

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторних робіт з дисципліни
«Технології Інтернету речей»
для здобувачів другого (магістерського) рівня вищої освіти
спеціальності F7 Комп'ютерна інженерія
усіх форм навчання.
Частина 1

Методичні вказівки до лабораторних робіт з дисципліни «Технології Інтернету речей» для здобувачів другого (магістерського) рівня вищої освіти спеціальності F7 Комп'ютерна інженерія усіх форм навчання. Частина 1 / Укл. Р. К. Кудерметов, М. Ю. Тягунова, А. В. Тіменко, О. В. Польська – Запоріжжя: НУ «Запорізька політехніка», 2025. – 42 с.

Укладачі: Р. К. Кудерметов, к.т.н., доцент
М. Ю. Тягунова, к.т.н., доцент
А. В. Тіменко, ст. викладач
О. В. Польська, ст. викладач

Рецензент: С. С. Грушко, к.т.н., доцент

Відповідальний
за випуск: О. В. Польська, ст. викладач

Затверджено:
на засіданні кафедри
«Комп'ютерні системи та мережі»
Протокол № 1 від 05.08.2025

Рекомендовано до видання
НМК факультету КНТ
Протокол № 1 від 11.08.2025

ЗМІСТ

Частина 1	
Вступ.....	4
1 Лабораторна робота № 1 Створення контурів керування IoT за допомогою мікрокомп'ютера Raspberry Pi	5
1.1 Загальні відомості	5
1.2 Приклади використання RPi	7
1.3 Завдання до лабораторної роботи.....	10
1.4 Контрольні питання	12
2 Лабораторна робота № 2 Створення контурів керування iot на базі плати Raspberry Pi Pico у середовищі Wokwi	13
2.1 Загальні відомості	13
2.2 Завдання до лабораторної роботи.....	21
2.3 Контрольні питання	22
3 Лабораторна робота № 3 Розробка прототипів IoT застосувань у середовищі Node-RED.....	23
3.1 Загальні відомості	23
3.2 Завдання до лабораторної роботи.....	24
3.3 Контрольні питання	30
4 Лабораторна робота №4 Використання операційної системи Home Assistant для створення розумного будинку	31
4.1 Загальні відомості	31
4.2 Завдання до лабораторної роботи.....	31
4.3 Контрольні питання	40
Перелік джерел посилань.....	41
Частина 2	

ВСТУП

Реалії сучасного світу інформаційних технологій тісно пов'язані із концепцією Інтернету речей (IoT, Internet of Things). Ця концепція полягає в наступному: тисячі (мільйони) «розумних» пристроїв взаємодіють один із одним без участі людини з метою автоматизації (поліпшення) бізнес-процесів різноманітних предметних областей: «розумне місто», «розумний будинок» тощо. Створення систем згідно названої концепції можливе на основі технології програмно-конфігурованих мереж (SDN, Software Defined Networking) [1, 2]. При цьому вводиться поняття «контролера» – засобу централізованого координуванні взаємодії компонентів системи. Для забезпечення узгодженої взаємодії останніх використовуються різноманітні протоколи передачі даних. Ключову роль при цьому відіграє протокол OpenFlow – протокол взаємодії контролера і комутаторів [3]. Типовим шляхом організації процесу проектування систем на основі програмно-конфігурованих мереж є використання спеціалізованих емуляторів [4]. Найпоширенішим таким засобом є емулятор Mininet [5, 6]. Для вивчення можливостей Інтернету речей використовують сімейство мікрокомп'ютерів Raspberry Pi [7, 8].

Комплекс лабораторних робіт з дисципліни «Технології Інтернету речей» складається з двох частин і охоплює наступні питання:

- налаштування середовища Mininet та його використання з протоколом OpenFlow для створення та тестування базових топологій програмно-конфігурованих мереж [9]–[11];
- створення прототипів контурів моніторингу та управління IoT-систем у середовищі Wokwi [12];
- розробка прототипів IoT застосувань у середовищі Node-RED [13];
- використання операційної системи Home Assistant для створення розумного будинку [14];
- програмування мереж з використанням Python API та питання їх конфігурування і тестування за допомогою графічного інтерфейсу.

1 ЛАБОРАТОРНА РОБОТА № 1

СТВОРЕННЯ КОНТУРІВ КЕРУВАННЯ ІОТ ЗА ДОПОМОГОЮ МІКРОКОМП'ЮТЕРА RASPBERRY PI

Мета роботи: навчитися використовувати одноплатний комп'ютер Raspberry Pi в проектах Інтернету речей.

1.1 Загальні відомості

Raspberry Pi (RPi) – одноплатний комп'ютер спочатку розроблений як бюджетна система для навчання інформатиці, що згодом одержав набагато більш широке використання в різноманітних системах у тому числі системах IoT.

RPi випускається в декількох комплектаціях: моделі A, B, B+, 2B, 3B, 4B. Версії A, B, B+ оснащені ARM11 процесором Broadcom BCM2835 700 МГц і модулем оперативної пам'яті 256МБ/512МБ, що розміщені за технологією «package-on-package» безпосередньо на процесорі. Модель 2B оснащується процесором з чотирма ядрами Cortex-A7 1 ГГц і оперативною пам'яттю 1 ГБ. Модель A має один порт USB 2.0, модель B – два, а моделі B+, 2B – чотири. Також в моделях B, B+, 2B присутній порт Ethernet. Крім основного ядра, BCM2835 включає в себе графічне ядро з підтримкою OpenGL ES 2.0, апаратного прискорення, FullHD-відео і DSP-ядро.

Вивід відеосигналу можливий через композитний роз'єм RCA або через цифровий HDMI-інтерфейс. У версії B+ і 2B вивід можливий через аудіороз'єм 3.5. Коренева файлова система, образ ядра і призначені для користувача файли розміщуються на карті пам'яті SD, MMC, MicroSD (тільки в моделі B+) або SDIOрен.

Підключення периферії можна проводити як через наявні інтерфейси, так і через контакти роз'єму GPIO (General Purpose Input/Output) – порти вводу/виводу загального призначення. Завдяки цьому цей RPi можна використовувати для управління різними пристроями. У моделі B встановлено 26-контактний GPIO, а в моделях B+, 2B, 3B, 4B – розширений 40-контактний GPIO. Графічне відображення плати представлено на рис. 1.1. Призначення контактів GPIO показано на рис. 1.2.

Raspberry Pi працює в основному на операційних системах, заснованих на Linux ядрі. ARM11 засновано на 6-ій версії ARM, який

підтримує не всі версії Linux. Для встановлення операційних систем існує інструмент NOOBS. На моделі Raspberry Pi 3B, яка надається для виконання лабораторних робіт, встановлено ОС Raspbian.

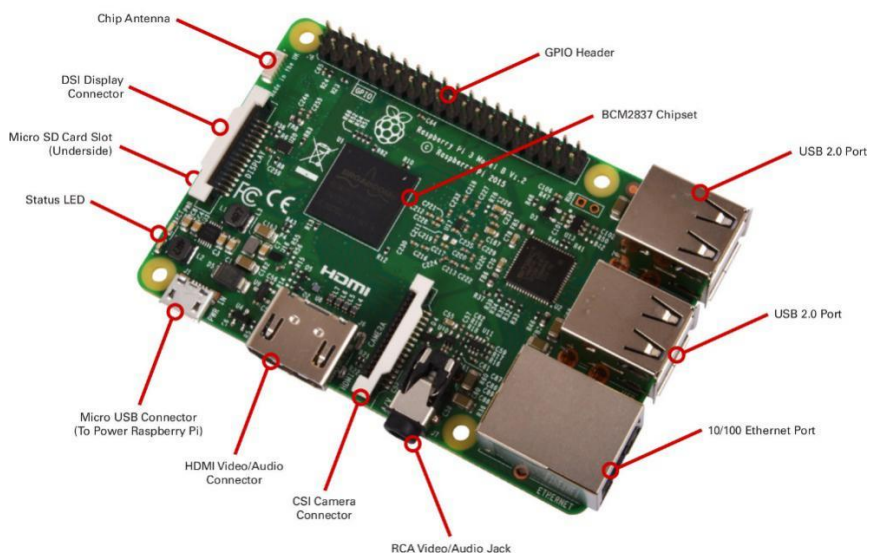


Рисунок 1.1 – Одноплатний комп’ютер Raspberry Pi 3B

5V power	5V power	GROUND	GPIO14 UART0_TXD	GPIO15 UART0_RXD	GPIO18 PCM_CLK	GROUND	GPIO23	GPIO24	GROUND	GPIO25	GPIO8 SPI0_CE0_N	GPIO7 SPI0_CE1_N	ID_SC I2c ID eeprom	GROUND	GPIO12	GROUND	GPIO16	GPIO20	GPIO21
2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40
1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33	35	37	39
3V3 power	GPIO2 SDA112C	GPIO3 SCL112C	GPIO4	GROUND	GPIO17	GPIO27	GPIO22	3V3 power	GPIO10 SPI0_MOSI	GPIO9 SPI0_MISO	GPIO11 SPI0_SCLK	GROUND	ID_SD I2c ID eeprom	GPIO5	GPIO6	GPIO13	GPIO19	GPIO26	GROUND

Рисунок 1.2 – Призначення контактів GPIO

1.2 Приклади використання RPi

Використання RPi продемонструємо на найпростіших прикладах, в яких периферійними пристроями є світлодіод і кнопка. Контакти (порти) GPIO можуть бути налаштовані як «входи» та як «виходи». При цьому в режимі «вхід» значення максимально допустимої напруги є 3.3 В, а максимальне значення електричного струму – 0.5 мА. У режимі «вихід» максимальне значення електричного струму – 16 мА. Після включення RPi та завантаження операційної системи всі порти GPIO встановлені в стан логічного нуля, якому відповідає низький рівень напруги 0. Підключення до портів елементів електронних схем рекомендується виконувати через обмежувальні резистори з опором у діапазоні 680 Ом – 2 кОм.

У наших прикладах ми будемо підключати до портів GPIO світлодіоди і кнопку. На рис. 1.3 показані рекомендовані електричні схеми підключення світлодіоду та кнопки до портів GPIO (номери портів можуть бути будь-якими з допустимих).

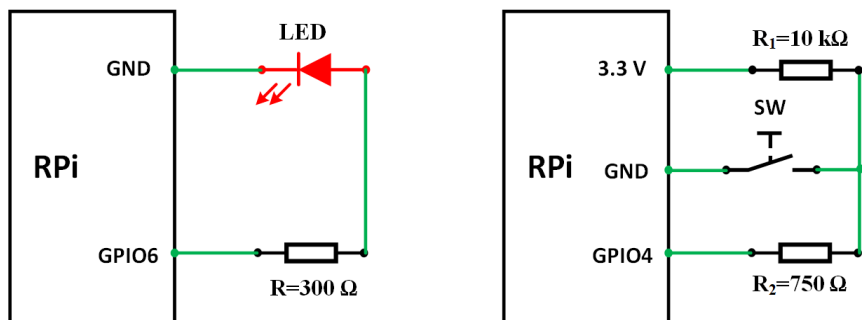


Рисунок 1.3 – Підключення світлодіоду та кнопки до портів GPIO

При розробці різноманітних схем із використанням RPi зручно скористатися макетною платою. Приклад такої плати показано на рис. 1.4.

Програмування RPi GPIO будемо здійснювати на мові Python. Мова Python найчастіше використовується для програмування GPIO. В операційній системі Raspbian мова Python і модуль RPi.GPIO встановлені за замовчуванням. Для розробки, налагодження та виконання програм на лабораторних зразках RPi встановлено

середовище розробки Integrated Development and Learning Environment (IDLE).

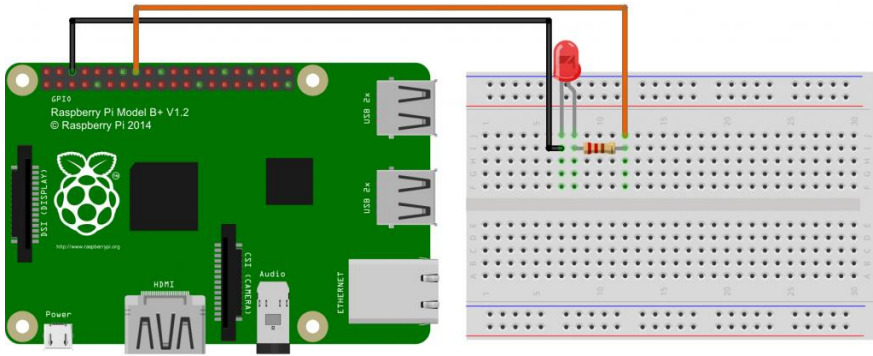


Рисунок 1.4 – Використання макетної плати

Приклад 1. Управління світлодіодом.

Розробимо схему з підключенням світлодіоду та програму, яка керує миготінням світлодіоду. Виберемо порт GPIO 6 до якого через резистор підключимо анод світлодіоду. Катод світлодіоду підключимо до контакту GND (земля). Ця схема буде подібною до схеми на рис. 1.3. та рис. 1.4.

Код програми для керування світлодіодом представлений в лістингу 1.1. У перших двох рядках програми підключаються бібліотека (модуль) функції керування GPIO та модуль для керування часовими затримками. Функція `GPIO.setmode(GPIO.BCM)` встановлює режим нумерації портів за назвою. Замість цього режиму можна використовувати режим нумерації за номерами контактів `GPIO.setmode(GPIO.BOARD)`. Далі ми встановлюємо порт GPIO 6 у режим «вихід» (`GPIO.OUT`) і задаємо його початковий рівень низьким (`GPIO.LOW`). Логіка нескінченного миготіння світлодіоду з частотою (1 Гц) міститься в блоці `try`, який викликає виняток при натисканні на клавіатурі комбінації `Ctrl+C`, що завершує роботу програми. Останній рядок програми призначений для скидання всіх портів GPIO у вихідний (початковий) стан.

Лістинг 1.1 – Управління миготінням світлодіоду у циклі

```

from RPi import GPIO
from time import sleep

GPIO.setmode(GPIO.BCM)
GPIO.setup(6, GPIO.OUT)
GPIO.output(6, GPIO.LOW)

try:
    while True:
        GPIO.output(6, True)
        sleep(0.5)
        GPIO.output(6, False)
        sleep(0.5)
except KeyboardInterrupt:
    print("program stop")
GPIO.cleanup()

```

Приклад 2. Схема зі світлодіодом та кнопкою.

У цьому прикладі світлодіод буде миготіти лише тоді, коли кнопка натиснута. Схема використовує кнопку, яка підключена до порту GPIO 4 так, як показано на рис. 1.3. Електричну схему для даного прикладу пропонується розробити самостійно. Реалізація програми представлена у лістингу 1.2. У цій програмі для наочності портам GPIO 4 та GPIO 6 призначені ідентифікатори BUTTON та LED відповідно. Порт BUTTON встановлений у режимі «вхід». Коли кнопку натиснуто, на порті BUTTON (GPIO 4) буде низький рівень напруги, що відповідає значенню логічній змінній False.

Лістинг 1.2 – Управління світлодіодом за допомогою кнопки

```

from RPi import GPIO
from time import sleep

LED = 6
BUTTON = 4
GPIO.setmode(GPIO.BCM)
GPIO.setup(LED, GPIO.OUT)
GPIO.output(LED, GPIO.LOW)
GPIO.setup(BUTTON, GPIO.IN)
print('program: button+led')

```

```

try:
    while True:
        if GPIO.input(BUTTON) == False:
            print('button pressed')
            GPIO.output(LED, True)
            sleep(0.5)
            GPIO.output(LED, False)
            sleep(0.5)
        else:
            print('button is not pressed')
            sleep(1.0)

except KeyboardInterrupt:
    print('the program has stopped')
GPIO.cleanup()

```

Можливий результат роботи програми, який виводиться на панелі командного рядка IDLE:

```

program: button+led
button is not pressed
button is not pressed
button is not pressed
button is not pressed
button pressed
button pressed
button pressed
button pressed
button is not pressed
button is not pressed
button is not pressed
the program has stopped

```

1.3 Завдання до лабораторної роботи

Результат виконання завдань повинен містити схему електричну принципову (аналогічну з рис. 1.3) або фотографію макета схеми (аналогічно з рис. 1.4), код програми та за наявності результат роботи програми, що виведений на консоль.

Варіант 1. Розробити схему, в якій до портів RPi GPIO підключено три світлодіоди (червоний, жовтий та зелений). Програма

управління схемою повинна в нескінченному циклі керувати увімкненням та вимкненням світлодіодів у наступній послідовності:

- одну секунду увімкнено лише червоний світлодіод;
- дві секунди увімкнено лише жовтий світлодіод;
- три секунди увімкнено лише зелений світлодіод;
- далі цикл повторюється.

Варіант 2. Розробити схему, в якій до портів RPi GPIO підключено три світлодіоди (червоний, жовтий та зелений). Програма управління схемою повинна у нескінченному циклі керувати увімкненням та вимкненням світлодіодів так, щоб червоний світлодіод миготів із частотою 1 Гц, жовтий – із частотою 2 Гц, зелений – із частотою 0.5 Гц.

Варіант 3. Розробити схему, в якій до портів RPi GPIO підключено два світлодіоди (червоний та зелений) та кнопка. Програма керування схемою повинна в нескінченному циклі керувати увімкненням та вимкненням світлодіодів так, щоб при ненависнутій кнопці світлодіоди миготіли з частотою 1 Гц, а при натиснутій кнопці червоний світлодіод миготів із частотою 0.5 Гц, зелений – з частотою 2 Гц.

Варіант 4. Розробити схему, в якій до портів RPi GPIO підключено один світлодіод та дві кнопки. Програма керування схемою повинна в нескінченному циклі керувати увімкненням та вимкненням світлодіодів за таким правилом: якщо кнопки не натиснуті, світлодіод миготить з частотою 1 Гц; якщо натиснута тільки перша кнопка, світлодіод миготить з частотою 2 Гц; якщо натиснута тільки друга кнопка, світлодіод миготить з частотою 3 Гц; Якщо обидві кнопки натиснуті, світлодіод світить, але не миготить.

Варіант 5. Розробити схему, в якій до портів RPi GPIO підключено два світлодіоди (червоний та зелений) та дві кнопки. Програма керування схемою повинна в нескінченному циклі керувати увімкненням та вимкненням світлодіодів за таким правилом: якщо кнопки не натиснуті, обидва світлодіоди миготять із частотою 0.5 Гц; якщо натиснута лише перша кнопка, червоний світлодіод повинен миготіти із частотою 1 Гц, а зелений згаснути; якщо натиснута лише друга кнопка. Зелений світлодіод має миготіти з частотою 2 Гц, а червоний – з частотою 0.5 Гц.

1.4 Контрольні питання

- 1 Чому Raspberry Pi називають мікрокомп'ютером, а не мікроконтролером?
- 2 Поясніть схему підключення світлодіоду до GPIO порту.
- 3 Поясніть схему підключення кнопки до GPIO порту.
- 4 Перелічіть інтерфейси, що забезпечують функціонування Raspberry Pi.
- 5 Перелічіть інтерфейси GPIO з описом їх призначення.
- 6 Чи може той самий контакт GPIO працювати як вхідний так і вихідний порт?
- 7 Якою буде напруга на портах GPIO після завантаження операційної системи?
- 8 Яка максимальна напруга може бути на виході GPIO порту?
- 9 Яку максимально допустиму напругу можна подати на вхід порту?
- 10 Для чого призначені контакти GPIO з напругою 5 В і 3.3 В?
- 11 Чи є у GPIO Raspberry Pi порт аналого-цифрового перетворювача?
- 12 Чи можна керувати портами GPIO з інтерфейсу командного рядка?
- 13 У чому різниця режимів нумерації GPIO портів GPIO.BCM та GPIO.BOARD?
- 14 Чи можна працювати з Raspberry Pi віддалено (за допомогою комп'ютерної мережі)?
- 15 Як відбувається завантаження операційної системи у Raspberry Pi ?

2 ЛАБОРАТОРНА РОБОТА № 2

СТВОРЕННЯ КОНТУРІВ КЕРУВАННЯ ІОТ НА БАЗІ ПЛАТИ RASPBERRY PI PICO У СЕРЕДОВИЩІ WOKWI

Мета роботи: навчитися створювати прототипи контурів моніторингу та управління IoT-систем у середовищі Wokwi.

2.1 Загальні відомості

Wokwi – це онлайн-симулятор призначений для моделювання електронних схем на базі мікроконтролерів Arduino, ESP32, STM32, Pi Pico, в яких використовуються моделі сенсорів (датчиків), актуаторів (виконавчих приладів) та інших електронних компонентів. Wokwi – це вебзастосунок і середовище проектування, що працює у веббраузері. Wokwi безкоштовний для особистого користування.

Вхід у Wokwi здійснюється за посиланням wokwi.com. Wokwi є зручним інструментом для проектування IoT-підсистем, моделювання їх функціонування та сценаріїв використання.

Інструкція користувача, приклади використання доступні за посиланням docs.wokwi.com. Для роботи в wokwi слід зареєструватися, у цьому випадку ви будете входити в wokwi під своїм профілем і вам будуть доступні ваші розроблені проекти та деякі інші функції.

Розглянемо створення схеми IoT-контурів на прикладі проектів з використанням мікроконтролера Raspberry Pi Pico та мови MicroPython, який є розширенням мови Python 3 для програмування мікроконтролерів.

На головній сторінці Wokwi (рис. 2.1) ви можете вибрати тип мікроконтролера, на якому ви плануєте проектувати схему контуру IoT. Але якщо опуститись нижче на цій сторінці, ви побачите опцію вибору мови програмування. Виберіть MicroPython, в результаті відкриються деякі базові приклади проектів, стартові шаблони, нещодавно створені проекти, а також створені вами проекти, якщо ви зайшли до Wokwi під своїм профілем. Виберіть такий приклад, який найбільше відповідає ідеї вашого проекту. Після відкриття цього прикладу ви можете його перейменувати та редагувати.

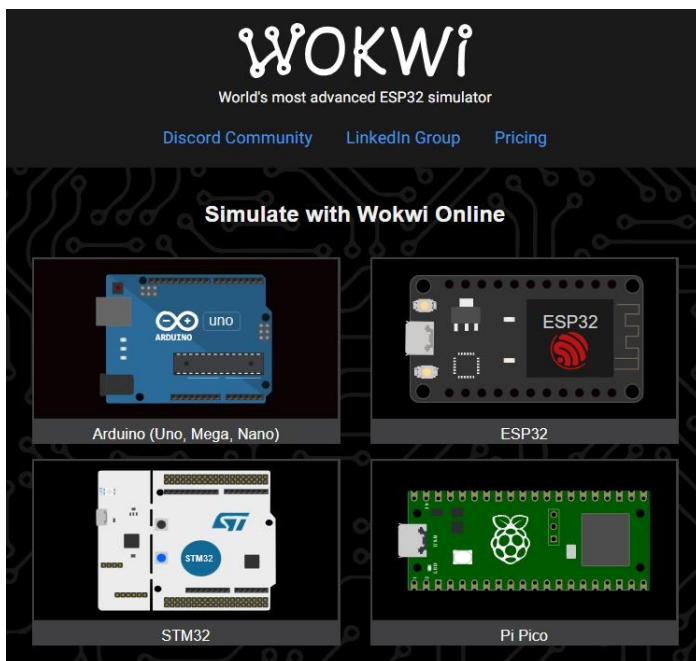


Рисунок 2.1 – Початок проектування у Wokwi

Raspberry Pi Pico (RPi Pico) – це плата з мікроконтролером RP2040. RP2040 – це мікроконтролер, розроблений Raspberry Pi. Він оснащений двоядерним процесором ARM Cortex-M0+ 133 МГц, 264 КБ вбудованої SRAM та підтримкою до 16 МБ зовнішньої флеш-пам'яті, а також включає програмовану підсистему вводу/виводу (PIO). Зовнішній вигляд плати RPi Pico та розміщення контактів (pins) вводу/виводу показано на рис. 2.2. Вхідне живлення плати може бути до 5,5 В.

Як видно з рисунку RPi Pico має багато інтерфейсів для підключення сенсорів, актуаторів та інших приладів, зокрема:

- три 12-розрядні аналого-цифрові перетворювачі (ADC);
- два порти SPI інтерфейсу;
- два порти I²C інтерфейсу;
- два порти UART інтерфейсу;
- 16 каналів PWM (широтна-імпульсна модуляція).

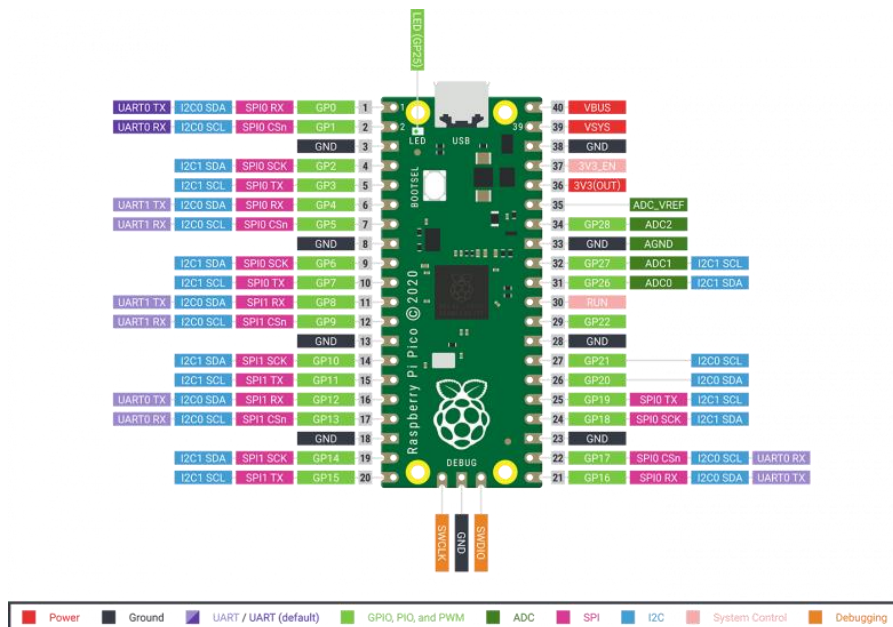


Рисунок 2.2 – Raspberry Pi Pico

У застосунку Wokwi передбачена палітра периферійних приладів (сенсорів, актуаторів) та електронних елементів, які можна включити до схеми за допомогою опції **Add a new part** (синя кнопка «+»).

В якості сенсорного вузла у схемі будемо використовувати цифровий датчик температури та вологості DHT-22, а актуаторами будуть світлодіоди (рис. 2.3).

Примітка 1. Вибір елементів із палітри пристроїв та графічний синтез схеми здійснюється за допомогою миші. При наведенні фокуса (маркера) на контакти плати мікроконтролера відображається найменування цього контакту. При виділенні пристрою на схемі з'являється знак питання, натиснувши на який можна отримати довідку про цей пристрій. Розташування пристроїв, з'єднання їх контактів провідниками також здійснюється за допомогою миші. Кут положення пристрою можна вибрати за допомогою клавіші 'R'.

Примітка 2. Відредагувати схему та задати деякі атрибути можна інтерактивно у файлі diagram.json (див. лістинг 2.1).

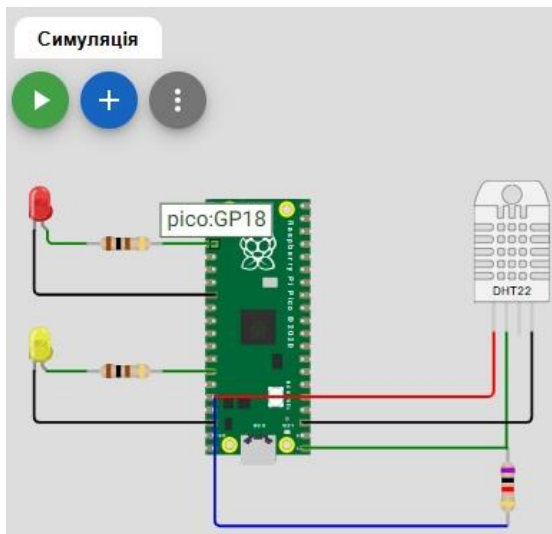


Рисунок 2.3 – Приклад схеми

Лістинг 2.1 – Опис схеми прикладу на мові JSON

```
{
  "version": 1,
  "author": "Ravil' Kudermetov",
  "editor": "wokwi",
  "parts": [
    {
      "type": "wokwi-pi-pico",
      "id": "pico",
      "top": -52.85,
      "left": -82.6,
      "rotate": 180,
      "attrs": { "env": "micropython-20241129-v1.24.1" }
    },
    {
      "type": "wokwi-resistor",
      "id": "r1",
      "top": 158.4,
      "left": 114.65,
      "rotate": 90,
      "attrs": { "value": "7000" }
    }
  ]
}
```

```

    },
    {
      "type": "wokwi-dht22",
      "id": "dht1",
      "top": -66.9,
      "left": 119.4,
      "attrs": { "temperature": "42", "humidity": "68" }
    },
    {
      "type": "wokwi-led",
      "id": "led1",
      "top": -70.8,
      "left": -226.6,
      "attrs": { "color": "red" }
    },
    {
      "type": "wokwi-resistor",
      "id": "r2",
      "top": -24.85,
      "left": -172.8,
      "attrs": { "value": "100" }
    },
    {
      "type": "wokwi-led",
      "id": "led2",
      "top": 34.8,
      "left": -226.6,
      "attrs": { "color": "yellow" }
    },
    {
      "type": "wokwi-resistor",
      "id": "r3",
      "top": 71.15,
      "left": -172.8,
      "attrs": { "value": "100" }
    }
  ],
  "connections": [
    [ "r1:2", "pico:3V3", "blue", [ "h0" ] ],
    [ "r1:1", "dht1:SDA", "green", [ "v0" ] ],
    [ "dht1:VCC", "pico:3V3", "red", [ "v0" ] ],
    [ "dht1:GND", "pico:GND.1", "black", [ "v0" ] ],

```

```
[ "led1:A", "r2:1", "green", [ "v0" ] ],
[ "pico:GP18", "r2:2", "green", [ "h0" ] ],
[ "pico:GND.6", "led1:C", "black", [ "h0" ] ],
[ "pico:GP0", "r1:1", "green", [ "h0" ] ],
[ "pico:GND.8", "led2:C", "black", [ "h0" ] ],
[ "led2:A", "r3:1", "green", [ "v0" ] ],
[ "pico:GP28", "r3:2", "green", [ "h0" ] ]
],
"dependencies": {}
}
```

DHT-22 – цифровий датчик температури та вологості, в якому для вимірювання температури в діапазоні від 0 до 50°C використовується терморезистор та ємнісний датчик вологості з діапазоном від 20 до 80%. Напруга живлення 3.0-6.0 В. Частота поновлення вимірювання ≤ 5 Гц. (Один вимір у 2 секунди).

Дані температури та вологості передаються в контролер по одному провіднику та мають послідовний інтерфейс, де одиниця та нуль кодуються різною тривалістю імпульсу. Пакет даних складається з 40 біт і його передача триває приблизно 4 мс. Передача даних синхронізується контролером, який для зчитування кожного пакета даних вимірювання надсилає датчику сигнал старту. Після передачі даних датчик переходить у режим низького електроспоживання.

Корпус датчика DHT-22 показано на рис. 2.4. Він має чотири виводи: 1 – напруга живлення (VCC), 2 – дані вимірювання (OUT), 3 – не використовується, 4 – підключення до землі (GND). Якщо довжина провідника від датчика до контролера менше 20 м, рекомендується підключати підтягуючий (pull up) резистор 4.7-10 кОм між виводами OUT і VCC.

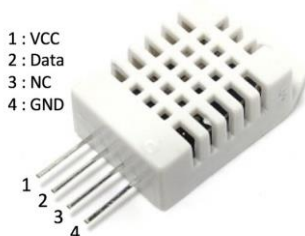


Рисунок 2.4 – Датчик температури та вологості DHT-22

Логіка функціонування схеми програмується на мові MicroPython, а також CircuitPython, C++. Для програмування використовується вбудований редактор коду. Після запуску програми на консолі можна побачити результат її роботи (рис. 2.5). Код даного прикладу наведено в лист. 2.2.

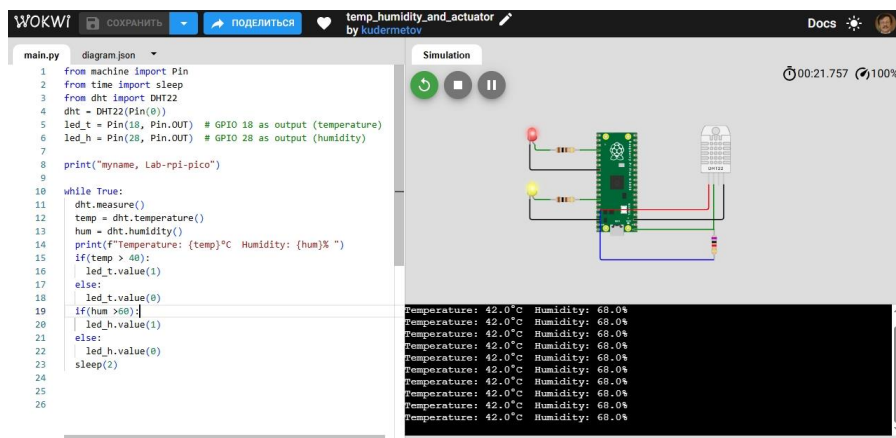


Рисунок 2.5 – Приклад симуляції

Лістинг 2.2 – Програма функціонування схеми

```

from machine import Pin
from time import sleep
from dht import DHT22
dht = DHT22(Pin(0))
#GPIO 18 as output (temperature)
led_t = Pin(18, Pin.OUT)
#GPIO 28 as output (humidity)
led_h = Pin(28, Pin.OUT)

print("myname, Lab-rpi-pico")

while True:
    dht.measure()
    temp = dht.temperature()
    hum = dht.humidity()
    print(f"Temperature: {temp} C Humidity: {hum}% ")
    if(temp > 40):

```

```

    led_t.value(1)
else:
    led_t.value(0)
if(hum > 60):
    led_h.value(1)
else:
    led_h.value(0)
sleep(2)

```

Відповідно до коду програми схема функціонує таким чином. Кожні 2 секунди вимірюються температура і вологість навколишнього середовища. Їх початкові значення задані у файлі `primer.json` (42°C, 68%). Якщо температура вище 40°C загоряється червоний світлодіод, якщо вологість більше 68% – загоряється жовтий світлодіод.

Для перевірки логіки роботи схеми можна в *процесі симуляції* змінити параметри довкілля. Для цього на схемі слід кнопкою миші натиснути на зображення датчика. В результаті на екрані з'явиться вікно з поточними значеннями температури та вологості, які можна мишкою змінити та переконатися у зміні цих параметрів по роботі схеми (ввімкнення/вимкнення світлодіодів) та значенням, що виводяться в консолі (рис. 2.6).

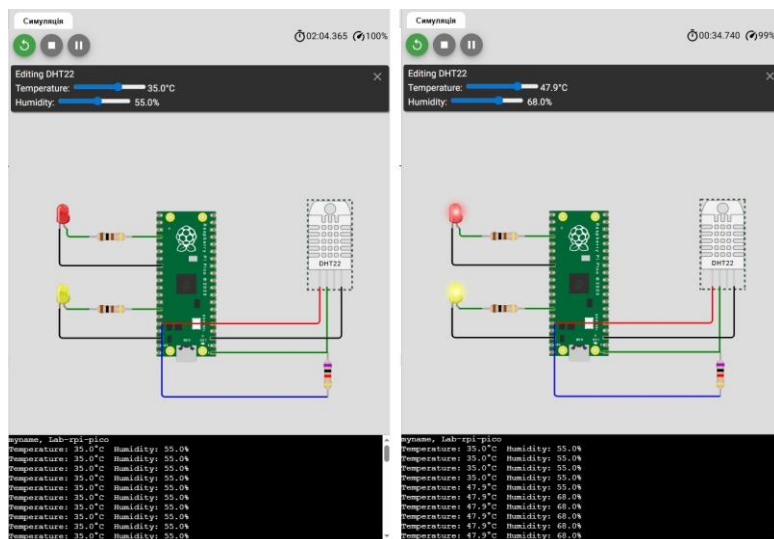


Рисунок 2.6 – Зміна температури та вологості довкілля

2.2 Завдання до лабораторної роботи

Завдання 1. Контроль температури та вологості у приміщенні.

Додати у схему прикладу зелений світлодіод. Для опису логіки роботи схеми введемо такі позначення (алфавіт входів та виходів): Т – температура; Н – вологість; N (normal) – нормальний діапазон температури/вологості; AN (above normal) – вище за норму; BN (below normal) – нижче за норму; G, R, Y – колір світлодіоду (зелений, червоний, жовтий); «+/-» – стани світлодіоду (включений/вимкнений). Нормальним діапазоном температури вважається 18-24°C, нормальним діапазоном вологості – 40-60%. Схема має виконувати наступну логіку (див. табл. 2.1).

Таблиця 2.1 – Логіка роботи схеми

T	H	G	R	Y
N	N	+	-	-
AN	N	+	+	-
N	AN	+	-	+
AN	AN	-	+	+
BN	N	-	+	-
N	BN	-	-	+
BN	BN	-	-	-

Завдання 2. Контроль освітлення у приміщенні.

Розробіть проект, у якому плата RPi Pico керує освітленням у кімнаті.

Проект повинен включати фоторезисторний датчик (LDR, light-dependant resistor), який вимірює рівень освітлення, та світлодіод, який імітує ввімкнення світла, коли темніє.

Технічні характеристики фоторезистора наведені у посиланні <https://docs.wokwi.com/parts/wokwi-photoresistor-sensor>.

Фоторезистор має два основні параметри: $r/10$ – опір фоторезистора при рівні освітленості 10 лк (для цієї моделі приладу він дорівнює 50 кОм); γ – визначає нахил графіку $\gamma = \log(R_{LDR})/\log(L)$, де R_{LDR} – опір фоторезистора (кОм), L – значення освітленості (лк). Для цієї моделі приладу $\gamma = 0.7$ (рис. 2.7).

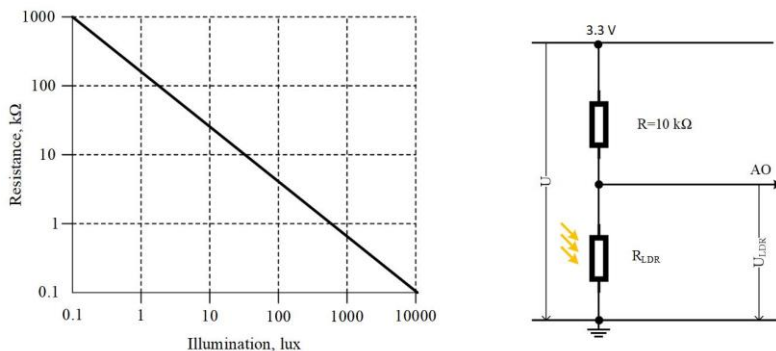


Рисунок 2.7 – Модель фоторезистора

Електрична схема фоторезисторного датчика наведена на рис. 2.7. Відповідно до цієї схеми опір фоторезистора та освітленість можна обчислити за формулами:

$$R_{LDR} = R \frac{U_{LDR}}{U - U_{LDR}}$$

$$L = \left(\frac{r l 10 \cdot 10^\gamma}{R_{LDR}} \right)^{1/\gamma}$$

Напруга U_{LDR} вимірюється шляхом підключення виходу АО до входу 16-розрядного аналого-цифрового перетворювача (ADC) плати RPi Pico (контакти GP26, GP27 або GP28). Перетворення 16-розрядного коду (позначимо як $u16$), отриманого з ADC, у значення напруги виконується за формулою:

$$U_{LDR} = U \cdot u16 / 65535$$

Світлодіод, що імітує актуатор, повинен включатись при освітленості менше 10 лк.

2.3 Контрольні питання

- 1 Яке призначення Wokwi?
- 2 Які можливості надає середовище Wokwi?
- 3 Яке призначення плати Raspberry Pi Pico?
- 4 Які проекти розумного будинку можна реалізувати з використанням Raspberry Pi Pico? Назвіть три або п'ять прикладів. Перерахуйте, що ще потрібно для реалізації таких проєктів.

3 ЛАБОРАТОРНА РОБОТА № 3

РОЗРОБКА ПРОТОТИПІВ ІОТ ЗАСТОСУВАНЬ У СЕРЕДОВИЩІ NODE-RED

Мета роботи: навчитися інсталювати та застосовувати графічний інструмент Node-RED при розробці власних додатків.

3.1 Загальні відомості

Node-RED – це конфігуратор IoT, графічний інструмент, який дозволяє створити просту схему для об'єднання кількох програм, вебсайтів і апаратного забезпечення. Node-RED є візуальним інструментом, який спрощує передачу і обробку повідомлень в IoT.

Одним з найбільш важливих факторів, які обмежують розвиток IoT, є відсутність зручних засобів розробки правил взаємодії пристроїв IoT між собою. Для вирішення цього завдання розроблено Node-RED, який дозволяє через браузер побудувати схему взаємодії пристроїв між собою і зовнішніми системами.

Дане рішення зручне, як проміжне, для зв'язку пристроїв різного типу між собою та/або з системою автоматизації, або СКБД, або іншою хмарою тощо. З використанням додаткових пакетів Node-RED можна використовувати для створення простих систем автоматизації розумного будинку, але це рішення буде відносно обмеженим через неповне покриття функціональних потреб такого проєкту.

Node-RED працює на програмній платформі Node.js, і розроблений для роботи на відносно малопродуктивних системах, таких як: RaspberryPi, BeagleBoneBlack, Arduino.

З урахуванням вищезазначених факторів, Node-RED зручно використовувати на шлюзах між різними мережами пристроїв IoT, які функціонують на власних, як правило, більш простих протоколах, і традиційним Інтернетом, що побудовані на TCP/IP, UDP. У цьому випадку він дозволить більш оптимально використовувати вільні ресурси шлюзу, який працює, як правило, на Linux.

З документацією Node-RED можна ознайомитись на офіційному сайті (<https://nodered.org/>).

3.2 Завдання до лабораторної роботи

Завдання 1. Виконати інсталяцію Node-RED в операційній системі Raspberry Pi OS.

Порядок виконання роботи.

1. Для інсталяції Node-RED має використовуватися наступне програмне забезпечення:

- у якості засобу віртуалізації встановити та використати програмний продукт VirtualBox 4.3.10 – файл VirtualBox-4.3.10-93012-Win.exe (<https://www.virtualbox.org/>);

- образ операційної системи 2021-01-11-raspbios-buster-i386.

Після того, як засіб віртуалізації VirtualBox буде встановлено, образ raspbios-buster-i386 слід використати у якості образу ISO файлу.

Обов'язково під час створення віртуальної машини увімкнути режим UEFI (Unified Extensible Firmware Interface), так як зображено на рисунку 3.1. Також в налаштуваннях мережевих інтерфейсів слід обрати опцію «Проміжний адаптер».

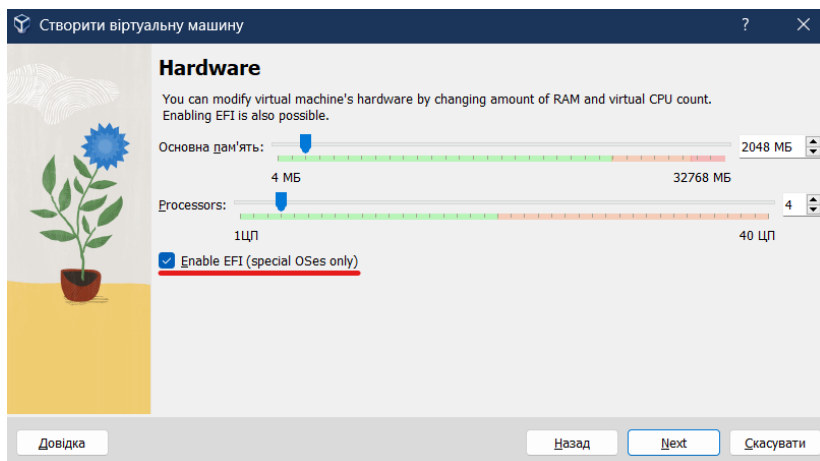


Рисунок 3.1 – Увімкнення UEFI режиму

Після створення віртуальної машини, інстальовати систему згідно вказівок інсталятора. Робочий стіл Raspberry Pi OS див. рис. 3.2.

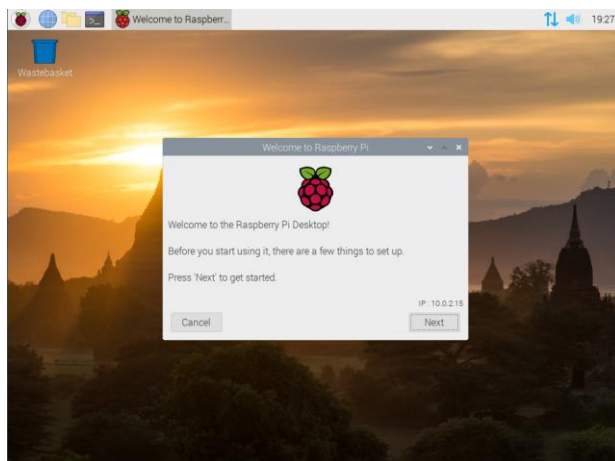


Рисунок 3.2 – Робочий стіл Raspberry Pi OS

2. Інсталиувати Node-RED на Raspberry Pi OS. Для цього необхідно відкрити термінал та вписати команду:

```
bash<(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered) .
```

3. Під час встановлення погодьтеся з усіма етапами (рис. 3.3).

```
Running Node-RED install for user pi at /home/pi on debian

This can take 20-30 minutes on the slower Pi versions - please wait.

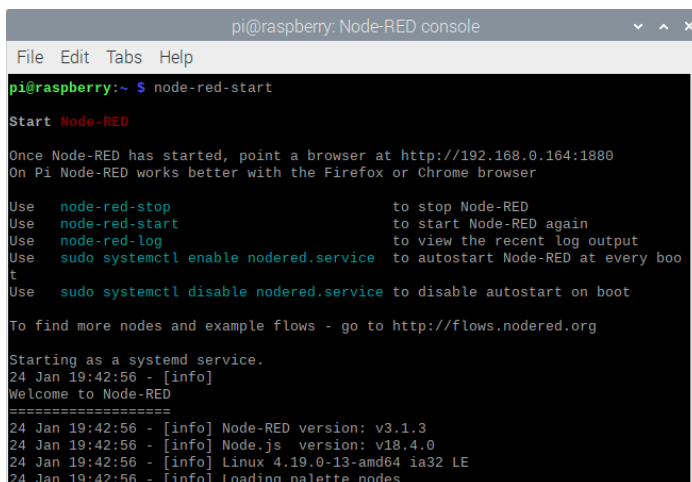
  Stop Node-RED
  Remove old version of Node-RED
  Remove old version of Node.js
  Install Node.js
  Clean npm cache
  Install Node-RED core
  Move global nodes to local
  Npm rebuild existing nodes
  Install extra Pi nodes
  Add shortcut commands
  Update systemd script

Any errors will be logged to /var/log/nodered-install.log
```

Рисунок 3.3 – Процес інсталяції Node-RED

4. Запускаємо сервіс командою: `node-red-start`. В терміналі виведеться адреса, яку потрібно буде ввести в браузері на вашому ПК.

На цьому етапі запуску графічного конфігуратора Node-RED завершено. Процес запуску Node-RED див. рис. 3.4.



```

pi@raspberrypi: Node-RED console
File Edit Tabs Help
pi@raspberrypi:~ $ node-red-start
Start Node-RED

Once Node-RED has started, point a browser at http://192.168.0.164:1880
On Pi Node-RED works better with the Firefox or Chrome browser

Use node-red-stop to stop Node-RED
Use node-red-start to start Node-RED again
Use node-red-log to view the recent log output
Use sudo systemctl enable nodered.service to autostart Node-RED at every boot
Use sudo systemctl disable nodered.service to disable autostart on boot

To find more nodes and example flows - go to http://flows.nodered.org

Starting as a systemd service.
24 Jan 19:42:56 - [info]
Welcome to Node-RED
=====
24 Jan 19:42:56 - [info] Node-RED version: v3.1.3
24 Jan 19:42:56 - [info] Node.js version: v18.4.0
24 Jan 19:42:56 - [info] Linux 4.19.0-13-amd64 ia32 LE
24 Jan 19:42:56 - [info] Loading palette nodes
  
```

Рисунок 3.4 – Процес запуску Node-RED

Завдання 2. Створення потоків у графічному конфігураторі Node-RED.

Робоче вікно конфігуратора Node-RED складається з декількох областей (рис. 3.5):

- потоки (можна створювати їх необхідну кількість за допомогою кнопки «+»);
- перелік вузлів, які можна пересувати до області потоків;
- відлагоджувач (є можливість фільтрування відповідною кнопкою у цій області вікна перегляду результатів роботи поточного потоку або всіх наявних);
- кнопка розгортання схеми «Deploy».

Завдання 2а. Зберіть схему, яка складається із вхідного елемента (inject) та елемента відлагодження (debug). При цьому на вихід кожні 5 сек буде виводитись посилання «hello».

Завдання 2б. Зберіть схему, яка забезпечує перевірку мережного підключення і доступності ресурсу google.com кожні 3 с, і

зберігає результат перевірки у файлі на жорсткому диску (використовуються елементи ping, file).

Завдання 2в. Доповніть схему із завдання 2а ще одним вхідним вузлом та функціональним вузлом. При цьому, якщо на вхід подається питання «How are you?», то виводиться відповідь «I am good», в іншому випадку – «Hello». Результат наведено на рис. 3.6.

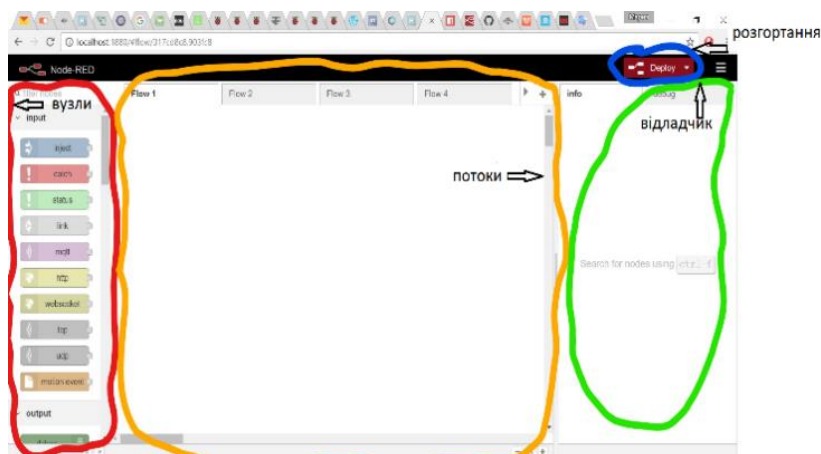


Рисунок 3.5 – Робоче вікно Node-RED у браузері з поміченими областями

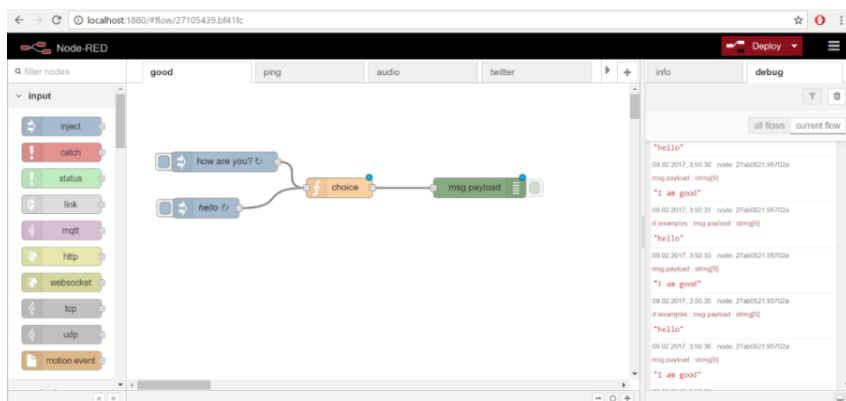


Рисунок 3.6 – Результат виконаного завдання

Завдання 2г. Отримайте дані із зовнішнього ресурсу.

Перед початком імпорту потрібно завантажити вузол OpenWeatherMap. Для цього у верхньому правому куту натисніть на «Керування палітрою» та встановіть node-red-node-openweathermap.

Для імпорту прикладу наведеного у ліст. 3.1 натискаємо Ctrl-I та копіюємо текст лістингу у вікно, яке при цьому відкривається (рис. 3.7). Обираємо «newflow» та натискаємо кнопку «Import».

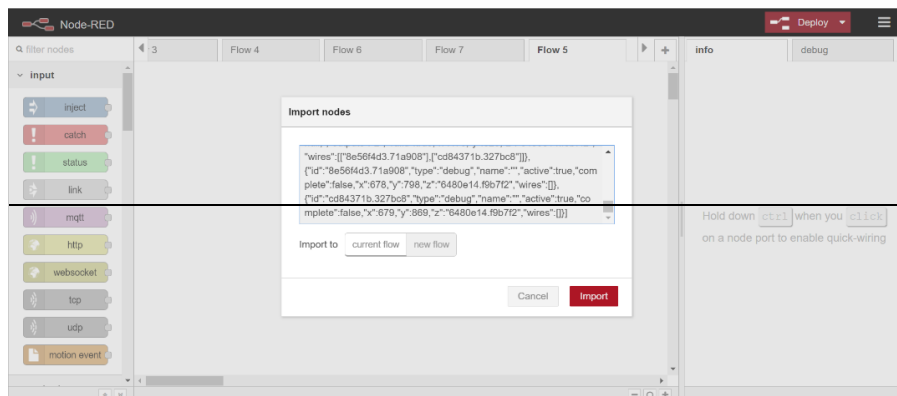


Рисунок 3.7 – Імпорт даних

Лістинг 3.1 – Отримання даних із зовнішнього ресурсу

```
[
  {
    "id": "b211d23335b8d910",
    "type": "tab",
    "label": "Weather",
    "disabled": false,
    "info": "",
    "env": []
  },
  {
    "id": "ae5cd3dd63bc0e5e",
    "type": "inject",
    "z": "b211d23335b8d910",
    "name": "",
    "props": [
      {
        "p": "payload"
```

```

    }
  ],
  "repeat": "600",
  "crontab": "",
  "once": true,
  "onceDelay": "1",
  "topic": "",
  "payload": "",
  "payloadType": "str",
  "x": 130,
  "y": 160,
  "wires": [
    [
      "774d8fe0d371a50e"
    ]
  ]
},
{
  "id": "bf97f0dedc5b197b",
  "type": "debug",
  "z": "b211d23335b8d910",
  "name": "debug 1",
  "active": true,
  "tosidebar": true,
  "console": true,
  "tostatus": false,
  "complete": "payload",
  "targetType": "msg",
  "statusVal": "",
  "statusType": "auto",
  "x": 570,
  "y": 160,
  "wires": []
},
{
  "id": "774d8fe0d371a50e",
  "type": "openweathermap",
  "z": "b211d23335b8d910",
  "name": "Weather",
  "wtype": "current",
  "lon": "35.19976172607571",
  "lat": "47.81367667784276",

```

```

    "city": "",
    "country": "",
    "language": "uk",
    "x": 360,
    "y": 160,
    "wires": [
      [
        "bf97f0dedc5b197b"
      ]
    ]
  }
]

```

Отримаємо схему з імпортованими даними (див. рис. 3.8).

Для роботи цієї схеми потрібно зареєструватися на сайті <https://openweathermap.org/>, у вкладці «Apikeys» створити новий ключ та вставити його в налаштуваннях «Weather». Після цього натиснути «Розгорнути» та побачити у консолі прогноз погоди у Запоріжжі на поточний момент.

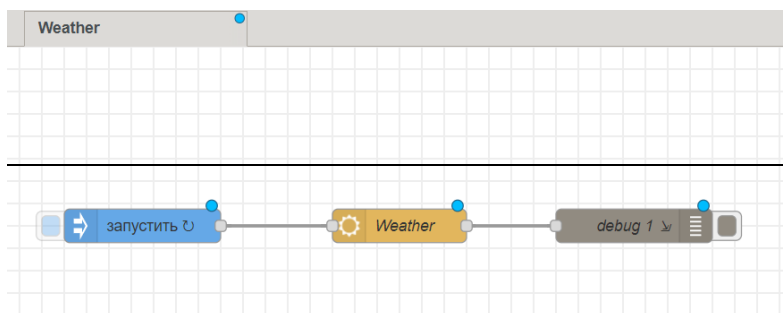


Рисунок 3.8 – Схема з імпортованими даними

3.3 Контрольні питання

- 1 Що собою являє Node-RED?
- 2 З яким програмним та апаратним забезпеченням є сумісним Node-RED?
- 3 Які є аналоги Node-RED?
- 4 У чому основні недоліки аналогів Node-RED?
- 5 У чому основні переваги Node-RED?

4 ЛАБОРАТОРНА РОБОТА №4

ВИКОРИСТАННЯ ОПЕРАЦІЙНОЇ СИСТЕМИ HOME ASSISTANT ДЛЯ СТВОРЕННЯ РОЗУМНОГО БУДИНКУ

Мета роботи: ознайомитися з особливостями роботи Home Assistant.

4.1 Загальні відомості

Home Assistant – вільне програмне забезпечення з відкритим вихідним кодом (<https://www.home-assistant.io/>), програмна платформа, яка виконує роль центрального вузла вашого розумного будинку. Вона дозволяє підключати, контролювати та автоматизувати різні пристрої, незалежно від їхньої марки або виробника, та сервіси з єдиного інтерфейсу, від освітлення і комфортного мікроклімату до систем безпеки будинку. Системою можна керувати з одного додатку, інформаційної панелі або навіть голосом.

Можна автоматизувати виконання необхідних повсякденних завдань без участі людини на основі тригерів, умов і дій. Наприклад, налаштувати автоматичне ввімкнення світла, коли ви повертаєтесь додому, або температурний режим відповідно до погоди.

На відміну від деяких інших платформ розумного дому, Home Assistant зберігає дані локально. Це означає, що інформація про пристрій і шаблони використання не надсилаються до хмари, захищаючи конфіденційність.

Завдяки відкритості Home Assistant, будь-хто може зробити свій внесок у його розвиток і налаштувати його під свої потреби. Для цієї платформи існує велика кількість доповнень, інтеграцій та автоматизацій, що надає можливість створення дійсно персоналізованого розумного будинку.

4.2 Завдання до лабораторної роботи

Завдання 1. Виконати інсталяцію Home Assistant в операційній системі Raspberry Pi OS.

Порядок виконання роботи.

1. Запустити віртуальну машину с операційною системою Raspberry Pi OS і перейти до вкладки «Пристрої» → «Оптичні диски», вибравши 2021-01-11-raspbios-buster-i386. На етапі запуску оберіть

пункт «System setup» → «Boot Manager» → «UEFI VBOX CD-ROM». Після цього у новому вікні оберіть тип запуску «Run with persistence».

2. В операційній системі у терміналі виконати команду `sudo apt install gnome-disk-utility` для встановлення Gnome Disk Utility.

3. Завантажити образ Home Assistant за наданим посиланням (https://github.com/home-assistant/operating-system/releases/download/11.4/haos_generic-x86-64-11.4.img.xz).

4. Відкрити Gnome Disk Utility через меню «Пуск» → «Accessories» → «Disks» (рис. 4.1), у верхньому куті натиснути «Детальніше» та обрати опцію «Restore Disk Image». Обрати завантажений образ та натиснути «Start Restoring». Ввести пароль «raspberrypi». Після завершення процесу відновлення перезапустити систему. Вилучити образ системи з віртуального CD-ROM.

5. Відкрийте браузер на вашому комп'ютері та перейдіть за адресою, яку ви отримали від Home Assistant. Завершіть початкову реєстрацію (рис. 4.2, 4.3).

6. Після реєстрації, у браузері з'явиться відображення нашого пристрою (рис. 4.4).

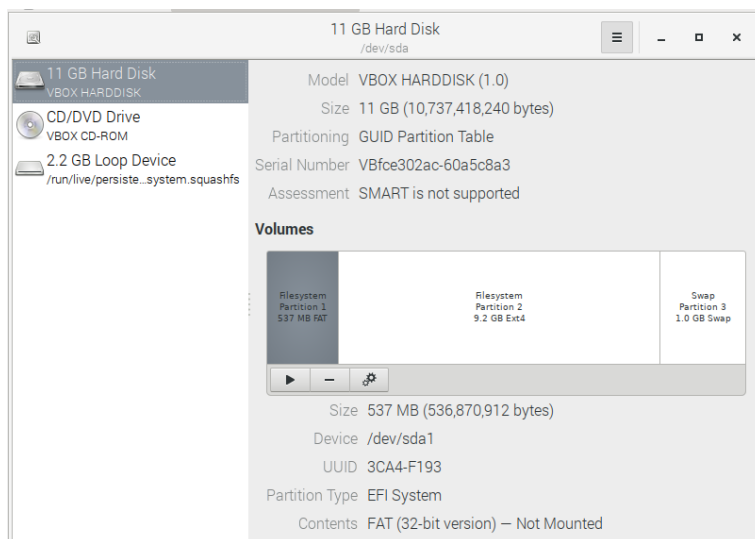


Рисунок 4.1 – Вікно Gnome Disk Utility

```

Waiting for the Home Assistant CLI to be ready...

[ASCII Art: Home Assistant logo made of dashes]

Welcome to the Home Assistant command line.

Waiting for Supervisor to startup...
System information
  IPv4 addresses for enp0s3: 192.168.0.164/24
  IPv6 addresses for enp0s3: fe80::d81e:7ada:3c55:b851/64

  OS Version:                Home Assistant OS 11.4
  Home Assistant Core:       landingpage

  Home Assistant URL:         http://homeassistant.local:8123
  Observer URL:               http://homeassistant.local:4357
ha > _

```

Рисунок 4.2 – Вікно Home Assistant

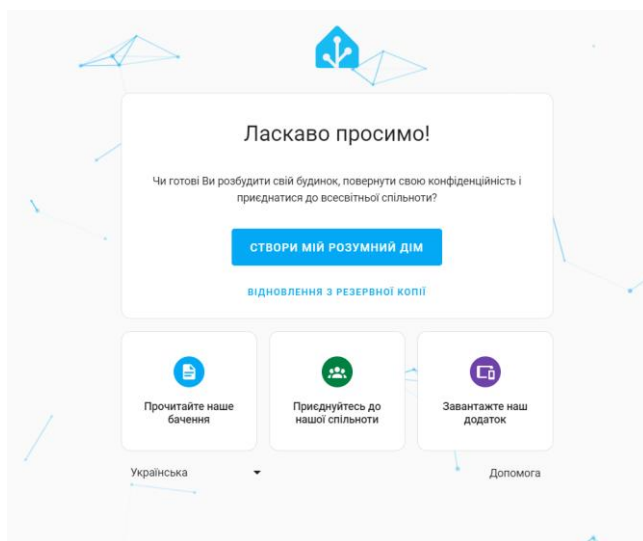


Рисунок 4.3 – Початкове вікно налаштування Home Assistant

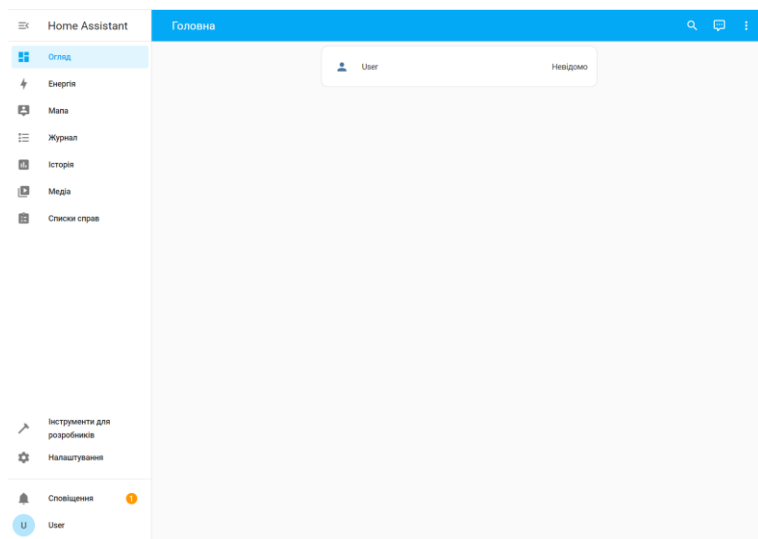


Рисунок 4.4 – Інтерфейс управління пристроєм

Завдання 2. Налаштування Home Assistant для роботи з сервісами.

1. Щоб додавати служби на головний екран спочатку потрібно взяти його під контроль. Для цього на головному екрані натиснути на три крапки у правому верхньому куті, обрати «Редагувати панель», знов три крапки та обрати «Взяти під контроль» (рис. 4.5).

2. Переглянути можливості наданого інтерфейсу управління пристроєм можна у розділах: «Енергія» (рис.4.6), «Медіа джерела» (рис. 4.7), «Служби» (рис. 4.8), «Магазин доповнень» (рис. 4.9) тощо.

3. Спробуйте встановити сервіс та додати його на головний екран. Перейдіть до розділу «Налаштування» → «Пристрої та сервіси» → «Додати інтеграцію». У вікні оберіть «Open Weather Map». У полі «Ключ API» введіть ключ Open Weather Map, нижче введіть широту та довготу (наприклад, 47.81367667784276, 35.19976172607571). Поверніться до розділу служб, виберіть «Open Weather Map» та додайте інформацію про погоду на інформаційну панель (рис. 4.10).

4. Перегляньте інформацію щодо навантаження процесора через «Налаштування» → «Система» → «Устаткування» (рис. 4.11) та зайняте місце на диску у розділі «Зберігання» (рис. 4.12).

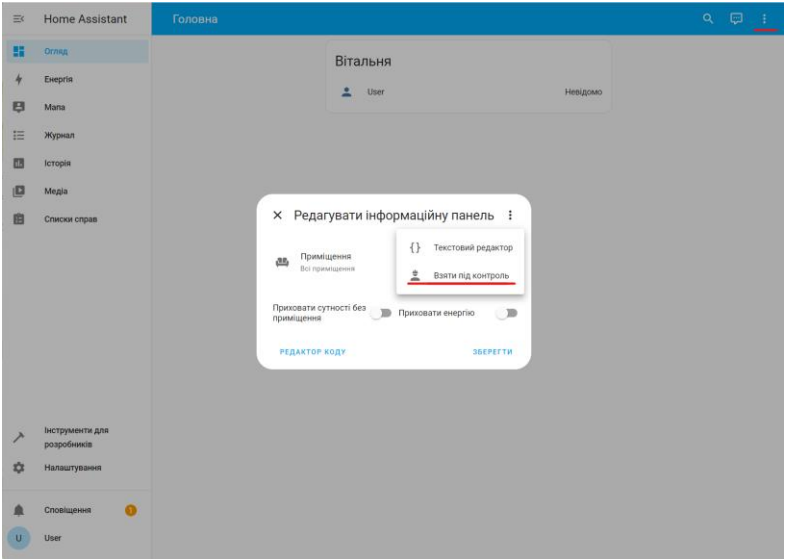


Рисунок 4.5 – Вікно налаштування

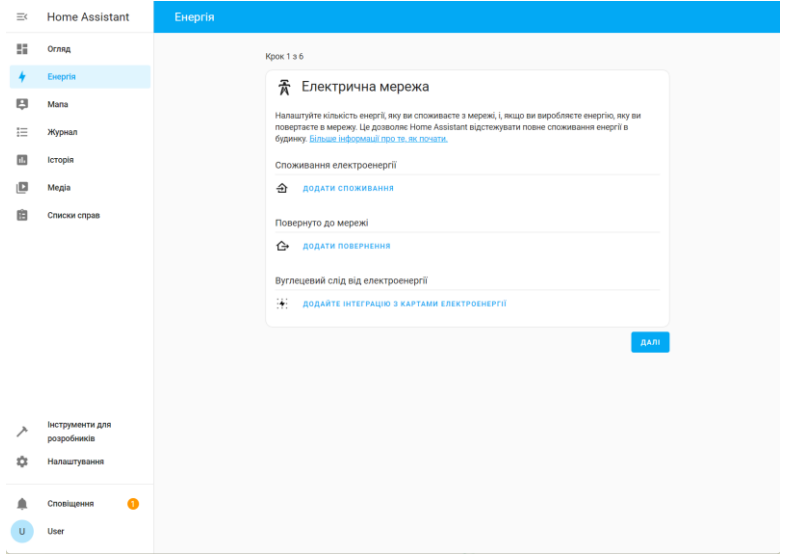


Рисунок 4.6 – Розділ «Енергія»

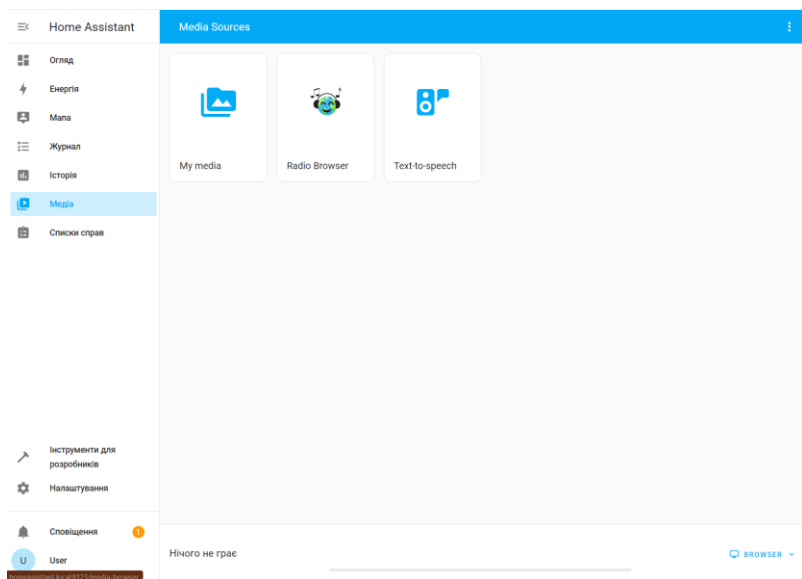


Рисунок 4.7 – Розділ «Медіа джерела»

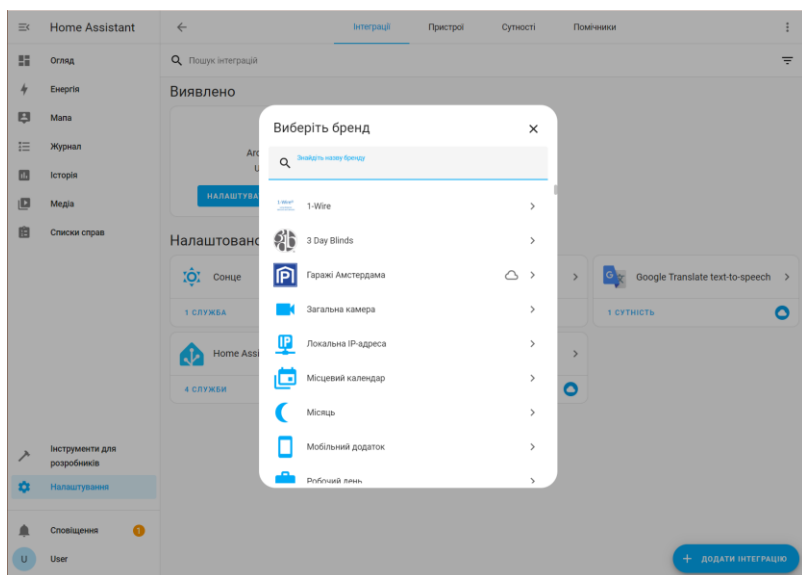


Рисунок 4.8 – Розділ «Служби»

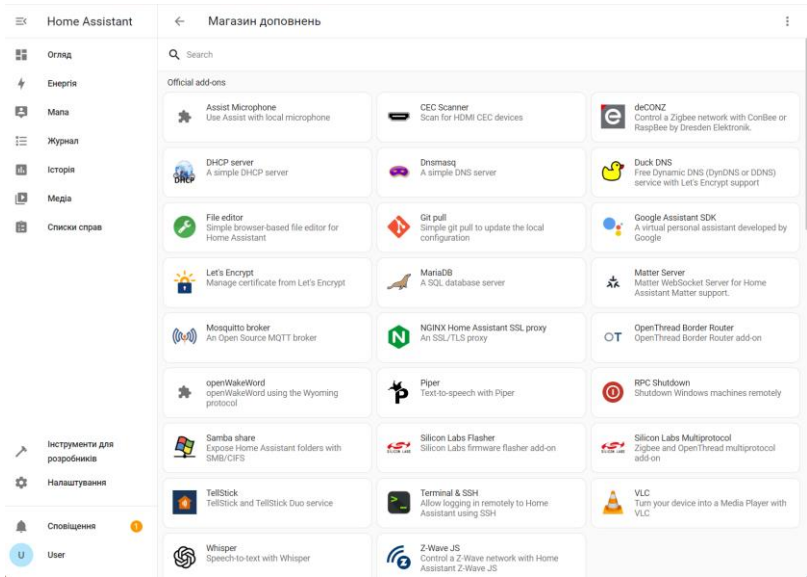


Рисунок 4.9 – Розділ «Магазин доповнень»

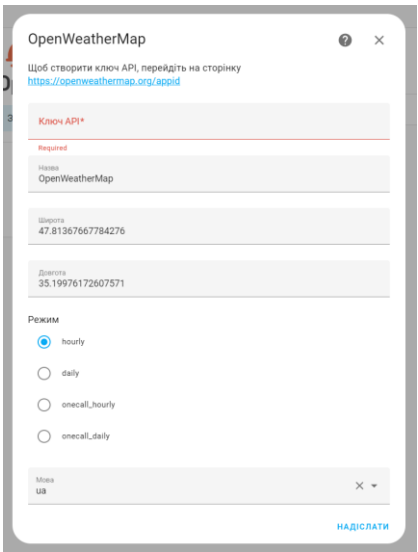


Рисунок 4.10 – Налаштування служби OpenWeatherMap

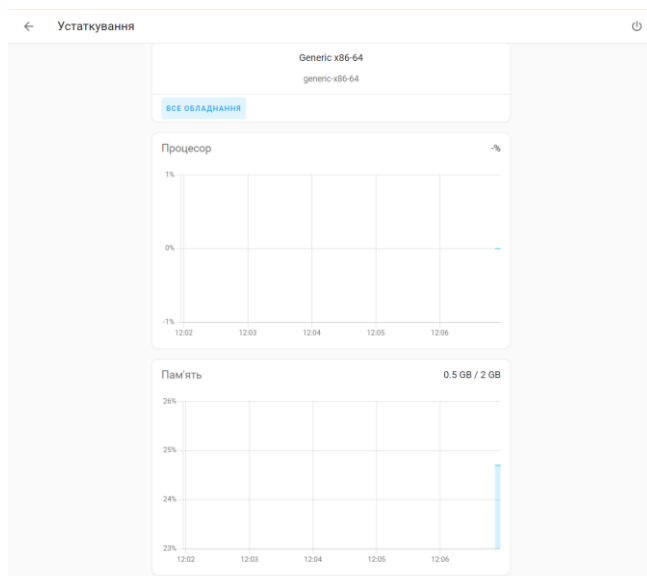


Рисунок 4.11 – Діаграма відображення навантаження оперативної пам'яті та процесора

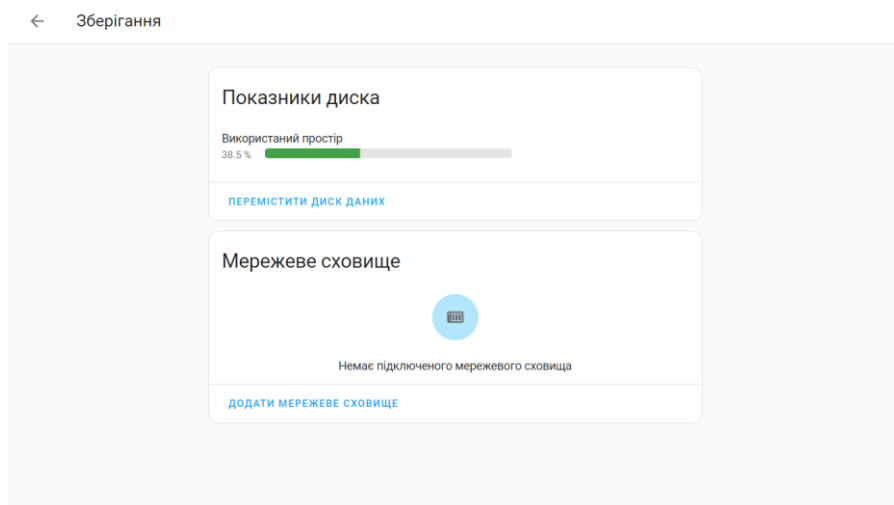


Рисунок 4.12 – Зайняте місце на диску

Після закінчення роботи з Home Assistant перейдіть у розділ «Система», у верхньому правому куті натисніть на кнопку вимкнення → «Розширені опції» → «Вимкнути систему» (рис. 4.13). Результат виконання дії показано на рис. 4.14.

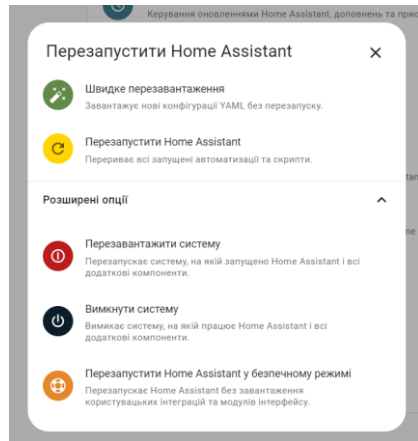


Рисунок 4.13 – Варіанти перезавантаження і вимкнення Home Assistant

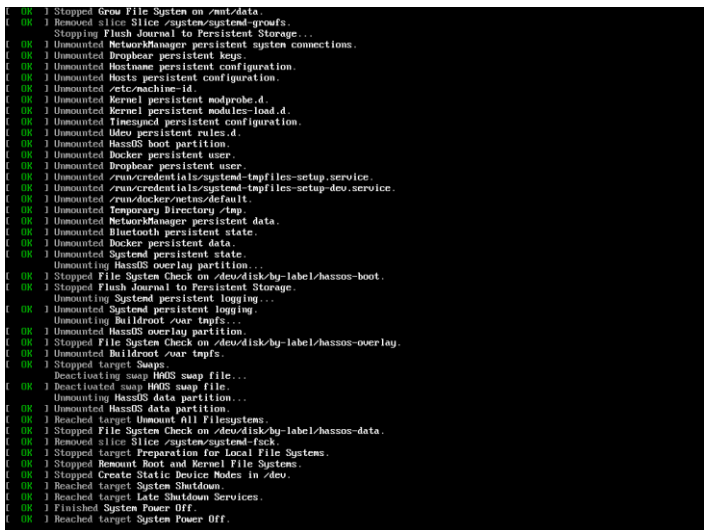


Рисунок 4.14 – Процес вимкнення HomeAssistant

4.3 Контрольні питання

- 1 Що таке HomeAssistant?
- 2 Які можливості надає HomeAssistant?
- 3 На яких пристроях може працювати HomeAssistant?
- 4 Які комунікаційні стандарти підтримує HomeAssistant?
- 5 Які функції HomeAssistant неможливо налаштувати з користувацького інтерфейсу?
- 6 Які можливості містить в собі конфігураційний файл `configuration.yaml`, яким чином можна його редагувати і як налаштувати доступ до файлу?
- 7 Які є аналоги HomeAssistant?
- 8 У чому основні переваги та недоліки HomeAssistant перед аналогами?

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1 Feamster, N. The road to SDN: an intellectual history of programmable networks [Text] / N. Feamster, J. Rexford, E. Zegura // ACM SIGCOMM Computer Communication Review. – 2014. – 44(2). – P. 87–98. doi: 10.1145/2602204.2602219

2 Nadeau, T. D. SDN: Software Defined Networks [Text] / T. D. Nadeau, K. Gray. – Sebastopol, CA, USA :O'Reilly Media, 2013. – 384 p. ISBN: 978-1-449-34230-2

3 OpenFlow [Electronic resource]. – Access mode: <https://www.opennetworking.org/sdn-resources/openflow>

4 Goransson, P. Software Defined Networks: A Comprehensive Approach [Text] / P. Goransson, C. Black, T. Culver. – [2nd ed.]. – Waltham, MA, USA :Elsevier, 2016. – 436 p. ISBN: 9780128045558

5 Mininet: An Instant Virtual Network on your Laptop (or other PC) [Electronic resource]. – Access mode: <http://mininet.org>

6 Ketu F. Emulation of Software Defined Networks Using Mininet in Different Simulation Environments / F. Ketu, S. Askar // Proc. of 2015 6th International Conference on Intelligent Systems, Modelling and Simulation (Kuala Lumpur, Malaysia, February 9–12, 2015). – P. 205–210. doi: 10.1109/ISMS.2015.46

7 Raspberry Pi Foundation [Electronic resource]. – Access mode: <https://www.raspberrypi.org/>

8 Raspbian Documentation [Electronic resource]. – Access mode: <https://www.raspbian.org/RaspbianDocumentation>

9 Wang, S-Y. Comparison of SDN OpenFlow network simulator and emulators: EstiNet vs. Mininet [Text] / S-Y. Wang // Proc. of the 2014 IEEE Symposium on Computers and Communication (Funchal, Portugal, June 23–26, 2014). doi: 10.1109/ISCC.2014.6912609

10 Ivey, J. Comparing a Scalable SDN Simulation Framework Built on ns-3 and DCE with Existing SDN Simulators and Emulators [Text] / J. Ivey, H. Yang, C. Zhang, G. Riley // Proc. of the 2016 annual ACM Conference on SIGSIM Principles of Advanced Discrete Simulation (Banff, Alberta, Canada, May 15–18, 2016). – P. 153–164. doi: 10.1145/2901378.2901391

11 Hu, F. Network Innovation through OpenFlow and SDN: Principles and Design [Text] / F. Hu. – Boca Raton, FL, USA : CRC Press, 2014. – 520 p. ISBN 9781466572096

12 Wokwi: World's most advanced ESP32 simulator [Electronic resource]. – Access mode: <https://wokwi.com/>

13 Node-RED [Electronic resource]. – Access mode: <https://nodered.org/>

14 HomeAssistant [Electronic resource]. – Access mode: <https://www.home-assistant.io/>