

# OBIEKTY INTERNETU RZECZY

## Laboratorium

Ćwiczenie 3 zdalne

Temat: Współpraca obiektów IoT z aplikacjami w chmurze.

### 1. Pobranie obrazów maszyn wirtualnych

Laboratorium 3 realizowane zdalnie z przedmiotu OBIR, wykonywane jest w głównej części na maszynie wirtualnej przygotowanej przez autorów tego laboratorium. Przed przystąpieniem do pracy nad zadaniem laboratoryjnym trzeba przygotować sobie środowisko pracy. Najważniejsze to zainstalowanie pakietu VirtualBox który jest nadzorcą dla uruchamianej w ramach laboratorium maszyny wirtualnej. Dla poprawnego działania maszyn wirtualnych konieczne jest zainstalowanie: tego nadzorcy w wersji przynajmniej 6.1 lub nowszej, do pobrania ze strony:

<https://www.virtualbox.org/wiki/Downloads>

właściwy link do instalatora to np.:

<https://download.virtualbox.org/virtualbox/6.1.4/VirtualBox-6.1.4-136177-Win.exe>

Proszę także zainstalować VirtualBox Extension Pack w wersji odpowiedniej dla zainstalowanego nadzorcy np.: w wersji 6.1.4 do pobrania z adresu (jedna linia):

[https://download.virtualbox.org/virtualbox/6.1.4/Oracle\\_VM\\_VirtualBox\\_Extension\\_Pack-6.1.4.vbox-extpack](https://download.virtualbox.org/virtualbox/6.1.4/Oracle_VM_VirtualBox_Extension_Pack-6.1.4.vbox-extpack)

Pliki z obrazami maszyny wirtualnej (rozszerzenie OVA) dla VirtualBox'a dostępny jest pod adresem:

[https://secure.tele.pw.edu.pl/~apruszko/obir/obir\\_20201124.ova](https://secure.tele.pw.edu.pl/~apruszko/obir/obir_20201124.ova)

Do pobrania obu plików z podanej lokalizacji należy użyć poświadczeń:

login: obirvm

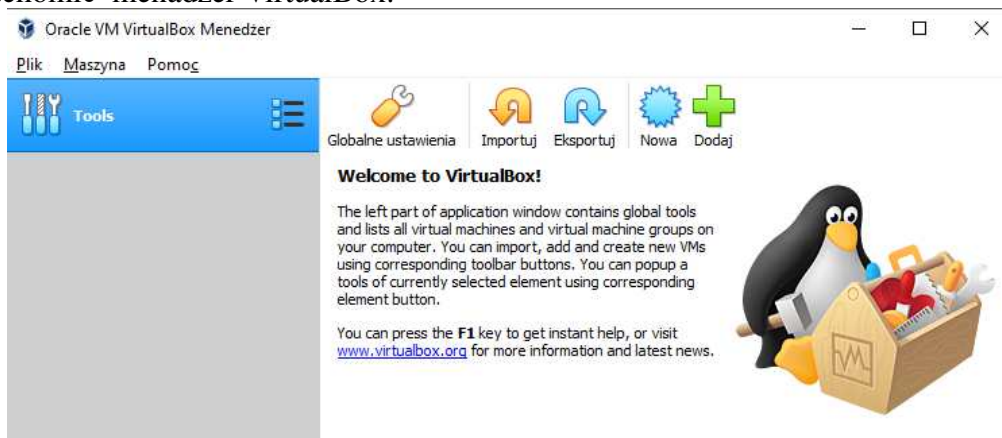
hasło: obir20vm20

### **Uwaga!**

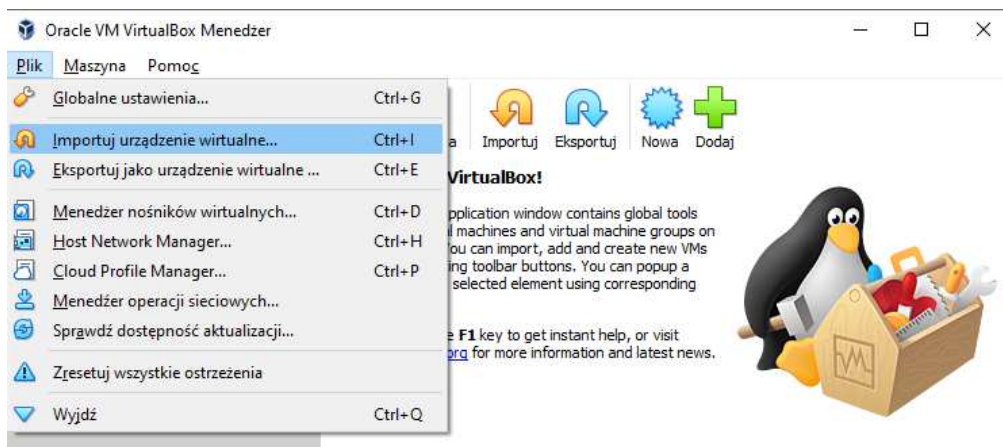
Plik `obir_20201124.ova` „waży” nie cały 1GB, więc jego pobieranie może zająć pewien czas a miejsce do którego zapisujemy ten plik musi posiadać odpowiednią wolną przestrzeń. Po zaimportowaniu maszyna wirtualna zajmie około 3GB.

### 2.Import i ustawienie maszyny wirtualnej

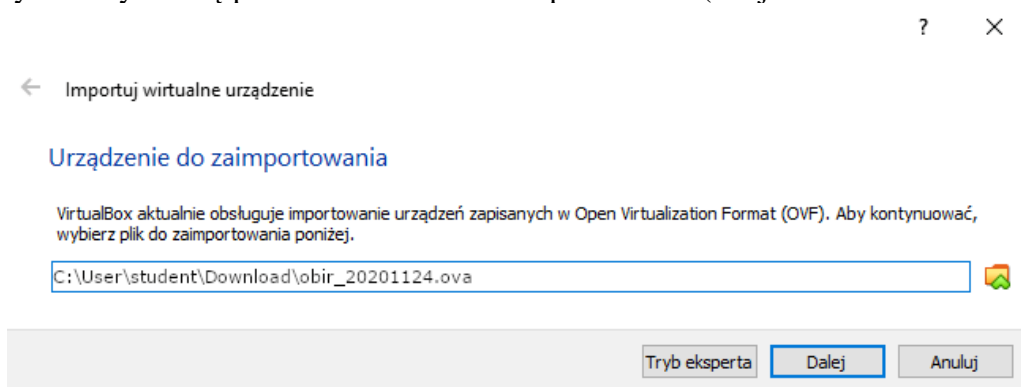
Po pobraniu na dysk lokalny swojego komputera pliku z obrazem maszyny wirtualnej (`obir_20201124.ova`), możliwe jest jej zaimportowanie. Dla zaimportowania maszyny wirtualnej należy uruchomić menadżer VirtualBox:



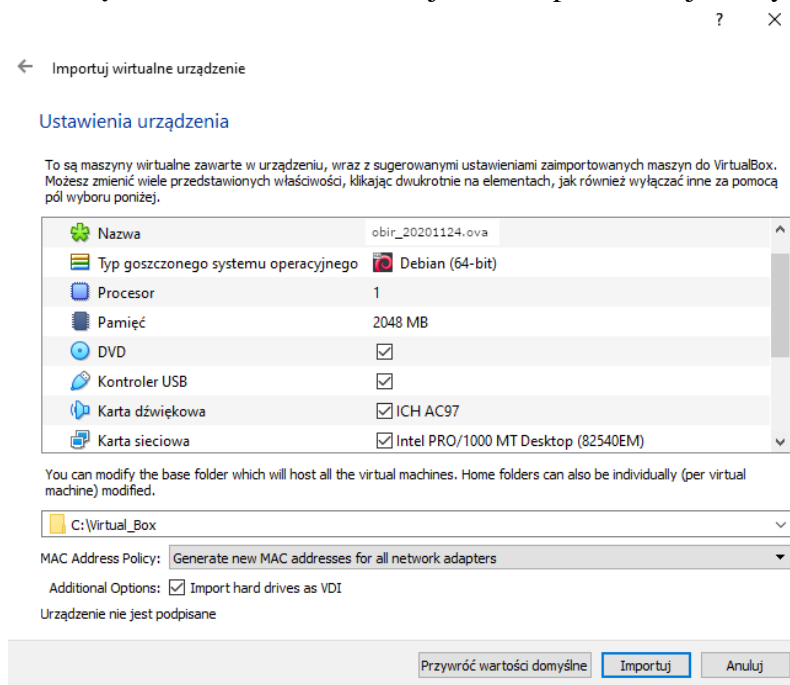
A następnie wybieramy „File” → ”Importuj urządzenie wirtualne”:



po czym wybieramy nazwę pliku z obrazem do zaimportowania (tutaj: obir\_20201124.ova):



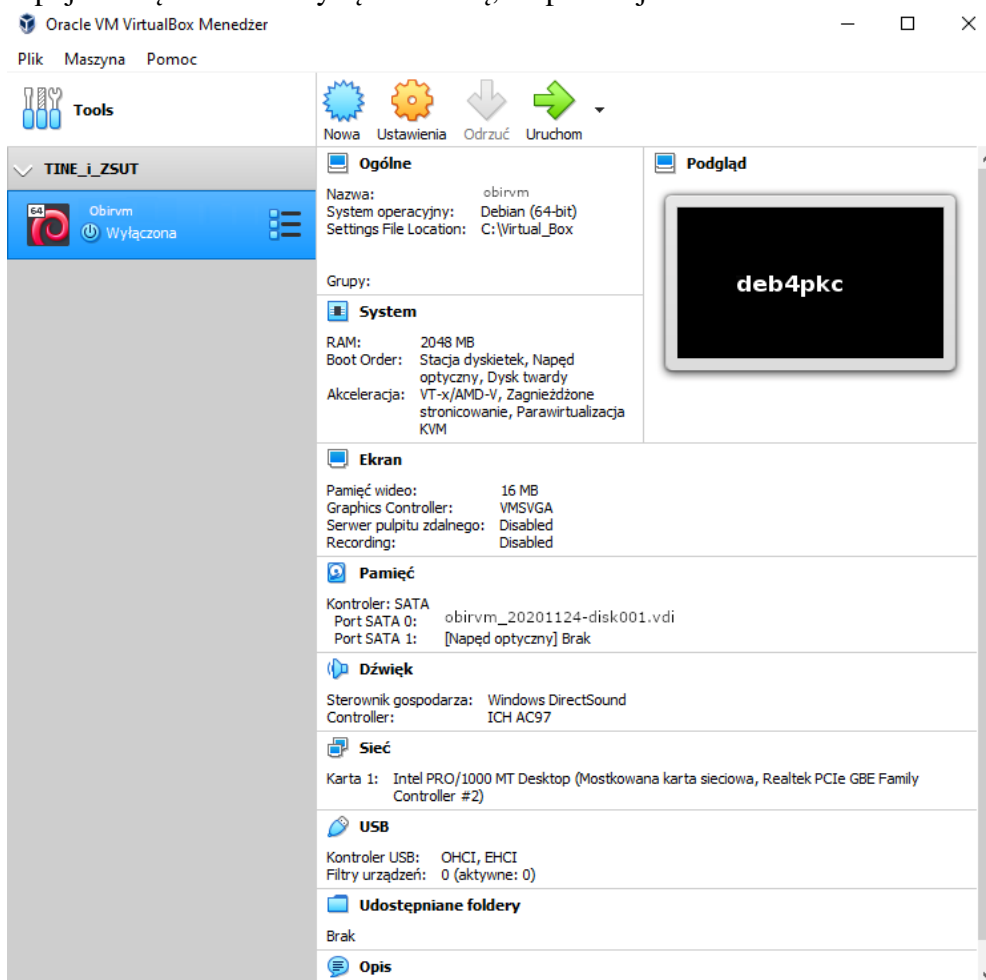
Po zatwierdzeniu ukaże się nam okienko z informacjami o importowanej maszynie wirtualnej:



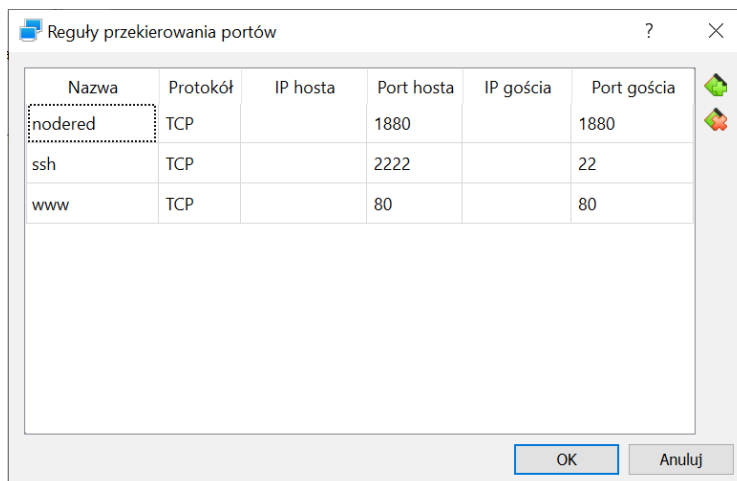
Należy zwrócić uwagę na niektóre parametry, w szczególności – na wielkość pamięci oddanej do dyspozycji systemu wirtualizowanego. Generalnie powinno się ją ustawić na wartość jak najniższą, lecz jeszcze wystarczającą (tutaj: 1024MB), tak, żeby pozostałej pamięci komputera wystarczyło dla innych maszyn wirtualnych i dla macierzystego systemu operacyjnego. Dla pokazanej wyżej

wartości 1024MB, komputer, na którym wykonuje się VirtualBox, musi być wyposażony w pamięć RAM o wielkości co najmniej 2GB (a najlepiej ponad 4GB).

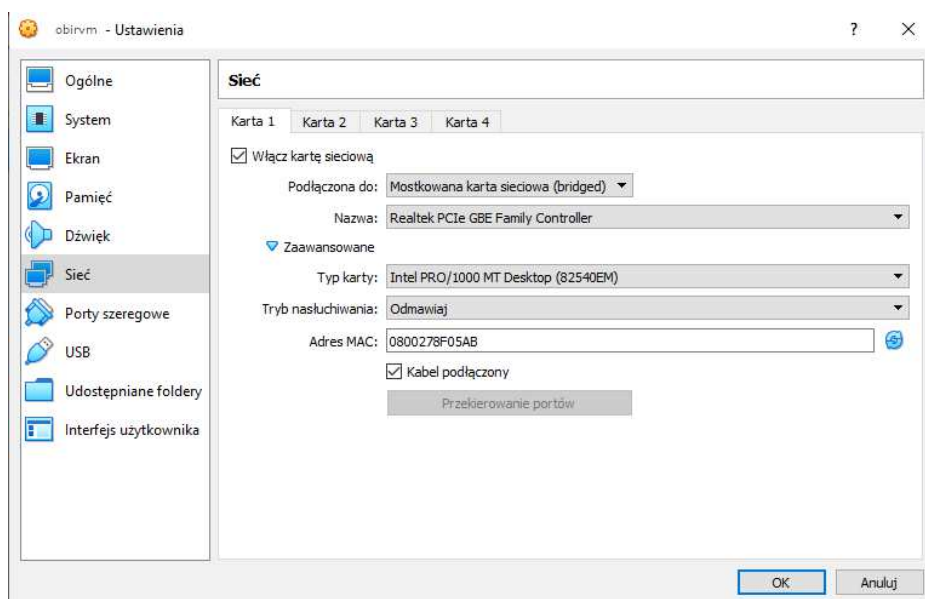
Po dokonaniu ewentualnych modyfikacji możemy przejść do dalszych kroków – poprzez kliknięcie „Importuj”, system przez dłuższą chwilę będzie kopiował dane, po czym pojawi w menadżerze VirtualBox’a pojawi się nowa maszyna wirtualna, co pokazuje obrazek:



Przed uruchomieniem tej maszyny wirtualnej, dokładniejszej uwagi wymaga ustawienie jej wirtualizowanych kart sieciowych. Dla dalszych prac wskazane jest ustawić nowo zaimportowanej maszynie wirtualnej jej kartę sieciową w trybie pracy NAT. Dodatkowo można sprawdzić czy poprawnie ustawione jest przekazywanie ruchu na określonych numerach portów TCP (wybieramy Ustawienia->Sieć->Karta 1->Zaawansowane->Przekierowanie portów):



Po ustawieniu karty sieciowej w trybie pracy NAT należy zmienić zgodnie z przydzielonym sobie zadaniem adresu MAC jest możliwe poprzez kliknięcie w „Zaawansowane” i wpisanie w polu „Adres MAC” wartości zapisanej w formacie HEX zawierającej 12 znaków, co pokazuje rysunek:



Na powyższym rysunku adres MAC to „0800278F05AB”, dzięki takiemu ustawieniu po uruchomieniu maszyny wirtualnej, jej system operacyjny uruchomi kartę sieciową z właśnie takim numerem MAC. Ustawienie adresu MAC jest jednym z kluczowych aspektów tego laboratorium, źle ustawiona wartość może uniemożliwić realizację pozostałych zadań tego laboratorium

Po uruchomieniu maszyny wirtualnej po niedługiej chwili (zależnej od wydajności maszyny wirtualnej) system Linux w oknie o nazwie: „obirvm...Oracle VM VirtualBox” udostępni nam konsolę systemową, logujemy się poprzez tę konsolę do systemu Linux, używając poświadczeń:

login: student

hasło: student77

Po zalogowaniu się do systemu z użyciem powyższych poświadczeń, możemy sprawdzić czy maszyna działa z właściwym numerem MAC. Wykonujemy to przez wydanie z konsoli tekstowej polecenia: „ip a” – co ukazuje rysunek:

```
student@obirvm:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:8f:05:ab brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic enp0s3
        valid_lft 86373sec preferred_lft 86373sec
    inet6 fe80::a00:27ff:fe73:5bff/64 scope link
        valid_lft forever preferred_lft forever
student@obirvm:~$ _
```

W pozycji 2: (urządzenie o nazwie „enp0s3”) widać raport narzędzia „ip” informujący jaki adres sieciowy ma to urządzenie tu 10.0.2.15 a karta sieciowa ma adres MAC: 08:00:27:8F:05:AB. Proszę pamiętać, iż nazwa: „enp0s3”, może w pewnych przypadkach być inna.

### 3. Wyłączenie maszyny wirtualnej

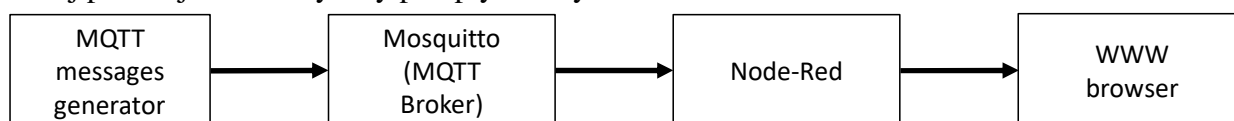
Po zakończeniu prac wysoce zalecane jest wyłączenie maszyny wirtualnej za pomocą polecenia „shutdown”. W przypadku maszyny obirvm uczynić to można najwygodniej przez zalogowanie się do konsoli tekstowej systemu operacyjnego tej maszyny wirtualnej i wydać polecenie (dzięki odpowiedniej konfiguracji mimo, iż użytkownik student ma prawo wykonać poniższe polecenie ze swoimi uprawnieniami):

```
sudo /usr/sbin/shutdown -h now
```

Po wydanym tak poleceniu i niedługiej chwili maszyna wirtualna powinna zakończyć swoje działanie.

### 4. Praca przygotowawcza z systemem Node-Red

Na maszynie wirtualnej obirvm został system Node-Red[1] wraz z dodatkami takimi jak m.in. node-red-dashboard[2]. Dodatkowo na tej maszynie zainstalowano broker MQTT o nazwie mosquitto[3] oraz automatycznie uruchamiany emulator źródła danych sensorycznych. Rysunek poniżej pokazuje schematyczny przepływ danych:



Emulator oznaczono tu jako „MQTT message generator” i publikuje on cyklicznie wiadomości w trzech różnych tematach. Do jego właściwej pracy wymagane jest poprawne ustawienie adresu MAC maszyny obirvm – procedura opisana jest w pkt. 2.

Aby sprawdzić jakie wiadomości generuje opisany emulator należy zalogować się do konsoli maszyny wirtualnej (lub połączyć się z maszyny goszczącej z maszyną wirtualną poprzez protokół SSH[4] np.: PuTTY[5] podając adres 127.0.0.1 i port 2222) i uruchomić z linii poleceń:

```
mosquitto_sub -h 127.0.0.1 -p 1883 -t "#" -v
```

Po maksymalnie 2 minutach powinny nam ukazać się komunikaty podobne do poniższych:

```
...
sensor/6553/temperature {time: 1506290088, type: temperature, value: 36}
sensor/2038/humidity {time: 1506290088, type: humidity, value: 14}
sensor/1032/light {time: 1506290088, type: light, value: 44}
...
```

Jeżeli takowych komunikatów przez dłuższy czas nie widać należy uruchomić z linii poleceń:

```
tail -n20 /run/obir_log
```

które powinno ukazać się nam coś podobnego do treści:

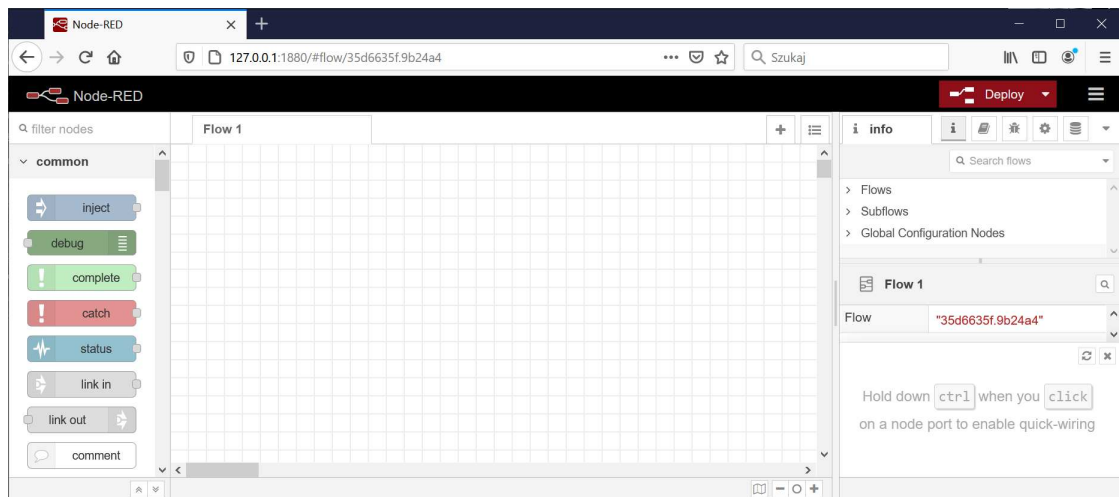
```
...
Obir executable start working at 2020-11-23 07:43:09
Unknown MAC address: 08:00:27:bb:ff:ff
...
```

Każdy z wygenerowanych komunikatów po wydaniu tego polecenia należy przesłać (w trybie awaryjnym, czyli jak najszybciej) do prowadzącego ćwiczenie laboratoryjne – ustali co poszło nie tak. Bez poprawnie działającego „MQTT message generator” nie można kontynuować dalszych prac.

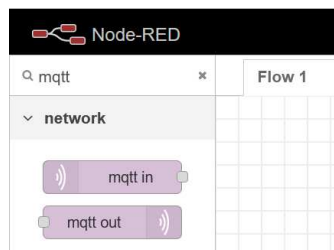
### 5. Tworzenie wizualizacji danych z wykorzystaniem systemu Node-Red

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 rozpyływ danych (tzw: „flow”) i konfigurację użytych w tym przepływie danych komponentów oraz panel wizualizacji (dashboard). Widok nie skonfigurowanego główny 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:



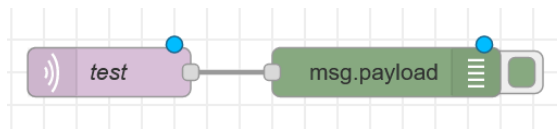
Widać zatem że mamy do dyspozycji dwa komponenty: „mqtt in” i „mqtt out”. Pierwszy będzie subskrybował wiadomości a drugi publikował – w tym ćwiczeniu istotny jest tylko „mqtt in” – który wybieramy, a następnie klikamy na ten komponent i konfigurujemy, tak aby ustawienia były zgodne z poniższymi:


#### Edycja podstawowych parametrów komponentu

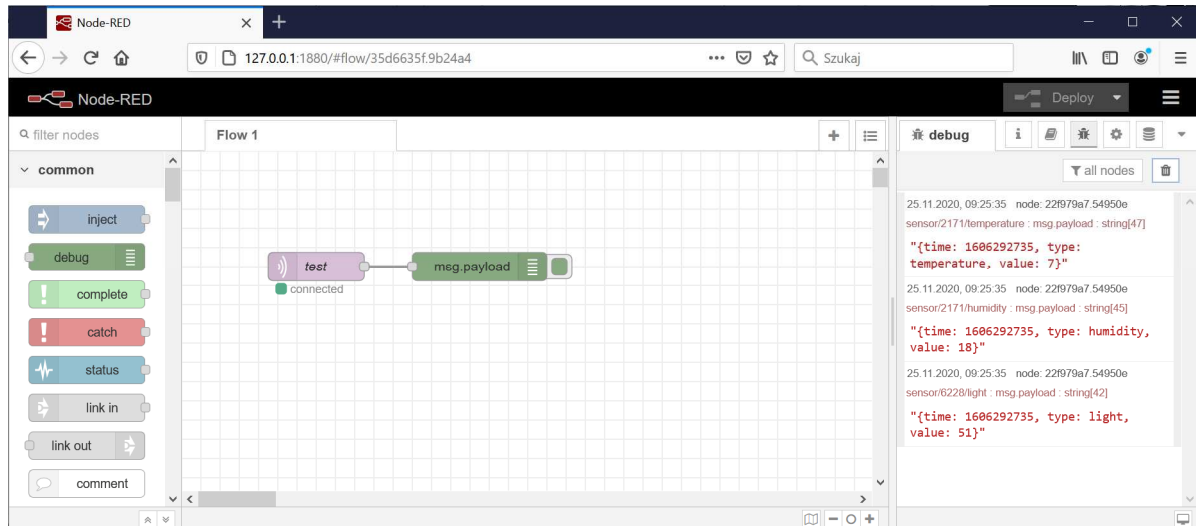
#### Ustawienia aspektów sieciowych komponentu

Proszę zauważyć, że komponent „mqtt in” subskrybuje wszelkie wiadomości – a spodziewamy się przynajmniej trzech tematów wiadomości. Należy zatem w polu „Topic” wpisać właściwy temat subskrypcji a w polu „Name” wpisujemy słowo „test”.


Aby sprawdzić czy użyty komponent działa wybieramy komponent „debug” i łączymy go z „mqtt in” tworząc prosty system:

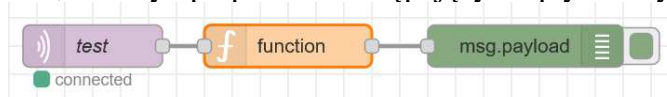


Aby „flow 1” rozpoczął działanie należy kliknąć na klawisz „Deploy” (po wszelkich zmianach zmienia on kolor na czerwony). Od tego momentu o ile nie zostanie zaważony błąd w opisach i konfiguracjach, system uruchomi aplikację wirtualną opisaną przez „flow 1”. Dzięki komponentowi „debug” (który na powyższym rysunku zmienił nazwę na „msg.payload”) można uruchamiać swój „flow”, kliknąwszy na symbol „” (pod klawiszem „Deploy”) i obserwować wygenerowane komunikaty, co pokazuje rysunek:

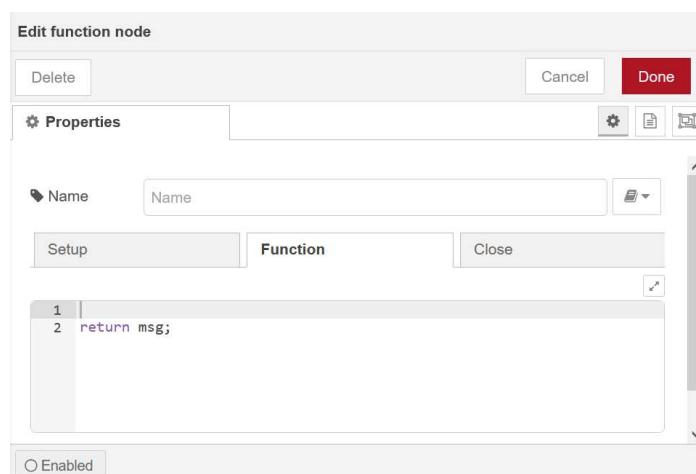


Podczas dłuższej pracy systemu w prawej części okna z uruchomionym Node-Red pokazane zostaną odebrane przez komponent „mqtt in” (tu nazwanym „test”).

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



W domyślnym ustawieniu taki układ komponentów będzie dalej działał poprawnie. Komponent „function” opisany jest w JavaScript i opis jak ma działać komponent „function” można zmienić klikając na ten komponent, a domyślnie ma on zawartość:





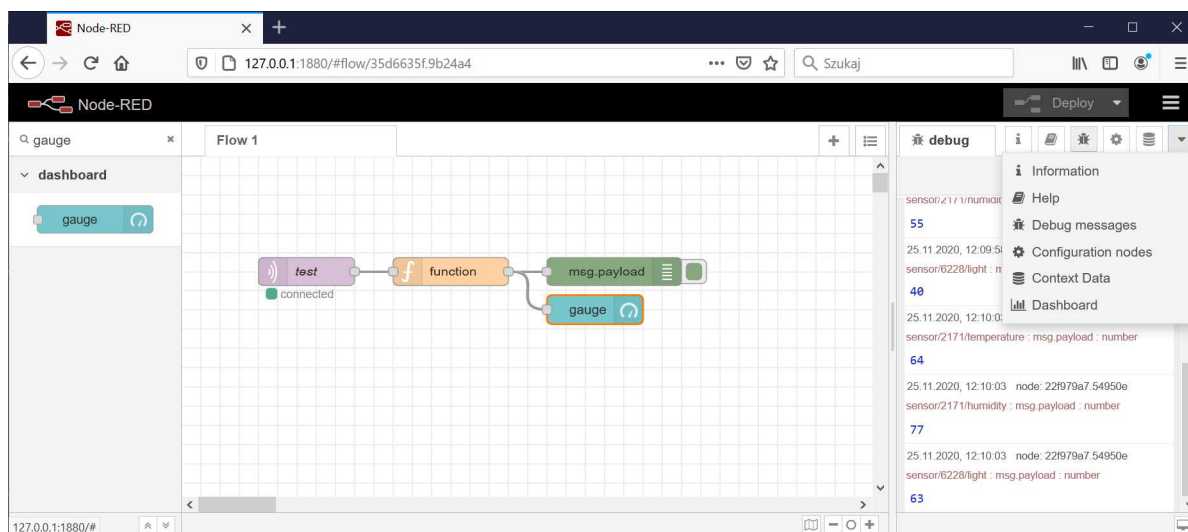
Aby usunąć niechcianą część wiadomości (tu są to nawiasy: { i }) należy - zgodnie ze składnią języka JavaScript wspieraną przez Node-red[9] - poddać ją obróbce, za pomocą kodu np.:

```
var t1=msg.payload.split("{}");//podziel względem znaku {
var t2=t1[1];                //w t2 zapisz część za znakiem { do końca wiersza
var t3=t1.split("{}");       //podziel względem znaku }
msg.payload=t3[0];           //w finalnej wiadomości umieść tylko znaki przed }
return msg;
```

wpisując powyższy kod do okna edycji węzła function. Działanie funkcji split() polega na utworzeniu obiektu wielowierszowego tak aby każdy wiersz odpowiadał polu wiadomości wejściowej rozdzielonej szukanym znakiem. W podobny sposób można pobierać poszczególne wartości, dla pola value, kod mógłby wyglądać następująco:

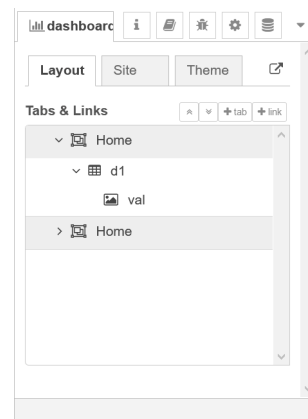
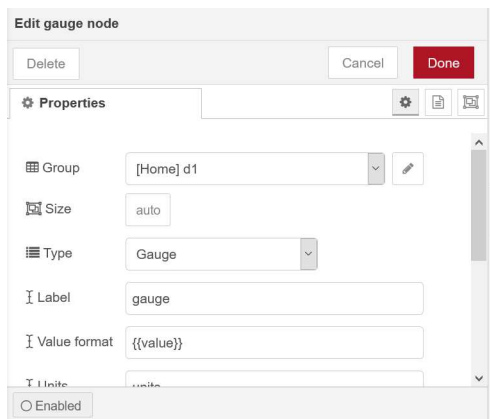
```
var t1=msg.payload.split("{}");//podziel względem znaku {
var t2=t1[1];                //w t2 zapisz część za znakiem { do końca wiersza
var t3=t1.split("{}");       //podziel względem znaku }
var t4=t3[0];               //w finalnej wiadomości umieść tylko znaki przed }
var all_val=t4.split(",");   //podziel treść względem znaku ,
var val=all_val[2].split(":")[1].trim(); //wyciągnij z pozostałego wartość
msg.payload=parseInt(val);   //zamień tekst na postać liczbowa stałoprzecinkową
return msg;
```


Po takiej obróbce wartość można przekazać do wyświetlenia za pomocą tzw. miernika (ang gauge). Aby to było możliwe dodajemy komponent „gauge” i łączymy go z pozostałymi elementami:

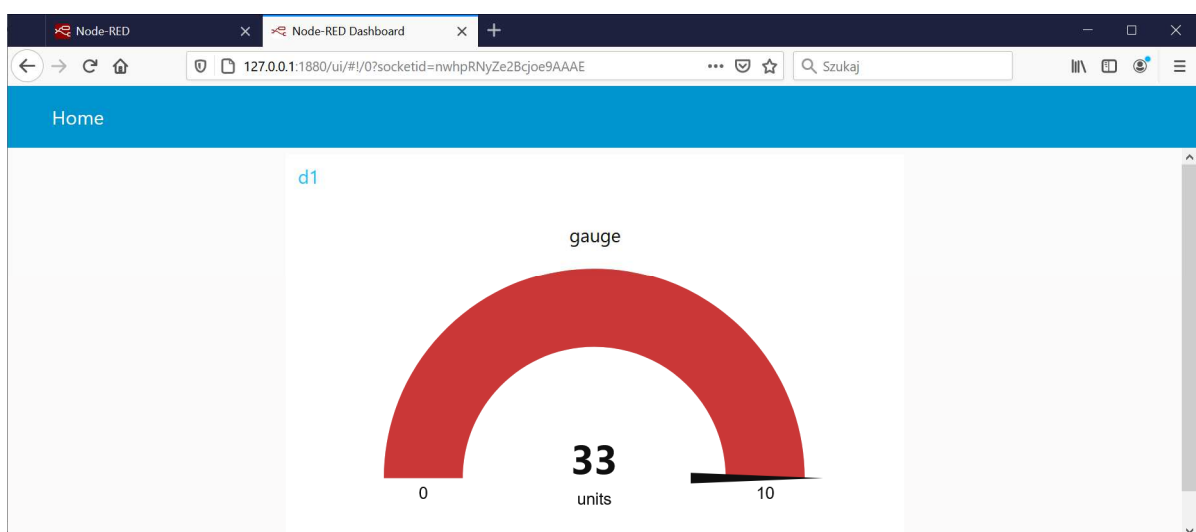


Pozostawiając komponent debug (tu jako msg.payload) możemy nadal obserwować treści przekazywane przez brokera MQTT. Aby jednak obejrzeć właściwy miernik trzeba wybrać panel wizualizacji tzw. dashboard (na powyższym rysunku widać rozwinięte w prawym górnym rogu – zasłania pole wartości diagnostycznych)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

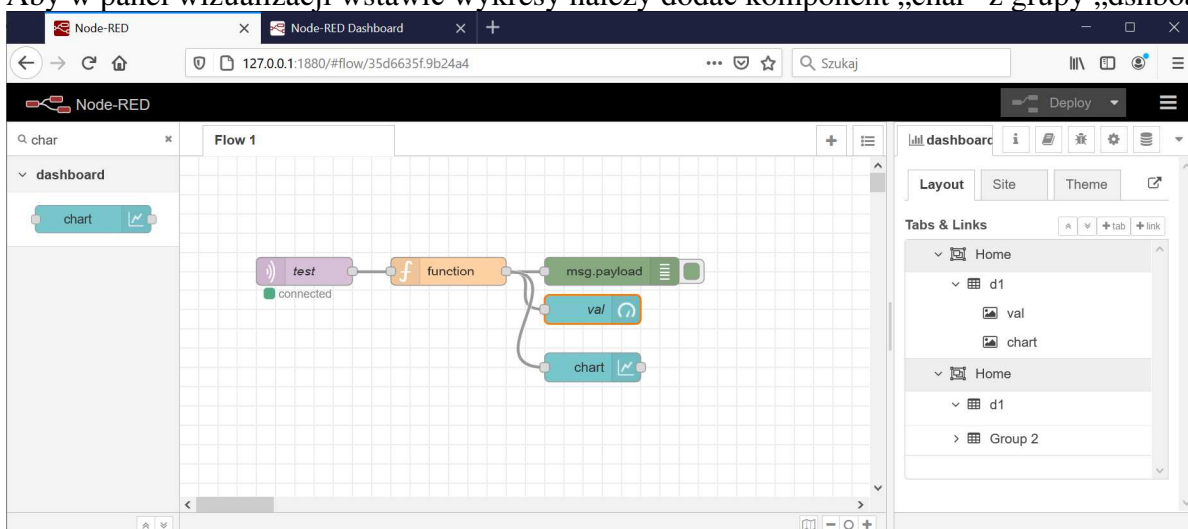




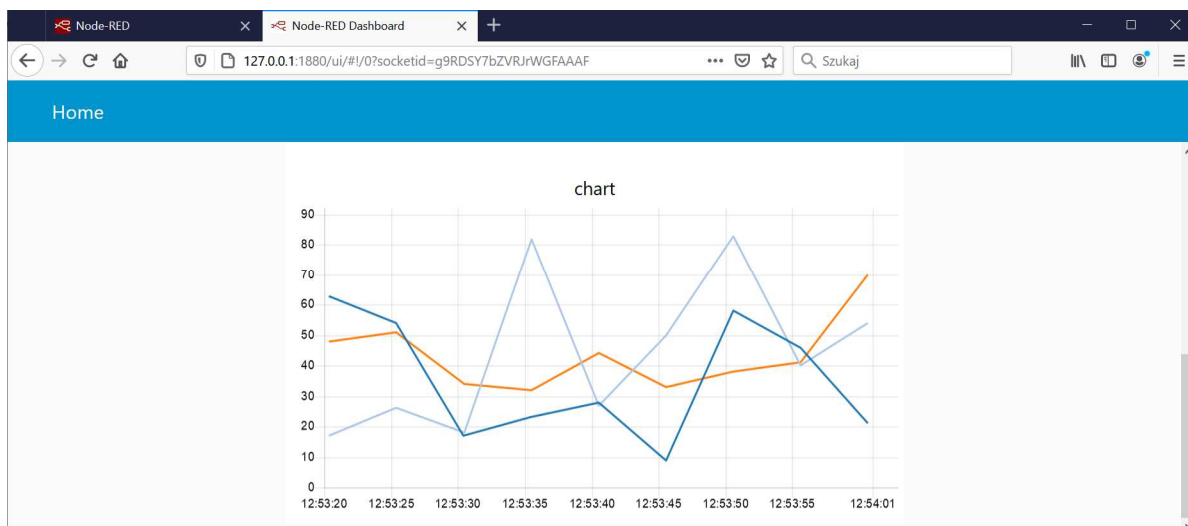
Po kliknięciu w części „Dashboard” na znak: , system otworzy nowe okienko przeglądarki a w nim pokaże miernik:



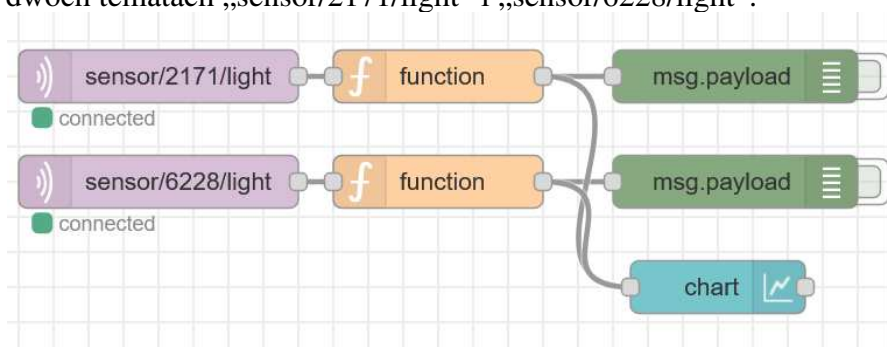
Aby w panel wizualizacji wstawić wykresy należy dodać komponent „char” z grupy „dshboard”:



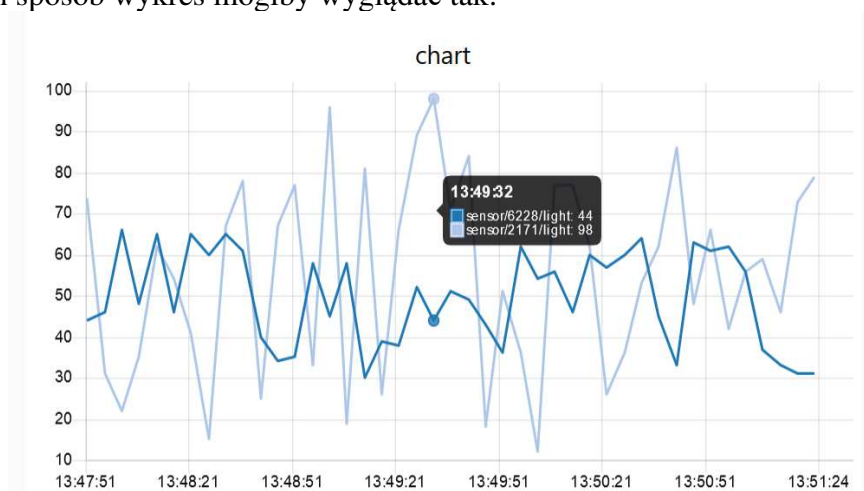
Bezpośrednie podłączenie komponentu „function” z komponentem „char” choć nie jest to zalecane widać tutaj że działa:



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 pod wyłącznie jednym tematem. W praktyce taka sytuacja jest raczej nie spotykana podczas dłuższego działania takiego systemu. Praktyczniejszym podejściem jest ustalenie tematów w jakich chcemy subskrybować i tylko takie dane dalej przetwarzać. Rysunek poniżej pokazuje przykładowe podejście z przetwarzaniem dwóch wiadomości w dwóch tematach „sensor/2171/light” i „sensor/6228/light”:



Uzyskany w ten sposób wykres mógłby wyglądać tak:



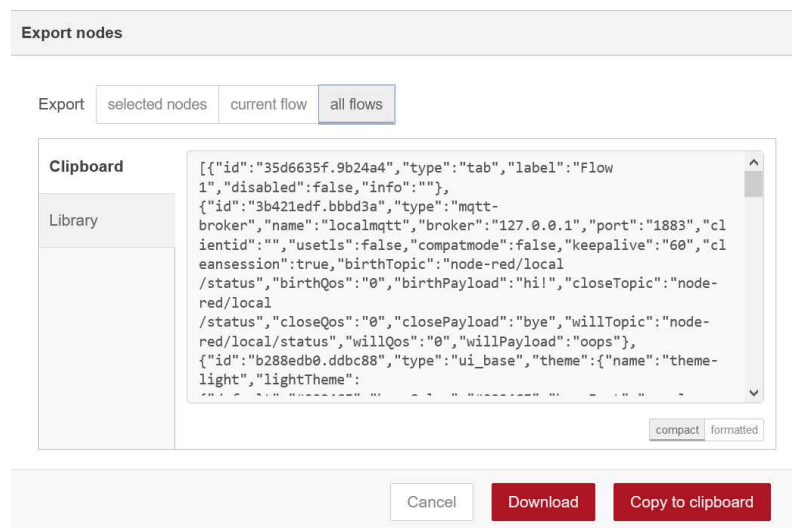
Wykres ten jest ciekawy – lecz nieco zbyt techniczny. Użytkownik widzi w opisach tematy w jakim następowała subskrypcja. Choć twórcy aplikacji opartej na Node-Red te informacje mogą być przydatne, w użytkowaniu przez osoby „nie techniczne” może być uciążliwe. Użytkownik taki chciałby raczej widzieć np.: opisy „pokoj” i „kuchnia”.



Aby system dodawał takie etykiety, należy w kodzie zapisanym w komponencie function przed zwróceniem wyniku zapisać:

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

Po wykonanych czynnościach konfiguracyjnych warto rozważyć archiwizację zbudowanego przepływu danych. Aby to zrobić wybieramy prawe menu (klawisz „☰”) a następnie wybieramy „Export” po czym pojawi się okno „Export nodes”:



Wybierając klawisz „Download” system zapyta gdzie umieścić plik flow.json. Plik ten zawiera opis przepływu danych.

### Literatura:

1. <https://nodered.org>, ostatnia wizyta 2020.11.24
2. <https://flows.nodered.org/node/node-red-dashboard>, ostatnia odsłona 2020.11.24
3. <https://mosquitto.org>, ostatnia odsłona 2020.11.24
4. [https://pl.wikipedia.org/wiki/Secure\\_Shell](https://pl.wikipedia.org/wiki/Secure_Shell), ostatnia odsłona 2020.11.24
5. <https://www.putty.org>, ostatnia odsłona 2020.11.24
6. <https://nodered.org/docs/user-guide/messages>, ostatnia odsłona 2020.11.24
7. <https://nodered.org/docs/user-guide/writing-functions>, ostatnia odsłona 2020.11.24
8. <https://pl.wikipedia.org/wiki/JavaScript>, ostatnia odsłona 2020.11.24
9. <https://www.steves-internet-guide.com/node-red-functions>, ostatnia odsłona 2020.11.24