

Міністерство освіти і науки України
Запорізький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних робіт з дисципліни
"Основи Інтернету речей"
Частина 1
для студентів спеціальності
123 "Комп'ютерна інженерія"
всіх форм навчання

Методичні вказівки до виконання лабораторних робіт з дисципліни "Основи Інтернету речей" Частина 1 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. – 38 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 № 1. ЗНАЙОМСТВО З RASPBERRY PI 3	4
1.1 Загальні відомості	4
1.2 Завдання до лабораторної роботи	6
1.3 Контрольні питання	14
2 ЛАБОРАТОРНА РОБОТА № 2 ЗАСТОСУВАННЯ КОНФІГУРАТОРА ІНТЕРНЕТУ РЕЧЕЙ NODE-RED ..	15
2.1 Загальні відомості	15
2.2 Завдання до лабораторної роботи	16
2.3 Контрольні питання	21
3 ЛАБОРАТОРНА РОБОТА № 3 ВИКОРИСТАННЯ ХМАРНОЇ ПЛАТФОРМИ BLUEMIX В ІОТ	22
3.1 Загальні відомості	22
3.2 Завдання до лабораторної роботи	22
3.3 Контрольні питання	36
ЛІТЕРАТУРА	36

1 ЛАБОРАТОРНА РОБОТА № 1. ЗНАЙОМСТВО З RASPBERRY PI 3

Мета роботи: ознайомитися з особливостями роботи одноплатного комп'ютера Raspberry Pi 3.

1.1 Загальні відомості

Raspberry Pi - одноплатний комп'ютер розміром з банківську карту, спочатку розроблений як бюджетна система для навчання інформатиці, що згодом одержав набагато більш широке застосування і популярність, ніж очікували його автори.

Raspberry Pi випускається в декількох комплектаціях: модель «А», модель «В», модель «В+», модель «2В», «Zero» і «3В». Перші три версії оснащені ARM11 процесором Broadcom BCM2835 [1] з тактовою частотою 700 МГц і модулем оперативної пам'яті на 256МБ / 512МБ, розміщеними за технологією «package-on-package» безпосередньо на процесорі. Модель «2В» оснащується процесором з 4 ядрами Cortex-A7 з частотою 1 ГГц і оперативною пам'яттю розміром 1 ГБ. Модель «А» оснащується одним USB 2.0 портом, модель «В» двома, а моделі «В +» і «2В» - чотирма. Також в моделях «В», «В +» і «2В» присутній порт Ethernet. Крім основного ядра, BCM2835 включає в себе графічне ядро з підтримкою OpenGL ES 2.0, апаратного прискорення і FullHD-відео і DSP-ядро [2-4]. Однією з особливостей є відсутність годинника реального часу [5].

Вивід відеосигналу можливий через композитний роз'єм RCA або через цифровий HDMI-інтерфейс. У версії «В +» і «2В» 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 «В +») або SDIO [6].

Однією з найцікавіших особливостей Raspberry Pi є наявність портів GPIO (general-purpose input/output – порти вводу/виводу загального призначення). Завдяки цьому цей комп'ютер можна використовувати для управління різними пристроями. У моделі «В» встановлено 26-піновий GPIO, а в моделі «В +» і «2 В» - 40-піновий роз'єм GPIO.



Рисунок 1.1 – Роз’єм GPIO

Raspberry Pi працює в основному на операційних системах, заснованих на Linux ядрі. Також можлива установка Windows 10 IOT [7]. ARM11 засновано на 6 версії ARM, який підтримує не всі версії Linux. Для установки операційних систем існує інструмент NOOBS. На моделі Raspberry Pi 3 B, яка надається для виконання лабораторних робіт, встановлено ОС Raspbian.

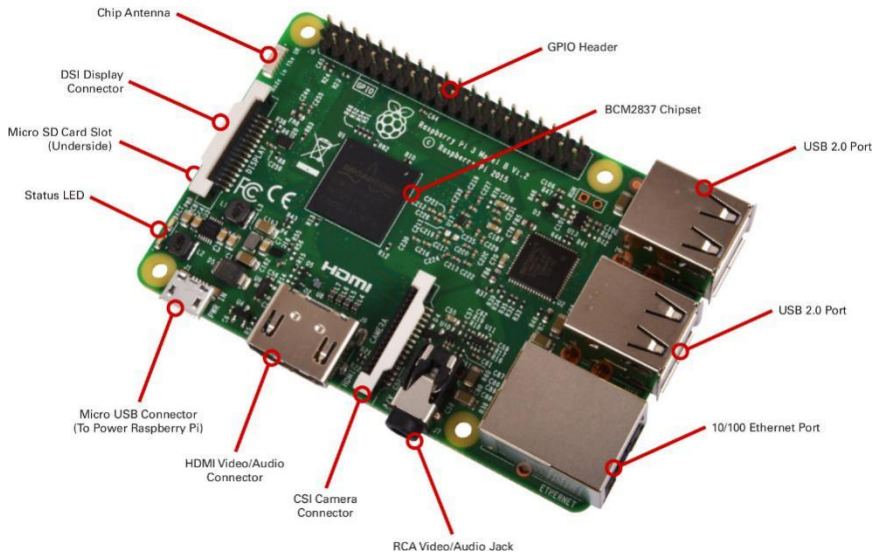


Рисунок 1.2 – Одноплатний комп'ютер Raspberry Pi 3 B

Графічне відображення плати представлено на рис.1.2.

1.2 Завдання до лабораторної роботи

Завдання 1. Робота з датчиком температури.

Хід роботи.

1. Підключення датчика до плати

Дистрибутив Raspbian підтримує DS18B20 - 1-провідний температурний сенсор. Такі сенсори є 3-піновими, як транзистори, і є точними цифровими пристроями. Оскільки Raspberry Pi не має аналогово-цифрового перетворювача, використання аналогового температурного сенсора, наприклад TMP3, неможливо. Тому DS18B20 - хороший вибір для вимірювання температури.

Для вимірювання температури необхідні такі пристрої (рис.1.3 – 1.5):

- плата Raspberry Pi 3 B;
- цифровий температурний сенсор DS18B20;
- резистор 4.7-10КОм;
- макетна плата;

- набір джамперів;
- Pi Cobber.



Рисунок 1.3 – Цифровий температурний сенсор DS18B20

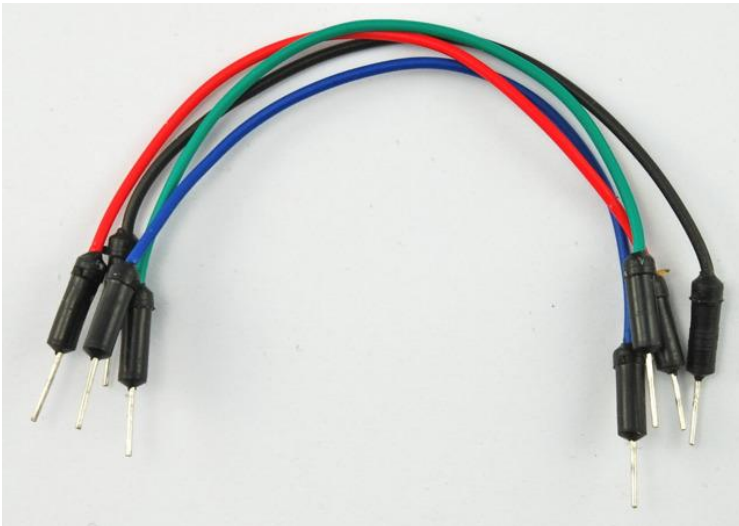


Рисунок 1.4 – Набір джамперів

Рисунок 1.6 – Схема підключення елементів

Схема макетної плати для DS18B20 показана на рис.1.6, а підключені вже елементи на рис.1.7. Однопровідні сенсори DS18B20 можуть бути підключені паралельно на відміну від більшості сенсорів. Усі сенсори використовують однакові роз'єми. Нам потрібен всього один резистор 4.7K на усі. Резистор потрібен, щоб передача даних по шині даних була стабільною. Будьте уважні при установці DS18B20. Вигнутий край повинен бути зліва, як показано на зображенні нижче. Якщо Ви встановите чіп навпаки, то він нагріється і зламається. Якщо Ви використовуєте водонепроникну версію DS18B20, то у Вашого пристрою 3 дроти: червоний, чорний і жовтий. Крім цього є ще провід відбракування з голою міді, який не потрібно підключати. Якщо Ви використовуєте високотемпературну версію DS18B20, підключіть помаранчеву стрічку до 3,3, білі дроти до землі, а синю стрічку для даних до гнізда # 4. Вам потрібен резистор $\sim 4.7\text{-}10\text{K}$ від даних до 3,3 DS18B20 Хоча DS18B20 виглядає зовсім як звичайний транзистор, він виконує більше функцій. У чіпі: однопровідний серійний інтерфейс, керуюча логічна схема і сам температурний сенсор. Вивід чіпа посилає цифрові повідомлення, а інтерфейс Raspbian їх зчитує. Працювати з пристроєм можна через командний рядок або SSH, до повного запуску програми.

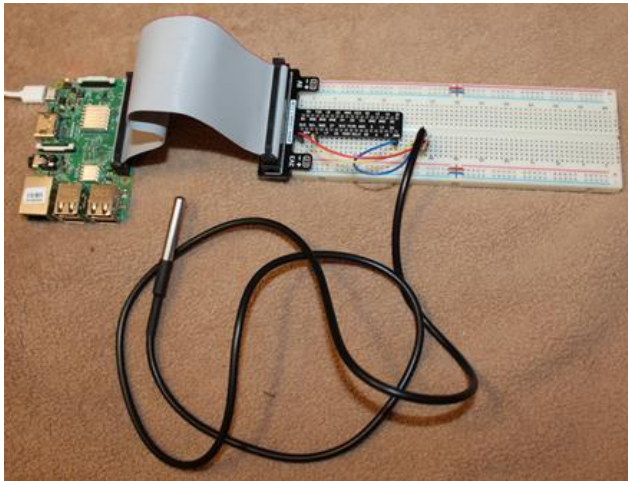


Рисунок 1.7 – Підключені елементи на макетній платі

2. Далі додамо підтримку одного проводу Для початку додайте такий рядок в **/boot/config.txt**. Ви можете редагувати файл за допомогою nano. Запустіть **sudo nano /boot/config.txt**, перейдіть в самий низ і введіть наступне: **dtoverlay = wl-gpio**. Далі перезапустіть систему (**sudo reboot**). Після цього увійдіть ще раз і введіть команди, представлені у лістингу 1.1, у вікно терміналу. У папці "devices", папка з назвою на '28' - 'може мати інше ім'я, тому використовуйте CD, щоб задати потрібне ім'я папки.

Лістинг 1.1 – Підключення сенсора

```
sudo modprobe wl-gpio
sudo modprobe wl-therm
cd /sys/bus/wl/devices ls
cd 28-xxxx
cat wl_slave
```

У результаті в кінці рядка буде ТАК чи НІ. Якщо написано так, то температура в кінці другого рядка в 1/000 градусах Цельсія (рис.1.8).

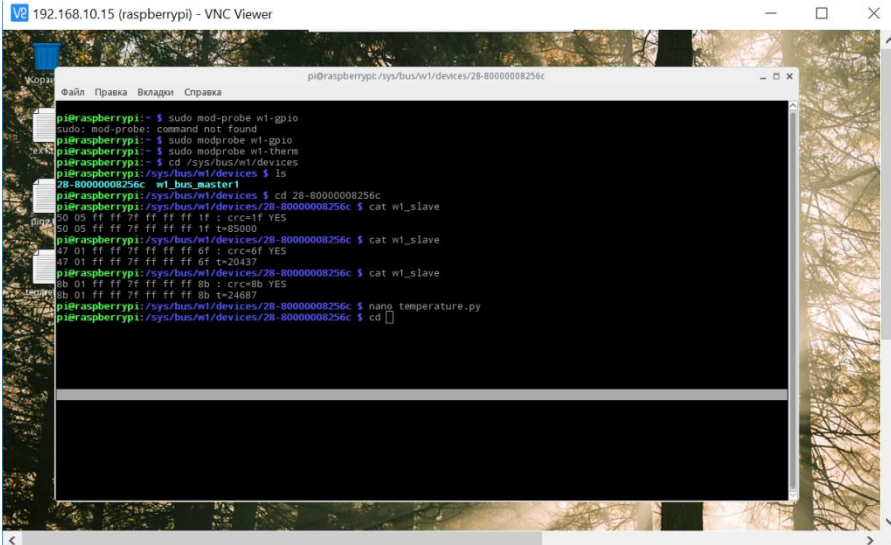


Рисунок 1.8 – Підключення сенсора

Розробка програмного забезпечення.

Програмне забезпечення розроблено з використанням мови Python та відповідає за невдалі повідомлення і повідомляє про температуру в С і F кожну секунду (лістинг 1.2).

Лістинг 1.2 – Обробка даних з сенсора

```
import os
import glob
import time
os.system('modprobe wl-gpio')
os.system('modprobe wl-therm')
base_dir = '/sys/bus/wl/devices/'
device_folder = glob.glob(base_dir + '28*')[0]
device_file = device_folder + '/wl_slave'
def read_temp_raw():
    f = open(device_file, 'r')
    lines = f.readlines()
    f.close()
    return lines
def read_temp():
    lines = read_temp_raw()
    while lines[0].strip()[-3:] != 'YES':
        time.sleep(0.2)
        lines = read_temp_raw()
    equals_pos = lines[1].find('t=')
    if equals_pos != -1:
        temp_string = lines[1][equals_pos+2:]
        temp_c = float(temp_string) / 1000.0
        temp_f = temp_c * 9.0 / 5.0 + 32.0
        return temp_c, temp_f
    while True:
        print(read_temp())
        time.sleep(1)
```

На початку програма подає команди 'modprobe' для запуску інтерфейсу. Наступні 3 рядки потрібні для того, щоб знайти файл з якого слід читати повідомлення. Зчитування температури відбувається в двох функціях. read_temp_raw викликає два рядки повідомлення з інтерфейсу. read_temp виконує перевірку повідомлень до тих пір, поки не отримає "YES" у кінці першого рядка. Функція отримує два результату: температуру в градусах С і температуру в градусах F. Якщо Ви хочете, Ви можете розділити їх навпіл таким чином:

1. Підключення камери. Для цього потрібно Raspberry Pi вимкнути. Далі акуратно підняти (на 1-2мм) пластикову засувку на роз'ємі, а потім акуратно вставити шлейф від камери в роз'єм, таким чином, щоб контактні площадки шлейфу дивилися в сторону від Ethernet-роз'єму, заклацнути пластикову засувку на роз'ємі. Тепер шлейф повинен бути затиснутий в роз'ємі і можна включати Raspberry Pi.

2. Налаштування камери. Після завантаження ОС, запускаємо налаштування

```
sudo raspi-config
```

Обираємо пункт Enable camera, тиснемо Enter, а потім Finish. Після цього вам запропонують перезавантаження.

3. Перевірка роботи камери. Для перевірки роботи модуля камери можна скористатися утилітою raspistill. виконавши команду

```
raspistill -o cam.jpg
```

У результаті ви отримаєте в поточній директорії, файл cam.jpg з виконаним камерою знімком.

За замовчуванням, знімок буде 2592 x 1944 (5038848 пікселів - адже камера 5 Мп)

Для доступу до камери можна використовувати MMAL і V4L API.

MMAL (через OpenMAX) - Broadcom specific API used only on Videocore 4 systems. raspistill для захвату картинки використовує / DEV / vchiq.

4. Навчившись отримувати знімки за допомогою камери, напишемо скрипт, що дозволяє знімати зображення з камери щохвилини (лістинг 1.3):

Лістинг 1.3 – Зміст файлу camera.sh

```
#!/bin/bash
```

```
DATE=$(date +"%Y-%m-%d_%H%M")
```

```
#raspistill -vf -hf -o /home/pi/camera/$DATE.jpg
```

```
# No vertical and horazontal flip
echo -n $DATE
raspistill -o /home/pi/camera/$DATE.jpg
echo " - ok"
```

Створюємо в домашньому каталозі дефолтного користувача pi, директорію camera для зберігання знімків.

А тепер, запускаємо скрипт camera.sh на виконання зйомки кожну хвилину:

```
while ;; do ./camera.sh; sleep 1m; done
```

У каталозі camera повинно щохвилини з'являтися нове зображення із назвою у форматі рік-місяць-день година-хвилина.jpg

Завдання 3*. Робота з відео

1.3 Контрольні питання

1. Яке призначення Raspberry Pi 3?
2. Які структурні особливості Raspberry Pi 3?
3. Які аналоги Raspberry Pi 3?
4. У чому основні переваги Raspberry Pi 3 перед її аналогами?

2 ЛАБОРАТОРНА РОБОТА № 2 ЗАСТОСУВАННЯ КОНФІГУРАТОРА ІНТЕРНЕТУ РЕЧЕЙ NODE-RED

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

2.1 Загальні відомості

Node-RED - це конфігуратор Інтернету речей (Internet of Things, IoT), графічний інструмент, який дозволяє створити просту схему для об'єднання кількох програм, веб-сайтів і апаратного забезпечення. Node-RED - це візуальний інструмент, який спрощує передачу і обробку повідомлень в Інтернеті речей.

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

Дане рішення зручно як проміжне для зв'язку пристроїв різного типу між собою і / або ж з системою автоматизації або, наприклад, СУБД або іншою хмарою. З використанням додаткових пакетів Node-RED можна використовувати для створення простих систем автоматизації розумного будинку, але рішення буде відносно обмеженим через неповне покриття функціональних потреб розумного будинку.

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

- Raspberry Pi;
- BeagleBone Black;
- Arduino.

З урахуванням озвучених факторів Node-RED зручно використовувати на шлюзах між різними мережами пристроїв інтернету речей, які функціонують на власних, як правило, більш простих протоколах, і традиційним інтернетом, побудованих на TCP/IP, UDP. У цьому випадку він дозволить більш оптимально використовувати вільні ресурси шлюзу, який працює, як правило, на Linux [1].

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

2.2 Завдання до лабораторної роботи

Завдання 1. Запустити графічний конфігуратор Node-RED на робочому комп'ютері.

Хід роботи.

1. Інсталювати node.js на робочий комп'ютер. Для цього необхідно перейти за посиланням <https://nodered.org/docs/getting-started/installation> та закачати версію, відповідну вашій операційній системі.

2. Інсталювати із завантаженого файлу Node.js.

3. Знайти інсталюваний продукт у меню «Пуск» робочого комп'ютеру та запустити командний рядок (рис.2.1).

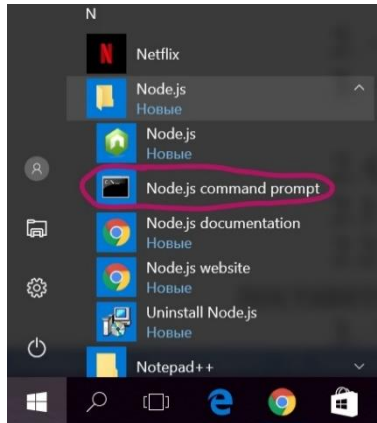


Рисунок 2.1 – Меню «Пуск» із встановленою платформою Node.js

Далі усі дії з інсталювання виконуємо у запущеному вікні командного рядка.

4. Обновлюємо версію менеджера пакету вузлів npm (скороченно від node's package manager) командою
`npm install -g npm@2.x`

5. Інсталуємо node-gyp - крос-платформний інструмент командного рядка написаний у Node.js для компіляції власних модулів для Node.js командою

```
npm install -g node-gyp
```

6. Безпосередньо інсталуємо Node-RED командою

```
npm install -g --unsafe-perm node-red
```

7. Запускаємо Node-RED. Для цього у командному рядку (рис.2.2) вводимо

```
node-red,
```

а у браузері (рис.2.3) вводимо

```
http://localhost:1880
```

```
C:\Users\Мария>node-red
8 Feb 01:52:34 - [info]

Welcome to Node-RED
=====

8 Feb 01:52:34 - [info] Node-RED version: v0.16.2
8 Feb 01:52:34 - [info] Node.js version: v6.9.5
8 Feb 01:52:34 - [info] Windows_NT 10.0.14393 x64 LE
8 Feb 01:52:35 - [info] Loading palette nodes
8 Feb 01:52:41 - [warn] -----
8 Feb 01:52:41 - [warn] [rpi-gpio] Info : Ignoring Raspberry Pi specific node
8 Feb 01:52:41 - [warn] [tail] Not currently supported on Windows.
8 Feb 01:52:41 - [warn] -----
8 Feb 01:52:41 - [info] Settings file : \Users\Мария\.node-red\settings.js
8 Feb 01:52:41 - [info] User directory : \Users\Мария\.node-red
8 Feb 01:52:41 - [info] Flows file : \Users\Мария\.node-red\flows_MARY.json
8 Feb 01:52:41 - [info] Server now running at http://127.0.0.1:1880/
8 Feb 01:52:41 - [info] Starting flows
8 Feb 01:52:41 - [info] Started flows
```

Рисунок 2.2 – Запуск Node-RED у командному рядку

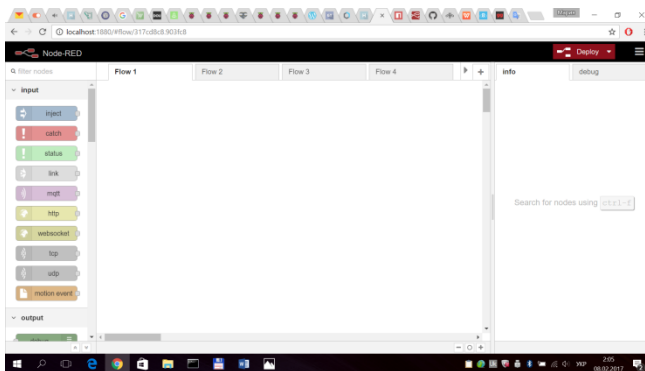


Рисунок 2.3 – Робоче вікно Node-RED у браузері

На цьому етапі запуску графічного конфігуратора Node-RED завершено.

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

Робоче вікно конфігуратора Node-RED складається з декількох областей (рис.2.4):

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

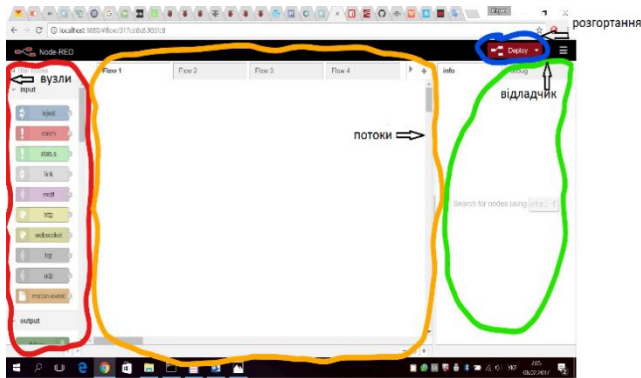


Рисунок 2.4 – Робоче вікно Node-RED у браузері з поміченими областями

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 пінгування сайту yandex.ru кожні 3 сек та результат пінгування зберігає у файлі на жорсткому диску (використовуються елементи ping, file).

Завдання 2в. Отримати дані із зовнішнього ресурсу.

Для цього імпортуємо приклад наведений нижче у лістингу 2.1. Для імпорту натискаємо Ctrl-I та копіюємо дані лістингу у вікно, яке при цьому відкривається (рис.2.5). Обираємо «new flow» та натискаємо кнопку «Import».

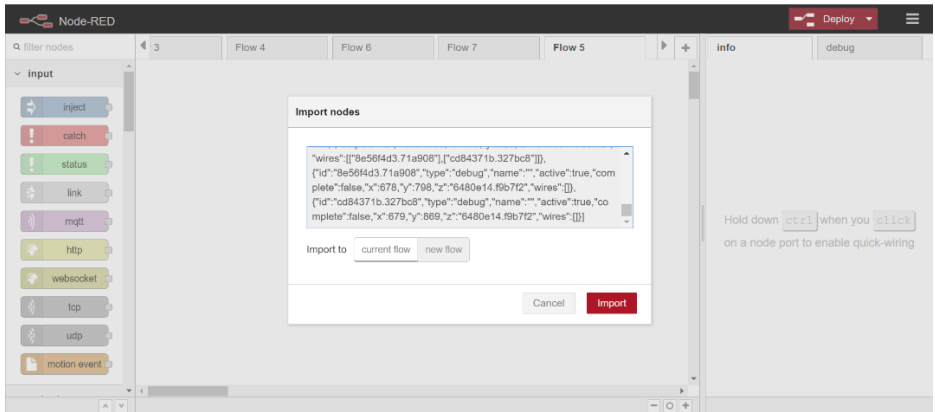


Рисунок 2.5 – Імпорт даних

Лістинг 2.1 – Отримання даних із зовнішнього ресурсу

```
[{"id":"11b032a3.ee4fcd","type":"inject","name":"Tick","
topic":"","payload":"","repeat":"","crontab":"*/5 * * *
*","once":false,"x":161,"y":828,"z":"6480e14.f9b7f2","wires":
[["a2b3542e.5d4ca8"]]}, {"id":"a2b3542e.5d4ca8","type":"http
request","name":"UK
Power","method":"GET","url":"http://realtimeweb-
prod.nationalgrid.com/SystemData.aspx","x":301,"y":828,"z":"6
480e14.f9b7f2","wires":[["2631e2da.d9ce1e"]]}, {"id":"2631e2da
.d9ce1e","type":"function","name":"UK Power
Demand","func":"// does a simple text extract parse of the
http output to provide an object containing the uk power
demand, frequency and time\n\nif
(~msg.payload.indexOf('<span')) {\n    var dem =
msg.payload.split('Demand:') [1].split(\"MW\") [0];\n    var
fre =
msg.payload.split('Frequency:') [1].split(\"Hz\") [0];\n\n
msg.payload = {};\n    msg.payload.demand =
parseInt(dem.split(">\" [1].split("<\" [0]);\n
msg.payload.frequency =
parseFloat(fre.split(">\" [1].split("<\" [0]);\n    \n
msg2 = {};\n    msg2.payload = (msg.payload.frequency >= 50)
? true : false;\n\n    return [msg,msg2];\n}\n\nreturn
null;","outputs":"2","valid":true,"x":478,"y":828,"z":"6480e1
4.f9b7f2","wires":[["8e56f4d3.71a908"], ["cd84371b.327bc8"]]},
{"id":"8e56f4d3.71a908","type":"debug","name":"","active":tru
e,"complete":false,"x":678,"y":798,"z":"6480e14.f9b7f2","wire
s":[]}, {"id":"cd84371b.327bc8","type":"debug","name":"","acti
```

```
ve":true,"complete":false,"x":679,"y":869,"z":"6480e14.f9b7f2", "wires":[]}]}
```

При цьому буде створено наступну схему (рис.2.6):

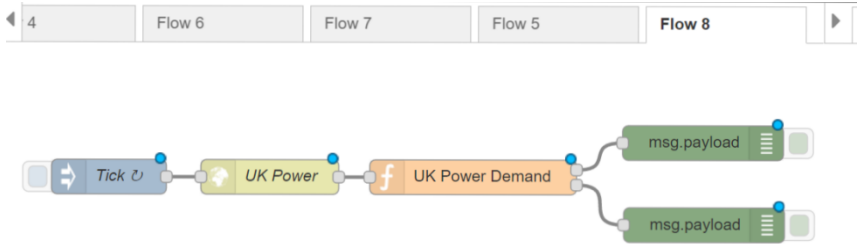


Рисунок 2.6 – Схема за імпортованими даними

На отриманій схемі є елементи `HttpRequest node` та `function node`. `HttpRequest node` використаний для отримання веб-сторінки під час запуску. `function node` – використовується для опису дій, які необхідно виконати.

Проаналізуйте властивості представлених на схемі елементів.

Завдання 2г. Синтез мовлення.

Перетворімо зібрану інформацію у попередньому завданні до аудіо формату. Для цього необхідно інсталиувати новий вузол, який забезпечує синтез мовлення, `node-red-contrib-play-audio`.

Інсталиємо вузол командою

`npm install node-red-contrib-play-audio`.

Далі оновлюємо схему згідно рис.2.7.

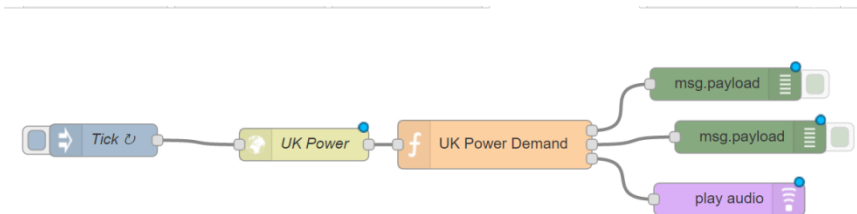


Рисунок 2.7 – Оновлена схема за імпортованими даними

У полі TTS Voice вузла play audio обираємо «Google русский». Функцію доповнюємо записом `msg3.payload = ('Частота составляет: ' + msg.payload.frequency);`

Запускаємо проект.

Завдання 2д. 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.2.8.

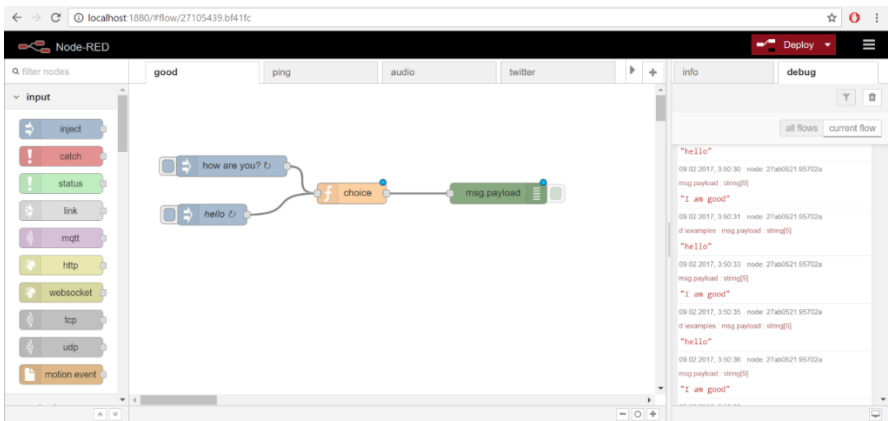


Рисунок 2.8 – Результат виконаного завдання

2.3 Контрольні питання

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

3 ЛАБОРАТОРНА РОБОТА №3 ВИКОРИСТАННЯ ХМАРНОЇ ПЛАТФОРМИ BLUEMIX В ІОТ

3.1 Загальні відомості

Bluemix - це відкрита хмарна платформа IBM, за допомогою якої є можливість налаштувати інфраструктуру, підготувати ресурси і хостинг, дозволяючи безпосередньо зосередитися на створенні додатків для Web або мобільних пристроїв. Bluemix підтримує всі популярні мови програмування, дозволяючи розробнику обрати необхідну платформу: iOS, Android або HTML з JavaScript, та мову програмування: Ruby, PHP або Java.

Bluemix - це реалізація архітектури IBM Open Cloud Architecture на основі відкритого ПЗ Cloud Foundry, що працює за принципом «платформа як послуга» (Platform as a Service - PaaS). Більш детальна інформація на офіційному сайті <https://www.ibm.com/developerworks/ru/library/cl-bluemixfoundry/>.

Для роботи необхідно зареєструватися на сайті https://developer.ibm.com/sso/bmregistration?lang=ru_RU&ca=dwrussia-_bluemix-_cl-bluemixfoundry-_article или <https://console.ng.bluemix.net/> для використання безкоштовної пробної версії продукту. Безкоштовне використання продукту можливо протягом 30 днів, що цілком достатньо для виконання лабораторної роботи.

3.2 Завдання до лабораторної роботи

Завдання. За допомогою платформи Bluemix і служби IBM Watson IoT Paltform розробити Bluemix-додаток, який дозволить обробляти, візуалізувати і зберігати дані, отримані зі смартфона, який виступатиме у ролі датчика, підключеного до мережі.

Загальна схема завдання представлена на рис.3.1.

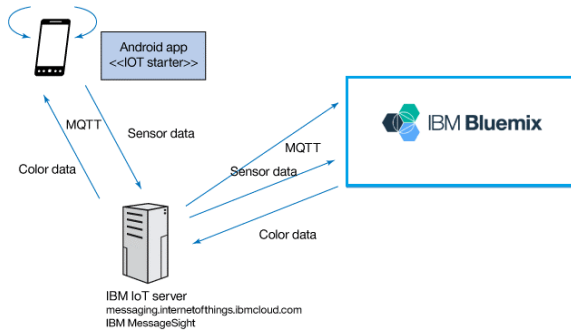


Рисунок 3.1 – Загальна схема завдання

Хід виконання

- 1. Увійдіть до Bluemix під вашим профілем**
- 2. Оберіть Catalog.**
- 3. Клацніть по Internet of Things Platform Starter (рис. 3.2).**

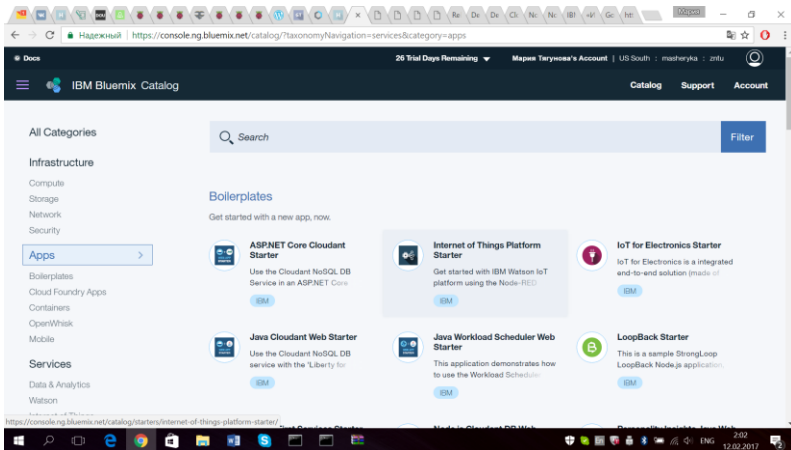


Рисунок 3.2 – Створення додатку

4. Вкажіть ім'я додатку (рис. 3.3). Це ім'я буде також використовуватися як ім'я хоста, тому воно повинно бути унікальним в Bluemix. У прикладі використовується lab3-blum.

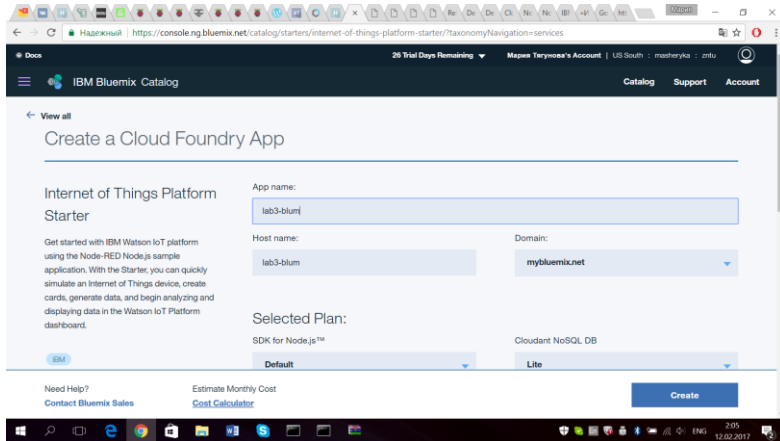


Рисунок 3.3 – Надання імені додатку

5. Натискаємо “Create”, отримуємо наступну сторінку (рис.3.4):

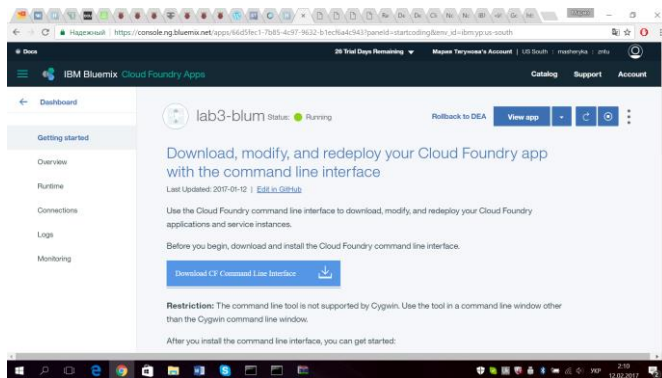


Рисунок 3.4 – Створений додаток

6. На панелі інструментів оберіть Dashboard (рис.3.5).

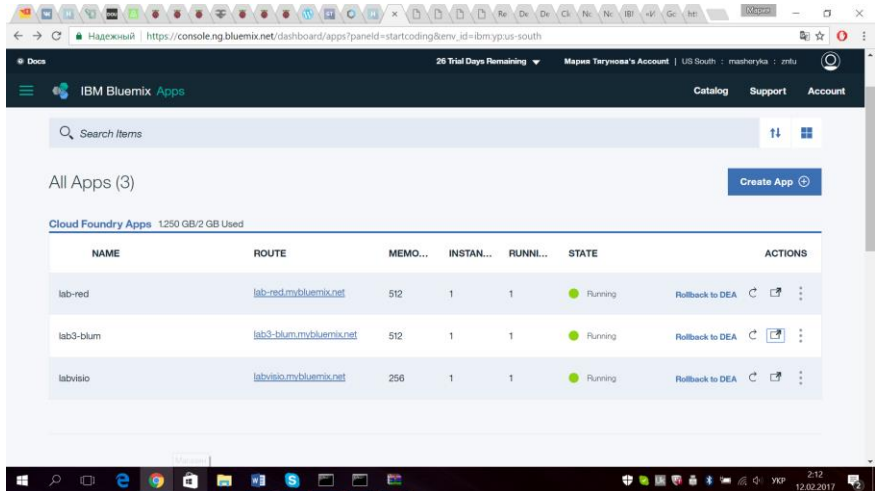


Рисунок 3.5 – Сторінка Dashboard

7. Натискаємо кнопку Create App та створюємо три служби (Service): Internet of Things Platform, Context Mapping, Driver Behavior. Для кожної із служб у Connection вказуємо зв'язок з додатком lab3-blum та оновлюємо кнопкою Restage (рис.3.6, 3.7).

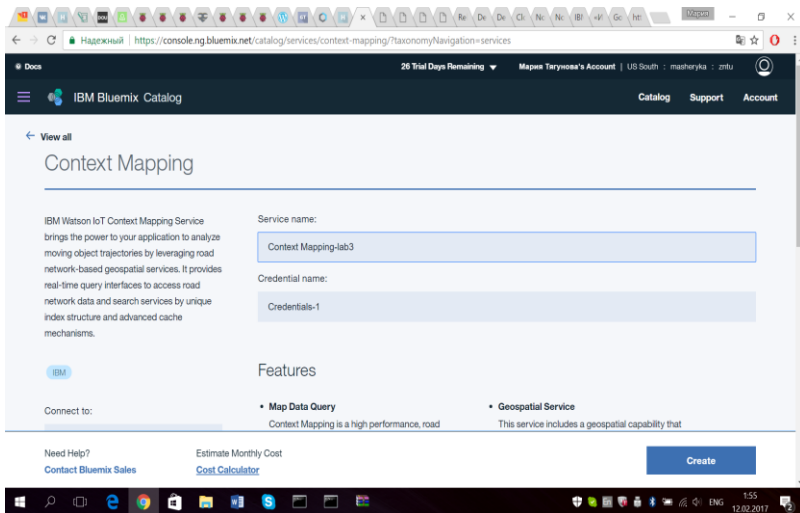


Рисунок 3.6 – Створення служби Context Mapping

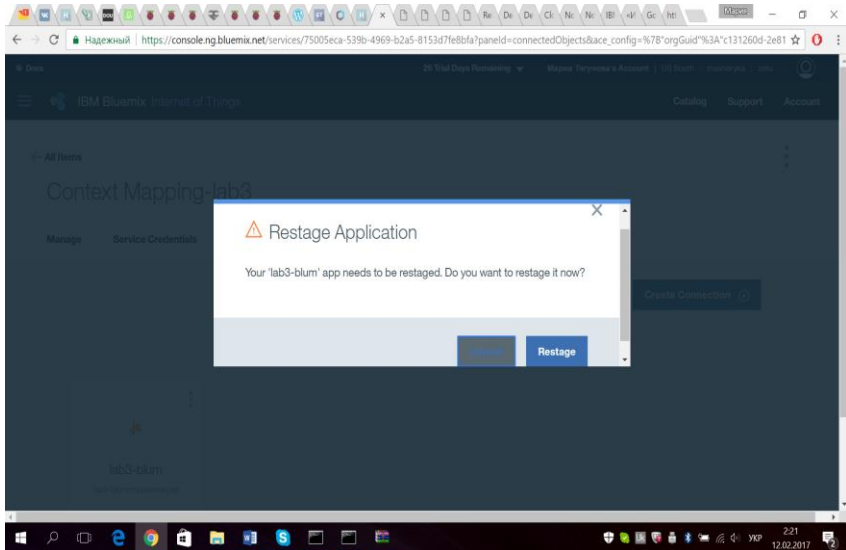


Рисунок 3.7 – Оновлення кнопкою Restage

8. Далі додаємо пристрої, які відправляють MQTT-повідомлення на IoT-сервер. Для цього обираємо службу Internet of Thing Platform-lab3 та натискаємо Launch Dashboard. Буде відкрита сторінка приборної панелі як показано на рис.3.8. Це панель сервера IBM Internet of Things- служби, незалежної від Bluemix. Додатком присвоюється ідентифікатор організації Organization ID. Він буде потрібен для розробки мобільного додатку. У прикладі надано ідентифікатор із значенням 3a0kb7.

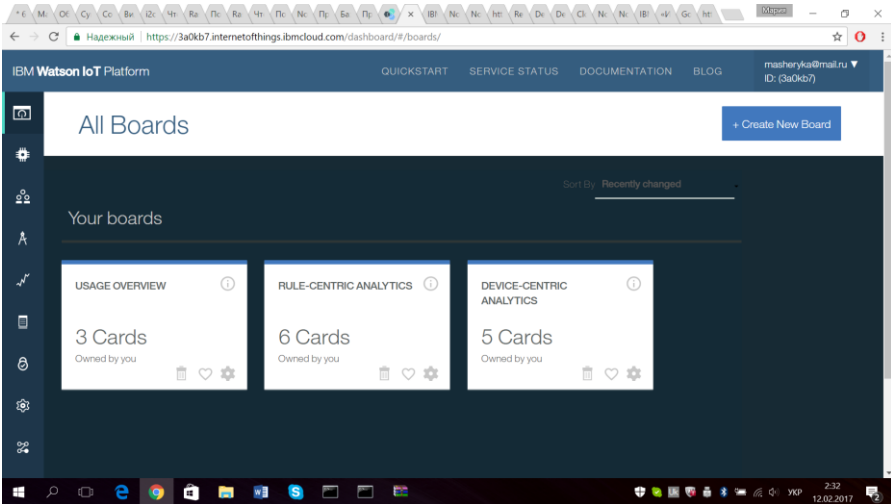


Рисунок 3.8 – Приборна панель

9. У лівому меню, яке з'являється при наведенні на нього покажчика миші, обираємо Device (рис.3.9).

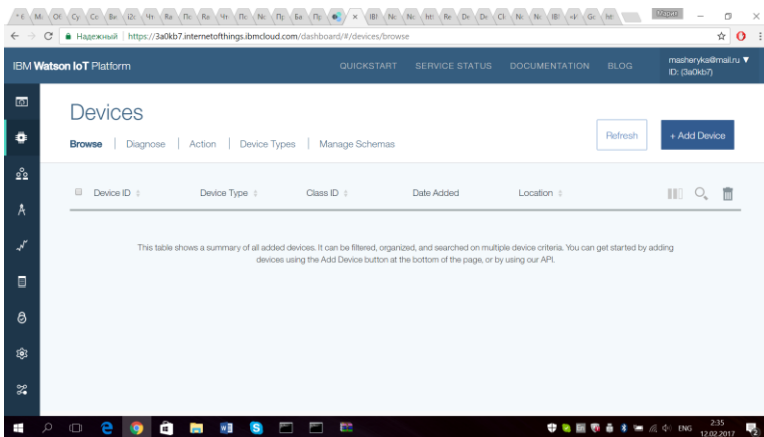


Рисунок 3.9 – Додавання пристрою

10. Після цього натискаємо кнопку Add Device і створюємо тип пристрою (рис.3.10). У вас може бути кілька типів пристроїв, а самих пристроїв може бути багато. Тип пристрою в даному випадку - це

група пристроїв із загальними характеристиками. У них можуть бути однакові датчики або ще якісь елементи. У поточному випадку тип пристрою повинен мати ім'я Android. Такий тип необхідно використовувати для додатка, яке будемо використовувати пізніше.

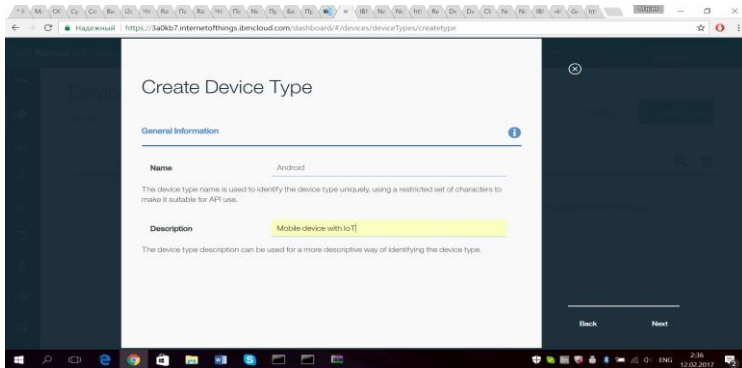


Рисунок 3.10 – Створення типу пристрою

11. Натискаємо кнопку Next. Після цього відкривається сторінка на якій можна задати різні властивості про пристрій, включаючи серійний номер або модель. Вводити ці дані не будемо, тому просто натискаємо Next (рис.3.11).

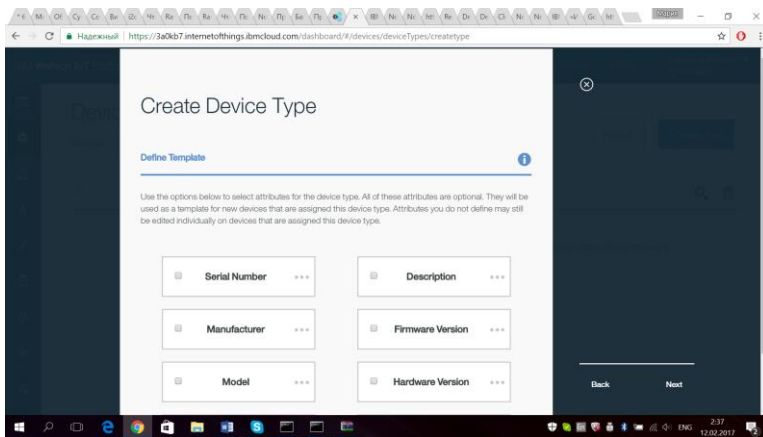


Рисунок 3.11 – Властивості пристрою

12. Далі вказуємо ідентифікатор пристрою. Це може бути, наприклад, MAC-адресу смартфона, або будь-яка назва. Проте ідентифікатор повинен бути унікальним. У прикладі показано назва «myphone» (рис.3.12).

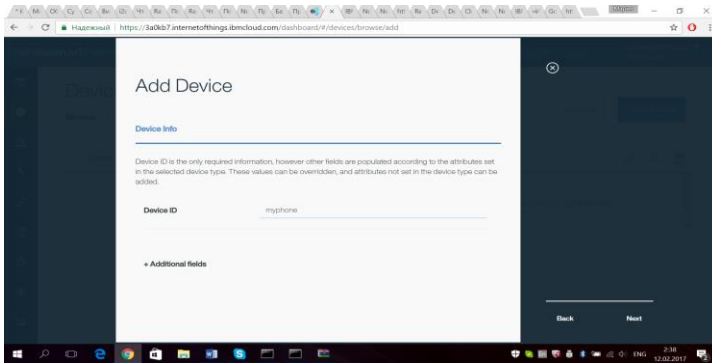


Рисунок 3.12 – Ідентифікатор пристрою

13. Натискаємо кнопку Next. Залишаємо порожнім поле Мета даних і натискаємо кнопку Next. Залишаємо порожнім ще й поле маркера аутентифікації і знову натискаємо на Next.

14. Далі вказуємо значення токена аутентифікації. Його краще запам'ятати або записати, оскільки воно знадобиться пізніше (рис.3.13). Натискаємо Next.

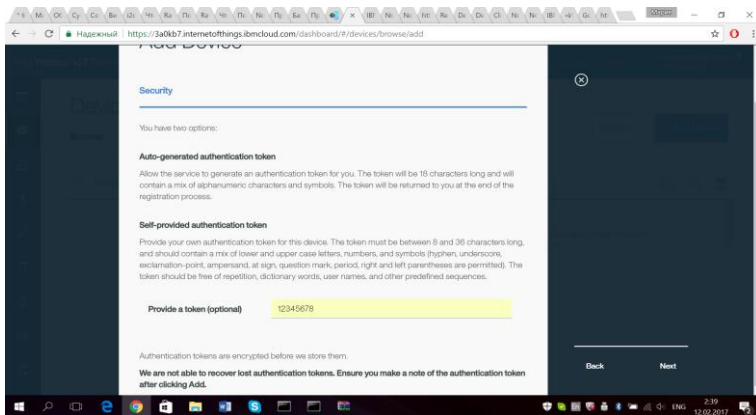


Рисунок 3.13 – Задавання токена аутентифікації

У результаті буде сгенеровано пристрій, як показано на рис. 3.14, 3.15. Після цього ми можемо відправляти MQTT-повідомлення з пристрою (смартфону) на сервер IBM IoT.

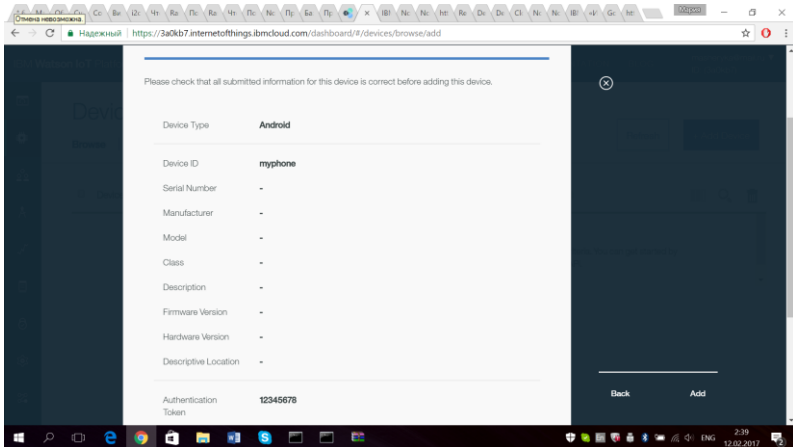


Рисунок 3.14 – Сгенерований пристрій

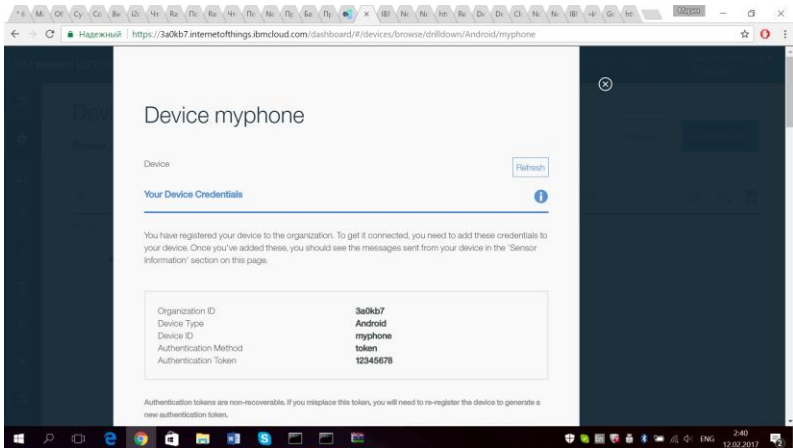


Рисунок 3.15 – Характеристики сгенерованого пристрою

15. Встановлюємо і конфігуруємо додаток для Android. Для того, щоб смартфон зчитував і відправляв показання датчика, необхідно використовувати додаток IoT Starter for Android. Але перед

чим в телефоні потрібно зайти в меню Налаштування> Безпека і від імені адміністратора пристрою дозволити встановлення програм з невідомих джерел. Для цього ставимо позначку в полі Невідомі джерела. Тепер з'явилася можливість встановлювати файли .apk не тільки з Google Play.

16. Відкриваємо браузер в телефоні і вводимо таку адресу: `ibm.box.com/v/iotstarterapp-v2`. Йде перенаправлення на сайт `m.box.com` і до файлу `iot-starter_501-v2.apk`.

17. Натискаємо Download. Далі за допомогою файлового менеджера шукаємо файл в каталозі завантажень; у цьому ж каталозі клацаємо по файлу `iot-starter_501-v2.apk` і натискаємо Install (Встановити).

18. Додаток IoT Starter після всіх названих дій буде встановлено на вашому пристрої з ОС Android. Після цього потрібно виконати ще один важливий етап - конфігурувати додаток для Android. Для цього запускаємо програму IoT Starter і вказуємо такі параметри:

- Organization: Ідентифікатор організації, який виводився на сервері IBM IoT (він був зазначений на початку етапу «Додавання пристрою, який буде відправляти MQTT-повідомлення на сервер IoT»). У нашому прикладі цей ідентифікатор має значення `3a0kd7`.

- Device ID: Ідентифікатор пристрою, який ви вказали в кінці етапу «Додавання пристрою, який буде відправляти MQTT-повідомлення на сервер IoT». У нашому прикладі цей параметр має значення `myphone`.

- Auth Токен: Токен авторизації, який ви вказали на етапі «Додавання пристрою, який буде відправляти MQTT-повідомлення на сервер IoT». У нашому прикладі цей параметр має значення `12345678`.

19. Обов'язково ставимо галочку напроти SSL та натискаємо на кнопку Activate Sensor.

Після цього додаток почне збирати свідчення датчика прискорення на смартфоні, відправляючи зібрану інформацію на сервер IBM IoT. Додаток також відображає інформацію з акселерометру і кількість отриманих або відправлених повідомлень.

20. Для перевірки відправлення повідомлення натискаємо «Send Text» та відкриваємо на комп'ютері сторінку сервера IBM IoT для організації (ці дані вказувалися в етапі «Додавання пристрою, який буде відправляти MQTT-повідомлення на сервер IoT»).

21. На вкладці Device відображається вже пристрій на базі Android OS (рис.3.15).

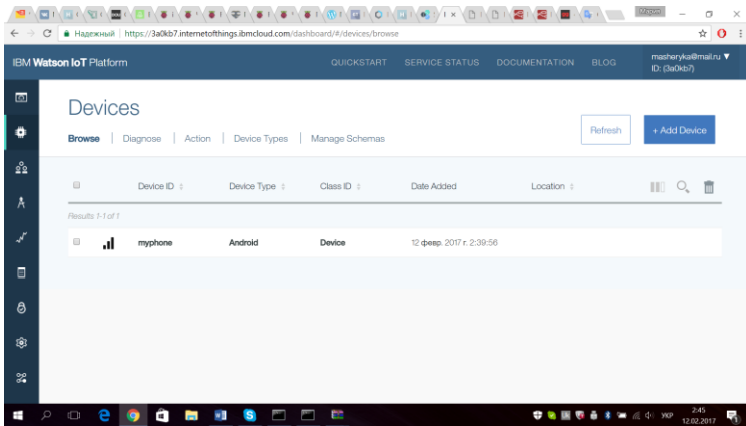


Рисунок 3.16 – Підключений пристрій

22. Клацаємо на ідентифікаторі пристрою, після цього має з'явитися вікно з повідомленнями, що надходять зі смартфона (рис.3.16). Повідомлення зі смартфона відправляються в форматі JSON формату. Вони містять інформацію про прискорення і позиції. Тепер можна працювати з даними повідомлень в середовищі Bluemix. На рис. 3.17 відображено текстове повідомлення «hello», надіслане зі смартфона.

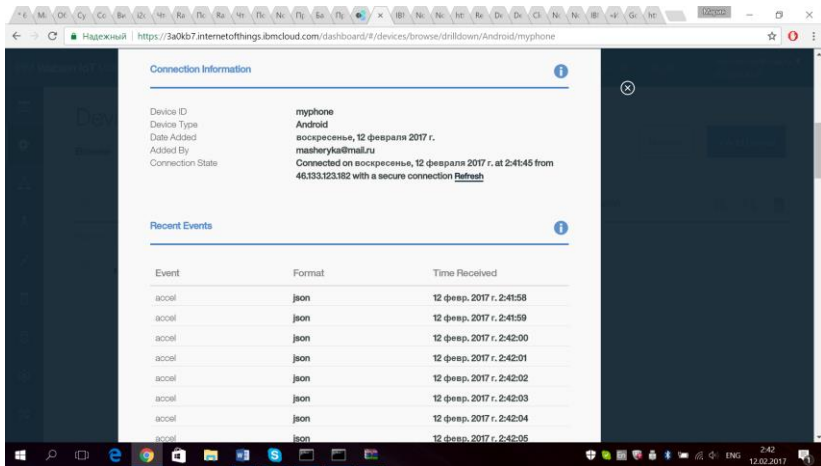


Рисунок 3.17 – Надіслана інформація у форматі json

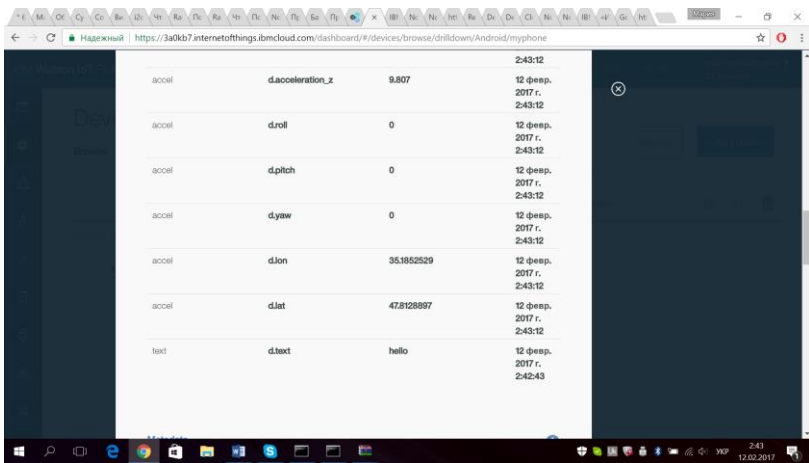


Рисунок 3.18 – Надіслана інформація за кнопкою Send Text

23. Далі покращимо додаток Bluemix IoT, використовуючи потік Node_RED для того, щоб обробляти повідомлення зі смартфона і відправляти повідомлення назад на смартфон. Телефон буде реагувати, змінюючи колір фону додатку. Для цього відкриваємо Bluemix, обираємо Dashboard і на панелі Bluemix Dashboard дивимося, чи виконується додаток Bluemix IoT (створений на етапі 1).

24. Далі відкриваємо в браузері сторінку <ім'я_вашого_додатку>.mybluemix.net, де - це назва програми Bluemix IoT, створеного на кроці 1. У редакторі Node-RED для додатка Bluemix натисніть на кнопку Go to your Node-RED flow editor (рис.3.18). Відкривається редактор з прикладом потоку.

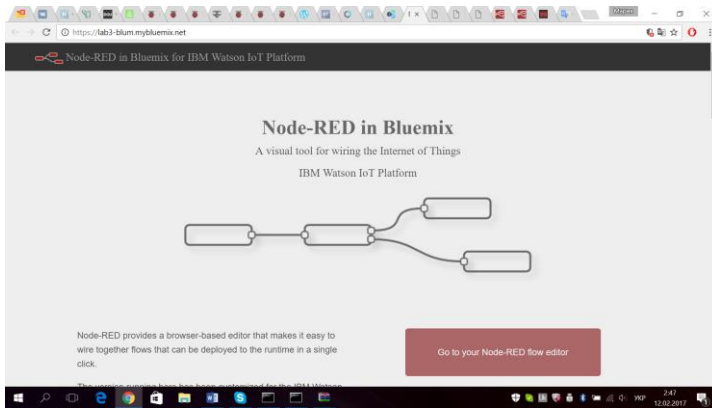


Рисунок 3.19 – Редактор Node-RED для додатка Bluemix

25. Можна використовувати в редакторі метод drag-and-drop, щоб почати конструювати потік повідомлень. Але можна і створювати такий потік самотійно, імпортуючи наведений нижче код у лістингу 3.1. Перед цим необхідно прибрати всі існуючі вузли, видаливши їх за допомогою клавіші Delete.

Лістинг 3.1 – Код потоку у Node-RED

```
[{"id":"215c6601.12d32a","type":"ibmiot
in","authentication":"boundService","apiKey":"","inputType":"
evt","deviceId"
:":"","applicationId":"","deviceType":"","eventType":"accel","c
ommandType":"","format":"json","name":"IBM IoT App In","s
ervice":"registered","allDevices":true,"allApplications":"","
allDeviceTypes":true,"allEvents":"","allCommands":"","all
Formats":"","x":264.1999969482422,"y":434.1999969482422,"z":
"dc6a0ca6.7993b","wires":[["ce21edbb.320628","fdfdaaa2.a45
678"]]}, {"id":"ce21edbb.320628","type":"function","name":"cal
c color","func":"var accelZ = msg.payload.d.acceleration_
z;var r = 0.0;var b = 0.0;var g = 0.0;if (accelZ > 0) {\tg =
Math.round(accelZ * 25);} else if (accelZ < 0 ) {\tr =
255.0;} else {\tr = 104;\tg = 109;\tb = 115;}a =
1.0;msg.eventOrCommandType = \"color\";m sg.payload =
```

```
JSON.stringify({"d":{"r":{"b":{"g":{"alpha":{"a}}}}};return msg;","outputs":1,"noerr":0,"x":437.1999969482422,"y":430.1999969482422,"z":"dc6a0ca6.7993b","wires":[["9df37edb.5d4588"]]},{"id":"9df37edb.5d4588","type":"ibmiotout","authentication":"boundService","apiKey":"","outputType":"cmd","deviceId":"112233445566","deviceType":"Android","eventCommandType":"text","format":"json","data":{"d":{"value":{"text{}}},"name":"IBM IoT App Out","service":"registered","x":629.1999969482422,"y":443.1999969482422,"z":"dc6a0ca6.7993b","wires":[],"id":"fdfdaaa2.a45678","type":"debug","name":"","active":false,"console":"false","complete":"false","x":442.1999969482422,"y":364.1999969482422,"z":"dc6a0ca6.7993b","wires":[]}]}
```

26. Копіюємо код лістингу в буфер обміну.

27. У редакторі Node-RED натискаємо Ctrl-I, щоб відкрити діалогове вікно Імпорт вузлів. Вставляємо код і натискаємо ОК (рис.3.19).

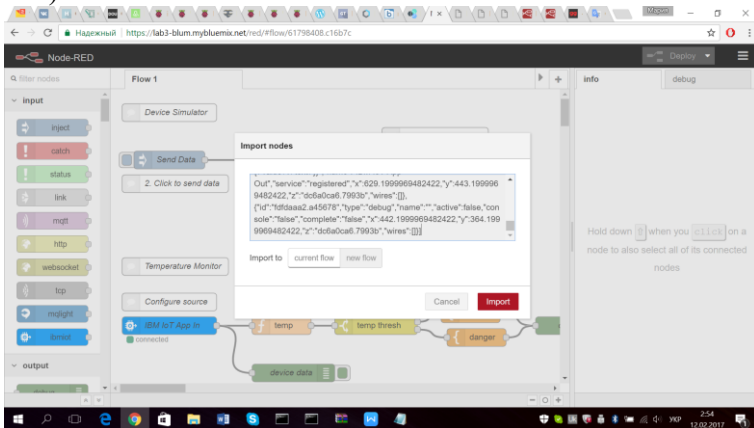


Рисунок 3.20 – Імпорт даних

28. Тепер потрібно адаптувати потік до ваших конкретних параметрів. Єдиним значущим параметром є ідентифікатор пристрою. Двічі клацаємо по вузлу IBM IoT App In. У спливаючому вікні вводим ідентифікатор пристрою, який використовували раніше (наприклад, turphone), і натискаємо ОК.

29. Так само змінюємо вузол IBM IoT App Out.

30. У редакторі клацаємо Deploy. Потік відразу ж повинен стати активним.

31. Переміщуємо смартфон, перевертаючи і нахиляючи його. У цей час фон повинен міняти свій колір, реагуючи на зміну орієнтації осі Z.

32. У редакторі Node-RED клацаємо по прямокутнику поряд з вузлом msg.payload. Після цього клацаємо по вкладці налагодження. Для того, щоб виконати налагодження. Тут ми повинні бачити повідомлення, що відправляються з телефону в форматі JSON (рис.3.20).

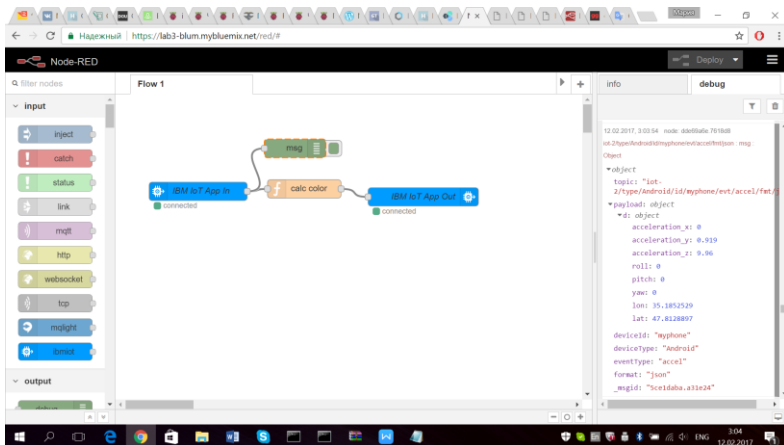


Рисунок 3.21 – Результати роботи у Node-RED

3.3 Контрольні питання

1. Що собою являє Bluemix?
2. Які основні структурні елементи Bluemix?
3. З яким програмним та апаратним забезпеченням є сумісним Bluemix?
4. Які аналоги Bluemix?
5. У чому основні переваги та недоліки Bluemix перед аналогами?

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. Ketı F. Emulation of Software Defined Networks Using Mininet in Different Simulation Environments / F. Ketı, 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

10. 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

11. Xming X Server for Windows [Electronic resource]. – Access mode: <https://sourceforge.net/projects/xming/>. – Title from the screen.

12. Cygwin [Electronic resource]. – Access mode: <http://cygwin.com>. – Title from the screen.