

Міністерство освіти і науки України
Запорізький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних робіт з дисципліни
"Основи Інтернету речей"
Частина 2
для студентів спеціальності
123 "Комп'ютерна інженерія"
всіх форм навчання

Методичні вказівки до виконання лабораторних робіт з дисципліни "Основи Інтернету речей" Частина 2 для студентів спеціальності 123 "Комп'ютерна інженерія" всіх форм навчання.
/ Укл. Р.К. Кудерметов, М.Ю. Тягунова, В.В.Шкарупило, А.В.Тіменко – Запоріжжя: ЗНТУ, 2019. – 42 с.

Укладачі: Р.К. Кудерметов, к.т.н., доцент
М.Ю. Тягунова, к.т.н., доцент
В.В.Шкарупило, к.т.н., доцент
А.В.Тіменко, асистент

Рецензент: С.Ю. Скрупський к.т.н., доцент

Відповідальний за випуск: А.В.Тіменко, асистент

Затверджено:
на засіданні кафедри
"Комп'ютерні системи та
мережі"
Протокол № 11 від 20.05.2019

Рекомендовано до видання
НМК ФКНТ
Протокол № 10 від 31.05.2019

ЗМІСТ

ЗМІСТ	3
1 ЛАБОРАТОРНА РОБОТА № 4 ЗНАЙОМСТВО З WEBІОРІ	4
1.1 Загальні відомості.....	4
1.2 Завдання до лабораторної роботи	4
1.3 Контрольні питання.....	14
2 ЛАБОРАТОРНА РОБОТА №5 КЕРУВАННЯ СВІТЛОДІОДОМ	15
2.1 Загальні відомості.....	15
2.2 Складання моделі.....	15
2.3 Завдання до лабораторної роботи	16
2.4 Контрольні питання.....	20
3 ЛАБОРАТОРНА РОБОТА № 6. НАЛАШТУВАННЯ СЕРЕДОВИЩА MININET	22
3.1 Загальні відомості.....	22
3.2 Завдання до лабораторної роботи	24
3.3 Зміст звіту.....	31
3.4 Контрольні питання.....	32
4 ЛАБОРАТОРНА РОБОТА № 7. РОБОТА З ГРАФІЧНИМ ІНТЕРФЕЙСОМ MINIEDIT	33
4.1 Загальні відомості.....	33
4.2 Завдання до лабораторної роботи	34
4.3 Зміст звіту.....	41
4.4 Контрольні питання.....	41

1 ЛАБОРАТОРНА РОБОТА № 4 ЗНАЙОМСТВО З WEBIOPi

Мета роботи: ознайомитися з особливостями роботи пакету WebIOPi.

1.1 Загальні відомості

Фреймворк WebIOPi представляє пакет програм, спеціально розроблений для Raspberry Pi для віддаленого управління пристроями. Спільно з Raspberry Pi він реалізує технологію Internet of Thing (Інтернет речей). Пакет WebIOPi дозволяє створювати різні додатки користувача.

WebIOPi має наступні можливості:

вбудований Web - сервер, реалізований на мові Python;

вбудована підтримка більш ніж 30 пристроїв з інтерфейсами UART, SPI, I2C, 1-Wire;

бібліотеки Javascript / HTML для створення веб-інтерфейсу;

бібліотеки Python / Java для створення додатків для Android;

підтримка протокола CoAP (Constrained Application Protocol), призначений для управління і взаємодії між простими електронними пристроями через мережу.

WebIOPi має відкритий код, який може бути змінений користувачем. Це дозволяє збільшити кількість завдань для вирішення. Для настройки пакета під конкретну задачу змінюється файл конфігурації.

Для роботи з пакетом необхідно зайти за посиланням webiopi.trouch.com та зареєструватися.

1.2 Завдання до лабораторної роботи

Завдання 1. Підключення Raspberry Pi 3 за допомогою WebIOPi.

Хід роботи.

1. Зайти за адресою webiopi.trouch.com під зареєстрованим власним акаунтом.

2. Далі клацаємо Next та переходимо до вибору пристрою – обираємо Raspberry Pi (рис.1.1).

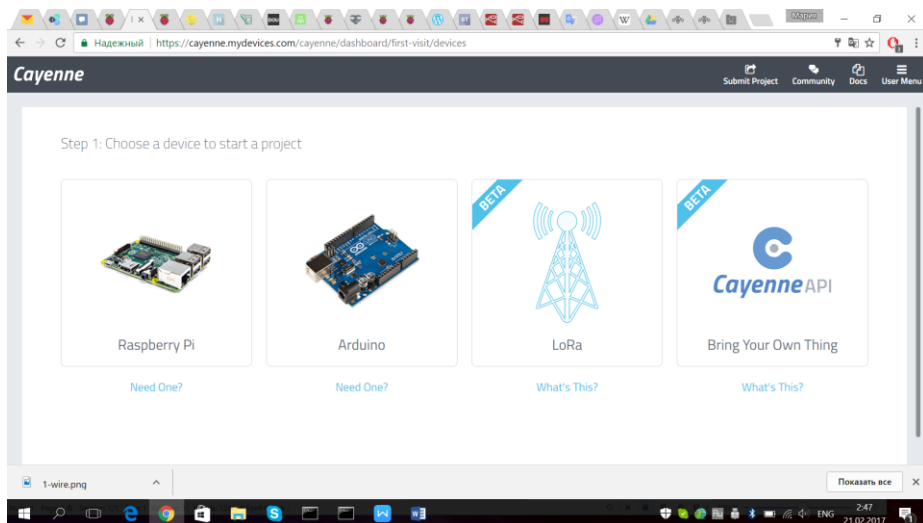


Рисунок 1.1 – Вибір пристрою

3. Клацаємо Next (рис.1.2) та підключаємо плату Raspberry Pi до електромережі (на фізичному рівні).

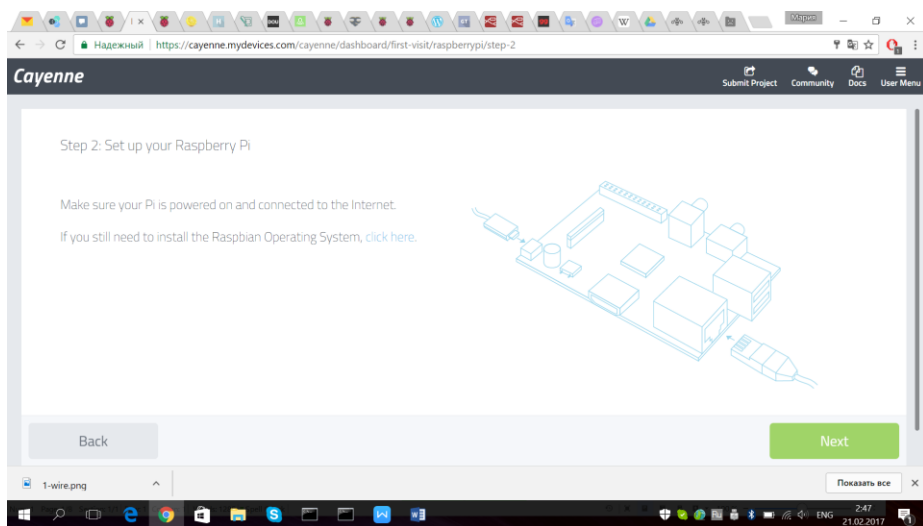


Рисунок 1.2 – Вибір підключеного пристрою

4. З'єднуємося із обраною платою (рис.1.3)

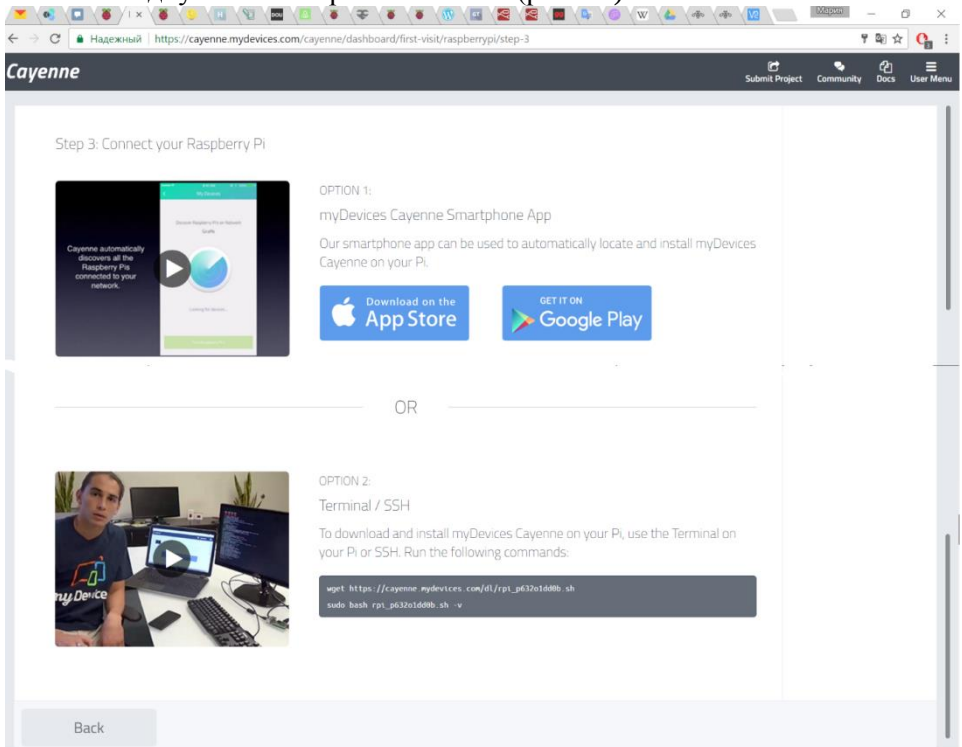


Рисунок 1.3 – Сторінка з'єднання з пристроєм

5. Запускаємо на комп'ютері VNC Viewer (забезпечує доступ до плати).

6. Відкриваємо командний рядок та вводимо (рис.1.4)

```
wget https://cayenne.mydevices.com/dl/rpi_p632o1dd0b.sh
```

```

pi@raspberrypi:~/Desktop $ wget https://cayenne.mydevices.com/dl/rpi_p632o1dd0b.sh
--2017-02-20 23:01:26-- https://cayenne.mydevices.com/dl/rpi_p632o1dd0b.sh
Распознаётся cayenne.mydevices.com (cayenne.mydevices.com)... 52.202.24.38, 54.87.151.232
Подключение к cayenne.mydevices.com (cayenne.mydevices.com)[52.202.24.38]:443...
соединение установлено.
HTTP-запрос отправлен. Ожидание ответа... 200 OK
длина: 1294 (1.3К) [application/x-sh]
Сохранение в: «rpi_p632o1dd0b.sh»

rpi_p632o1dd0b.sh 100%[=====] 1.26K ---KB/s за 0s

2017-02-20 23:01:28 (6.90 MB/s) - «rpi_p632o1dd0b.sh» сохранён [1294/1294]

pi@raspberrypi:~/Desktop $

```

Рисунок 1.4 – З'єднання плати з WebIOPi

7. Далі вводимо у командному рядку плати

```
sudo bash rpi_p632o1dd0b.sh -v
```

та отримуємо результати, що показано на рис.1.5

```

pi@raspberrypi:~/Desktop $ sudo bash rpi_p632o1dd0b.sh -v
continuing de new install...
--2017-02-20 23:01:31-- http://updates.mydevices.com/raspberrypi/mydevices-1.0.tar.gz
Размещение updates.mydevices.com (updates.mydevices.com): 54.140.225.117
Подключение к updates.mydevices.com (updates.mydevices.com)[54.140.225.117]:80... соединение установлено.
HTTP-запрос отправлен. Ожидание ответа... 200 OK
длина: 529417 (518К) [application/gzip]
Сохранение в: «mydevices/mydevices-1.0.tar.gz»

Home/~/Desktop/mydevices-1.0.tar.gz 100%[=====] 518K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 140%[=====] 726K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 180%[=====] 938K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 200%[=====] 1048K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 220%[=====] 1158K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 240%[=====] 1268K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 260%[=====] 1378K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 280%[=====] 1488K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 300%[=====] 1598K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 320%[=====] 1708K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 340%[=====] 1818K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 360%[=====] 1928K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 380%[=====] 2038K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 400%[=====] 2148K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 420%[=====] 2258K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 440%[=====] 2368K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 460%[=====] 2478K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 480%[=====] 2588K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 500%[=====] 2698K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 520%[=====] 2808K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 540%[=====] 2918K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 560%[=====] 3028K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 580%[=====] 3138K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 600%[=====] 3248K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 620%[=====] 3358K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 640%[=====] 3468K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 660%[=====] 3578K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 680%[=====] 3688K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 700%[=====] 3798K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 720%[=====] 3908K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 740%[=====] 4018K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 760%[=====] 4128K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 780%[=====] 4238K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 800%[=====] 4348K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 820%[=====] 4458K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 840%[=====] 4568K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 860%[=====] 4678K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 880%[=====] 4788K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 900%[=====] 4898K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 920%[=====] 5008K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 940%[=====] 5118K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 960%[=====] 5228K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 980%[=====] 5338K 1.2MB/s eta 10s
Home/~/Desktop/mydevices-1.0.tar.gz 1000%[=====] 5448K 1.2MB/s eta 10s

```

Рисунок 1.5 – Результати запуску файлу rpi_p632o1dd0b.sh

При цьому у WebIOPi автоматично відобразитиметься хід процесу підключення (рис.1.6)

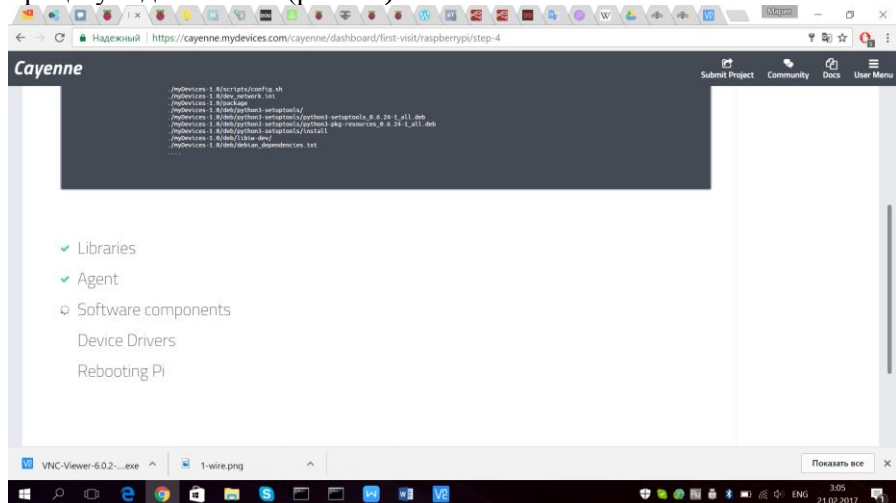


Рисунок 1.6 – Відображення ходу процесу у WebIOPi

8. Після автоматичного перезапуску системи, у браузері отримуюмо наступне відображення нашого пристрою.

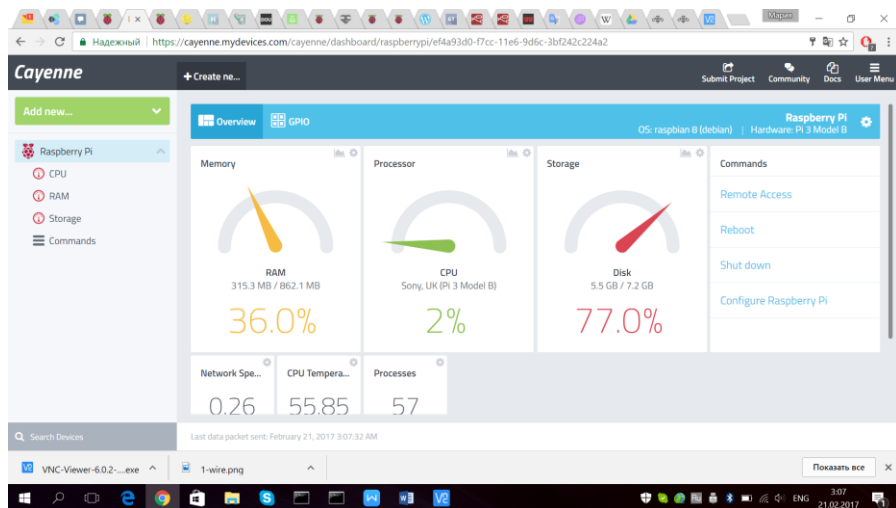


Рисунок 1.7 – Інтерфейс управління пристроєм

9. Переглядаємо можливості наданого інтерфейсу управління пристроєм (рис.1.8 - 1.9).

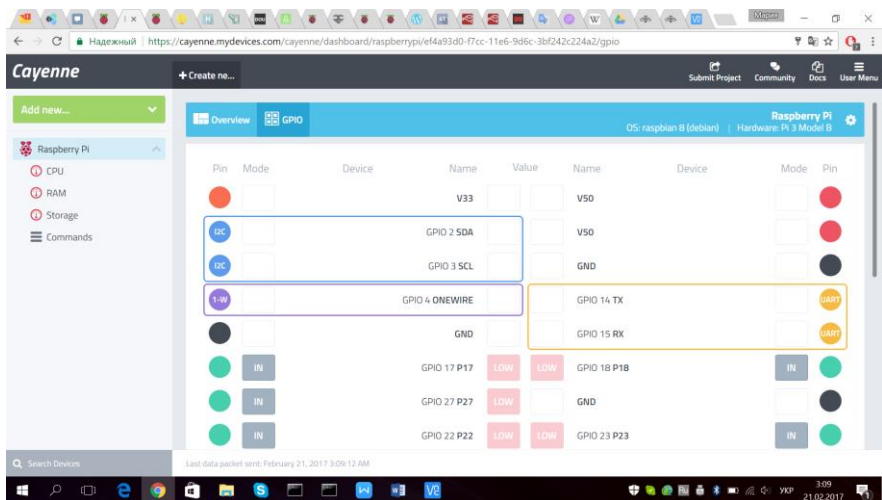


Рисунок 1.8 – Інтерфейс GPIO

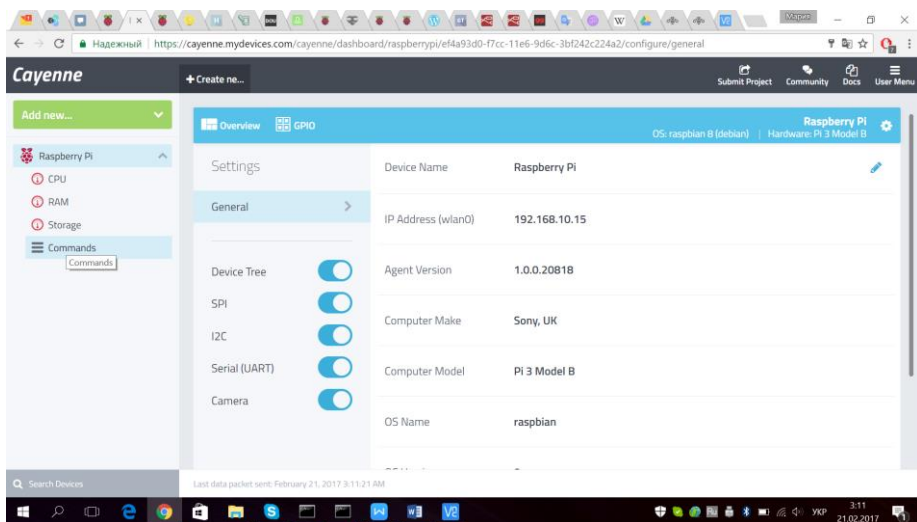


Рисунок 1.9 – Конфігурація пристрою (натискаємо Command – >Configure Raspberry Pi)

10. Натисніть на кнопку Details&Chart на відображенні CPU (рис.4.10).

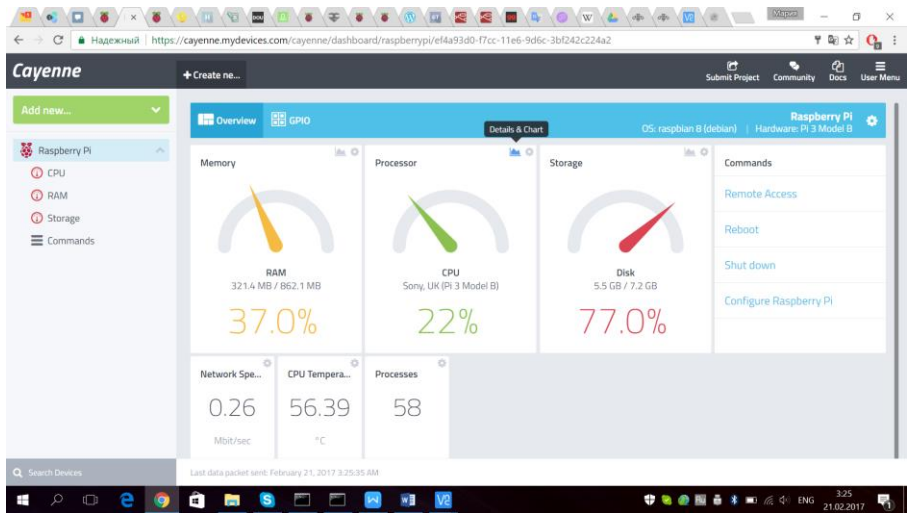


Рисунок 1.10 – Інтерфейс з підсвіченою кнопкою побудови діаграми

11. Перегляньте інформацію процесора (рис.1.11).

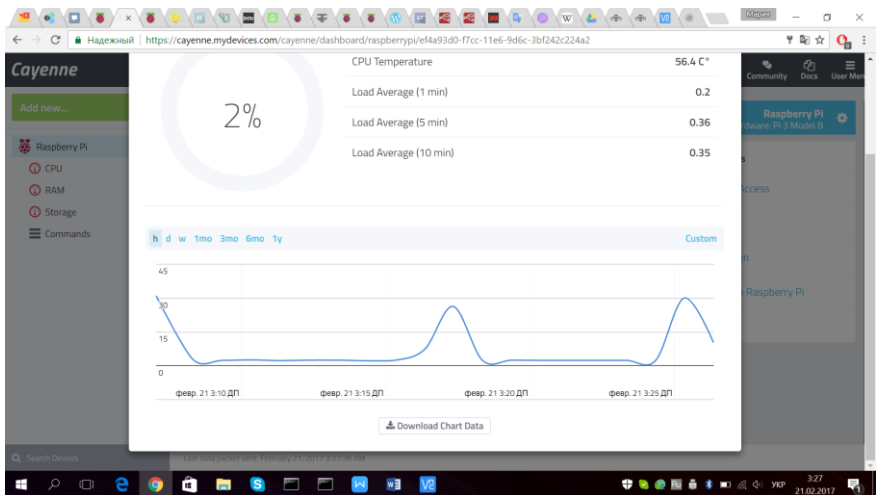


Рисунок 1.11 –Діаграма відображення зміни температури процесори

Завдання 2. Обробка даних з Raspberry Pi 3 за допомогою WebIOPi.

Хід роботи.

1. Додати датчик температури DS18B20 за допомогою натискання Add new → Sensors (рис.1.12).

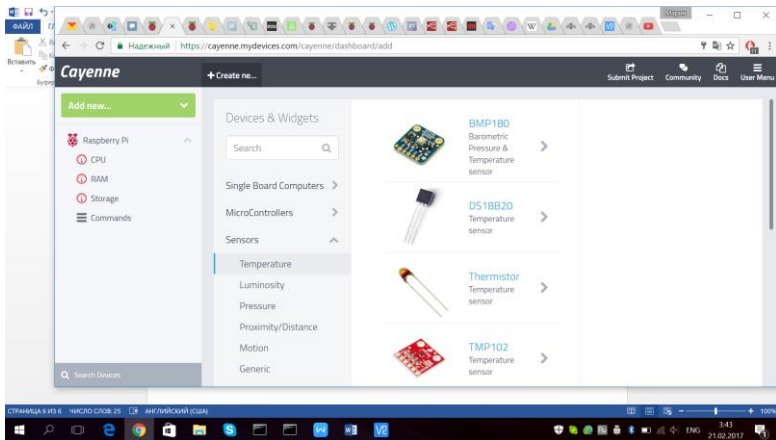


Рисунок 1.12 –Додавання датчику температури

2. Підключаємо датчик температури до Raspberry Pi 3 згідно схеми у л.р.1, та змінюємо властивості, як показано на рис. 1.13. Відображення датчика на панелі керування показано на рис.1.14.

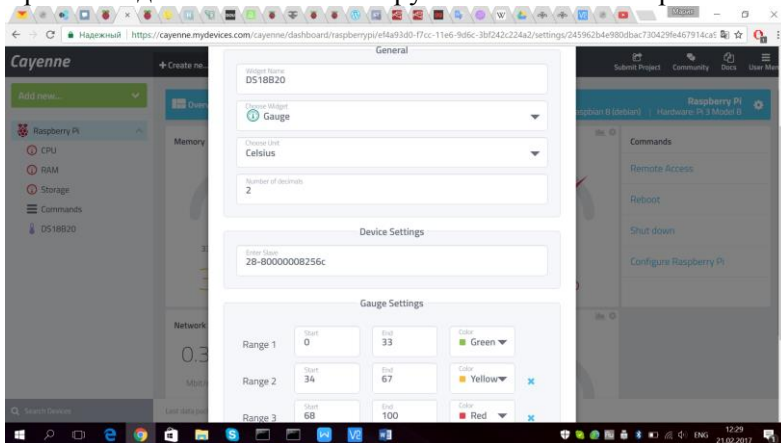


Рисунок 1.13 – Властивості датчика температури

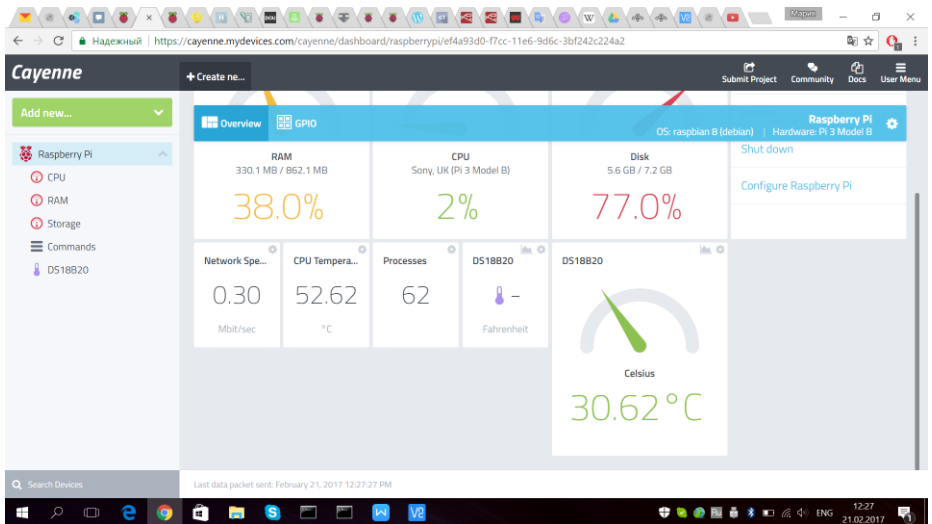


Рисунок 1.14 – Відображення датчика температури на панелі керування

3. Змінимо температуру затисканням датчика у руці та переглянемо змінення на графіку (рис.1.15).

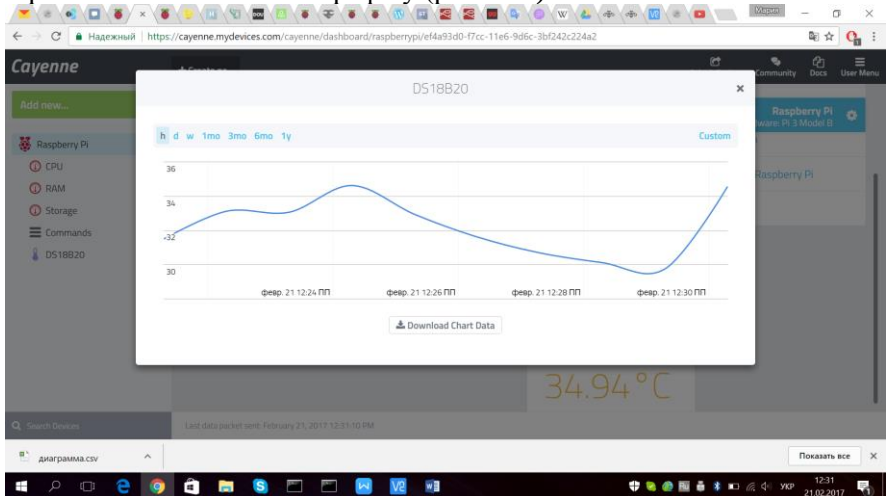


Рисунок 1.15 – Графік змінення температури, яка фіксується датчиком

4. Створення дії. Для цього необхідно клацнути на Add new та обрати Event. Далі налаштувати дію, наприклад, час вимкнення Raspberry Pi. Створіть свою дію аналогічно показаній на рис. 1.16-1.17.

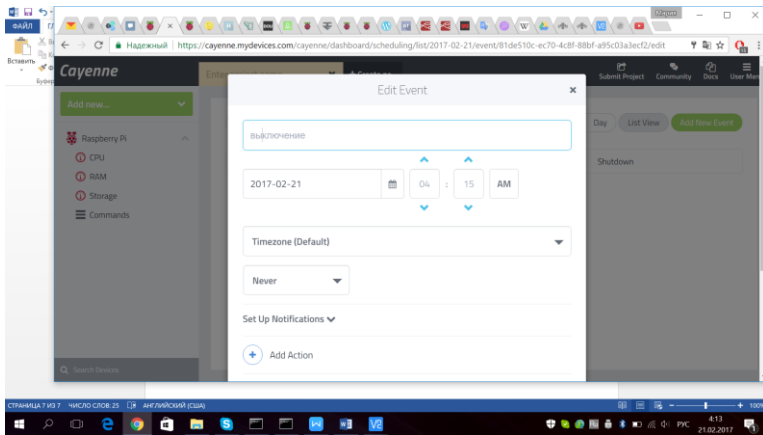


Рисунок 1.16 – Створення дії

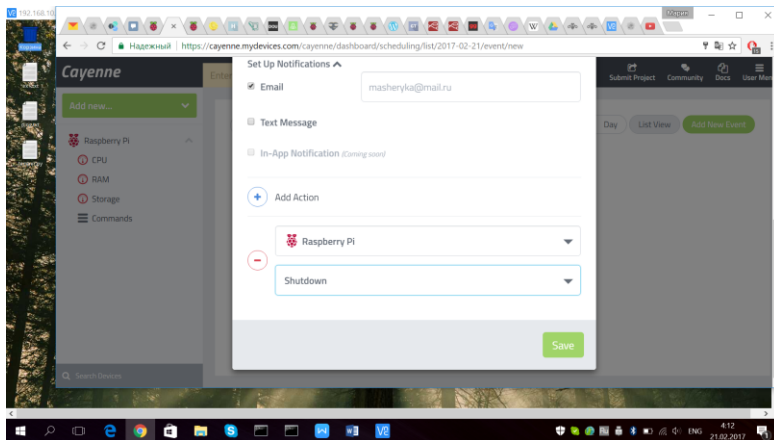


Рисунок 1.17 – Властивості створеної дії

Створену дію у меню показано на рис.1.18.

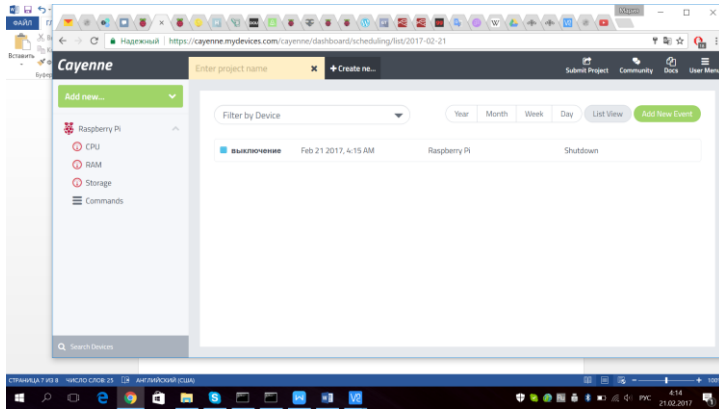


Рисунок 1.18 – Відображення створеної дії у WebIOPi

Результат виконання дії показано на рис.4.19.

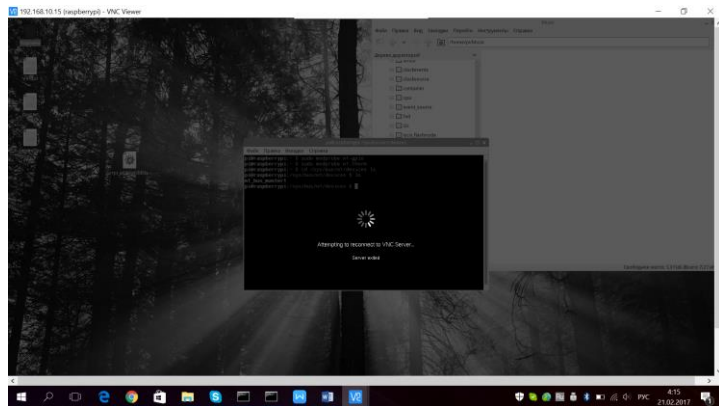


Рисунок 1.19 – Вимкнення Raspberry Pi

1.3 Контрольні питання

1. Що таке WebIOPi ?
2. З яким програмним та апаратним забезпеченням є можливість працювати у WebIOPi?
3. Які основні структурні елементи WebIOPi?
4. Які аналоги WebIOPi?
5. У чому основні переваги та недоліки WebIOPi перед аналогами?

2 ЛАБОРАТОРНА РОБОТА №5 КЕРУВАННЯ СВІТЛОДІОДОМ

Мета роботи: навчитися керувати виконавчими пристроями за допомогою плати Raspberry Pi на прикладі управління світлодіодом, підключеним до мінікомп'ютера.

2.1 Загальні відомості

По одному з описаних визначень Інтернету речей, Internet of Things - це датчики і приводи (виконавчі пристрої), вбудовані в фізичні об'єкти і пов'язані через дротові або бездротові мережі з використанням протоколу Internet Protocol (IP), який пов'язує Інтернет. В якості найпростішого виконавчого пристрою розглянемо в даній лабораторній роботі світлодіод, яким навчимося керувати з консолі, за допомогою написаної програми і за допомогою кнопки.

Для цього нам знадобиться наступне обладнання:

- плата Raspberry Pi;
- кабель живлення;
- USB-клавіатура;
- USB-миша;
- монітор з HDMI / RCA / DVI інтерфейсом;
- кабель, один кінець якого HDMI, а інший відповідає нашому монітору;
- SD-карта з уже встановленою ОС Raspbian;
- світлодіод;
- кнопка;
- резистор на 220 Ом
- 3 дроти «мама-тато»
- 2 дроти «тато-тато».

2.2 Складання моделі

Для роботи нам буде потрібно зібрати схему, подібну схемі на рис.2.1.

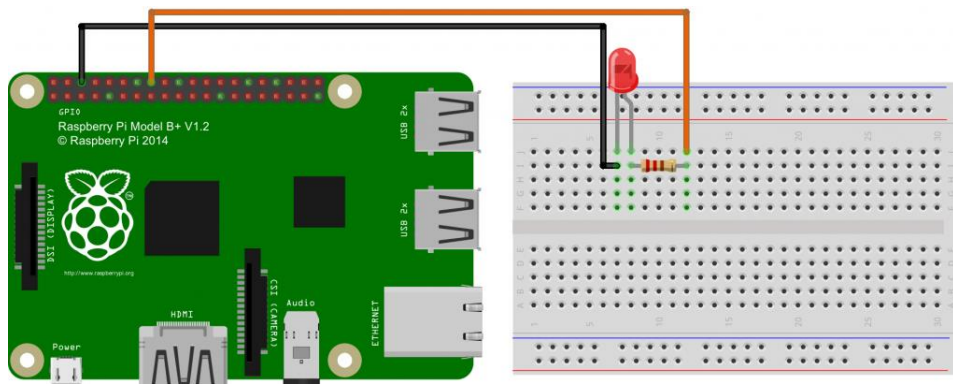


Рисунок 2.1 – Схема підключення світлодіода до Raspberry Pi

Розпіновка портів GPIO на Raspberry Pi представлена на рис.2.2. Підключати можна до будь-якого роз'єму, що підписаний як GPIO [номер_роз'єма] (позначені зеленим на рис.2.2).

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 2.2 – Распінровка Raspberry Pi.

2.3 Завдання до лабораторної роботи

Завдання 1. Управління світлодіодом на Raspberry Pi з консолі.

Для цього необхідно зібрати схему, аналогічну схемі на рис.5.1. Пін вибирається індивідуально. У прикладі розглянуто пін з номером 11.

Програмування Raspberry Pi GPIO будемо здійснювати на мові Python, тому що Python - сучасна об'єктно-орієнтована мова. Вона найбільш часто використовується для програмування GPIO на Raspberry Pi і входить до складу операційної системи Raspbian.

Заходимо в LXTerminal і набираємо:

```
sudo python
```

Після цього замість імені користувача на початку рядка має відобразитися >>>.

Вводимо наступні рядки:

```
import RPi.GPIO as GPIO # імпорт бібліотеки
GPIO.setmode(GPIO.BOARD) # "включення" GPIO
GPIO.setup(11, GPIO.OUT) # оголошення 11-го піна як вихід
```

Потім для включення світлодіода можна використовувати команду:

```
GPIO.output(11, 1)
```

А для виключення:

```
GPIO.output(11, 0)
```

Після роботи з GPIO бажано виконати команду:

```
GPIO.cleanup()
```

Завдання 2. Управління світлодіодом за допомогою написаної програми.

Програма для мигання світлодіодом на Raspberry Pi.

Для автономної роботи світлодіода нам буде потрібно написати і запустити програму. Для цього відкриємо встановлену програму IDLE 3 і в меню File натиснемо New. У вікні, ми можемо писати програму.

Тепер приступимо до складання програми, яка буде управляти вмиканням і вимиканням світлодіода за потрібним нам алгоритмом. Для прикладу, увімкнемо світлодіод на 5 секунд, а потім вимкнемо.

Перше, що нам потрібно зробити - це вибрати тип нумерації виводів загального призначення (номер, безпосередньо вказаний на піні на рис. 2.2).

У програмі тип нумерації задається за допомогою функції **setmode**:

```
GPIO.setmode(GPIO.BOARD)
```

Тепер потрібно задати режим роботи виводу №11. Всього є два режими: вивід (OUT) і ввід (IN). Ми хочемо керувати світлодіодом, значить нам потрібен режим «вивід». А ось якби ми хотіли прочитати якийсь сигнал, наприклад, з датчика, то слід було б вибрати режим «ввід».

```
GPIO.setup(11, GPIO.OUT)
```

Щоб подати на вивід №11 істину, використовуємо функцію **output**:

```
GPIO.output(11, True)
```

Крім цих функцій, нам знадобиться пауза **sleep**. Ця функція ставить на паузу виконання програми. Як аргумент вказується кількість секунд паузи.

```
sleep(5)
```

Нарешті, щоб функції роботи з GPIO і sleep працювали, на самому початку програми підключимо дві бібліотеки:

```
from RPi import GPIO
from time import sleep
```

Отже, вся програма представлена в лістингу 2.1.

Лістинг 2.1 - Управління миганням світлодіода.

```
from RPi import GPIO
from time import sleep
GPIO.setmode(GPIO.BOARD)
GPIO.setup(11, GPIO.OUT)
GPIO.output(11, True)
sleep(5)
GPIO.output(11, False)
GPIO.cleanup()
```

В кінці програми рекомендується викликати функцію **cleanup**, яка скине всі налаштування GPIO, щоб вони не заважали запуску чергової програми.

Миготіння світлодіода у циклі.

Вдосконалюємо програму таким чином, щоб світлодіод мигав безперервно з частотою 1 Герц (тобто один раз в секунду). Для цього нам потрібно буде додати в програму оператор циклу **while** (лістинг 2.2).

Лістинг 2.2 - Управління миганням світлодіода в циклі

```
from RPi import GPIO
from time import sleep

GPIO.setmode(GPIO.BOARD)
GPIO.setup(11, GPIO.OUT)

while True:
    GPIO.output(11, True)
    sleep(0.5)
    GPIO.output(11, False)
    sleep(0.5)
```

```
GPIO.cleanup()
```

Цикл **while** буде працювати поки істинна умова, вказана в ньому. Щоб цикл працював без зупинки, як умову ми поставили **True**.

Складена нами програма має один неприємний недолік. Якщо ми її запустимо, вона увійде в безкінечний цикл і ми не зможемо її зупинити штатними засобами. А якщо ми її зупинимо нештатно, то не виконається функція **cleanup**, що небажано.

Додамо в програму ще одну конструкцію, яка дозволить нам в будь-який момент перервати роботу циклу за допомогою комбінації клавіш Ctrl + C. При цьому програма завершиться в нормальному режимі і виконає функцію **cleanup**.

Лістинг 2.3 - Вдосконалена система розподілу ресурсів миганням світлодіода в циклі.

```
from RPi import GPIO
from time import sleep
GPIO.setmode(GPIO.BCM)
GPIO.setup(11, GPIO.OUT)
try:
    while True:
        GPIO.output(11, True)
        sleep(0.5)
        GPIO.output(11, False)
        sleep(0.5)
except KeyboardInterrupt:
    print 'program stop'
GPIO.cleanup()
```

Збережемо програму в папці /home/pi.

Тепер ми можемо запустити програму з LXTerminal за допомогою команди

```
sudo python led.py
```

Завдання 3. Управління світлодіодом за допомогою кнопки.

Поуправляти світлодіодом за допомогою зовнішньої кнопки: коли кнопка затиснута - світлодіод горить, коли відтиснута - не горить.

Для цього підключимо кнопку до порту 5.

Схема підключення світлодіода і кнопки до Raspberry Pi представлена на рис. 2.3.

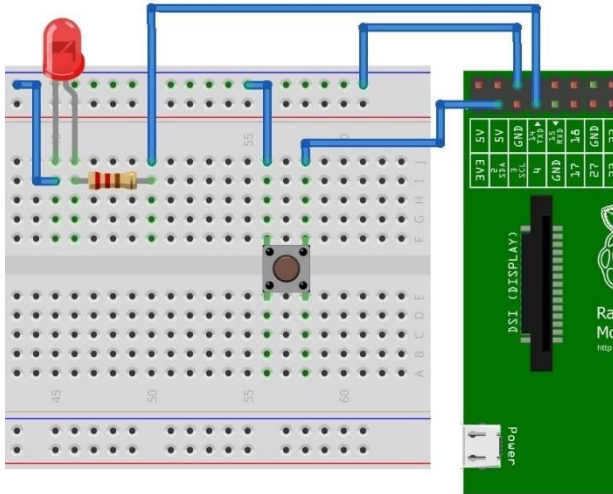


Рисунок 2.3 – Підключення до Raspberry Pi світлодіода і кнопки

Для управління нам буде потрібна наступна програма, представлена в лист.2.4.

Лістинг 2.4 - Управління світлодіодом за допомогою кнопки.

```
import RPi.GPIO as GPIO      #импорт библиотеки GPIO
GPIO.setmode(GPIO.BOARD)    # "включение GPIO"
GPIO.setup(7, GPIO.OUT)      #объявление порта 7 как выход
GPIO.setup(3, GPIO.IN)       #объявление порта 3 как вход
while True:                  #бесконечный цикл
    if GPIO.input(3) == False: #если кнопка зажата
        GPIO.output(7, 1)      #включаем светодиод
    else:                     #иначе
        GPIO.output(7, 0)      #выключаем
```

Завдання 4. Управління світлодіодом за допомогою фреймворка WebIOPi.

Виконайте завдання 3 за допомогою фреймворка WebIOPi. Особливості підключення детально описані в Л.Р. 4.

2.4 Контрольні питання

1. Що таке GPIO ?
2. Як запрограмувати елементарний світлодіод для мигання?
3. Дати пояснення коду програми:

```
from RPi import GPIO
from time import sleep
GPIO.setmode(GPIO.BOARD)
GPIO.setup(11, GPIO.OUT)
GPIO.output(11, True)
sleep(5)
GPIO.output(11, False)
GPIO.cleanup()
```

3 ЛАБОРАТОРНА РОБОТА № 6. НАЛАШТУВАННЯ СЕРЕДОВИЩА MININET

Мета роботи: набути практичних навичок встановлення, налаштування та використання емулятору Mininet.

3.1 Загальні відомості

Поточний стан розвитку глобальної мережі можна охарактеризувати наступним чином: існує нагальна потреба поліпшення засобів керування, моніторингу, конфігурування такої системи. Рішенням може слугувати використання програмно-конфігурованих мереж (SDN, Software Defined Networking). Ось деякі з ключових позицій [1]: виокремлення рівнів керування (Control Plane) та даних (Data Plane) [2], залучення уніфікованих комутаторів для перенаправлення потоків даних, застосування засобу централізованого координування таких комутаторів – контролеру (controller), слідування регламенту протоколу OpenFlow [3, 4]. Специфікація OpenFlow надає відкритий інтерфейс взаємодії компонентів SDN-мережі. Ключовими компонентами при цьому є контролер, комутатори (switches) та хости (hosts).

По причині відносної новизни SDN безпосередньо робота із відповідною мережею заданої топології не завжди є можливою. Рішенням може слугувати використання різноманітних засобів емуляції – комплексів програмно-апаратних засобів для відтворення SDN-мережі у межах віртуального середовища. Програмне забезпечення SDN-мереж базується на Linux-платформі. Ось приклади таких емуляторів: Mininet [5, 6], EstiNet [7], OpenNet [8], ns-3 [9]. Кожне із рішень має певні переваги та недоліки, проте важко не відзначити, що Mininet часто фігурує у якості зразку, з яким порівнюють інші емулятори. Через це будемо використовувати його у нашому циклі лабораторних робіт.

Середовище Mininet призначене слугувати засобом емуляції SDN-мережі, зокрема створювати віртуальні хости, комутатори, контролери та зв'язки між ними. Названі компоненти та зв'язки між ними разом формують топологію мережі.

Архітектуру SDN подано на рис. 3.1 [10].

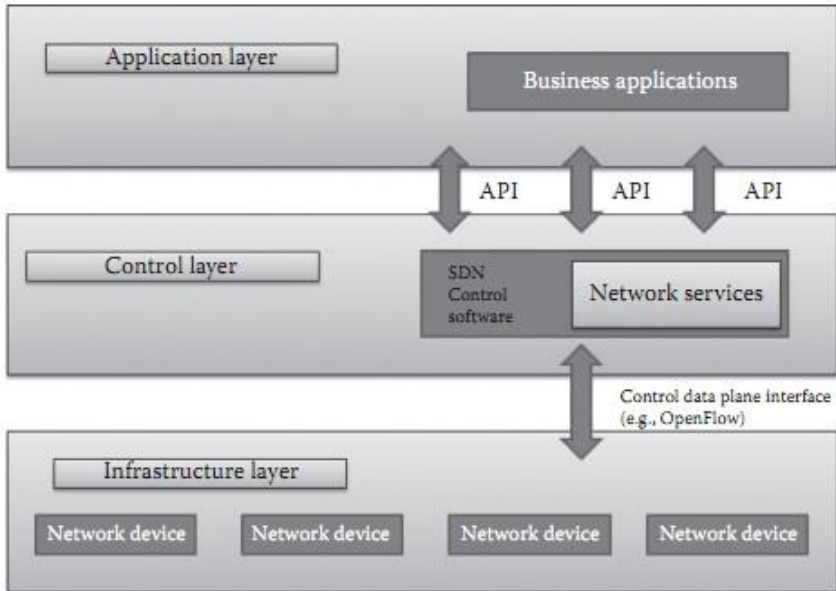


Рисунок 3.1 – Архітектура мережі SDN

Середовище Mininet надає засоби для здійснення розробки SDN-систем, проведення відповідних досліджень, тестувань, налаштувань програмної складової тощо.

Mininet, зокрема, забезпечує наступні можливості:

- дозволяє слугувати стендом для розробки SDN-додатків;
- надає підтримку одночасної роботи декількох розробників над спільною топологією;
- включає засобів комплексного тестування топології;
- надає спеціалізований API (Application Programming Interface), орієнтований на використання мови програмування Python.

У порівнянні із типовими підходами до віртуалізації, використання Mininet має наступні переваги:

- зручність інсталяції;
- більш швидкий час завантаження середовища;
- зручність реконфігурування системи.

У якості певного недоліку можна відзначити певні труднощі при роботі із графічною оболонкою емулятора із Windows-середовища, а

також певну лімітованість конфігурації мережі можливостями апаратного забезпечення базової платформи [6].

Виконання роботи полягає в наступному:

- інсталяція на Windows-платформі Linux-середовища Mininet у межах засобу віртуалізації VirtualBox;
- конфігурування мережевих інтерфейсів розгорнутого віртуального середовища;
- ознайомлення із базовими консольними командами Mininet, зокрема створення мереж із заданими топологіями.

Представлення виконаних завдань на перевірку викладачеві має здійснюватися одним з наступних шляхів:

- у порядку виконання завдань;
- за результатами виконання усіх завдань.

За результатами виконання завдання має бути підготовлено та захищено звіт.

3.2 Завдання до лабораторної роботи

Зауваження: варіант обирається за номером комп'ютера. Для непарних номерів виконується перший варіант, для парних – другий.

Завдання 1. Виконати інсталяцію середовища Mininet.

Для інсталяції емулятору Mininet 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, призначений для інсталяції на 32-бітну Windows-платформу;
- у якості Mininet-середовища – безпосередньо емулятору – версію 2.2.1 – вміст архіву mininet-2.2.1-150420-ubuntu-14.04-server-i386.zip, тобто цей файл слід попередньо розархівувати. Із назви файлу бачимо, що віртуалізації підлягає 32-бітне Linux-середовище ubuntu-14.04-server.

Після того, як засіб віртуалізації VirtualBox-4.3.10 буде встановлено, вміст архіву – файл mininet-vm-i386.vmdk – слід використати у якості жорсткого диску створюваної віртуальної системи.

Демонстрацією успішно виконаних дій має бути система вигляду, подібного до такого, поданого на рис. 3.2.

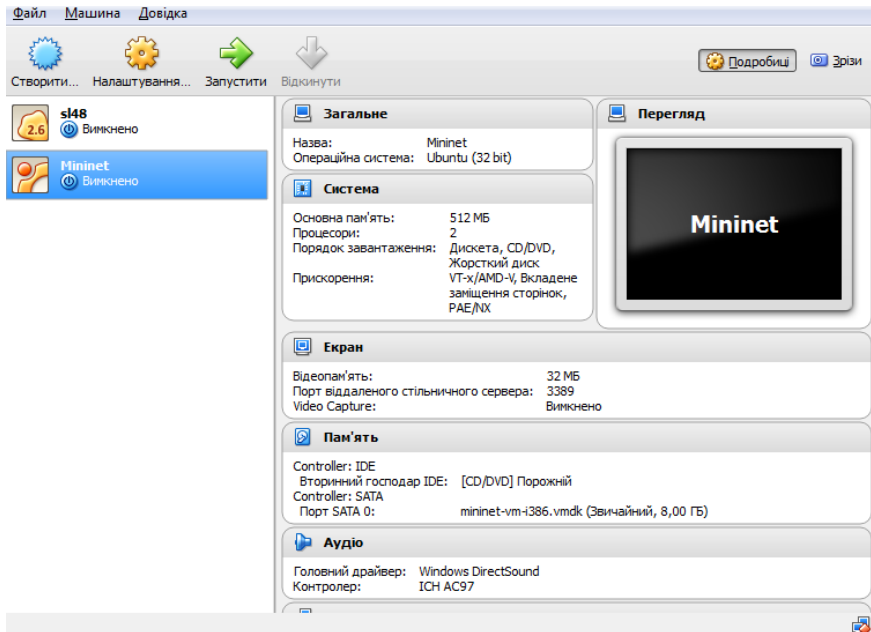


Рисунок 3.2 – Вікно інформації про створюване віртуальне середовище

Варто відзначити, що під створювану систему слід виділяти не менше 512 МБ оперативної пам'яті.

Завдання 2. Здійснити налаштування мережевих інтерфейсів віртуального середовища.

Для налаштування мережевих інтерфейсів віртуального середовища слід обрати опцію "Налаштування", та у пункті "Мережа" сконфігурувати "Адаптер 1" та "Адаптер 2" (рис. 3.3, 3.4).

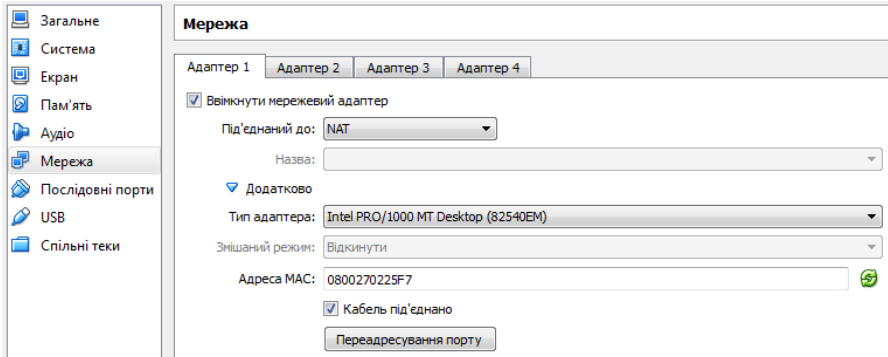


Рисунок 3.3 – Налаштування мережі ("Адаптер 1")

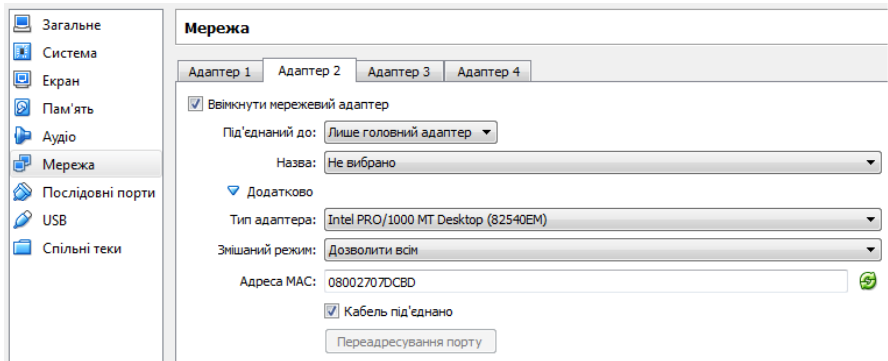


Рисунок 3.4 – Налаштування мережі ("Адаптер 2")

Завдання 3. Запустити середовище Mininet.

Натиснути на елемент керування "Запустити".

Через деякий час після запуску буде запропоновано ввести логін для входу в систему:

– у рядку mininet-vm login: слід ввести логін – mininet;

– у рядку password: ввести пароль – mininet.

У результаті успішних дій активується консоль системи – рядок

```
mininet@mininet-vm:~$
```

Завдання 4. Набути навичок роботи із базовими командами середовища Mininet.

Переглянути конфігурації мережевих інтерфейсів:

```
> sudo ifconfig
```

У результаті виконання названої команди перевірити IP-адресу інтерфейсу eth0. Нехай, наприклад, маємо адресу 10.0.2.15.

Створити топологію мінімальної конфігурації – один контролер (c0), один комутатор (s1) та два хости (h1, h2):

```
> sudo mn
```

Після виведеної інформації про компоненти та зв'язки створеної мережі з'явиться безпосередньо Mininet-консоль.

Переглянути інформацію про консольні команди Mininet:

```
> help
```

Переглянути інформацію про вузли мережі (має бути чотири вузли – комутатор (s1), два хости (h1, h2) та контролер (c0)):

```
> nodes
```

Перевірити зв'язки між вузлами, із зазначенням залучених портів:

```
> links
```

Одержати дані про IP-адресу та відповідну маску підмережі для зазначеного інтерфейсу хосту. Наприклад, у випадку інтерфейсу eth0 вузла h1 матимемо наступну команду:

```
> h1 ip addr show | grep eth0
```

Протестувати пропускну спроможність каналу зв'язку між зазначеними хостами за допомогою команди iperf. Наприклад, для вузлів h1 та h2 матимемо наступне:

```
> iperf h1 h2
```

Виконання команди займе певний час. При цьому кожний новий результат буде дещо відрізнятись від попереднього.

Переглянути інформацію про інтерфейси вузлів:

```
> net
```

Переглянути інформацію про конфігурації вузлів:

```
> dump
```

Вивести список мережевих інтерфейсів зазначеного вузла, виконавши команду `ifconfig`. Наприклад, у випадку вузла `h1` матимемо наступне:

```
> h1 ifconfig -a
```

Переглянути інформацію про процеси, що виконуються на вузлах. Наприклад, для вузла `s1` слід виконати наступну команду:

```
> s1 ps -a
```

Перевірити зв'язки між хостами.

Між заданими хостами:

```
> h1 ping -c 1 h2
```

Між усіма парами хостів:

```
> pingall
```

Запустити веб-сервер та відповідний клієнт на хостах.

Веб-сервер запустимо на вузлі `h1`, а його клієнт – на вузлі `h2`:

```
> h1 python -m SimpleHTTPServer 80 &
> h2 wget -O - h1
```

У результаті виконання останньої команди у консолі буде виведено HTML-код одержаної клієнтом веб-сторінки.

Завершити роботу веб-серверу:

```
> h1 kill %python
```

Продемонструвати викладачеві набуті навички.

Вийти із командного середовища Mininet до консолі віртуального Linux-середовища:

```
> exit
```

Очистити супутні конфігураційні дані, пов'язані із створеною топологією, виконавши команду mn із прапорцем -c:

```
> sudo mn -c
```

Завдання 5. Створення програмно-конфігурованої мережі із заданими характеристиками комунікаційних каналів.

Виконати завдання згідно варіанту, де створити програмно-конфігуровану мережу із заданими значеннями пропускної спроможності (параметр bw – bandwidth) та затримки (параметр delay) на комунікаційних каналах.

Варіант 1. Пропускна спроможність – 10 Мб/с, затримка – 20 мс:

```
> sudo mn --link tc,bw=10,delay=20ms
```

Варіант 2. Пропускна спроможність – 100 Мб/с, затримка – 40 мс:

```
> sudo mn --link tc,bw=100,delay=40ms
```

Для кожного із варіантів виконати наступне:

– заміряти пропускну спроможність каналу зв'язку між хостами п'ять разів, виконавши команду `iperf`. Середнє арифметичне значення замірів занести до звіту;

– визначити мінімальне значення параметру `rtt` (Round Trip Time), виконавши команду `ping`.

Зауваження: параметр `rtt` охоплює час, що витрачається на відправку пакета до вузла-одержувача та на одержання вузлом-відправником підтвердження про отримання пакета.

Для кожного із варіантів пояснити одержані результати, а саме – чому заміряні значення пропускну спроможності дещо менші заданого значення (параметр `bw`), а мінімальне значення показника `rtt` у середньому в чотири рази більше за встановлене значення затримки `delay` – пояснити, чому саме у чотири.

Завдання 6. Створити програмно-конфігуровану мережу із заданою топологією.

Поекспериментувати із зміною типу та розміру створюваної топології:

```
> sudo mn --test pingall --topo single,3
```

У результаті буде створено конфігурацію, що включатиме вже 3, а не 2 хости.

Повторити раніше виконані дії по відношенню до нової конфігурації.

Створити лінійну топологію із чотирма комутаторами та чотирма вузлами:

```
> sudo mn --test pingall --topo linear,4
```

Така топологія вже включатиме сім зв'язків.

Створити мережу деревовидної топології мінімальної конфігурації (один контролер, один комутатор, два хости) та протестувати її командою `pingall`:

```
> sudo mn --topo tree,depth=1,fanout=2 --test pingall
```

Параметр `--topo tree` задає деревовидну топологію. Атрибут `depth` задає кількість рівнів комутаторів (у нашому випадку – один – як вершина дерева комутаторів; на другому рівні вже буде два комутатори, на третьому – чотири, і т.д.). Атрибут `fanout` регламентує кількість розгалужень від (підключень до) кожного із комутаторів. У нашому випадку `fanout=2`, тобто маємо два підключення (хости) по відношенню до єдиного комутатора.

Наприклад, якщо б ми мали `depth=2`, то топологія би містила два рівні комутаторів. Перший рівень був би представлений єдиним комутатором, до якого би були підключені два комутатори з наступного рівня, а до кожного із них, у свою чергу, було би підключено по два хости. Тобто усього мали б три комутатори та чотири хости.

Параметр `--test pingall` означає, що після того як мережу заданої конфігурації буде створено, необхідно, щоб кожний із хостів виконав команду `ping` по відношенню до кожного із інших хостів.

Процедура створення та тестування такої мережі доволі ресурсоемна, і триватиме близько 5 с, про що сповіщатиметься у консолі.

Поекспериментувати із створенням та тестуванням мереж із різними значеннями параметрів `depth` та `fanout`.

3.3 Зміст звіту

Звіт має включати наступні складові:

1. Титульний лист.
2. Мета роботи.
3. Пояснення до результатів виконання завдання 5.
4. Відповіді на контрольні питання.

3.4 Контрольні питання

1. Технологія програмно-конфігурованих мереж SDN. Призначення, переваги та недоліки.
2. Дати визначення поняттю емуляції. Приклади засобів емуляції SDN-мереж.
3. Емулятор Mininet. Призначення, особливості.
4. Інсталяція та налаштування Mininet. Етапи виконання.
5. Кроки створення SDN-мережі з мінімальною топологією (за замовчанням) – один комутатор та два хости.
6. Команди взаємодії з хостами та комутаторами.
7. Команди перевірки зв'язків між хостами.
8. Команди запуску веб-сервера та відповідного клієнта.
9. Команди завдання затримок на комунікаційних зв'язках.
10. Параметр `rtt`. Визначення. Прокоментувати залежність його значення від значення параметру `delay`.
11. Засоби зміни конфігурації мережі.
12. Призначення параметрів `depth` та `fanout` при створенні мережі деревовидної топології. Охарактеризувати їх вплив на кількість вузлів мережі.

4 ЛАБОРАТОРНА РОБОТА № 7. РОБОТА З ГРАФІЧНИМ ІНТЕРФЕЙСОМ MINIEDIT

Мета роботи: набути практичних навичок налаштування та використання графічного інтерфейсу MiniEdit середовища Mininet.

4.1 Загальні відомості

Середовище Mininet, з метою поліпшення зручності та інтуїтивності його використання, має також і графічний інтерфейс – MiniEdit.

Для виконання роботи знадобиться наступне програмне забезпечення (ПЗ):

- встановлений засіб віртуалізації VirtualBox – збірка VirtualBox-4.3.10-93012-Win.exe;

- розгорнута віртуальна машина Mininet – вміст архіву mininet-2.2.1-150420-ubuntu-14.04-server-i386.zip використано у якості жорсткого диску віртуальної машини;

- утиліта PuTTY – виконуваний файл putty.exe – вільно розповсюджуване клієнтське ПЗ із підтримкою різноманітних протоколів віддаленого доступу, зокрема Telnet та SSH (Secure Shell). SSH – мережевий протокол прикладного рівня, що дозволяє здійснювати віддалене керування операційною системою. На відміну від Telnet, усі передавані дані шифруються;

- сервер Xming Server – збірка xming-x-server-6.9.0.38.exe – реалізація віконної системи X Window System для забезпечення стандартних інструментів і протоколів побудови графічного інтерфейсу користувача. У роботі використовуватимемо названий сервер у зв'язку із тим, що віртуальна машина, на якій встановлено Mininet, не містить графічної оболонки. Сервер надасть можливість візуалізувати у Windows-середовищі графічний інтерфейс MiniEdit для Mininet, що функціонує на віртуальній машині у Linux-середовищі [11].

Перші два зазначені пункти вже було пропрацьовано у попередній лабораторній роботі.

Безпосередньо виконання роботи полягатиме у здійсненні наступних кроків:

- інсталяція сервера Xming Server – для роботи у Windows-середовищі із графічним інтерфейсом MiniEdit для Mininet на основі віртуального Linux-середовища;
- запуск та конфігурування віртуального середовища Mininet;
- налаштування SSH-клієнту PuTTY – для захищеного підключення до віртуального Linux-середовища, на якому встановлено емулятор Mininet;
- підключення до віртуального Linux-середовища через інтерфейс PuTTY.

4.2 Завдання до лабораторної роботи

Завдання 1. Встановити сервер Xming.

Виконати інсталяцію сервера на диск E. При цьому слід обрати опцію "Full installation" та встановити прапорець навпроти опції "Normal PuTTY Link SSH client".

У результаті успішної інсталяції сервер має бути запущено автоматично (рис. 4.1).

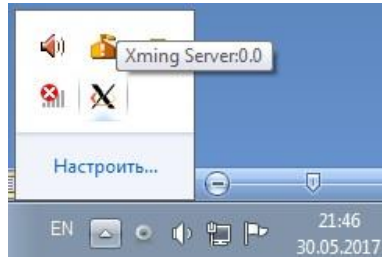


Рисунок 4.1 – Запущений сервер Xming

Для ручного запуску серверу необхідно використовувати утиліту XLaunch.exe, що знаходиться у директорії Xming інсталяції.

Завдання 2. Налаштувати віртуальне середовище Mininet.

Запустити на виконання віртуальне середовище Mininet через засіб віртуалізації VirtualBox. При цьому ввести логін і пароль (mininet, mininet) – як при виконанні попередньої роботи.

Потім слід виконати наступну послідовність кроків:

– (1-й крок) – спробувати одразу запустити графічне середовище MiniEdit у віртуальному середовищі Mininet, виконавши наступну команду:

```
> sudo python ./mininet/examples/miniedit.py
```

У результаті буде виведено повідомлення про помилку на кшталт наступного: "no display name and no \$DISPLAY environment variable". Аби позбутися його, необхідно виконати наступні кроки;

– (2-й крок) – переглянути конфігурацію мережових інтерфейсів:

```
> ifconfig -a
```

Для роботи використовуватимемо інтерфейс eth1. На даний момент зазначений інтерфейс ще не має IP-адреси у межах під мережі. Необхідно його вручну сконфігурувати у якості клієнту DHCP наступною командою:

```
> sudo dhclient eth1
```

Знову переглянути конфігурацію мережових інтерфейсів.

У результаті виконання команди зафіксувати IP-адресу для мережового інтерфейсу eth1. Має бути адреса, подібна до наступної: 192.168.56.101. Вона буде використовуватися для зовнішнього підключення до віртуального середовища Mininet із Windows-середовища з використанням утиліти PuTTY.

Завдання 3. Налаштувати SSH-підключення через PuTTY.

Для того, щоб PuTTY-клієнт мав можливість одержувати доступ до локально запущеного сервера Xming, необхідно виконати наступні кроки:

– відзначити прапорцем опцію "Enable X11 forwarding" у категорії "Connection -> SSH -> X11" налаштувань PuTTY (рис. 7.2);

– підключитися до віртуального Mininet-середовища у категорії Session, ввівши попередньо зафіксовану IP-адресу 192.168.56.101 та відкривши сесію, натиснувши на елемент управління Open.

Зауваження: порт підключення залишити за замовчанням (22).
У якості типу підключення прапорцем має бути зазначена опція SSH.

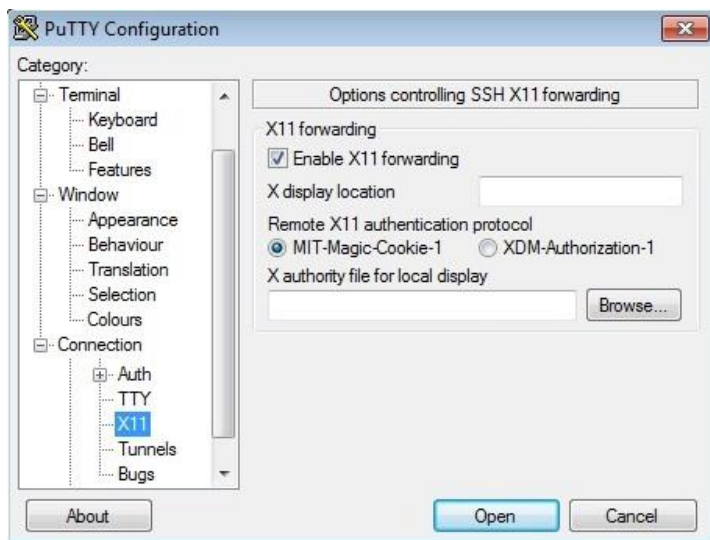


Рисунок 4.2 – Конфігурування SSH-клієнта

Завдання 4. Підключитися до консолі Mininet через клієнтське програмне забезпечення PuTTY.

Виконати наступні кроки:

- у PuTTY-консолі, що відкрилася, ввести логін та пароль для входу у Mininet-середовище (mininet, mininet);
- встановити SSH-з'єднання, зазначивши зафіксовану IP-адресу інтерфейсу eth1:

```
> ssh -Y mininet@192.168.56.101
```

- на запит введення паролю ввести пароль mininet;
- знову виконати 1-й крок завдання 2, ввівши через консоль PuTTY відповідну команду.

У результаті правильного виконання зазначених кроків у вигляді окремого вікна має з'явитися середовище графічної оболонки

MiniEdit (рис. 4.3). Це можливо за рахунок роботи серверу Xming Server.

Ключова перевага від використання серверу Xming Server – для роботи у Windows-середовищі не потрібно залучати додаткові бібліотеки, на відміну від серверу Cygwin [12].

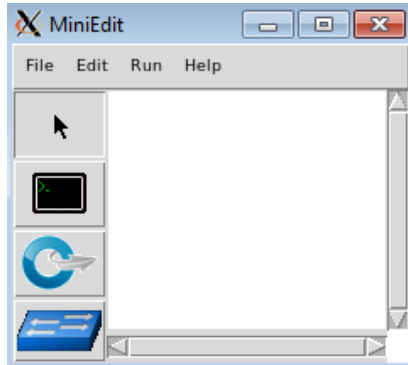


Рисунок 4.3 – Робоча область MiniEdit

Завдання 5. Створити SDN-мережу мінімальної топології засобами графічного інтерфейсу MiniEdit.

Результуюча мережа повинна мати вигляд, поданий на рис. 4.4.

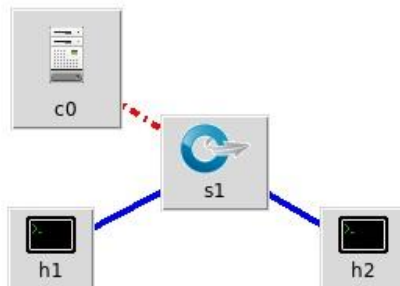


Рисунок 4.4 – Мережа, створена засобами MiniEdit

Кожний із вузлів такої мережі можна сконфігурувати потрібним чином (наприклад, задати потрібну IP-адресу, змінити назву тощо), натиснувши на відповідному елементі правою клавішею миші та

обравши опцію Properties. Наприклад, у випадку контролеру c0 матимемо налаштування, наведені на рис. 7.5.

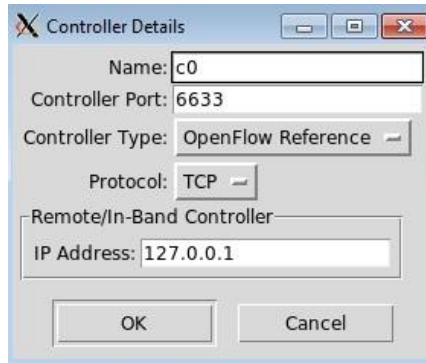


Рисунок 4.5 – Налаштування контролеру

На рис. 4.5 порт 6633 встановлюється за замовчанням.

У випадку, якщо контролерів декілька, їх порти мають відрізнятися. Наприклад, якщо маємо два контролери, номери портів можуть бути 6633 та 6634.

Для перевірки та встановлення налаштувань мережі у середовищі MiniEdit використовується елемент панелі інструментів Edit -> Preferences (рис. 4.6) [13].

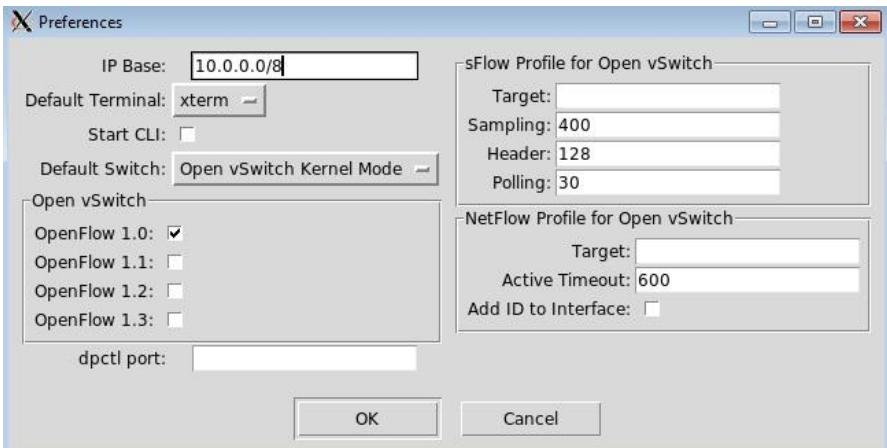


Рисунок 4.6 – Налаштування мережі

На рис. 7.6 видно, зокрема, що у якості протоколу OpenFlow за замовчанням використовується версія 1.0. Це є типовою картиною для поточного рівня впровадження мереж SDN. Ці налаштування справедливі лише для створеної топології (рис. 7.4).

Завдання 6. Зберегти створену топологію у файлі із назвою topo1.mn за адресою за замовчанням /home/mininet (файл топології обов'язково повинен мати розширення *.mn). Потім створену топологію можна буде за потреби завантажувати у середовище MiniEdit.

Для збереження топології виконати команду File -> Save, для завантаження – File -> Open.

Завдання 7. Перевірка створеної мережі.

На панелі керування Edit -> Preferences встановити прапорець навпроти опції Start CLI. Це дозволить працювати з командним рядком при перевірці мережі шляхом імітаційного моделювання.

Привласнити вузлам мережі IP-адреси.

Маска підмережі в рубриці Edit -> Preferences встановлена за замовчанням 10.0.0.0/8.

Перевірити зв'язок між хостами, запустивши процес імітаційного моделювання (Run -> Run) та, відкривши термінал вузла h1 (натиск правої клавіші миші на вузлі -> Terminal), виконати команду ping, вказавши IP-адресу цільового вузла, наприклад, наступним чином:

```
> ping 10.0.0.4
```

Зауваження: перед закриттям терміналу ввести команду exit.

Перевірити схему підключення комутатора. Для цього потрібно переконатися, що імітаційне моделювання виконується, та натиснути Run -> Show OVS Summary.

Завдання 8. Провести експерименти зі створеною мережею.

При запущеному процесі симуляції (імітаційного моделювання), відкрити термінал вузла h1 і виконати наступну команду:

```
> wireshark &
```

Це запустить на зазначеному хості h1 утиліту wireshark – програму-аналізатор трафіку.

У головному вікні програми – у полі Capture – у лівій нижній його частині виділити інтерфейс h1-eth0 та натиснути на елемент керування Start. Це дозволить налаштувати програму моніторингу на інтерфейс eth0 вузла h1.

У терміналі хоста h1 знову виконати команду ping:

```
> ping 10.0.0.4
```

У результаті виконання команди у робочій області програми відобразатиметься інформація про передані з інтерфейсу h1-eth0 пакети.

Створити мережу деревовидної топології. Конфігураційні дані задати згідно варіанту.

Зауваження:

– варіант обирається за номером по списку: перший варіант – для непарних номерів, другий – для парних.

Варіант 1: Мережа має включати 2 контролери, 7 комутаторів та 8 хостів. Сконфігурувати створену мережу згідно нижченаведених рекомендацій.

Варіант 2: Мережа має включати 3 контролери, 8 комутаторів та 10 хостів. Сконфігурувати створену мережу згідно нижченаведених рекомендацій.

Рекомендації до конфігурування та збереження мережі:

– для контролерів – задати унікальні номери портів;
– у панелі інструментів Edit -> Preferences встановити прапорець "Start CLI", що дасть змогу використовувати командний рядок;

– зберегти створену мережу у файлі з розширенням `"*.mn"`, виконавши `File -> Save`;

– зберегти відповідний Python-скрипт – у файлі з розширенням `"*.py"`. Для цього слід обрати опцію `File -> Export Level 2 Script`.

Зауваження: перш, ніж зупинити процес імітаційного моделювання, потрібно ввести команду `exit` у консолі, а потім вже натиснути елемент керування `Stop` на панелі керування.

Виконати дії, зазначені у завданнях 7 і 8, по відношенню до мережі, створеної згідно варіанту.

4.3 Зміст звіту

Звіт має включати наступні складові:

1. Титульний лист.
2. Мета роботи.
3. Відповіді на контрольні питання.

4.4 Контрольні питання

1. Послідовність кроків для організації роботи із графічною оболонкою MiniEdit з Windows-середовища.
2. Призначення утиліти PuTTY.
3. Призначення серверу Xming Server.
4. Переваги використання серверу Xming Server.
5. Переваги та недоліки використання для створення SDN-мережі заданої топології засобів командного рядку та графічної оболонки MiniEdit.
6. Призначення утиліти Wireshark.
7. Засоби тестування мережі у середовищі MiniEdit.
8. Призначення команди `Show OVS Summary`.
9. З урахуванням деревовидної топології SDN-мережі, пояснити залежності між кількостями контролерів, комутаторів і хостів.
10. Пояснити вплив скороченого формату запису маски підмережі на потенційну кількість хостів.

ЛІТЕРАТУРА

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>. – Title from the screen.
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>. – Title from the screen.
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. 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
8. Chan, M-C. OpenNet: A simulator for software-defined wireless local area network [Text] / M-C. Chan, C. Chen, J-X. Huang [et al.] // Proc. of the 2014 *IEEE Wireless Communications and Networking Conference* (Istanbul, Turkey, April 6–9, 2014). doi: 10.1109/WCNC.2014.6953088
9. 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