

Zad.1. W środowisku Arduino IDE należy dodać następujące biblioteki:

- ArduinoJson
- ESP8266HTTPClient
- OneWire.h
- DallasTemperature.h

Zad.2. JSON - deserializacja.

Uruchom dowolną przeglądarkę i w pasku adresu wpisz: <http://time.jsontest.com/>.

```
#include<ESP8266WiFi.h>
#include<ESP8266HTTPClient.h>
#include<ArduinoJson.h>

#define BAUDRATE 115200

const char* ssid = "Laboratorium-IoT";
const char* pass = "Tutaj wpisz hasło dostępowe";

void setup() {
  Serial.begin(BAUDRATE);
  WiFi.begin(ssid, pass);
  Serial.print("Connecting to WiFi ");
  while (WiFi.status() != WL_CONNECTED) {
    delay(250); Serial.print("."); }
  Serial.println("\nConnected to the WiFi network ");
  Serial.print("IP "); Serial.println(WiFi.localIP());
}


WiFiClient client;
HTTPClient http;
long p_millis = 0;
#define DELAY 5000


void loop() {
  if(millis() - p_millis > DELAY) {
    http.useHTTP10(true);
    http.begin(client,"http://time.jsontest.com");
    int httpStatusCode = http.GET();
    Serial.print("Code: ");
    Serial.println(httpStatusCode);
```

```
if(httpResponseCode > 0) {  
    JsonDocument doc;  
    DeserializationError error = deserializeJson(doc,  
                                                http.getStream());  
  
    if (error) {  
        Serial.print(F("deserializeJson() failed: "));  
        Serial.println(error.c_str());  
        return;  
    }  
    const char* date = doc["date"];    // "03-24-2023"  
    Serial.print("date: ");  
    Serial.println(date);  
    long long milliseconds_since_epoch =  
        doc["milliseconds_since_epoch"];  
    const char* time = doc["time"];    // "10:24:19 PM"  
    Serial.print("time: ");  
    Serial.println(time);  
}  
http.end();  
p_millis = millis();  
}  
}
```

Zaznaczony pogrubioną czcionką fragment kodu można wygenerować za pomocą asystenta JSON na stronie <https://arduinojson.org/v7/assistant/#/step1>. Parametry dla asystena są następujące:

Step 1: Configuration

Board	Espressif ESP8266 ESP-12E 32-bit 80KB	<div>Our sponsors</div> 
Mode	<input checked="" type="radio"/> Deserialize <input type="radio"/> Serialize	
Input	Stream	

 This is the Assistant for ArduinoJson 7.0.x. Make sure the same version is installed on your computer.

Next: JSON

Po wprowadzeniu ustawień należy nacisnąć przycisk *Next JSON*.

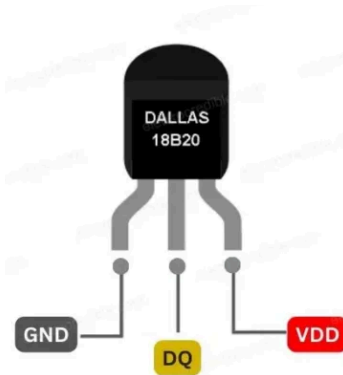
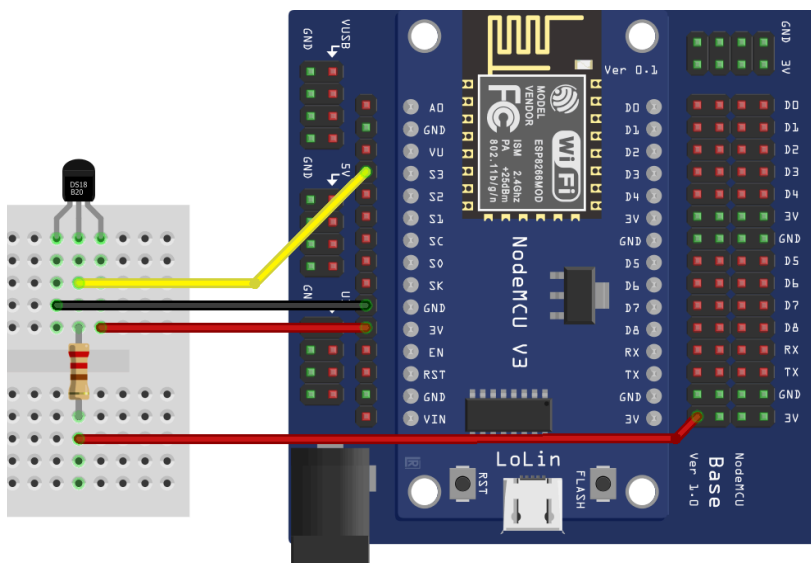
W drugim kroku należy wprowadzić przykładowe dane ze strony: <http://time.jsontest.com/>.



Po wprowadzeniu danych należy nacisnąć przycisk *Next Program*. Przygotowany program można skopiować, do wybranego narzędzia programistycznego, np. Arduino IDE i uzupełnić dodatkowymi liniami kodu, jak w przykładzie.

Zad.3. JSON - serializacja.

Połączyć układ jak na rysunku.



Przykład kodu:

```
#include<ESP8266WiFi.h>
#include<ArduinoJson.h>
#include <OneWire.h>
#include <DallasTemperature.h>

#define BAUDRATE 115200

const int oneWireBus = 10;
OneWire oneWire(oneWireBus);
DallasTemperature sensors(&oneWire);

void setup() {
  Serial.begin(BAUDRATE);
  sensors.begin();
}

long p_millis = 0;
#define DELAY 5000

JsonDocument doc;

void loop() {
  doc["name"] = "DS18B20";
  doc["temperature"] = 0.0;
  doc["unit"] = "deg C";

  if(millis() - p_millis > DELAY) {
    sensors.requestTemperatures();
    float temperatureC = sensors.getTempCByIndex(0);
    float temperatureF = sensors.getTempFByIndex(0);
    // Serial.print(temperatureC);
    // Serial.println("°C");
    // Serial.print(temperatureF);
    // Serial.println("°F");

    doc["temperature"] = temperatureC;
    serializeJson(doc, Serial);
    Serial.print("\n");
    p_millis = millis();
  }
}
```

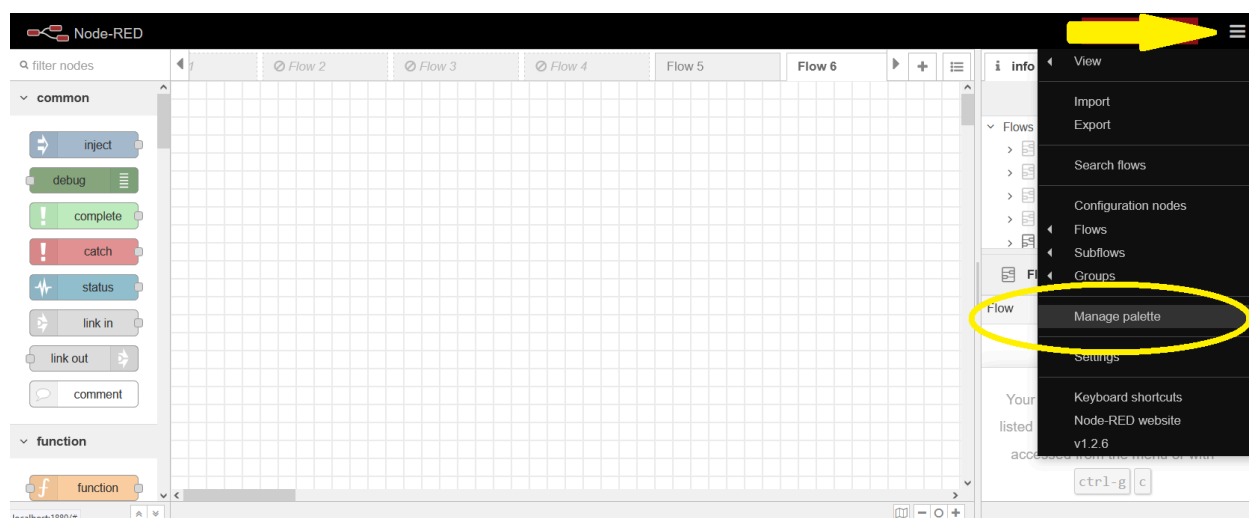
W celu sprawdzenia działania kodu można uruchomić *Moniator portu szeregowego* w środowisku Arduino IDE.

Zad.4. Środowisko Node-RED.

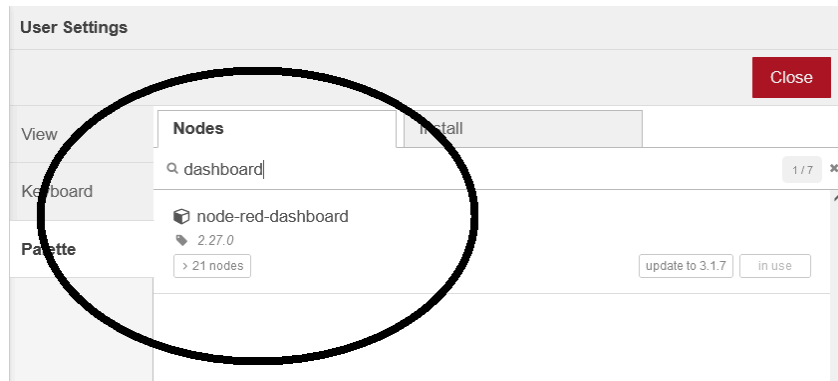
Node-RED jest narzędziem programistycznym przeznaczonym do powiązania urządzeń technicznych z interfejsem użytkownika i usługami sieciowymi. W celu instalacji środowiska Node-RED należy zapoznać się z instrukcją dostępną na stronie <https://nodered.org/docs/getting-started/local>.

Po instalacji, środowisko Node-RED można uruchomić w dowolnej przeglądarce stron www, wpisując w pasku adresu *localhost:1880*. Zbudowany interfejs użytkownika jest widoczny pod adresem *localhost:1880/ui*.

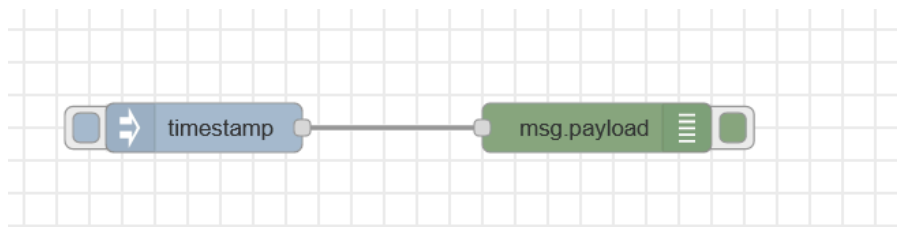
Przed przystąpieniem do pracy należy zainstalować potrzebne biblioteki - pakiety oprogramowania. Służy do tego funkcja Manage palette.



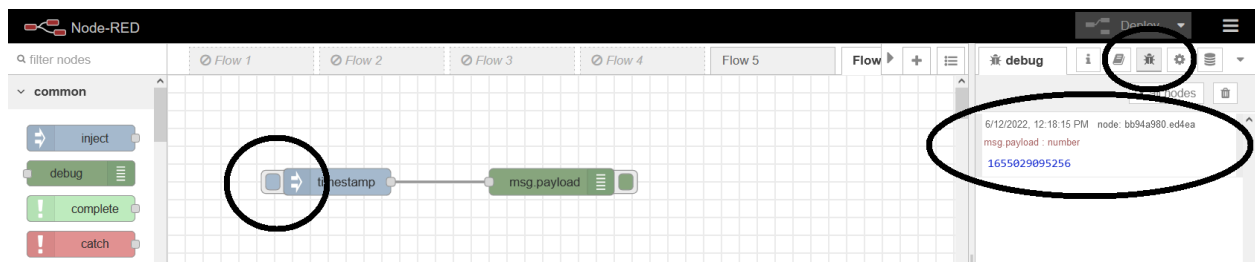
Zainstaluj *dashboard* i *node-red-node-serialport*.



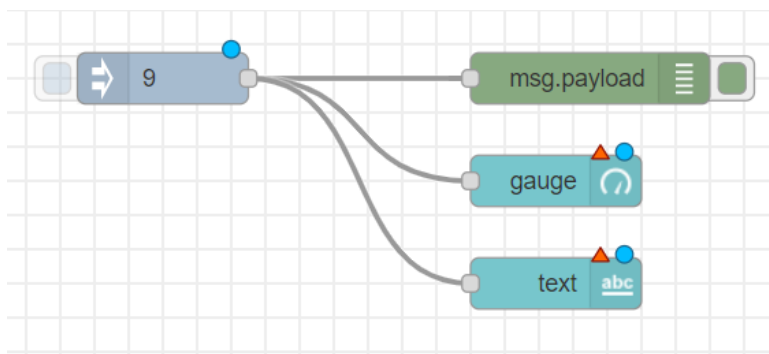
W celu sprawdzenia poprawności instalacji przygotuj diagram przepływów, jak na rysunku. Wszystkie potrzebne bloki (*nodes*), czyli *Inject* oraz *Debug*, są dostępne w paletce *Common*.



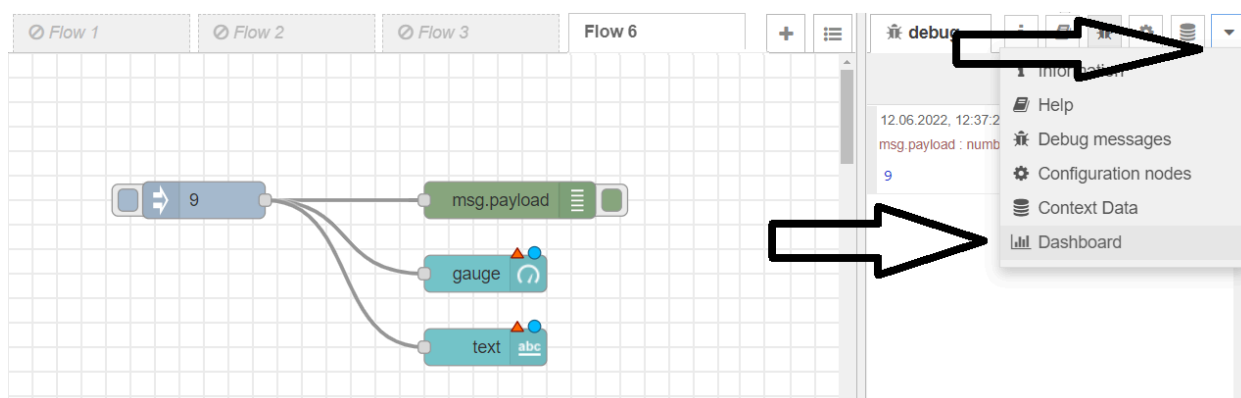
Wykonaj test działania korzystając z *Debug messages*.



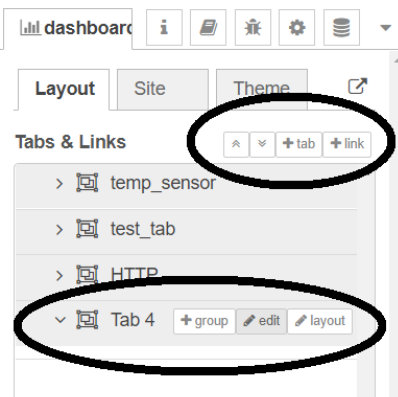
Dwukrotnie kliknij, lewym przyciskiem myszy, na bloku *Inject*. Zmień zawartość pola *msg.payload* z *timestamp* na *number*. Ustaw dowolną wartość liczbową. Dodaj bloki *gauge* i *text* z palety *dashboard*.



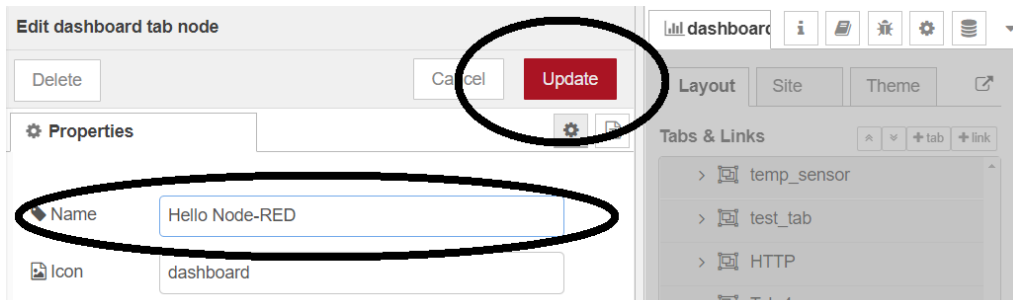
Naciśnij przycisk *Down* i wybierz, z listy rozwijalnej, *Dashboard*.



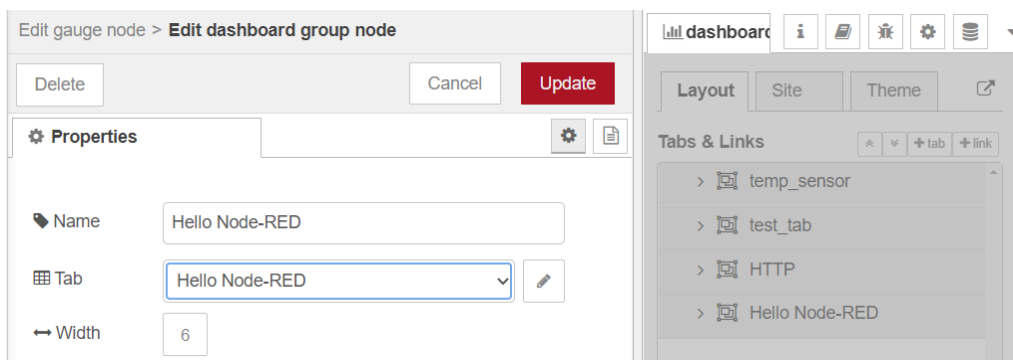
Wbierz zakładkę *Layout*. Naciśnij przycisk *+tab* w celu dodanie nowej zakładki. Zmiana nazwy jest możliwa po naciśnięciu przycisku *edit*.



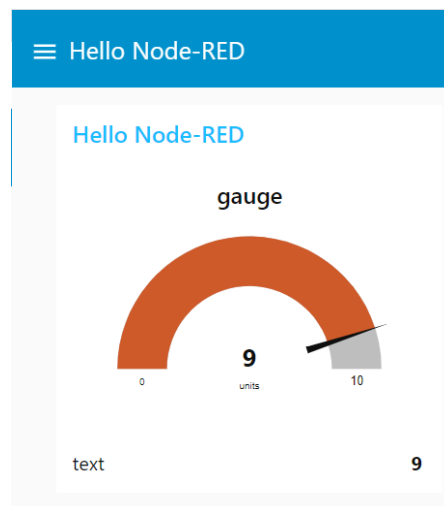
Wprowadź nazwę zakładki i naciśnij przycisk *Update*.



Dwukrotnie kliknij lewnym przyciskiem na bloku *gauge*. Edytuj grupę (*Group*) i wskaż zakładkę na liście rozwijalnej *Tab*. Zatwierdź zmiany za pomocą przycisku *Update*.



Powtórz operację dla bloku *Text*. Zatwierdź wprowadzone zmiany za pomocą przycisku *Deploy*. Wprowadź wartość liczbową do diagramy za pomocą bloku *Inject*. Sprawdź wyniki w zakładce *Debug messages*. Otwórz nową zakładkę w przeglądarce i wprowadź, w pasku adresu, *localhost:1880/ui*. Powinien zostać wyświetlony interfejs użytkownika.



Prześlij poniższy program do pamięci programu płytki NodeMCU.

```
#define BAUDRATE 115200

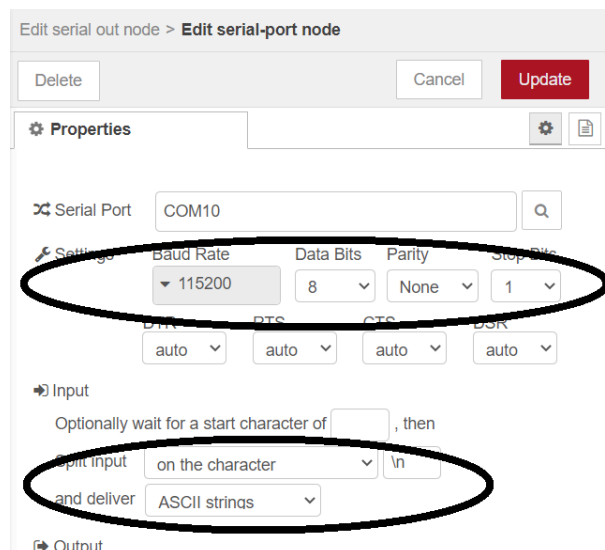
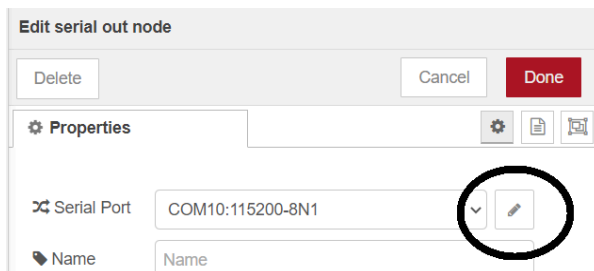
unsigned long p_millis = 0;

void setup() {
  Serial.begin(BAUDRATE); }

#define SERIAL_DELAY 1000

void loop() {
  if(millis() - p_millis > SERIAL_DELAY) {
    Serial.println(int(random(1,9)));
    p_millis = millis(); }
}
```

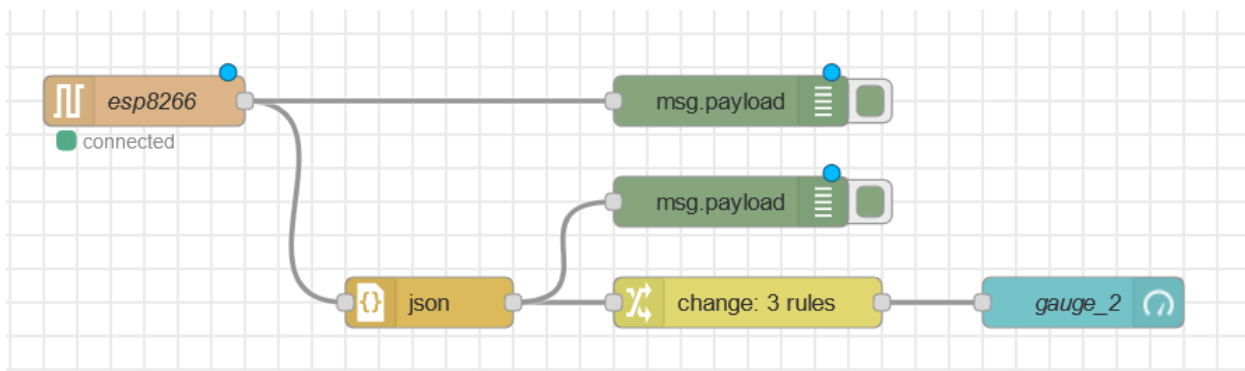
Zamień blok *inject* na blok *serial-in node* (z palety *network*). Skonfiguruj parametry komunikacji, zgodnie z ustawieniami pokazanymi na poniższych rysunkach.



Po zatwierdzeniu przyciskiem Deploy sprawdź wyniki w oknie *Debug messages* oraz na interfejsie (*localhost:1880/ui*).

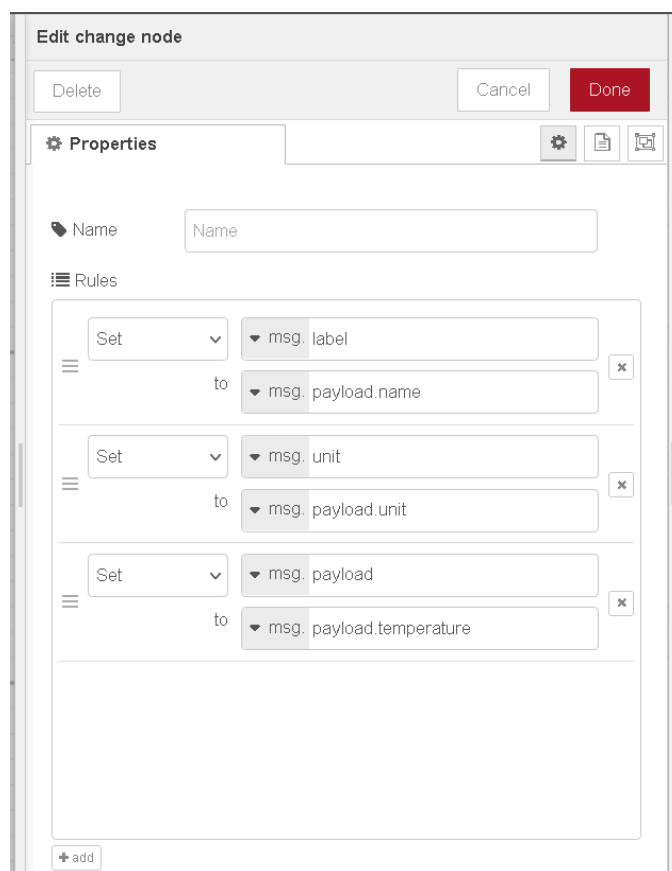
Zad.5. Node-RED - przesyłanie danych w formacie JSON.

Zbuduj poniższy diagram przepływów w środowisku Node-RED.



Wskazówki:

- blok *json* znajduje się na palecie *sequence*;
- blok *change* znajduje się na palecie *function*;
- konfiguracja bloku *json* została pokazana na poniższym rysunku.



- nową regułę można dodać za pomocą przycisku *+add*;

- konfiguracja bloku *gauge* została pokazana na poniższym rysunku - wartość pola *Group* zależy od wartości ustawionych w zadaniu 4.

Edit gauge node

Delete Cancel Done

Properties

Group [Number] BEC_classes

Size auto

Type Gauge

Label {{msg.label}}

Value format {{value}}

Units {{msg.unit}}

Range min 0 max 300

Colour gradient

Sectors 0 ... optional ... optional ... 300

Class Optional CSS class name(s) for widget

Name gauge_2

Enabled

Zad.6. Zadania do wykonania:

- (4.0) wykonaj i prześlij zadania od 1 do 6 - kody Arduino, pliki JSON (polecenie *Export*) diagramów wykonanych na platformie Node-RED;
- (4.5) w zadaniu 5 dodaj wykres obrazujący zmianę temperatury w czasie;
- (5.0) w zadaniu 5 dodaj przycisk, który wyśle do wskazanego użytkownika wiadomość email, jeżeli temperatura przekroczy zadaną na panelu użytkownika wartość.

Dla zainteresowanych:

1. Strona biblioteki ArduinoJson:
arduinojson.org/
2. Random Nerds Tutorial - czujnik DS18B20:
randomnerdtutorials.com/esp8266-ds18b20-temperature-sensor-web-server-with-arduino-ide/
3. Strona projektu Node-RED:
nodered.org
4. Programowanie w Node-RED:
noderedguide.com/nr-lecture-1
5. Kurs Node-RED (użytkownik ArturHome):
www.youtube.com/watch?v=9fTMpgr3EU0