

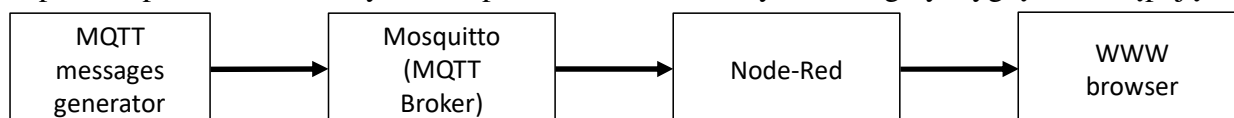
PBL5

Zajęcia zintegrowane

Temat: Node-Red tworzenie aplikacji

1. Praca przygotowawcza z systemem Node-Red

Do pracy z systemem Node-Red[1] przydatne mogą być dodatki m.in. node-red-dashboard[2]. Dla potrzeb przedstawiania wykresów przez to środowisko system mógłby wyglądać następująco:



„MQTT message generator” to przydatne tu urządzenie które publikuje cyklicznie wiadomości w zadanych tematach. Element ten jest przydatny na czas pierwszych eksperymentów z tym środowiskiem, gdy nie mamy jeszcze realnych elementów IoT publikujących określone wiadomości. Jednym z metod wytworzenia takiego elementu jest napisanie go w języku Python (plik: test.py):

```
#!/usr/bin/python

import sys
import datetime

def broker_address():
    return "127.0.0.1"
def broker_port():
    return 1883
def topicOut():
    return "test/testowy"

import paho.mqtt.publish as publish
mydata = str(datetime.date.today().strftime('%Y.%m.%d-'))
mydata = mydata + str(datetime.datetime.now().time().strftime('%H:%M:%S.%f'))

msg = 'DATA to: [' + mydata + ']'
publish.single(topicOut(),msg,hostname=broker_address(), port=broker_port())
```

Uruchamiamy tak utworzony program poprzez:

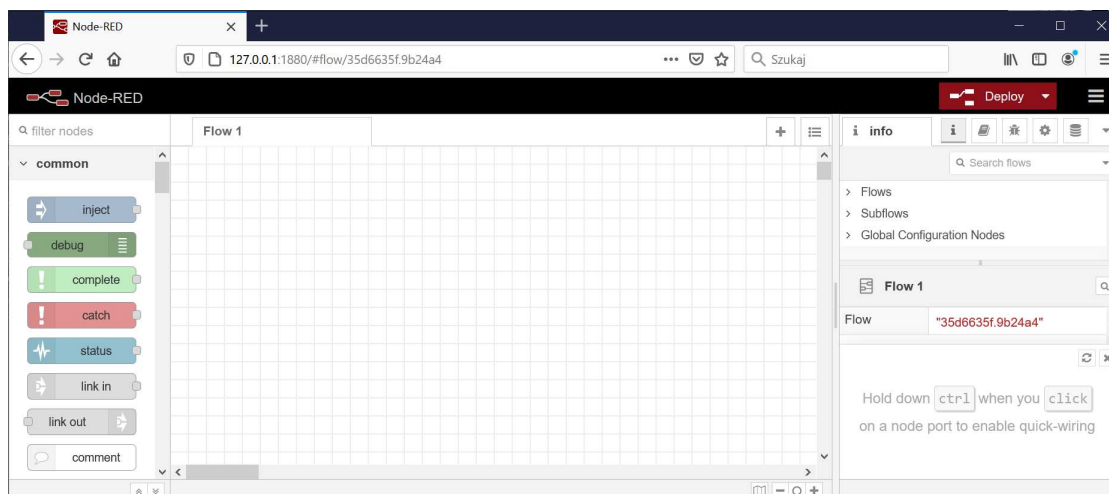
```
python test.py
```

Przerwanie lub zakończenie pracy realizujemy przez naciśnięcie CTRL i C.

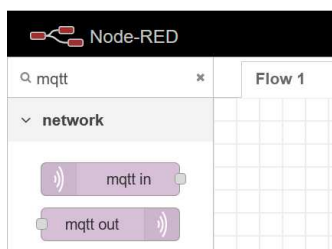
2. Tworzenie wizualizacji danych z wykorzystaniem systemu Node-Red

W przypadku kiedy obiektów Internetu Rzeczy mamy bardzo dużo trudno z biegiem czasu zorientować się jaki jest stan ich wszystkich. W takim przypadku wśród wielu możliwości systemu Node-Red tworzenie wizualizacji i wykresów jest bardzo pomocny.

Zakłada się na tym etapie że system Node-Red składa się z dwóch części: głównego panelu – gdzie opisujemy przepływ danych (tzw: „flow”) i konfigurację użytych w tym przepływie danych komponentów oraz panel wizualizacji (dashboard). Widok nie skonfigurowanego głównego panelu pokazuje obrazek:



Aby system Node-Red był gotów odbierać wiadomości z brokera MQTT należy dodać komponent wpisując w lewym górnym rogu w miejscu opisanym „filter nodes” słowo „mqtt” a pojawi się lista pasujących komponentów:



Widać zatem, że mamy do dyspozycji dwa komponenty: „mqtt in” i „mqtt out”. Pierwszy będzie subskrybował wiadomości a drugi publikował – na tym etapie istotny jest komponent „mqtt in” – który wybieramy, a następnie klikamy na ten komponent i konfigurujemy, tak aby ustawienia były zgodne z poniższymi:

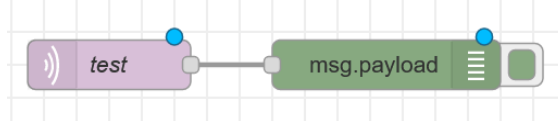
Edycja podstawowych parametrów komponentu

Ustawienia aspektów sieciowych komponentu

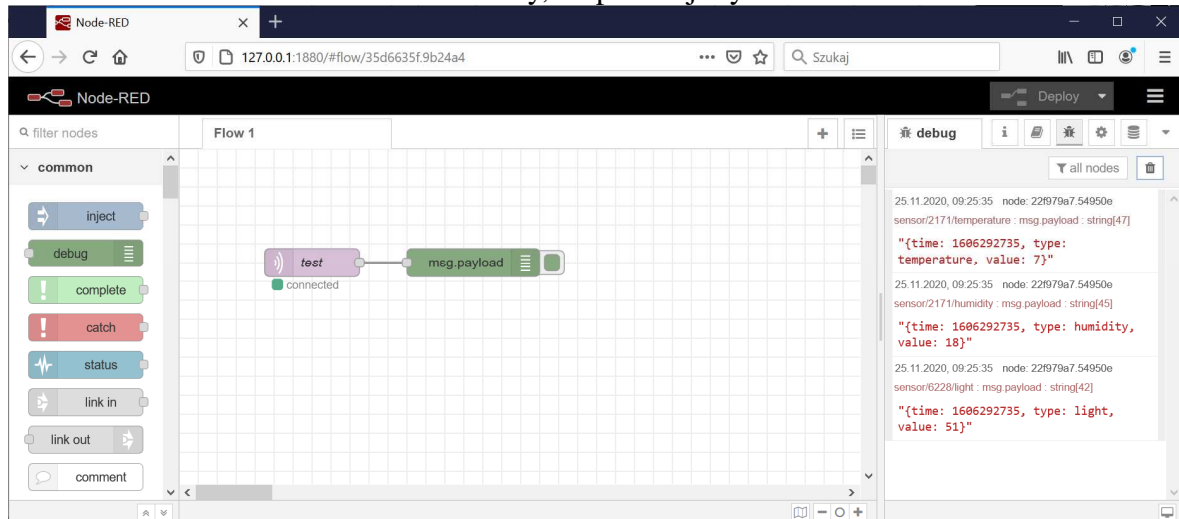
W przypadku stosowania konteneryzacji może być wymagane zamienienie adresu brokera z 127.0.0.1 na ten będący numerem IP karty sieciowej (10.0.126.196) faktycznie działającej w urządzeniu na jakim działa broker.

Proszę zauważyć także, że komponent „mqtt in” subskrybuje wszelkie wiadomości – a spodziewamy się przynajmniej określonych tematów wiadomości. Należy zatem w polu „Topic” wpisać właściwy temat subskrypcji. W polu „Name” wpisujemy dowolny tekst np.: „test”.


Aby sprawdzić czy użyty komponent poprawnie otrzymuje wiadomości od brokera, wybieramy komponent „debug” i łączymy go z „mqtt in” tworząc prosty system:

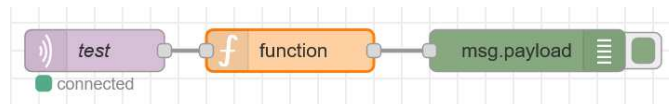


Aby tak powstały „flow 1” rozpoczął działanie należy kliknąć na klawisz „Deploy” (po wszelkich zmianach zmienia on kolor na czerwony). Od tego momentu o ile nie zostanie zaważony błąd w opisach i konfiguracjach, system uruchomi aplikację wirtualną opisaną przez „flow 1” i możemy obserwować odebrane od brokera komunikaty, co pokazuje rysunek:

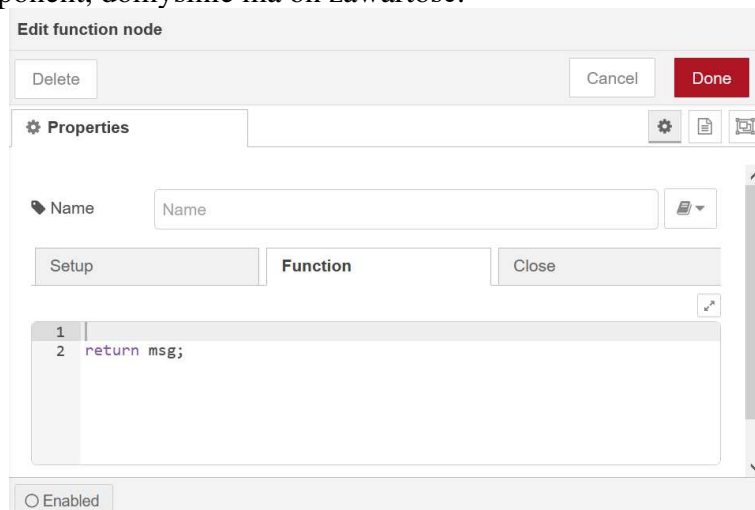


Podczas dłuższej pracy systemu w prawej części okna (część związana z „Debug”) pokazane zostaną odbierane wiadomości przez komponent „mqtt in” (tu nazwanym „test”).

Z reguły odbierane przez komponent „mqtt in” wiadomości nie nadają się bezpośrednio do wizualizacji na wykresach. Aby odpowiednie przetworzenie wiadomości[6] można było zrealizować należy użyć komponent „function” [7][8] , tak aby np.: powstał następujący rozpląt danych:



W domyślnym ustawieniu taki układ komponentów będzie dalej działał poprawnie. Funkcjonalność komponentu „function” jest opisana w języku JavaScript i jego implementacje można zmienić klikając na ten komponent, domyślnie ma on zawartość:



Założmy, że publikowana wiadomość będzie następującej treści:

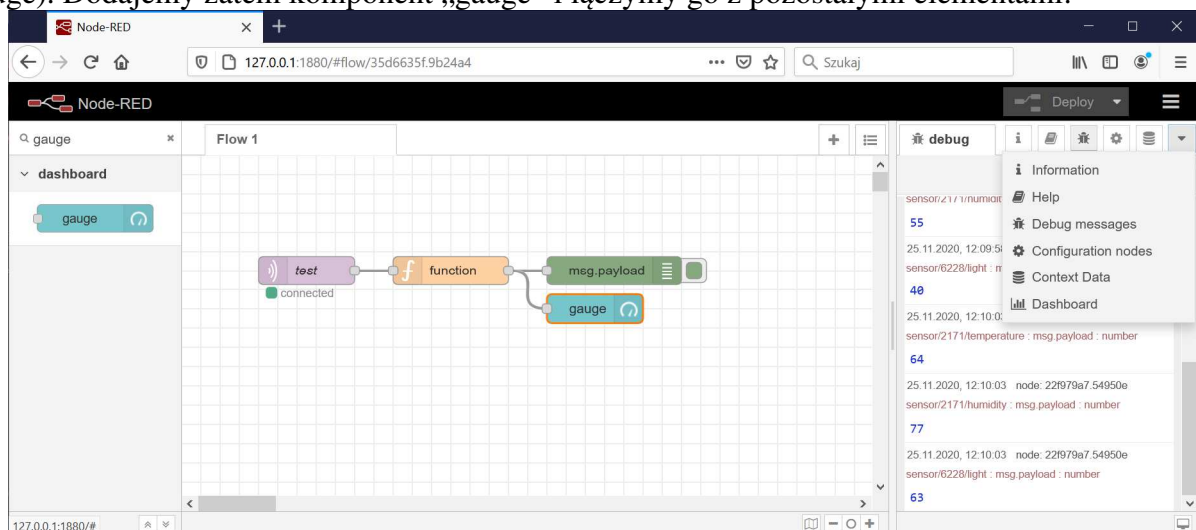
```
temp=42.8'C
```

Aby usunąć niechciane części wiadomości i „wyciągnąć” to co nas interesuje (tu: 42,8) należy - zgodnie ze składnią języka JavaScript wspieraną przez Node-Red[9] - podać wiadomość obróbce, np.: za pomocą kodu:

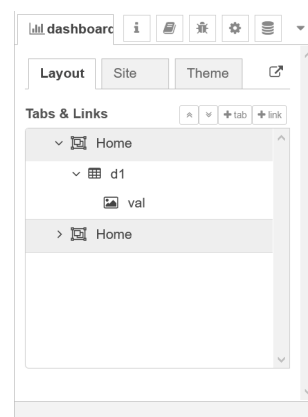
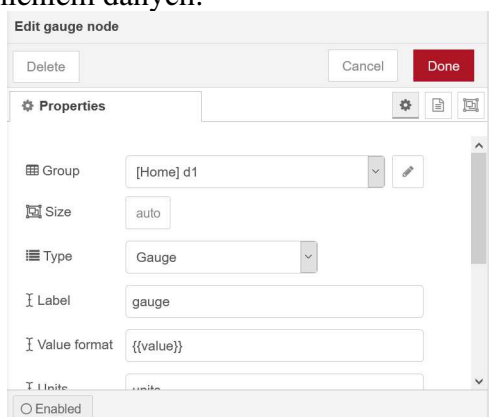
```
var t1=msg.payload.split("="); //podziel względem znaku =
var t2=t1[2];                  //w t2 zapisz część za znakiem = do końca wiersza
                                //(w przykładzie będzie to 42.8'C)
var t3=t2.split("'");          //podziel względem znaku '
msg.payload=parseInt(t3[0]);    //w finalnej wiadomości umieść tylko znaki przed `
return msg;
```


Działanie funkcji `split()` polega na utworzeniu obiektu wielowierszowego w którym każdy wiersz odpowiada polu wiadomości wejściowej rozdzielonej szukanym znakiem. Funkcja `parseInt()` ma za zadanie z postaci tekstowej (tu: 42.8) wygenerować liczbę zmiennoprzecinkową aby na jej podstawie można było wykreślać wykresy.

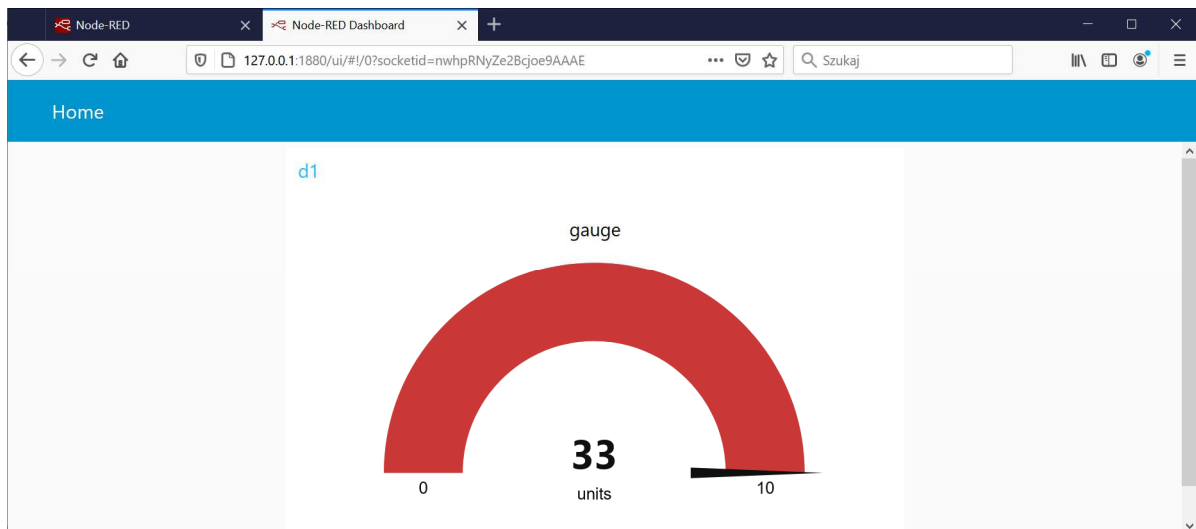
Po takiej obróbce wartość można przekazać do wyświetlenia np.: za pomocą tzw. miernika (ang gauge). Dodajemy zatem component „gauge” i łączymy go z pozostałymi elementami:



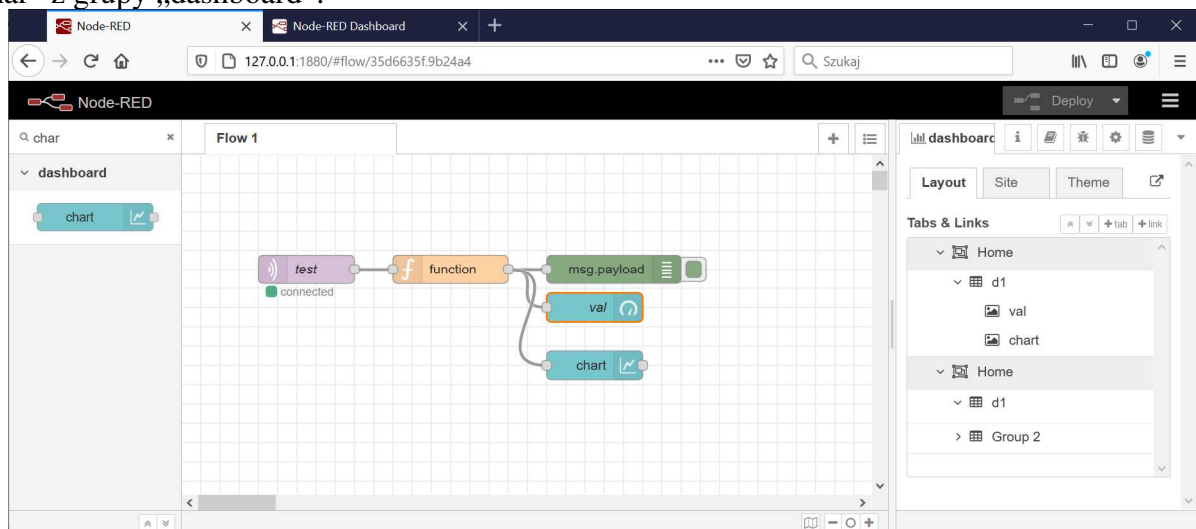
Pozostawiając aktywny component debug (tu jako `msg.payload`) możemy nadal obserwować treści przekazywane przez brokera MQTT. Aby jednak obejrzeć właściwy miernik trzeba wybrać panel wizualizacji tzw. dashboard (na powyższym rysunku widać to rozwinięte w prawym górnym rogu zasłaniające pole wartości diagnostycznych) i połączyć component miernika z właściwym strumieniem danych:



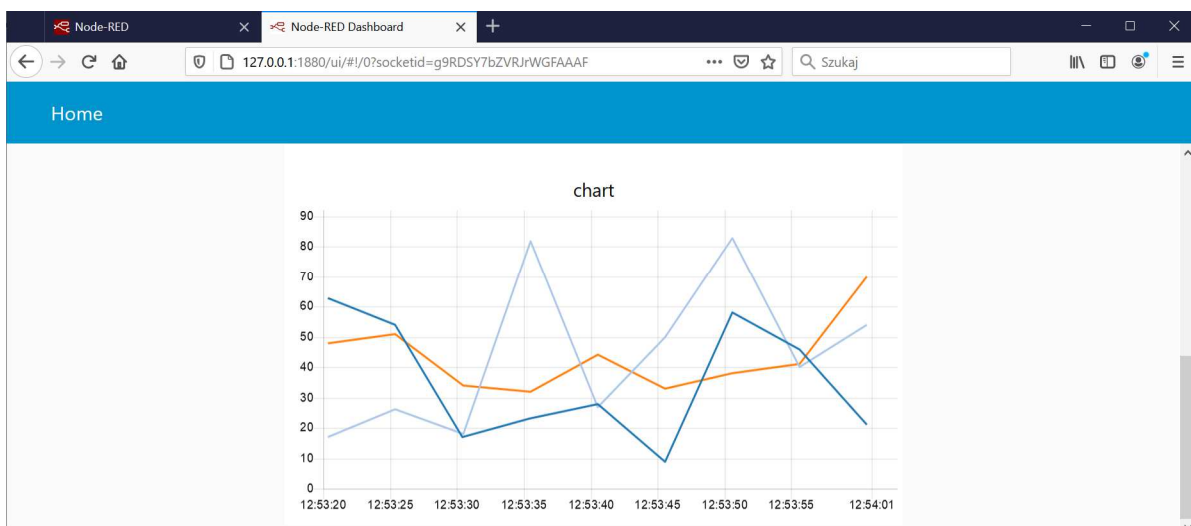
Po kliknięciu w części „Dashboard” na znak: , system otworzy nowe okienko lub zakładkę przeglądarki a w nim pokaże miernik:



Mając tak przygotowany rozptyw można dodać wykresy czasowe przez wybranie komponentu „char” z grupy „dashboard”:

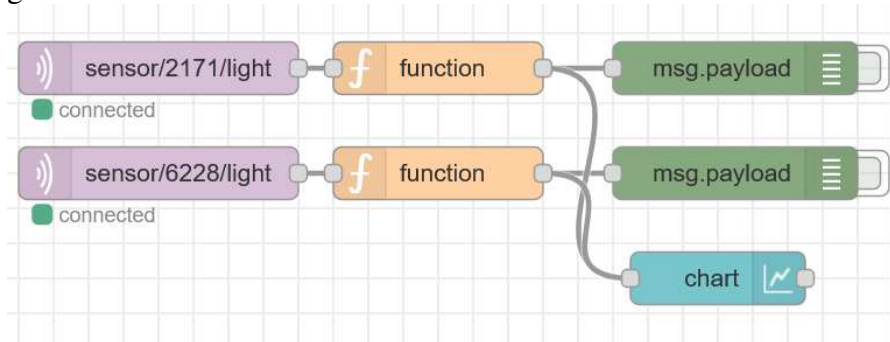


Bezpośrednie podłączenie komponentu „function” z komponentem „char” choć nie jest to zalecane widać tutaj że działa (i pokazuje że można rysować wykresy z różnych źródeł):



Powyższy wykres jest nieco chaotyczny, co więcej nie mamy kontroli nad danymi przetwarzanymi przez zdefiniowany przepływ danych. Takie podejście można uznać za poprawne wyłącznie dla prostych systemów które posiadają obiekt publikujący wiadomości wyłącznie w jednym temacie. W praktyce taka sytuacja jest raczej nie stosowana.

Praktyczniejszym podejściem jest ustalenie tematów w jakich chcemy subskrybować i tylko takie dane dalej przetwarzać. Rysunek poniżej pokazuje przykładowe podejście z przetwarzaniem dwóch hipotetycznych typów wiadomości w tematach odpowiednio: „sensor/2171/light” i „sensor/6228/light”:



Uzyskany w ten sposób wykres mógłby wyglądać tak:



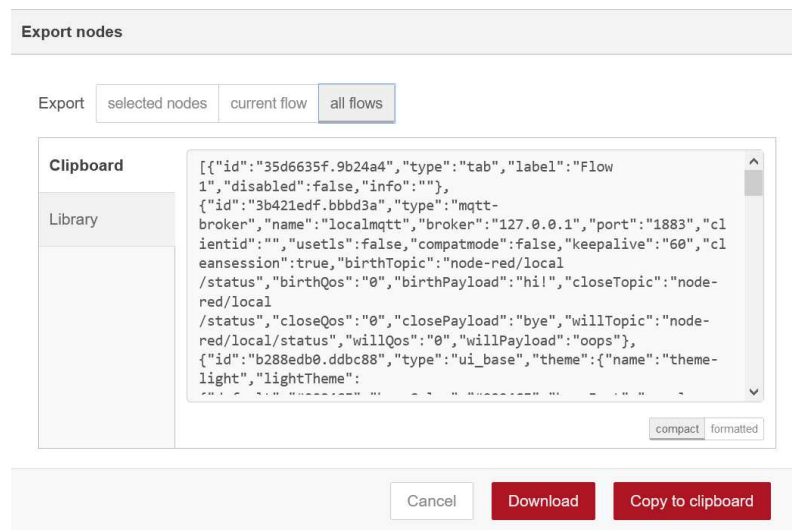
Wykres ten jest ciekawy – lecz nieco zbyt techniczny. Użytkownik widzi w opisach tematy w jakim następowała subskrypcja. Choć twórcy aplikacji opartej na Node-Red mogą taką wizualizację danych uznać za przydatną, w użytkowaniu przez osoby „nie techniczne” wykresy powinny być czytelnie opisane. Użytkownik taki chciałby raczej widzieć czytelne opisy dla każdego wykresu np.: „pokój” i „kuchnia”.



Dodanie odpowiednich etykiety realizuje się w kodzie zapisanym w komponencie function tuż przed zwróceniem wyniku:

```
msg.topic='kuchnia';  
msg.payload=parseInt(val); //zamień tekst na postać liczbowa stałoprzecinkową  
return msg;
```

Po wykonanych czynnościach konfiguracyjnych warto zarchiwizować zbudowane przepływy. Aby to zrobić wybieramy prawe menu (klawisz „☰”) a następnie wybieramy „Export” po czym pojawi się okno „Export nodes”:



Wybierając klawisz „Download” system Node-Red zapyta gdzie umieścić plik flow.json. Plik ten zawierać będzie implementacje przepływu danych.

Literatura:

1. <https://nodered.org>, ostatnia wizyta 2022.10.14
2. <https://flows.nodered.org/node/node-red-dashboard>, ostatnia odsłona 2022.10.14
3. <https://mosquitto.org>, ostatnia odsłona 2022.10.14
4. https://pl.wikipedia.org/wiki/Secure_Shell, ostatnia odsłona 2022.10.14
5. <https://www.putty.org>, ostatnia odsłona 2022.10.14
6. <https://nodered.org/docs/user-guide/messages>, ostatnia odsłona 2022.10.14
7. <https://nodered.org/docs/user-guide/writing-functions>, ostatnia odsłona 2022.10.14
8. <https://pl.wikipedia.org/wiki/JavaScript>, ostatnia odsłona 2022.10.14
9. <http://www.steves-internet-guide.com/node-red-functions>, ostatnia odsłona 2022.10.14