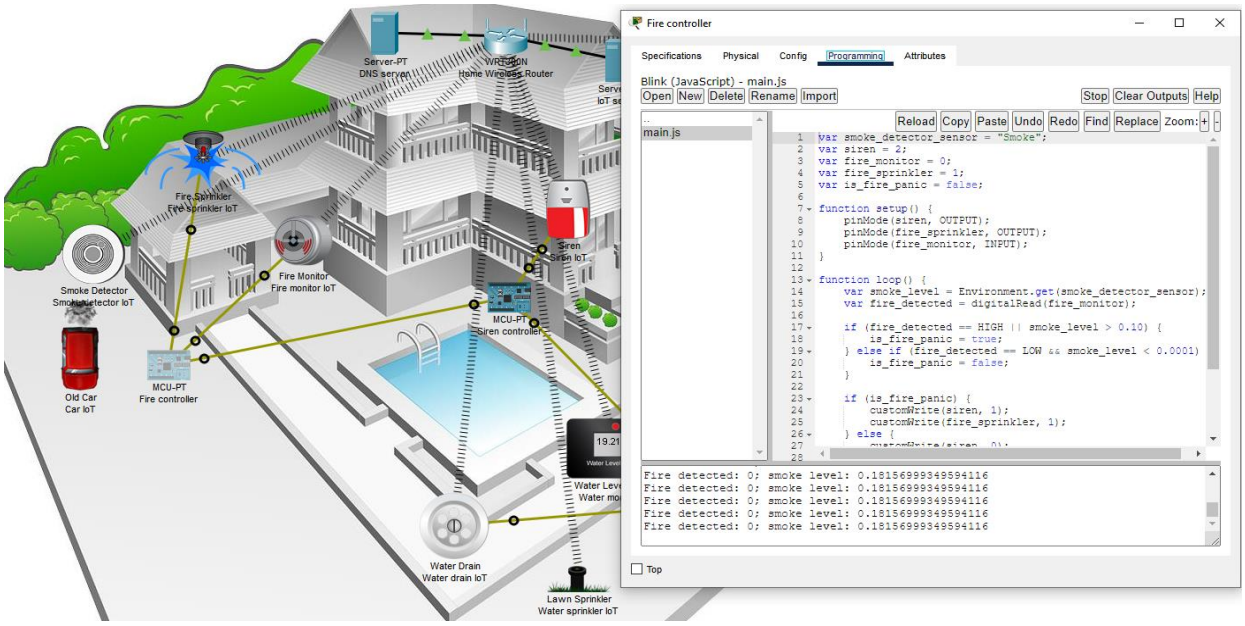


Рис. 5.28. Приклад роботи розумної системи запобігання пожежі

Приклад налаштування розумної системи контролю швидкості вітру

У середовище моделювання додаємо наступні пристрої: вікно, датчик вітру та програмовану плату MCU (рис. 5.29).

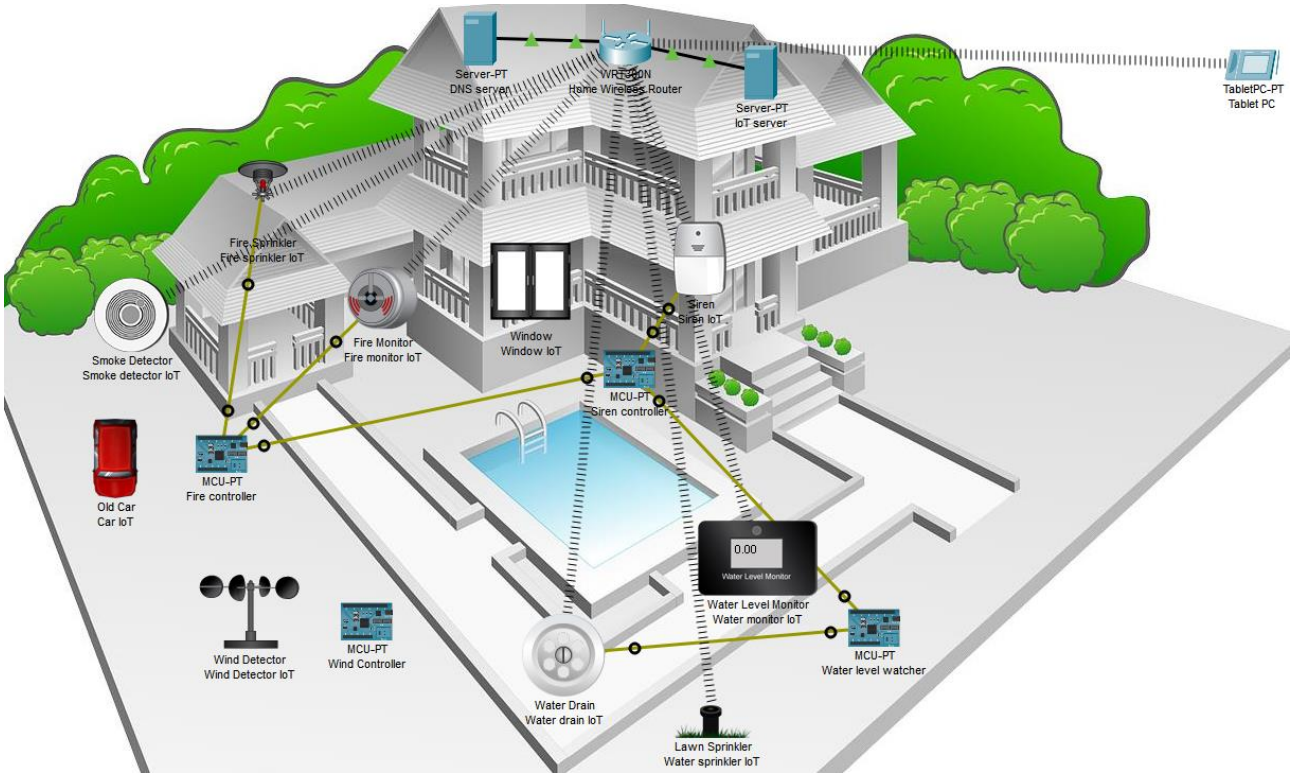


Рис. 5.29. Додання пристроїв для контролю швидкості вітру

Налаштуємо для всіх доданих пристроїв, окрім плат MCU, налаштування бездротової мережі WLAN. Також налаштуємо для всіх IoT пристроїв сервер IoT, а саме встановимо адресу серверу my.home.net, ім’я користувача та пароль – admin та admin відповідно, і натиснемо кнопку підключення.

З’єднаємо вікно кабелем з платою MCU. Вікно під’єднаємо до цифрового піну з номером 0 на платі MCU. Зчитувати швидкість вітру будемо зі змінної середовища "Wind Speed". Результат з’єднання зображено на рис. 5.30.

Напишемо код логіки контролю швидкості вітру для плати MCU на мові JavaScript. Якщо швидкість вітру буде більше 0.2, то вікно зачиняється. Як тільки швидкість вітру впаде нижче 0.1, то вікно відкриється. Код зображено на рис. 5.31.

Приклад роботи розумної системи контролю швидкості вітру за допомогою плати MCU зображено на рис. 5.32.

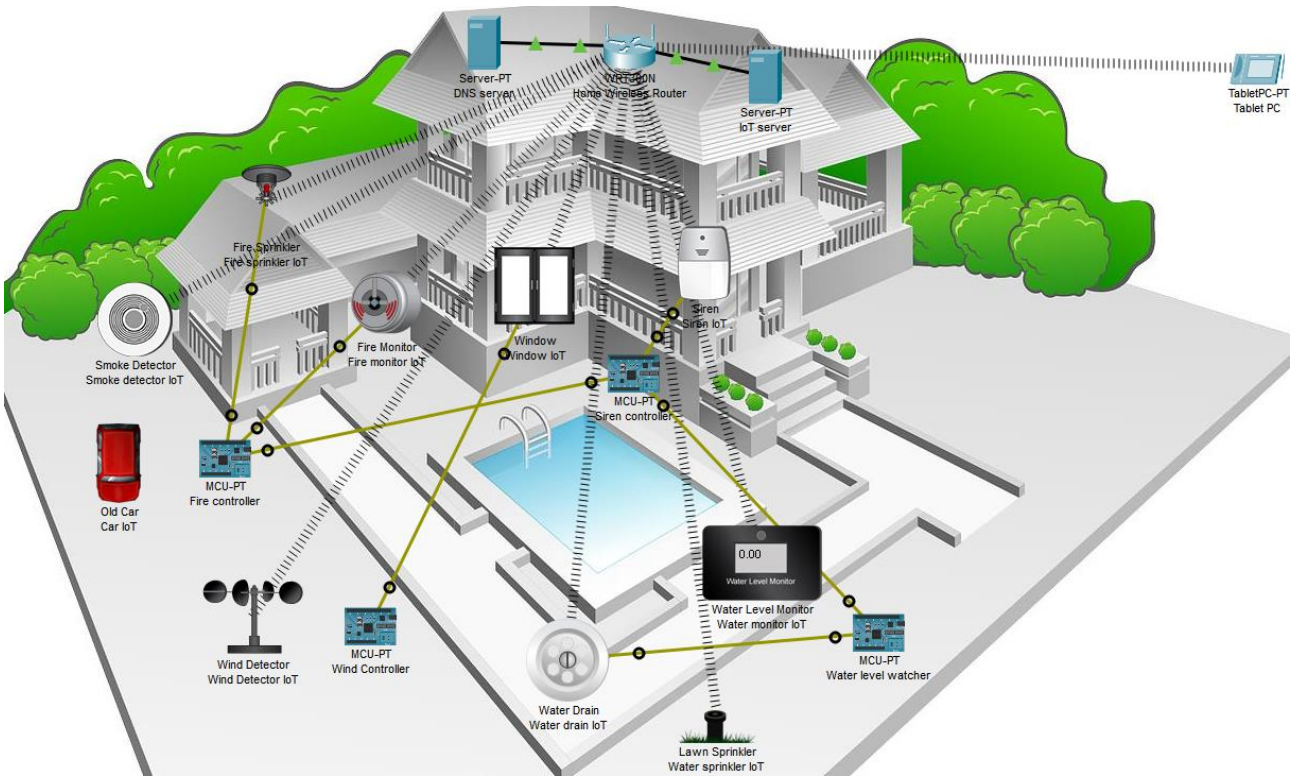


Рис. 5.30. З’єднання управляючих пристроїв IoT з платою MCU

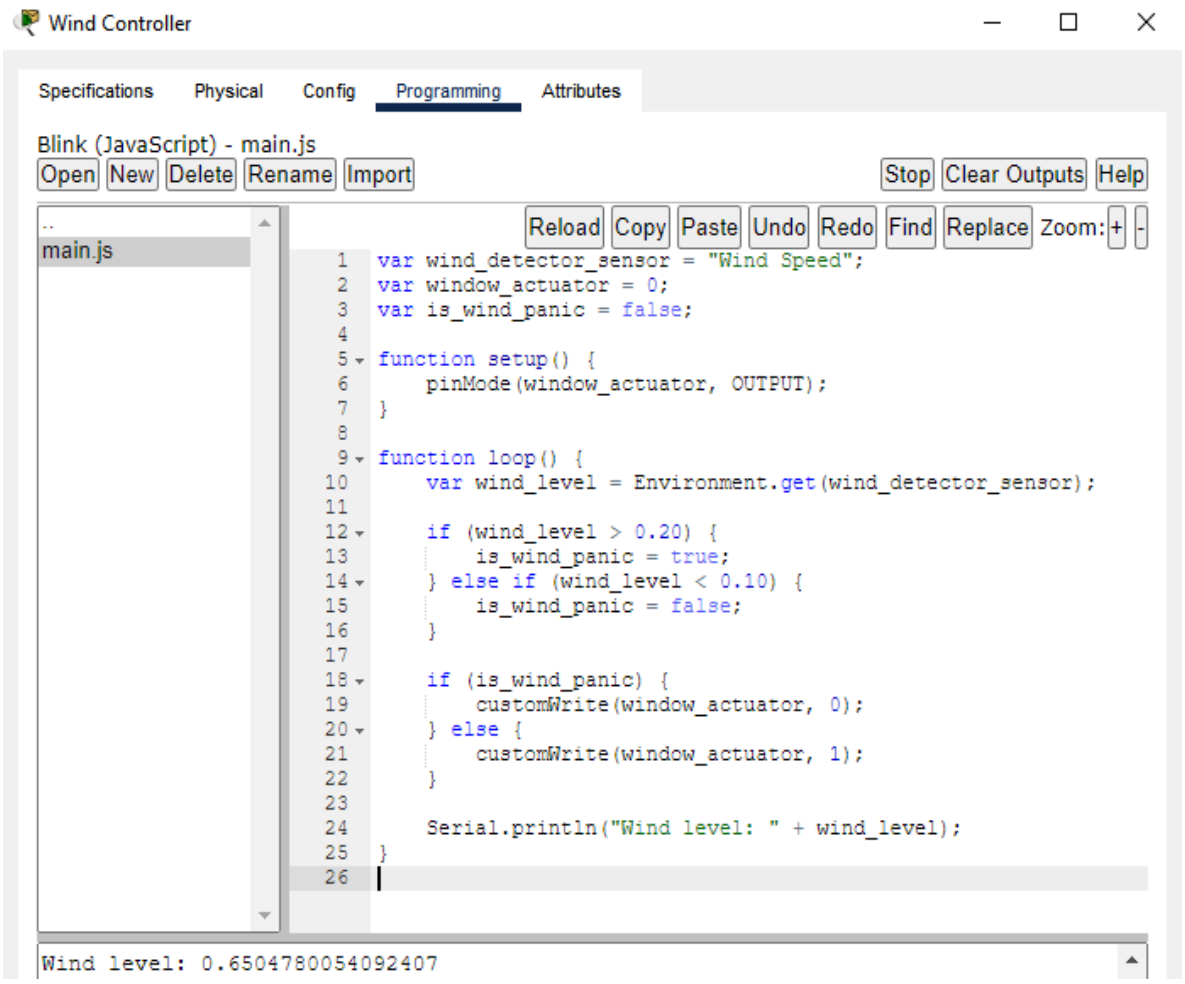


Рис. 5.31. Код контролю швидкості вітру для плати MCU

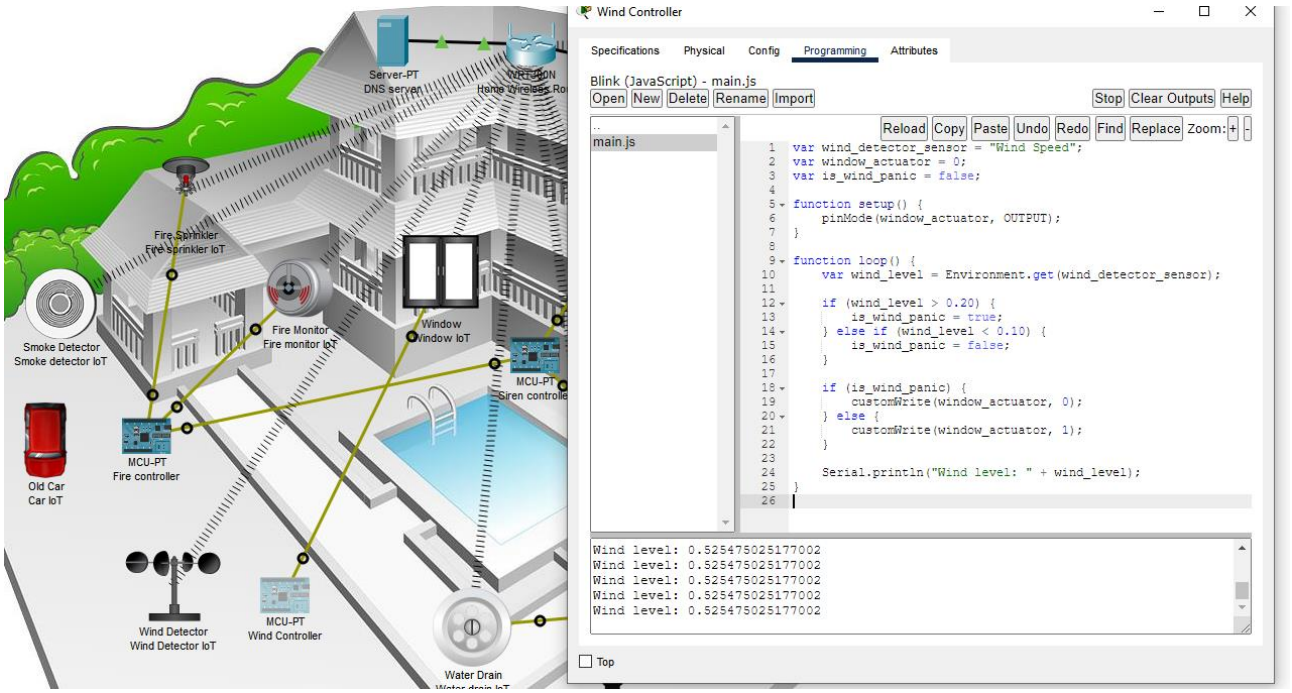


Рис. 5.32. Приклад роботи розумної системи контролю швидкості вітру

Приклад налаштування розумної системи контролю переміщень

У середовище моделювання додаємо наступні пристрої: вебкамеру, датчик руху, сенсор руху та програмовану плату MCU (рис. 5.33).

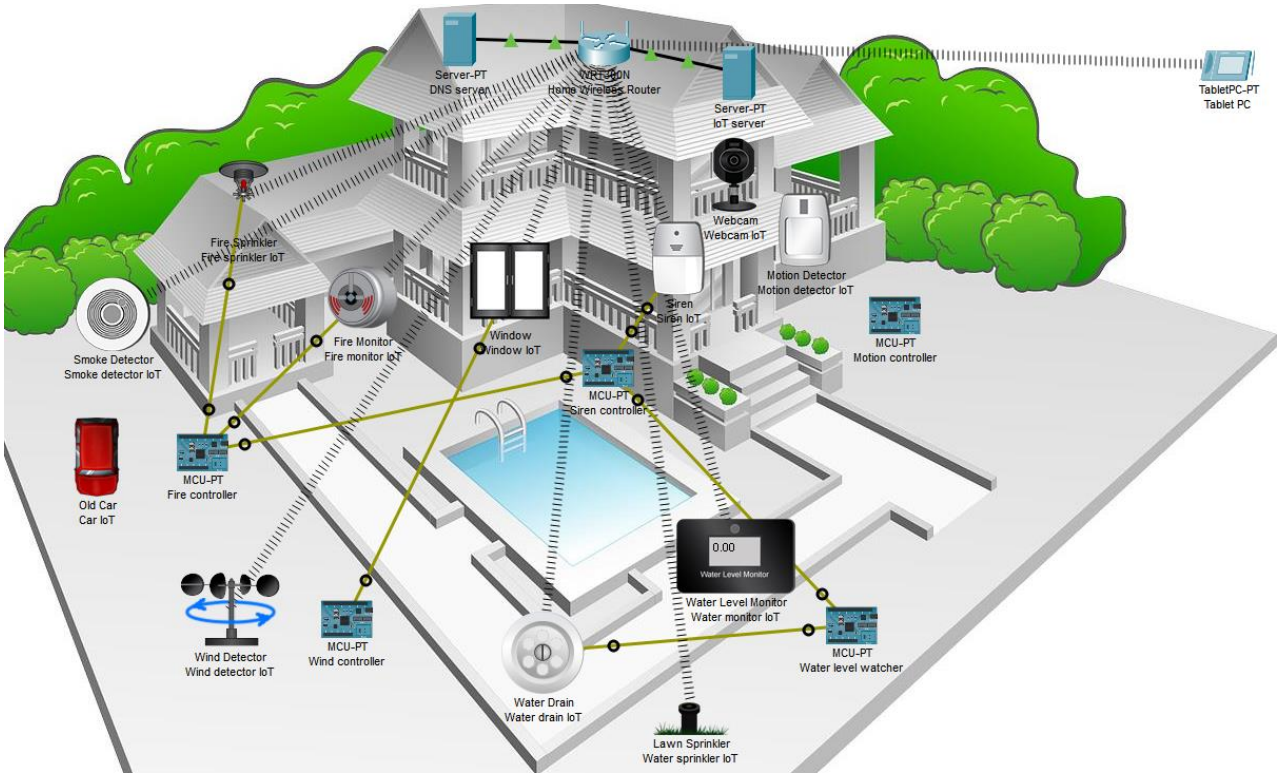


Рис. 5.33. Додавання пристроїв для контролю переміщень

Налаштуємо для всіх доданих пристроїв, окрім плат MCU та сенсору руху, налаштування бездротової мережі WLAN. Також налаштуємо для всіх IoT пристроїв сервер IoT, а саме встановимо адресу серверу my.home.net, ім’я користувача та пароль – admin та admin відповідно, і натиснемо кнопку підключення.

З’єднаємо вебкамеру та сенсор руху кабелем з платою MCU. Вебкамеру під’єднаємо до цифрового піну з номером 0 на платі MCU, а сенсор руху – до цифрового піну з номером 1. Результат з’єднання зображено на рис. 5.34.

Напишемо код логіки контролю переміщень для плати MCU на мові JavaScript. Якщо сенсор зафіксував переміщення, то вебкамера включиться. Як тільки будь-який рух припиниться, то вебкамера вимкнеться. Код програми зображено на рис. 5.35.

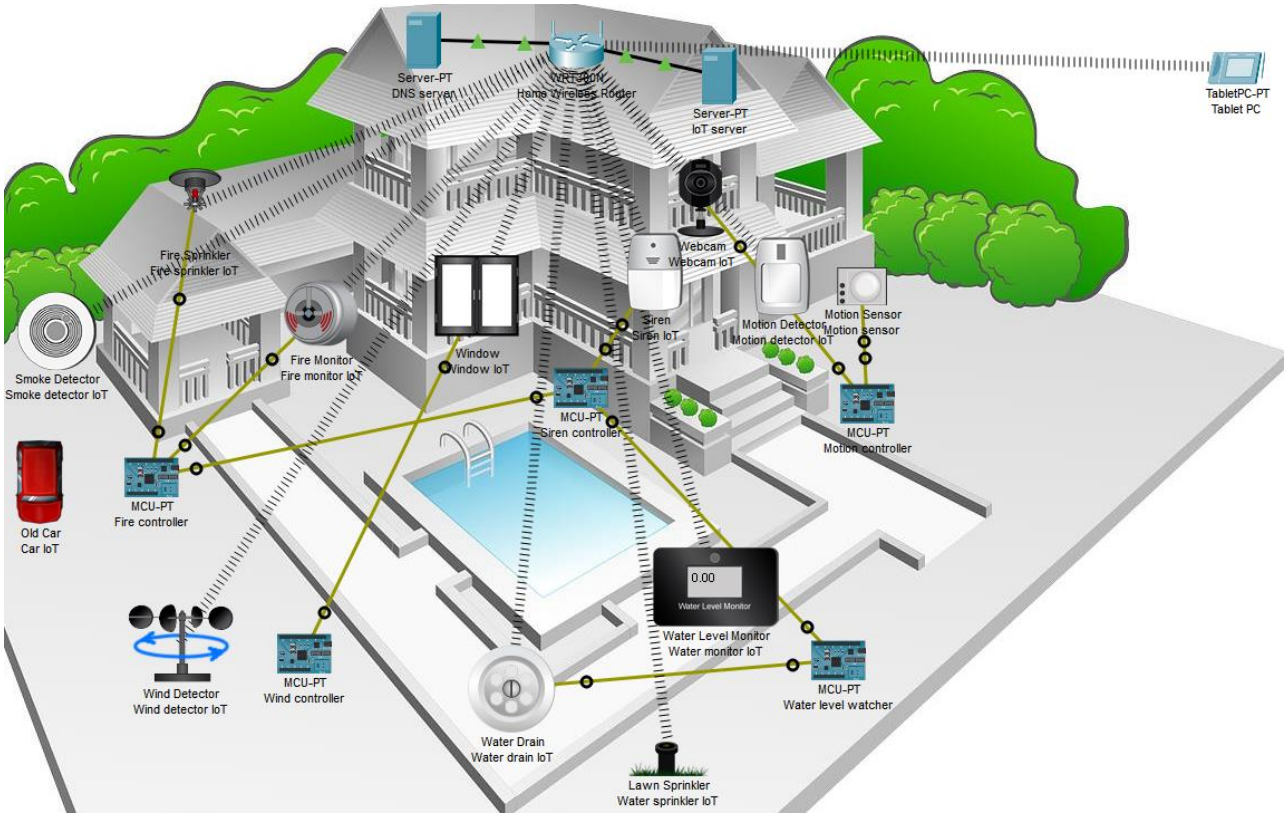


Рис. 5.34. З’єднання управляючих пристроїв IoT з платою MCU

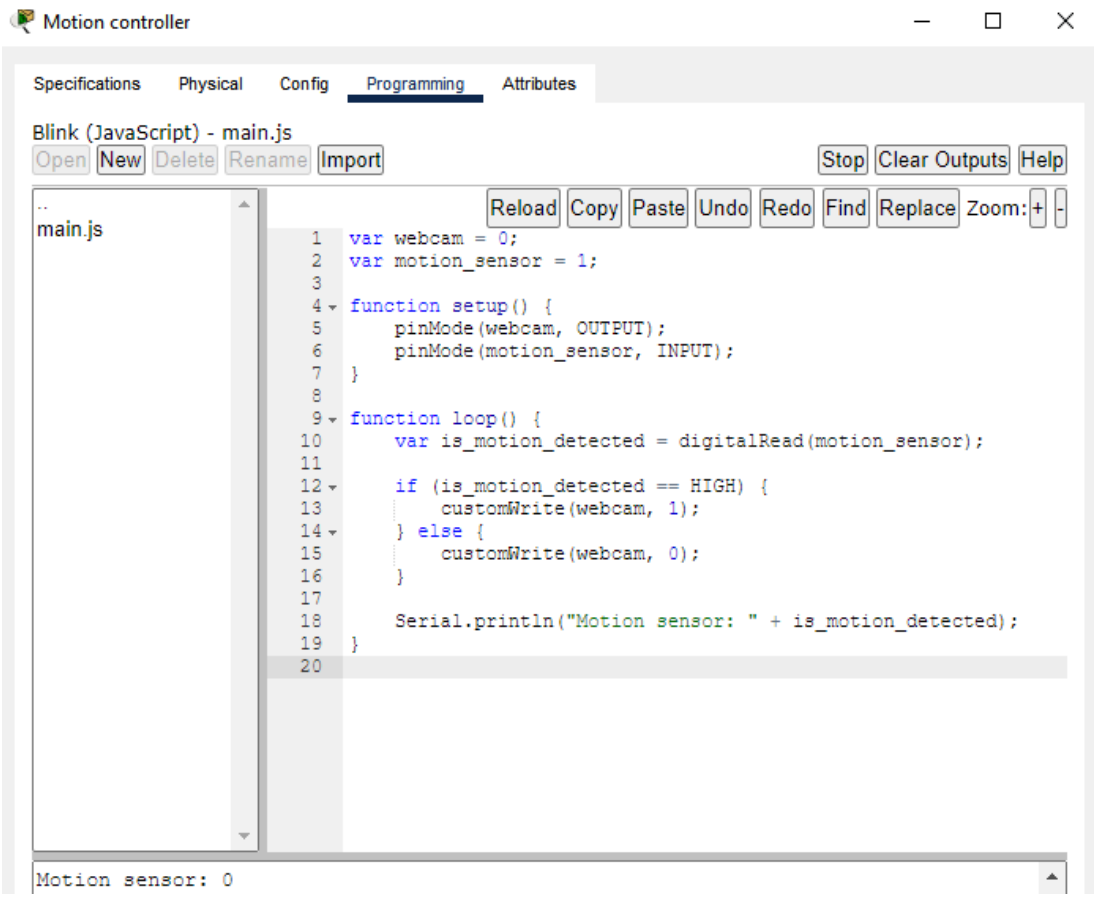


Рис. 5.35. Код контролю переміщень для плати MCU

Приклад налаштування розумної системи контролю шуму

У середовище моделювання додаємо наступні пристрої: світильник, сенсор звуку та програмовану плату MCU (рис. 5.36).

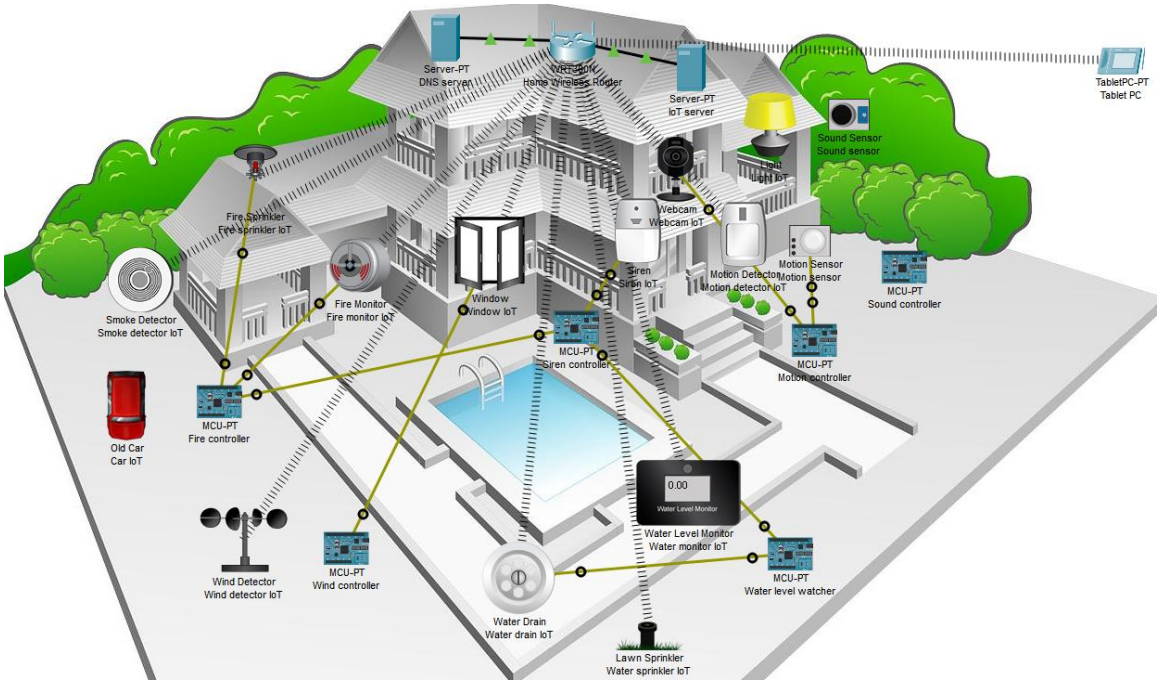


Рис. 5.36. Додавання пристроїв для контролю переміщень

Виконаємо для всіх доданих пристроїв, окрім плат MCU та сенсору звуку, налаштування бездротової мережі WLAN. Також налаштуємо для всіх IoT пристроїв сервер IoT, а саме встановимо адресу серверу my.home.net, ім’я користувача та пароль – admin та admin відповідно, і натиснемо кнопку підключення.

З’єднаємо світильник та сенсор звуку кабелем з платою MCU. Світильник під’єднаємо до цифрового піну з номером 0 на платі MCU, а сенсор звуку – до аналогового піну з номером 0. Результат з’єднання зображено на рис. 5.37.

Напишемо код логіки контролю шуму для плати MCU на мові JavaScript. Якщо сенсор звуку зафіксував звук більше 20 ДБ, то світло включається на максимальну потужність. Якщо від 20 ДБ до 10 ДБ – то наполовину, а якщо менше 10 ДБ, то світло виключається. Код програми зображено на рис. 5.38.

На рис. 5.39 зображено приклад роботи даної системи.

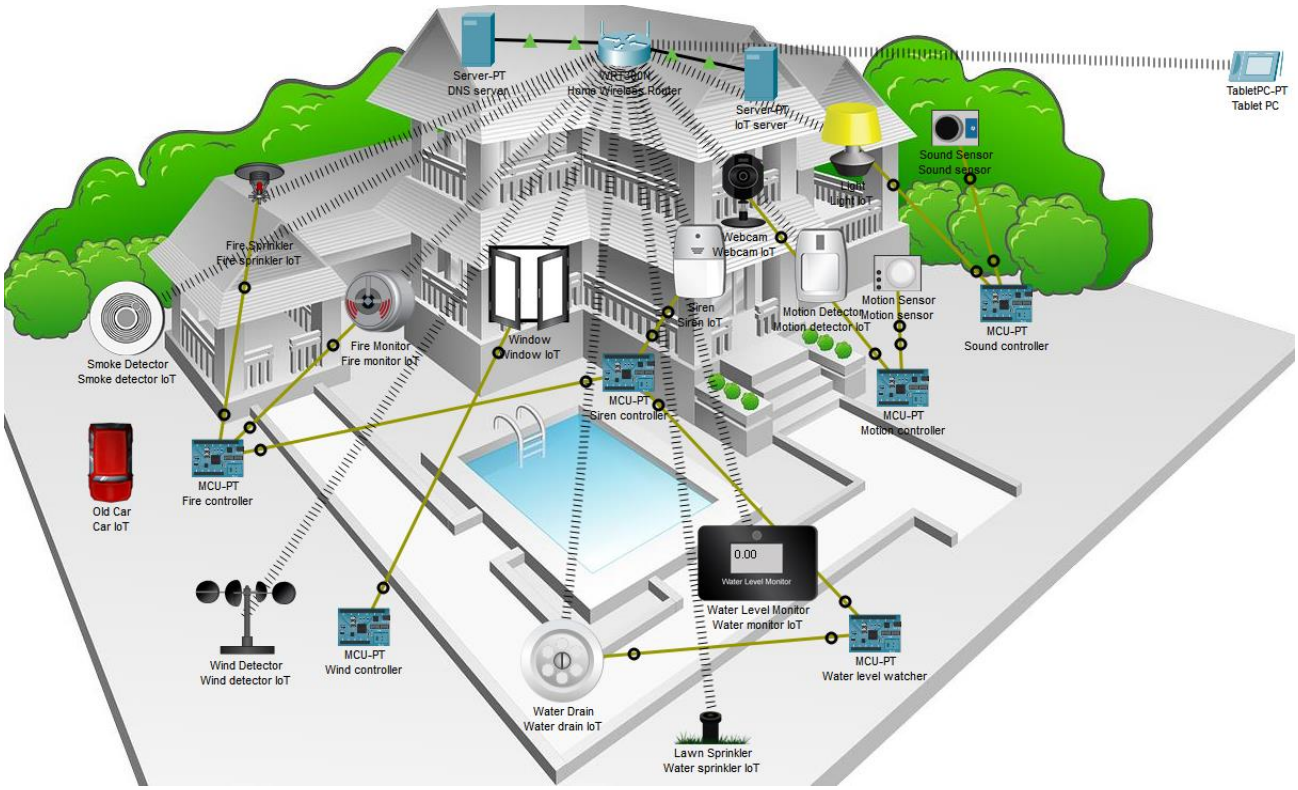


Рис. 5.37. З’єднання управляючих пристроїв IoT з платою MCU

Sound controller

Specifications | Physical | Config | Programming | Attributes

Blink (JavaScript) - main.js

Open | New | Delete | Rename | Import | Stop | Clear Outputs | Help

Reload | Copy | Paste | Undo | Redo | Find | Replace | Zoom: + -

main.js

```
1  var light = 0;
2  var sound_sensor = A0;
3
4  function setup() {
5      pinMode(light, OUTPUT);
6      pinMode(sound_sensor, INPUT);
7  }
8
9  function loop() {
10     var sound_level = analogRead(sound_sensor);
11
12     if (sound_level > 20) {
13         customWrite(light, 2);
14     } else if (sound_level > 5) {
15         customWrite(light, 1);
16     } else {
17         customWrite(light, 0);
18     }
19
20     Serial.println("Sound level: " + sound_level);
21 }
22
```

```
Sound level: 0
```

Рис. 5.38. Програмний код контролю шуму для плати MCU

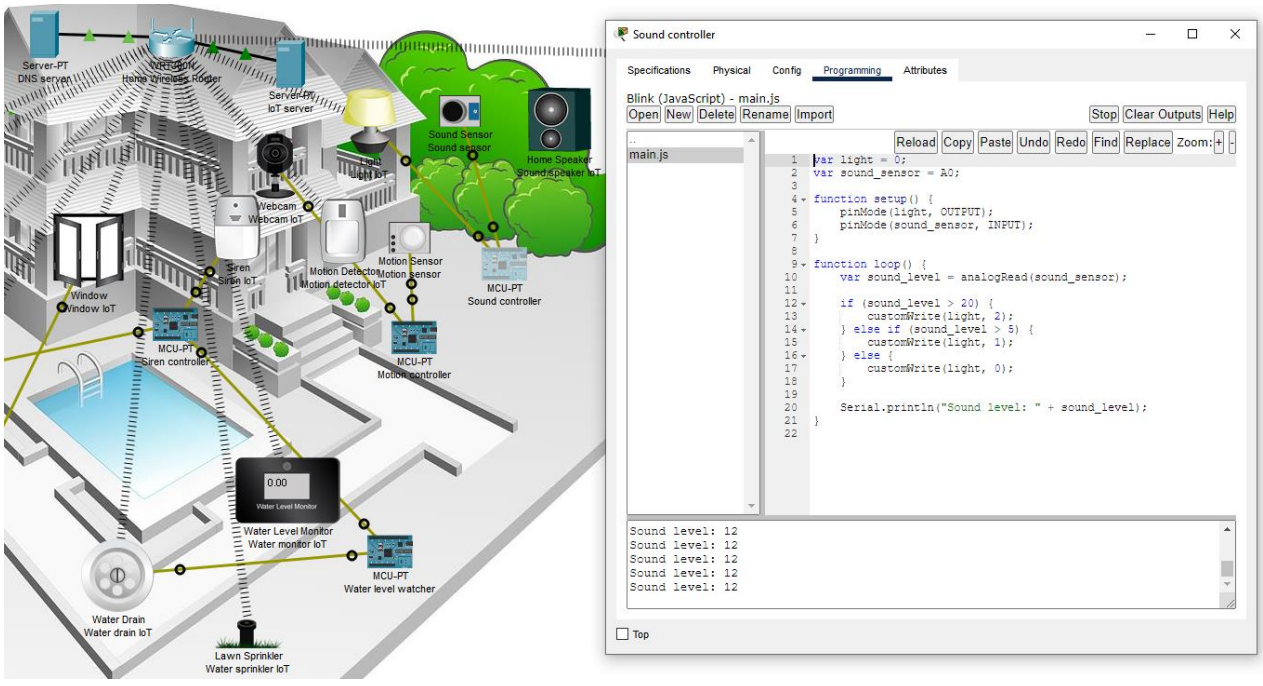


Рис. 5.39. Приклад роботи розумної системи контролю шуму

Завдання до лабораторної роботи 5

У середовищі моделювання Packet Tracer потрібно спроєктувати мережу для розумного будинку згідно сценарію за варіантом та запрограмувати пристрої для їх функціонування в заданій мережі. Усі пристрої потрібно підключити до бездротового маршрутизатора WLAN. Для усіх пристроїв налаштувати ідентифікатор SSID безпроводової мережі, налаштувати пароль, DHCP та DNS-сервер. Виконати тестування налаштованої мережі та пристроїв IoT. Власник розумного будинку після підключення через веббраузер повинен проходити аутентифікацію.

Варіанти сценаріїв для виконання лабораторної роботи 5

Варіант 1

Власник розумного будинку може управляти дверима гаражу, вентиляцією, перевіряти поточний стан системи сигналізації та рівень двоокису вуглецю у гаражі. Усі пристрої і сервер IoT підключені до однієї мережі WLAN. Сервер IoT налаштований для сервісів DNS.

Варіант 2

Змоделювати відкриття дверей гаража разом з включенням світла у розумному будинку. Датчик безпосередньо підключений до входу SBC. Якщо об'єкти виявлені, датчик подає значення на вхідний контакт SBC, тоді мікроконтролер порівнює його з попередньо встановленим пороговим значенням.

Варіант 3

Для розумного будинку використати наступні пристрої IoT: детектор руху, сирена, двері гаража. Використати датчик руху для тимчасової активації сирени сигналізації. При виявленні датчиком руху спрацьовує сирена. Якщо датчик не виявляє будь-якого руху після попередньо встановленого тайм-ауту, сирена налаштовується на вимкнення.

Варіант 4

Для розумного будинку налаштувати мережу пристроїв IoT, у якій датчик налаштований на включення вентилятора, відкриття дверей гаража і увімкнення сигналізаційної сирени у випадку досягнення рівня діоксиду вуглецю вище попередньо встановленої межі.

Варіант 5

Для розумного будинку встановити сонячні батареї та акумулятор. Вдень, коли світить сонце, сонячні панелі виробляють електроенергію, відбувається заряджання акумулятора. Якщо збільшується вологість в будинку, запускається кондиціонер. Акумулятор зберігає електропостачання, коли кондиціонер розряджається.

Варіант 6

Встановити детектор руху, який запускає трансляцію вебкамери у розумному будинку, якщо виявлено рух. Перевірити результат моделювання, натиснувши клавішу ALT на клавіатурі та виконати обробку на датчику детектора руху за допомогою миші. Вебкамера транслює зображення з домашньої сторінки IoT.

Варіант 7

Налаштувати інтелектуальне світло біля будинку. Використати датчик руху і датчик світла. Ліхтар вмикається автоматично, якщо поблизу рухається об'єкт та при цьому виявлено рівень видимого світла, нижчий заданого.

Варіант 8

Налаштувати детектори для безпеки розумного будинку: детектор для диму, детектор виявлення оксиду вуглецю та встановити пожежний сповіщувач. Налаштувати приведення в дію додаткової вентиляції та відкриття дверей і вікон у разі виявлення небезпек.

Варіант 9

Встановити датчик вологості IoT у розумному будинку. Якщо виявлена вологість перевищує або не досягає попередньо встановленого порогового значення, автоматично вмикається пристрій – зволожувач повітря, який регулює рівень вологості.

Варіант 10

Змоделювати систему зрошення саду, де датчик виявляє рівень вологості і, якщо показник вологості нижчий заданого діапазону, автоматично запускається система зрошування.

Варіант 11

Змоделювати датчик руху і вебкамеру, яка повинна включатися при спрацьовуванні датчика руху і вимикатися, коли рух припиняється. Створити правила включення і виключення для вебкамери, використовуючи головний шлюз через веббраузер і вкладку Conditions (правила If – Else).

Варіант 12

Змоделювати роботу газонного розприскувача. Використати тип з'єднання Copper Straight Through FastEthernet0 Sprinker з будь-яким вільним портом Ethernet головного шлюзу Home Gateway. Використати в Gateway / DNS Ip v4 – DHCP.

Варіант 13

Встановити в розумному будинку датчик рівня води. Якщо датчик фіксує перевищенння заданого порогу, вмикається сирена та підключається пристрій «розумний водостік».

Варіант 14

Забезпечити в розумному будинку систему реагування на вогонь з метою запобігання пожежі. При виявленні диму чи вогню спрацьовує система гасіння та сирена.

Варіант 15

Налаштувати детектор вимірювання швидкості вітру для бездротової мережі з домашнім шлюзом. Налаштувати закриття вікна в будинку при перевищенні значення рівня швидкості вітру вище значень заданого діапазону.

Варіант 16

Встановити датчик руху при відкритті дверей будинку і вебкамеру так, щоб вебкамера включалася при спрацьовуванні датчика руху і вимикалася, коли рух припинявся.

Варіант 17

Для розумного будинку встановити датчик руху в приміщенні. У разі виявлення руху детектором руху вмикати світло та сигналізацію.

Варіант 18

Для розумного будинку змоделювати систему обігріву приміщення, де датчик виявляє рівень тепла і, якщо показник рівня тепла нижче заданого, запускається система обігріву.

Варіант 19

Встановити в розумному будинку вебкамеру, яка починає трансляцію, коли вмикається кавоварка. Якщо показник рівня світла при цьому нижче заданого значення, вмикається лампа.

Варіант 20

Змоделювати систему зрошення саду, де датчик виявляє рівень вологості і, якщо показник вологості нижчий заданого діапазону, запускається система зрошування.

Варіант 21

Забезпечити в розумному будинку систему реагування на вогонь з метою запобігання пожежі. При виявленні диму чи вогню повинна спрацьовувати система гасіння та сирена.

Варіант 22

Для розумного будинку налаштувати мережу, у якій датчик налаштований на включення вентилятора, відкриття дверей гаража і увімкнення сигналізаційної сирени у випадку досягнення рівня діоксиду вуглецю вище попередньо встановленої межі.

Варіант 23

Для розумного будинку встановити датчик руху при відкритті дверей будинку і вебкамеру так, щоб вебкамера включалася при спрацьовуванні датчика руху і вимикалася, коли рух припинявся.

Варіант 24

Для розумного будинку змоделювати систему обігріву приміщення, де датчик виявляє рівень тепла і, якщо показник рівня тепла нижче заданого, запускається система обігріву.

Варіант 25

Використати смарт-пристрої IoT: детектор руху, сирена, двері гаража. Використати датчик руху для тимчасової активації сирени сигналізації. При виявленні датчиком руху спрацьовує сирена. Коли датчик не виявляє будь-якого руху після попередньо встановленого тайм-ауту, сирена налаштовується на вимкнення.

Питання для самоперевірки

1. Які можливості середовища моделювання Packet Tracer для створення та налаштування IoT рішень ви знаєте?
2. Яке призначення мікроконтролера?
3. Яким чином встановлюється зв'язок між датчиками та виконавчими елементами?
4. Як відбувається програмування MCU?
5. Як відбувається програмування SBC?

ЛАБОРАТОРНА РОБОТА 6.

РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОТРИМАННЯ ДАНИХ З MAPQUEST DIRECTIONS API

Мета роботи*:* розробити програмне забезпечення для отримання даних у форматі JSON з MapQuest Directions API, використовуючи мову програмування Python.

Теоретичні відомості

Інтерфейс прикладного програмування (application programming interface, API) – це програмний посередник, який дозволяє різним програмам спілкуватися між собою. Щоразу, коли ви користуєтеся, наприклад, програмою спільного використання поїздок, надсилаєте мобільний платіж або змінюєте температуру термостата зі свого телефону, ви використовуєте API. Термін «API» зазвичай використовується для опису інтерфейсів підключення до програмного забезпечення. Проте з роками сучасний API набув деяких унікальних характеристик, які змінили технологічний простір. По-перше, сучасні API відповідають певним стандартам (зазвичай HTTP і REST), крім того, сьогодні API розглядаються більше як продукти, ніж як код. Вони розроблені для використання певною аудиторією (наприклад, розробниками мобільних пристроїв), задокументовані та надають версії таким чином, щоб користувачі мали чіткі очікування щодо їх обслуговування та життєвого циклу.

REST API обмінюються даними через запити HTTP для виконання стандартних функцій бази даних, таких як створення, читання, оновлення та видалення записів у межах ресурсу. Наприклад, REST API використовуватиме запит GET для отримання запису, запит POST для його створення, запит PUT для оновлення запису та запит DELETE для його видалення. Усі методи HTTP можна використовувати у викликах API. Добре розроблений REST API схожий на вебсайт, який працює у веббраузері з вбудованою функцією HTTP.

Інформацію про стан ресурсу в будь-який конкретний момент або часову позначку (представлення ресурсу) можна надати клієнту практично в будь-якому форматі, включаючи JavaScript Object Notation (JSON), HTML, XLT, Python, PHP або звичайний текст. JSON популярний, тому що його читають як люди, так і машини, і він не залежить від мови програмування.

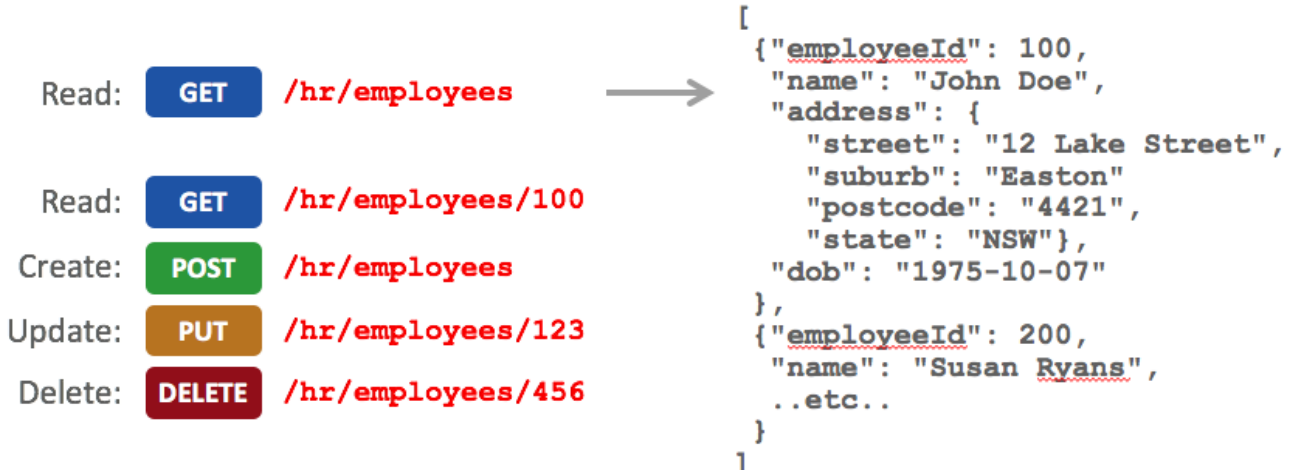


Рис. 6.1. Приклад ресурсу та набору URI для доступу до нього

Postman – це платформа API, на якій розробники програмного забезпечення можуть проектувати, створювати, тестувати та повторювати свої API. Postman є найбільшим у світі публічним центром API.

Репозиторій API дозволяє користувачам зберігати, каталогізувати та співпрацювати навколо артефактів API на центральній платформі в публічних, приватних або партнерських мережах.

Конструктор API допомагає реалізувати робочий процес розробки API за допомогою специфікацій, включаючи OpenAPI, GraphQL і RAML. Інструментами є клієнт API, API design, документація API, тестування API, макет серверів і виявлення API.

Безпека API є дуже важливою під час розробки загальнодоступного API. Поширені загрози включають SQL-ін’єкцію, атаку на відмову в обслуговуванні (DoS), порушення автентифікації та викрадення даних.

Завдання до лабораторної роботи 6

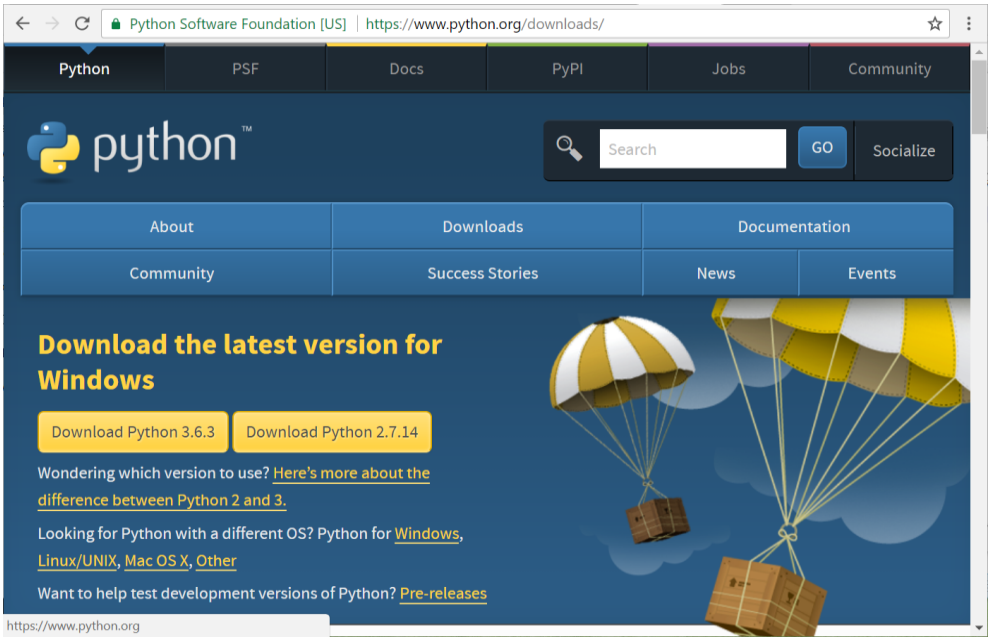
Необхідно виконати скріншоти виконання роботи на надати відповіді на поставлені питання. Для цієї лабораторної необхідно: встановити і перевірити Python; встановити Postman; встановити JSONView. Для розбору JSON даних необхідно буде виконати наступні задачі: отримати ключ API MapQuest; імпортувати необхідні модулі; створити змінні запиту API та побудувати URL-адресу; додати функціональність введення даних користувачем; додати функцію виходу, щоб користувач міг завершити роботу програми; відображати інформацію про поїздку, а саме: її тривалість, відстань та використання палива; перебирати дані JSON, щоб отримати та вивести напрямки; відображати повідомлення про помилки при невірному введенні даних.

У цій лабораторній роботі створюється програмне забезпеченн, яке отримує дані JSON з API MapQuest Directions, аналізує дані і форматує їх для виведення користувачеві з використанням запитів GET Route з API MapQuest Directions.

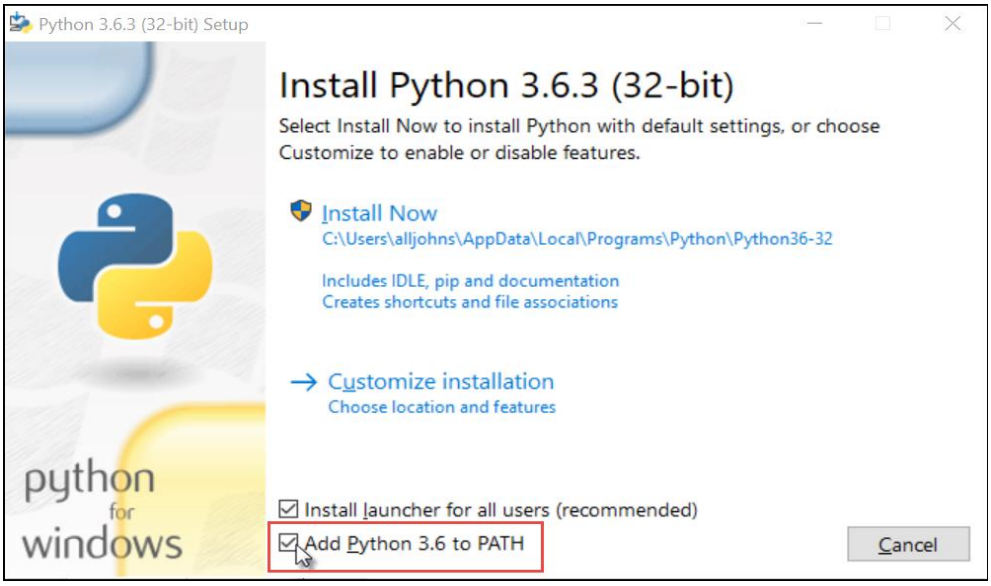
Виконайте наведені нижче дії, щоб установити та перевірити Python.

Завантажте та встановіть Python.

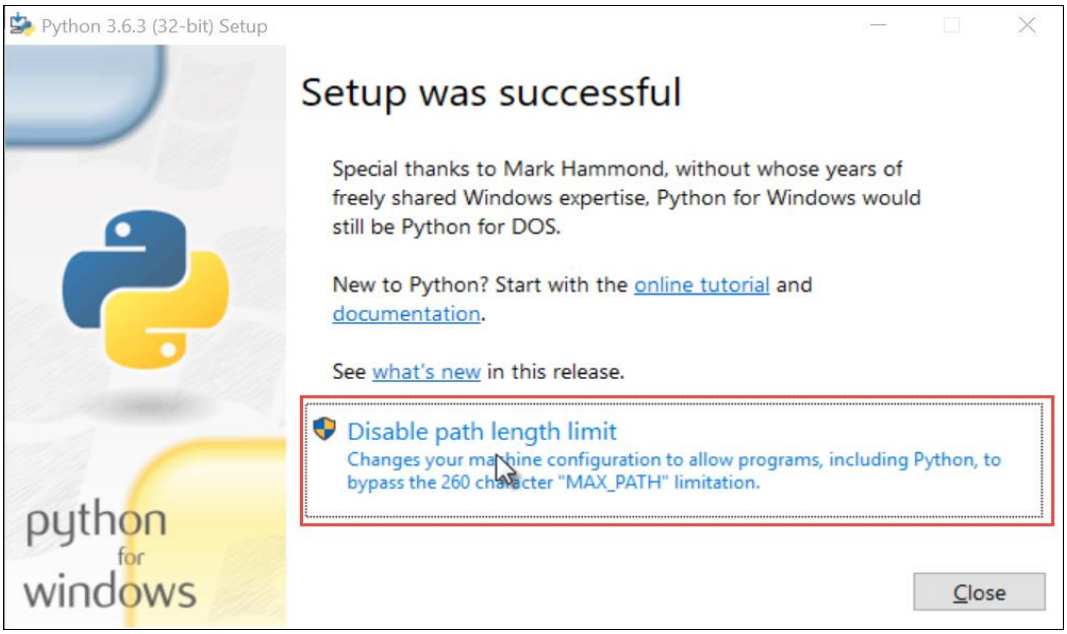
Перейдіть на сторінку https://www.python.org/downloads і завантажте найновішу версію Python.



Під час встановлення обов'язково встановіть прапорець «Додати Python 3.6 до шляху».



Після завершення налаштування натисніть «Вимкнути обмеження довжини шляху», якщо ця опція доступна для вашої інсталяції.



Перевірте інсталяцію Python.

Для ОС Windows відкрийте командний рядок і введіть команду **python**. Має відкритися інтерактивний інтерпретатор. Введіть **quit()**, щоб вийти з інтерактивного інтерпретатора Python.

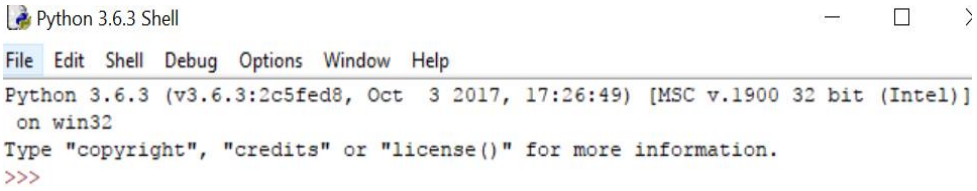
Для ОС Mac або Linux відкрийте командний рядок і введіть команду **python3**. Введіть **quit()** , щоб вийти з інтерактивного інтерпретатора Python.

Переконайтеся, що ви можете відкрити IDLE.

Для ОС Windows відкрийте IDLE з меню «Програми»: **Пуск > Python 3.6 > IDLE (Python 3.6 32-bit)**.

Для ОС Mac або Linux відкрийте командний рядок і введіть команду **idle3**.

Оболонка IDLE має відкритися.



Стандартна установка Python постачається з усіма стандартними модулями, які потрібні більшості програмістів для створення програм. Доступні також модулі, які розширюють функціональні можливості Python. Щоб початкова інсталяція Python була невеликою, ці модулі не включені в стандартний пакет.

Повернувшись до вікна командного рядка, скористайтеся індексом пакетів Python (до якого можна отримати доступ за допомогою команди **pip**), щоб завантажити запити модулів і таблицю. Для Linux і Mac використовуйте команду **pip3 install** *module_name*.

```
C:\> pip install requests
Collecting module
 Downloading module-0.2.1.tar.gz
Collecting requests
 Downloading requests-2.18.4-py2.py3-none-any.whl (88kB)
 100% |████████████████████████████████| 92kB 217kB/s
Collecting idna<2.7,>=2.5 (from requests)
 Downloading idna-2.6-py2.py3-none-any.whl (56kB)
 100% |████████████████████████████████| 61kB 1.2MB/s
Collecting certifi>=2017.4.17 (from requests)
 Downloading certifi-2017.11.5-py2.py3-none-any.whl (330kB)
 100% |████████████████████████████████| 337kB 139kB/s
Collecting chardet<3.1.0,>=3.0.2 (from requests)
 Downloading chardet-3.0.4-py2.py3-none-any.whl (133kB)
 100% |████████████████████████████████| 143kB 782kB/s
Collecting urllib3<1.23,>=1.21.1 (from requests)
 Downloading urllib3-1.22-py2.py3-none-any.whl (132kB)
 100% |████████████████████████████████| 133kB 1.2MB/s
Installing collected packages: module, idna, certifi, chardet, urllib3, requests
 Running setup.py install for module ... done
Successfully installed certifi-2017.11.5 chardet-3.0.4 idna-2.6 module-0.2.1
requests-2.18.4 urllib3-1.22
```

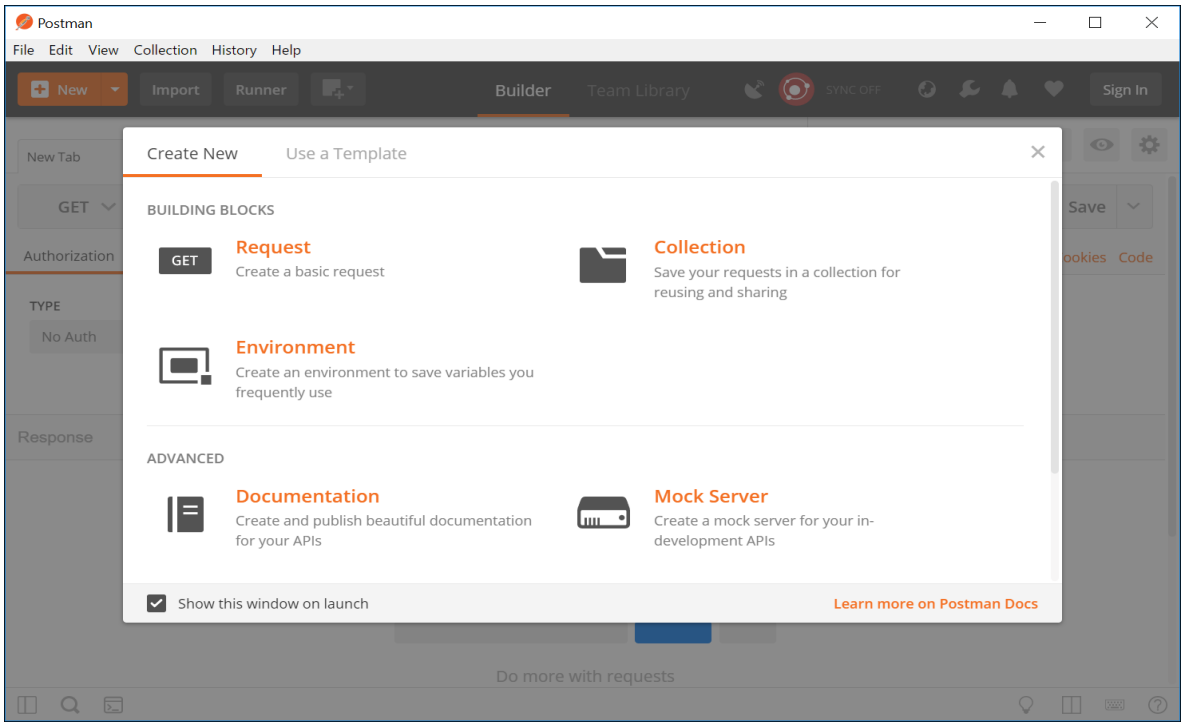
```
C:\> pip install tabulate
Collecting module
Collecting tabulate
 Downloading tabulate-0.8.1.tar.gz (45kB)
 100% |████████████████████████████████| 51kB 71kB/s
Installing collected packages: tabulate
 Running setup.py install for tabulate ... done
Successfully installed tabulate-0.8.1

C:\>
```

Встановлення програми Postman для надсилання запитів HTTP API

Завантажте додаток Postman.

Postman доступний як додаток для операційних систем Windows, Mac і Linux. Щоб установити Postman, перейдіть на сторінку програм за адресою https://www.getpostman.com/apps і натисніть «Завантажити» для вашої ОС.

Зареєструйтеся, щоб використовувати Postman, або виберіть створити обліковий запис в інший час. Переконайтеся, що програму Postman встановлено.

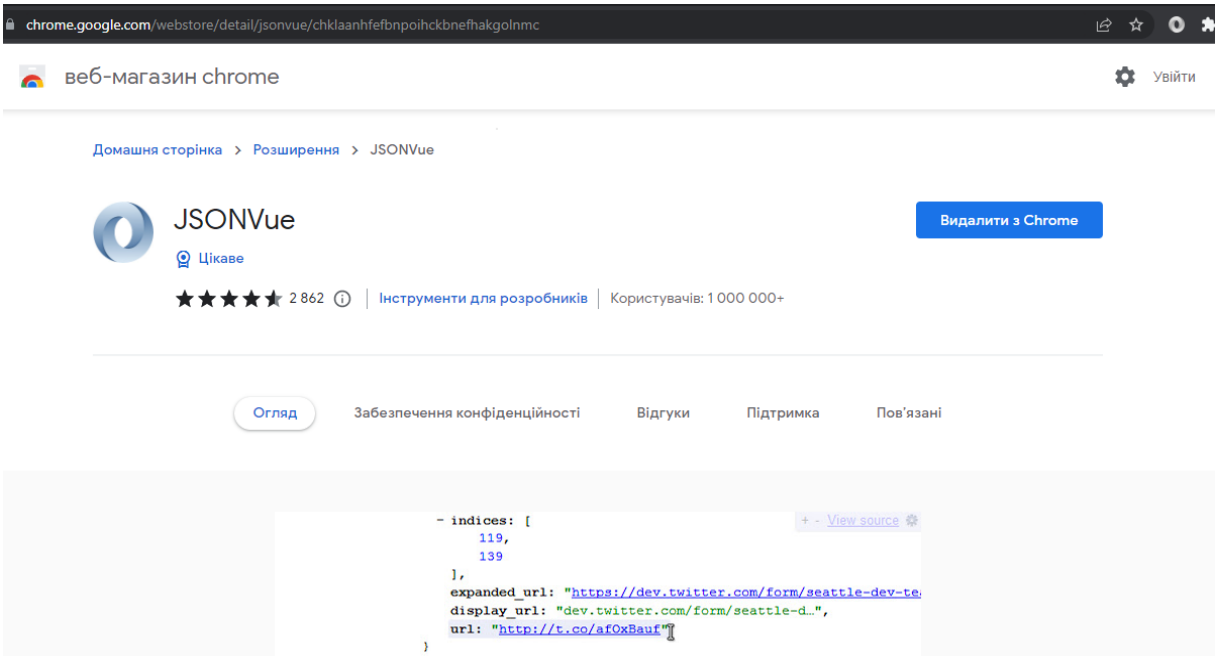


JSONView – це розширення Chrome, яке використовується для кращого перегляду даних JSON.

Встановіть розширення JSONView для Chrome.

За посиланням отримайте доступ до розширення JSONView:
https://chrome.google.com/webstore/detail/jsonview/chklaanhfefbnpoihckbnefhakgolnmc

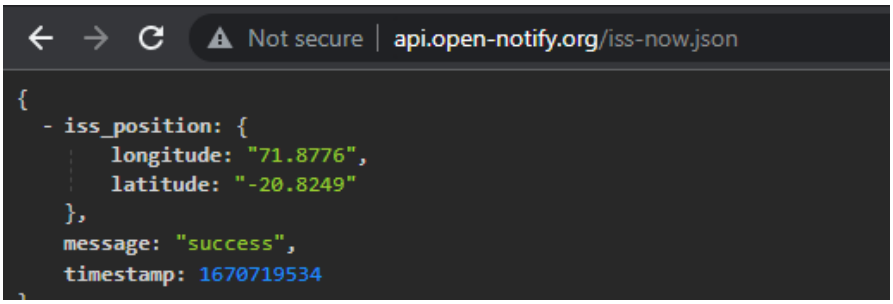
і додайте його до Chrome. Після встановлення ви повинні побачити піктограму JSONView у верхньому правому куті браузера Chrome. Якщо посилання не працює, знайдіть «розширення JSONView для Chrome», щоб знайти його.



Перевірте розширення JSONView.

Перевіримо, чи працює розширення JSONView належним чином, відкривши API, яке показує положення Міжнародної космічної станції. Ви повинні мати можливість згортати та розгортати розділи даних JSON.

http://api.open-notify.org/iss/v1/?lat=30.26715&lon=-97.74306



Розбір JSON за допомогою Python

Аутентифікація RESTful запиту виконується одним з чотирьох способів:

- None – аутентифікація для запитів відсутня взагалі. Кожен має право успішно виконати запит та отримати бажаний ресурс в даній ситуації.

• Basic HTTP – одна з найпростіших форм аутентифікації, в якій ім’я користувача та пароль передаються без додаткового захисту в самому запиті. Тобто запит містить поле заголовка у вигляді Authorization: Basic <credentials>, де credentials – це закодовані в Base64 ідентифікатор користувача та пароль, які з’єднані в один рядок двокрапкою.

• Token – це метод аутентифікації, який складається з токену. Токен – це зашифрований рядок, який зазвичай генерується сервером у відповідь на спеціальний запит.

• Open Authorization (OAuth) – це метод авторизації, який дозволяє користувачам сайтів оритмувати токени доступу до даних, що розміщуються на сайтах-сервісах.

Виконайте наступні дії.

1. Отримайте ключ MapQuest API.
2. Імпортуйте необхідні модулі.
3. Створіть змінні запиту API та побудуйте URL-адресу.
4. Додайте функцію введення користувача.
5. Додайте функцію виходу, щоб користувач міг завершити роботу програми.
6. Отримайте інформації про час, відстань і споживання палива для заданого маршруту.
7. Переглядайте дані JSON, щоб отримати та вивести маршрути.
8. Відобразьте повідомлення про помилки для неправильного введення користувача.

Вам потрібно створити програму, яка отримує дані JSON з MapQuest Directions API, аналізує дані та форматує їх для виведення користувачеві, використовуючи запит GET Route від MapQuest Directions API. Перегляньте документацію GET Route Directions API за посиланням:

https://developer.mapquest.com/documentation/directions-api/route/get/

Якщо посилання вище не працює, знайдіть «Документація API MapQuest».

Продемонструйте роботу додатку MapQuest Directions API.

Програма запитує початкове місце та пункт призначення, потім запитує дані JSON від MapQuest Directions API, аналізує їх і відображає корисну інформацію.

Для того, щоб побачити JSON для наведеного вище виведення, потрібно замінити **your_api_key** ключем MapQuest API, отриманим на попередному кроці.

Автентифікація запиту RESTful.

Перед створенням програми вам потрібно отримати ключ із сайту розробника MapQuest, що ви зробите на наступному кроці. Раніше ви не використовували автентифікацію для доступу до API ISS Pass Predictions. Однак багато API вимагають певної форми автентифікації.

Автентифікація запиту RESTful виконується одним із чотирьох способів. Опишіть кожен з методів:

1. None;
2. Basic HTTP;
3. Token;
4. Open Authorization (OAuth).

Для API MapQuest використайте автентифікацію за маркером.

Отримайте ключ MapQuest API.

Щоб отримати ключ MapQuest API, виконайте такі дії:

Перейдіть за адресою: https://developer.mapquest.com/ .

Натисніть Sign Up (**Зареєструватися)** у верхній частині сторінки.

Заповніть форму, щоб створити новий обліковий запис.

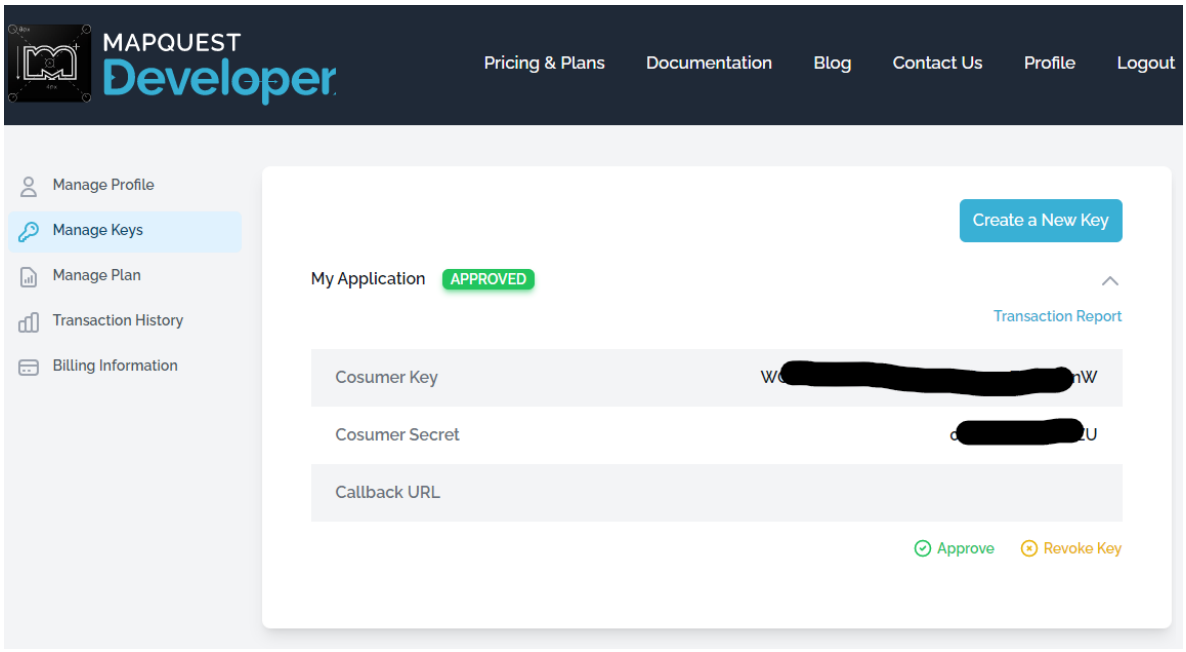
Після натискання Sign Me Up (**Зареєструватися)** вас буде перенаправлено на сторінку Manage Keys (**Керування ключами)**.

Натисніть Approve All Keys (**Схвалити всі ключі)** та розгорніть My Application (**Моя програма)**.

Скопіюйте свій **Consumer Key** в Блокнот для подальшого використання. Це буде ключ, який ви використовуватимете у подальшому.

MapQuest може змінити процес отримання ключа. Якщо наведені вище кроки більше не діють, знайдіть “steps to generate mapquest api key” (кроки для генерації ключа API mapquest).

Для того, щоб мати змогу отримувати дані з API MapQuest, необхідно зареєструватися та згенерувати відповідний токен. Надалі цей токен необхідно використати в усіх майбутніх запитах, щоби успішно пройти авторизацію.



Імпорт модулів для програми.

Щоб запустити сценарій аналізу даних JSON, вам потрібно імпортувати два модулі з бібліотеки Python: **requests** і **urllib.parse**. Модуль **запиту** надає функції для отримання даних JSON з URL-адреси.

Модуль **urllib.parse** надає різноманітні функції для аналізу та маніпулювання даними JSON, які ви отримуєте від запиту до URL-адреси.

Відкрийте порожній файл сценарію та збережіть його **08_parse-json1.py**.

Імпортуйте модулі **urllib.parse** та **requests**:

```
import urllib.parse
import requests
```

Створіть змінні для запиту API.

Першим кроком у створенні вашого запиту API є створення URL-адреси, яку ваша програма використовуватиме для здійснення виклику. Спочатку URL-адреса буде комбінацією таких змінних:

- **main_api** – основна URL-адреса, до якої ви маєте доступ;
- **orig** – параметр для визначення вашого місцеположення;
- **dest** – параметр для визначення місця призначення;
- **ключ** – ключ MapQuest API, який ви отримали з вебсайту розробника.

Створіть змінні для створення URL-адреси, яка надсилатиметься в запиті. Скопіюйте ваш ключ MapQuest у змінну **key**.

```
main_api = "https://www.mapquestapi.com/directions/v2/route?"
orig = "Washington"
dest = "Baltimaore"
key = "your_api_key"
```

Поєднайте чотири змінні **main_api**, **orig**, **dest** і **key**, щоб відформатувати запитану URL-адресу. Використовуйте метод **urlencode**, щоб правильно відформатувати значення адреси. Ця функція створює частину параметрів URL-адреси та перетворює можливі спеціальні символи у значення адреси (наприклад, пробіл у «+», а кому у «%2C»).

```
url = main_api + urllib.parse.urlencode({"key": key, "from":orig, "to":dest})
```

Створіть змінну для зберігання відповіді на запитану URL-адресу та друку повернених даних JSON. Змінна **json_data** містить представлення словника Python відповіді **json** методу **get** модуля **requests.** Оператор **print** використовується для перевірки повернутих даних.

```
json_data = requests.get(url).json()
print(json_data)
```

Перевірте URL-запит.

Запустіть свій сценарій **08_json-parse1.py** і переконайтеся, що він працює. За потреби усуньте проблеми з кодом. Хоча ваш результат може дещо відрізнятися, ви повинні отримати відповідь JSON, подібну до наведеної нижче.

```
{'route': {'distance': 38.089, 'hasHighway': True, 'hasUnpaved': False,
'hasAccessRestriction': False, 'options': {'mustAvoidLinkIds': [],
'maxWalkingDistance': -1, 'manmaps': 'true', 'urbanAvoidFactor': -1,
'stateBoundaryDisplay': True, 'cyclingRoadFactor': 1, 'routeType': 'FASTEST',
'countryBoundaryDisplay': True, 'drivingStyle': 2, 'highwayEfficiency': 22,
'narrativeType': 'text', 'routeNumber': 0, 'tryAvoidLinkIds': [], 'generalize': -
1, 'returnLinkDirections': False, 'doReverseGeocode': True, 'avoidTripIds': [],
'timeType': 0, 'sideOfStreetDisplay': True, 'filterZoneFactor': -1,
'walkingSpeed': -1, 'useTraffic': False, 'unit': 'M', 'tr
[output omitted]
>>>
```

Змініть змінні **orig** і **dest**. Перезапустіть сценарій, щоб отримати інші результати.

Роздрукуйте URL-адресу та перевірте статус запиту JSON.

Тепер, коли ви знаєте, що запит JSON працює, ви можете додати додаткові функції до програми.

Збережіть свій сценарій як **08_json-parse2.py**.

Видаліть оператор **print(json_data)**, оскільки вам більше не потрібно перевіряти, чи запит правильно відформатовано.

Роздрукуйте створену URL-адресу, щоб користувач міг побачити точний запит, зроблений програмою.

Проаналізуйте дані JSON, щоб отримати значення коду стану **statuscode**.

Запустіть цикл if, який перевіряє успішний виклик, значення якого дорівнює 0. Додайте оператор друку, щоб відобразити значення коду стану **statuscode** та його значення. **\n** додає порожній рядок під результатом.

```
print("URL: " + (url))
json_data = requests.get(url).json()
json_status = json_data["info"]["statuscode"]
if json_status == 0:
print("API Status: " + str(json_status) + " = A successful route call.\n")
```

У цій лабораторній роботі ви також додасте оператори **elif** і **else** для різних значень коду стану **statuscode**.

Перевірте статус і команди друку URL-адреси.

Запустіть свій сценарій **08_json-parse2.py** і переконайтеся, що він працює. За потреби усуньте проблеми з кодом. Ви повинні отримати результат виконання програми, подібний до наступного.

```
URL:
https://www.mapquestapi.com/directions/v2/route?key=your_api_key&to=Baltimore&from
=Washington
API Status: 0 = A successful route call.


>>>
```

Додайте дані користувача для початкового місця та пункту призначення.

Збережіть свій сценарій як **08_json-parse3.py**.

Видалити поточні змінні **orig** і **dest**.

Перепишіть вихідний **orig** і **dest**, щоб вони були в циклі while, у якому запитується введення користувача для початкового розташування та кінцевого призначення маршруту. Цикл while дозволяє користувачеві продовжувати робити запити для різних напрямків.

Переконайтеся, що весь код, що залишився, має відступ у циклі while.

```
while True:
    orig = input("Starting Location: ")
    dest = input("Destination: ")
    url = main_api + urllib.parse.urlencode({"key": key, "from":orig, "to":dest})
    print("URL: " + (url))

    json_data = requests.get(url).json()
    json_status = json_data["info"]["statuscode"]

    if json_status == 0:
        print("API Status: " + str(json_status) + " = A successful route call.\n")
```

Перевірте функцію введення користувача.

Запустіть сценарій **08_json-parse3.py** і переконайтеся, що він працює. За потреби усуньте проблеми з кодом. Додайте функцію виходу на наступному етапі. Наразі введіть **Ctrl+C**, щоб вийти з програми.

Додайте до програми функцію виходу.

Замість введення **Ctrl+C** для виходу з програми додайте можливість для користувача ввести **q** або **quit** як ключове слово для виходу з програми.

Щоб оновити програму, виконайте наступні дії. Збережіть свій сценарій як **08_json-parse4.py**. Додайте оператор **if** після кожної змінної розташування, щоб перевірити, чи вводить користувач **q** або **quit**.

```
while True:
    orig = input("Starting Location: ")
    if orig == "quit" or orig == "q":
        break
    dest = input("Destination: ")
    if dest == "quit" or dest == "q":
        break
```

Перевірте функцію виходу.

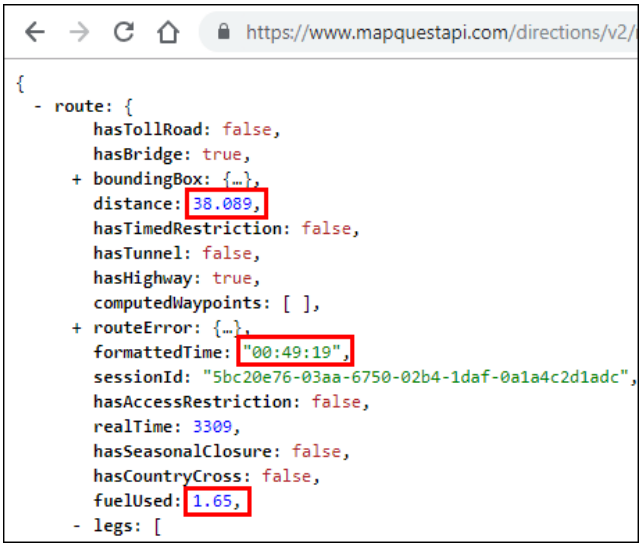
Запустіть свій сценарій **08_json-parse4.py** чотири рази, щоб перевірити кожну змінну розташування. Переконайтеся, що і **quit**, і **q** завершать роботу програми. За потреби усуньте проблеми з кодом.

Проаналізуйте та відобразіть деякі дані про подорож.

Скопіюйте свою URL-адресу у вебпереглядач. Якщо ви згорнете всі дані JSON, ви побачите, що є два кореневих словники: **route та info**.

Розширте словник **route** та вивчіть дані. Існують значення, які вказують, чи є на маршруті платні дороги, мости, тунелі, шосе, перекриття чи переїзди в інші країни. Ви також маєте побачити значення відстані, загального часу, який займе поїздка, споживання палива, як зазначено нижче.

Щоб проаналізувати та відобразити це, вкажіть словник **route** та виберіть пару ключ/значення, яку потрібно надрукувати.



Збережіть свій сценарій як **08_json-parse5.py**.

Під командою друку статусу API додайте оператори друку, які відображають місця відправлення та відправлення, а також ключі **formattedTime**, **distance** та **fuelUsed**.

Додайте оператор друку, який відображатиме подвійний рядок перед наступним запитом початкового розташування.

Запустіть **08_json-parse5.py**, щоб побачити такий результат:

```
if json_status == 0:
        print("API Status: " + str(json_status) + " = A successful route call.\n")
        print("=============================================")
        print("Directions from " + (orig) + " to " + (dest))
        print("Trip Duration:   " + (json_data["route"]["formattedTime"]))
        print("Miles:           " + str(json_data["route"]["distance"]))
        print("Fuel Used (Gal): " + str(json_data["route"]["fuelUsed"]))
        print("=============================================")
```

MapQuest використовує імперську систему, і немає параметра запиту для зміни даних на метричну систему. Тому вам, ймовірно, слід перетворити програму на відображення значень показників, як показано нижче.

```
print("Kilometers:      " + str((json_data["route"]["distance"])*1.61))
print("Fuel Used (Ltr): " + str((json_data["route"]["fuelUsed"])*3.78))
```

Запустіть змінений сценарій **08_json-parse5.py.**

Використовуйте аргумент **"{:.2f}".format,** щоб відформатувати значення з плаваючою крапкою до 2 знаків після коми перед перетворенням їх на рядкові значення. Кожне твердження має бути в одному рядку.

```
print("Kilometers:      " +
str("{:.2f}".format((json_data["route"]["distance"])*1.61)))
    print("Fuel Used (Ltr): " +
str("{:.2f}".format((json_data["route"]["fuelUsed"])*3.78)))
```

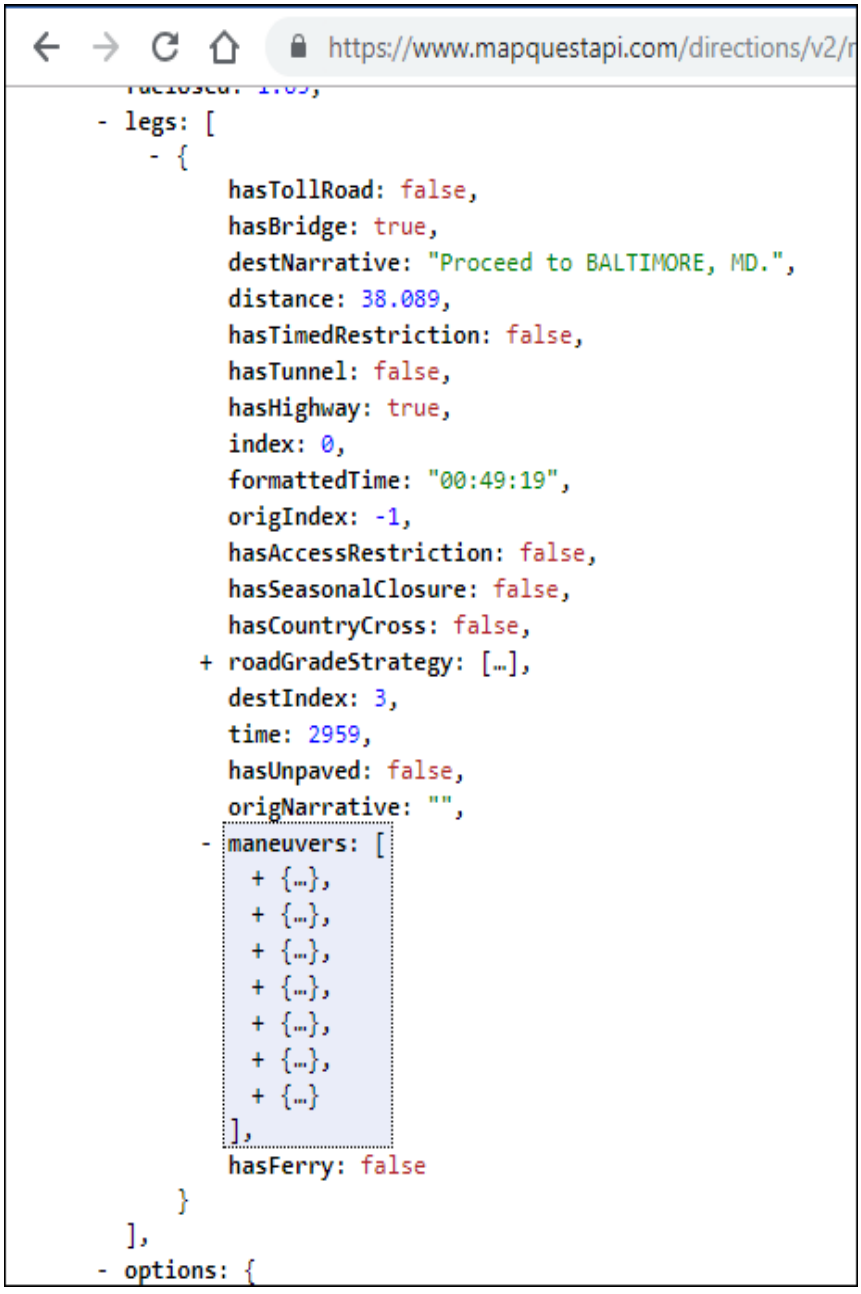
Перевірте функціональність аналізу та форматування.

Запустіть свій сценарій **08_json-parse5.py**, щоб переконатися, що він працює. За потреби усуньте проблеми з кодом. Переконайтеся, що у вас є всі правильні відкриваючі та закриваючі дужки.

Перевірте дані JSON про маневри.

Тепер ви готові відобразити покрокові вказівки від початкового місця до пункту призначення заданого маршруту. Знайдіть список етапів **legs** у словнику маршруту. Список **legs** включає один великий словник з більшістю даних JSON.

Знайдіть список маневрів **maneuvers** і згорніть кожен із семи словників усередині, як показано нижче.



Розгорніть перший словник у списку маневрів **maneuvers.** Кожен словник містить описовий ключ **narrative key** зі значенням опису руху, наприклад, «Почати на північ...», як показано нижче. Вам потрібно проаналізувати дані JSON, щоб отримати значення описового ключа **narrative key** для відображення у програмі.

Додайте цикл for для перегляду даних JSON маневрів.

Щоб оновити програму, виконайте такі дії:

Збережіть свій сценарій як **08_json-parse6.py**.

Додайте цикл **for** під другим оператором друку подвійного рядка. Цикл for повторює кожен список маневрів **maneuvers** і виконує наступне:

1) Друкує значення **narrative.**

2) Перетворює милі на кілометри за допомогою ***1,61**.

3) Форматує значення кілометрів для друку лише двох знаків після коми за допомогою функції **"{:.2f}".format**.

Додайте оператор друку, який відображатиме подвійний рядок перед наступним запитом початкового розташування.

Другий дворядковий оператор друку не має відступу в циклі **for**. Тому він є частиною попереднього оператора **if**, який перевіряє параметр **statuscode**.

```
print("Fuel Used (Ltr): " + str("{:.2f}".format((json_data["route"]["fuelUsed"])*3.78)))
print("=============================================")
for each in json_data["route"]["legs"][0]["maneuvers"]:
    print((each["narrative"]) + " (" + str("{:.2f}".format((each["distance"])*1.61) + "
km)"))
print("=============================================\n")
```

Перевірте ітерацію JSON.

Запустіть свій сценарій **08_json-parse6.py** і переконайтеся, що він працює. За потреби усуньте проблеми з кодом.

Перевірте, чи не введено користувачем неправильне значення.

Тепер ви готові додати останню функцію до своєї програми, щоб повідомляти про помилку, коли користувач вводить недійсні дані. Пам'ятайте, що ви запустили цикл if, щоб переконатися, що повернутий код стану **statuscode** дорівнює 0 перед аналізом даних JSON.

```
json_status = json_data["info"]["statuscode"]
    if json_status == 0:
        print("API Status: " + str(json_status) + " = A successful route call.\n")
```

Але що станеться, якщо код стану **statuscode** не дорівнює 0? Наприклад, користувач може ввести недійсне місце розташування або може не ввести одне або кілька місць. Якщо так, то програма відображає URL-адресу та запитує нове початкове місце. Збережіть свій сценарій як **08_jsont-parse7.py**.

```
for each in json_data["route"]["legs"][0]["maneuvers"]:
            print((each["narrative"]) + " (" +
str("{:.2f}".format((each["distance"])*1.61) + " km)"))
        print("=============================================\n")
    elif json_status == 402:
        print("**********************************************")
        print("Status Code: " + str(json_status) + "; Invalid user inputs for one or
both locations.")
        print("**********************************************\n")
    else:

print("************************************************************************")
        print("For Staus Code: " + str(json_status) + "; Refer to:")
        print("https://developer.mapquest.com/documentation/directions-api/status-
codes")

print("************************************************************************\n")
```

Щоб надати інформацію про помилку, коли це станеться, додайте оператори **elif** та **else** до свого циклу **if**. Після останнього оператора друку подвійного рядка під **if json_status == 0** додайте оператори **elif** та **else**.

Інструкція **elif** друкується, якщо значення **коду стану** становить 402 для недійсного розташування. Інструкція **else** друкується для всіх інших значень коду стану, таких як відсутність запису для одного чи кількох місць. Оператор else завершує цикл **if/else** і повертає програму до циклу **while**.

Перевірте повну функціональність програми.

Запустіть свій сценарій **08_json-parse7.py** і переконайтеся, що він працює. За потреби усуньте проблеми з кодом. Протестуйте всі можливості програми.

Питання для самоперевірки

1. Для чого використовується API?
1. Що є специфікацією API?
2. Для чого використовується вебінтерфейс API?
3. Поясніть призначення стандартів GraphQL, JSON RPC, SOAP і REST API.
4. Яке призначення платформи Postman?

ВИМОГИ ДО ОФОРМЛЕННЯ ЗВІТІВ ЛАБОРАТОРНИХ РОБІТ

Звіт до лабораторної роботи повинен включати:

1. Титульний аркуш з зазначенням ПІБ, № групи, кафедри, факультету та дати виконання лабораторної роботи.

2. Завдання для лабораторної роботи.

3. Хід лабораторної роботи. Цей розділ складається з послідовного опису виконуваних кроків згідно інструкцій до лабораторної роботи з наведенням скріншотів результатів виконання програми.

4. Відповіді на питання для самоперевірки.

5. Висновки до лабораторної роботи.

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Simulate IoT Projects. Wokwi. Електронний ресурс. Режим доступу: https://wokwi.com/projects/340367397829476948
2. Олещенко Л.М., Хіцко Я.В. Програмування пристроїв Інтернету речей: лабораторний практикум: навч. посіб. – Київ: КПІ ім. Ігоря Сікорського, 2019. – 47 с. Електронний ресурс. Режим доступу: https://dut.edu.ua/uploads/l_2146_81381740.pdf
3. Internet of Things System Design with Integrated Wireless MCUs. Електронний ресурс. Режим доступу: https://www.silabs.com/documents/public/white-papers/Internet-of-Things-System-Design-with-Integrated-Wireless-MCUs.pdf
4. ESP8266 NodeMCU with HTU21D Temperature and Humidity Sensor. Електронний ресурс. Режим доступу: https://microcontrollerslab.com/esp8266-nodemcu-htu21d-temperature-humidity-sensor/
5. What Is Fog Computing? Components, Examples, and Best Practices. Електронний ресурс. Режим доступу: https://www.spiceworks.com/tech/edge-computing/articles/what-is-fog-computing/
6. SSID. Електронний ресурс. Режим доступу: https://www.security.org/vpn/ssid/
7. Building trust in IoT devices with powerful IoT security solutions. Електронний ресурс. Режим доступу: https://www.thalesgroup.com/en/markets/digital-identity-and-security/iot/iot-security
8. Parsing JSON with a Python Application. Електронний ресурс. Режим доступу: https://vsip.info/1331-lab-parsing-json-with-a-python-application-pdf-free.html
9. What is an API. Електронний ресурс. Режим доступу: https://www.ibm.com/topics/api
10. What is REST. Електронний ресурс. Режим доступу: https://restfulapi.net/