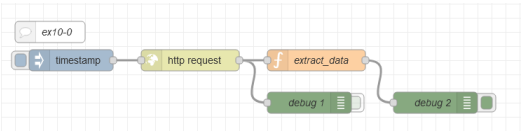
物聯網實務第十一周作業

電機四乙10828241 陳大荃

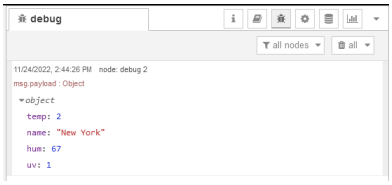
November 25, 2022

Exercise 10-0 Request and separate data from https://weatherstack.com/.

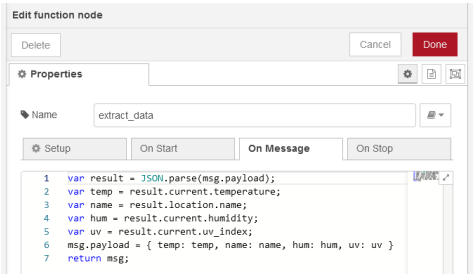
Using API from `https://weatherstack.com/` with the specified city, New York. If there is space between the city name, add "%20" to replace it. Then extract the name, temperature, humidity, and UV numbers from returned data.



(a) flow



(b) debug



(c) node: extract data

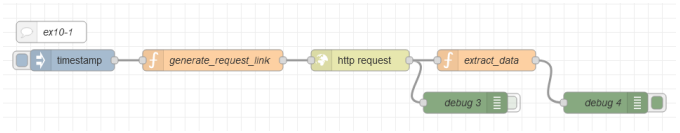
(d) node: http request

Figure 1: Exercise 10-0.

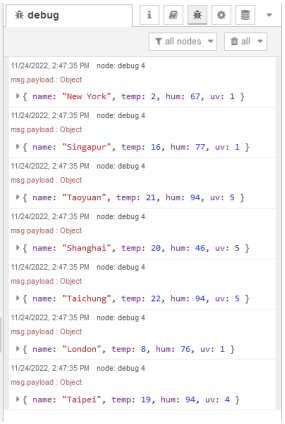
Exercise 10-1 Check seven cities' weather.

Instead of manually changing the API request link seven times, seven API request links are generated accordingly by a for loop in node "generate_request_link". These links are sent as streamline messages and processed one by one. The processing steps are the same with exercise 10-0.

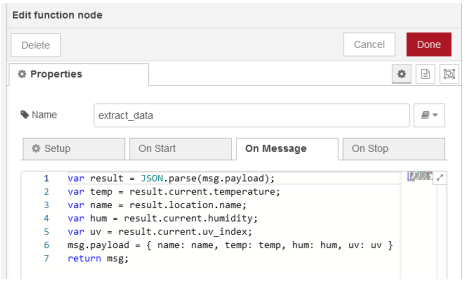
Note: node "http request" is in its default settings.



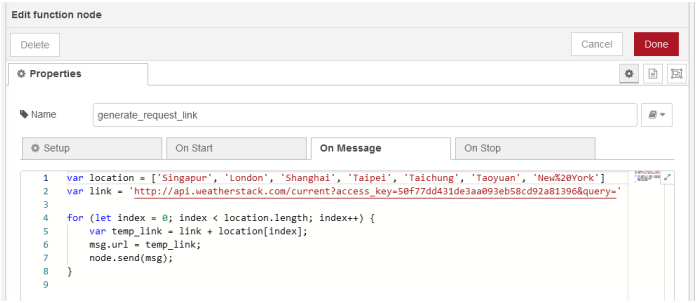
(a) flow



(b) debug



(c) node: extract data



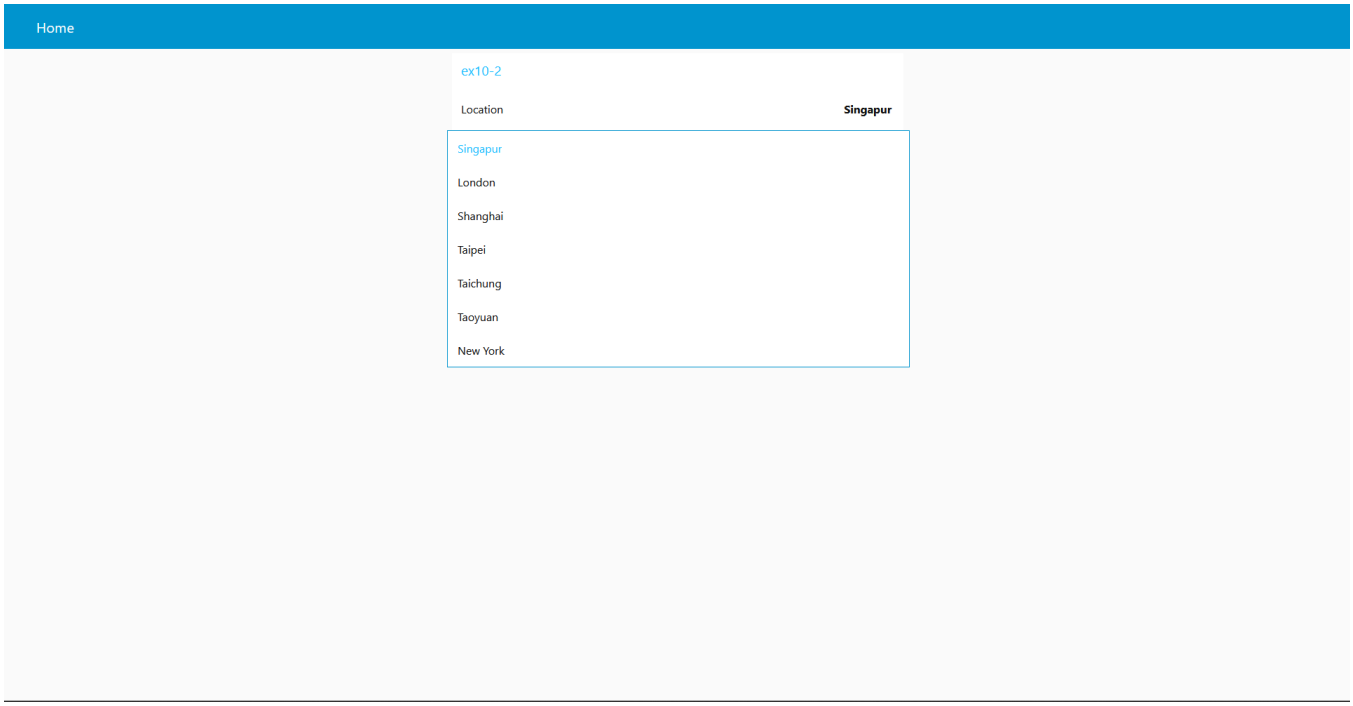
(d) node: generate request link

Figure 2: Exercise 10-1.

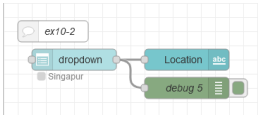
Exercise 10-2 Using dropdown to make an interface for city selection.

Generate the 7 cities' names as options in a dropdown menu by manually specifying their keys and values.

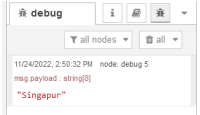
Note: node "dropdown" is manually filled with options.



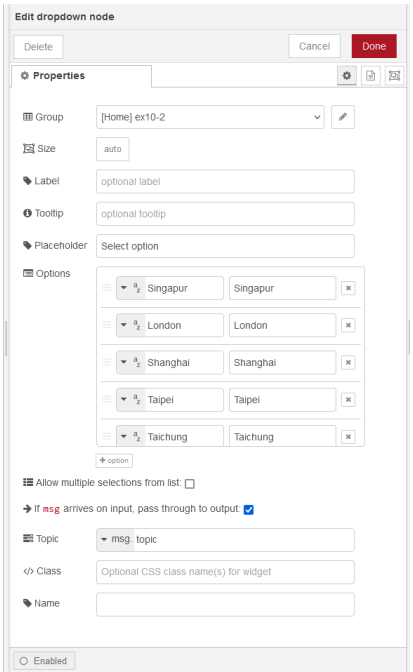
(a) dashboard



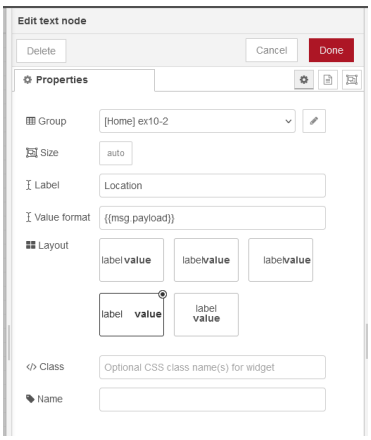
(b) flow



(c) debug



(d) node: dropdown



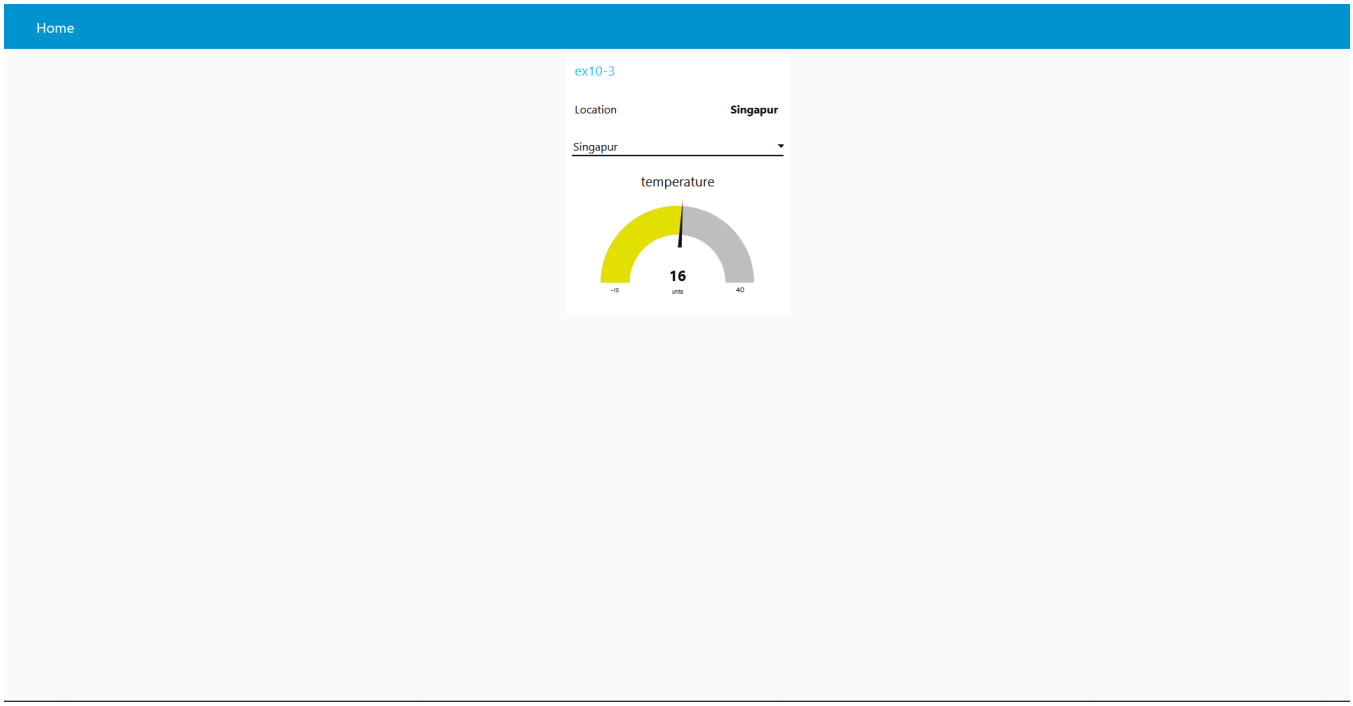
(e) node: location

Figure 3: Exercise 10-2.

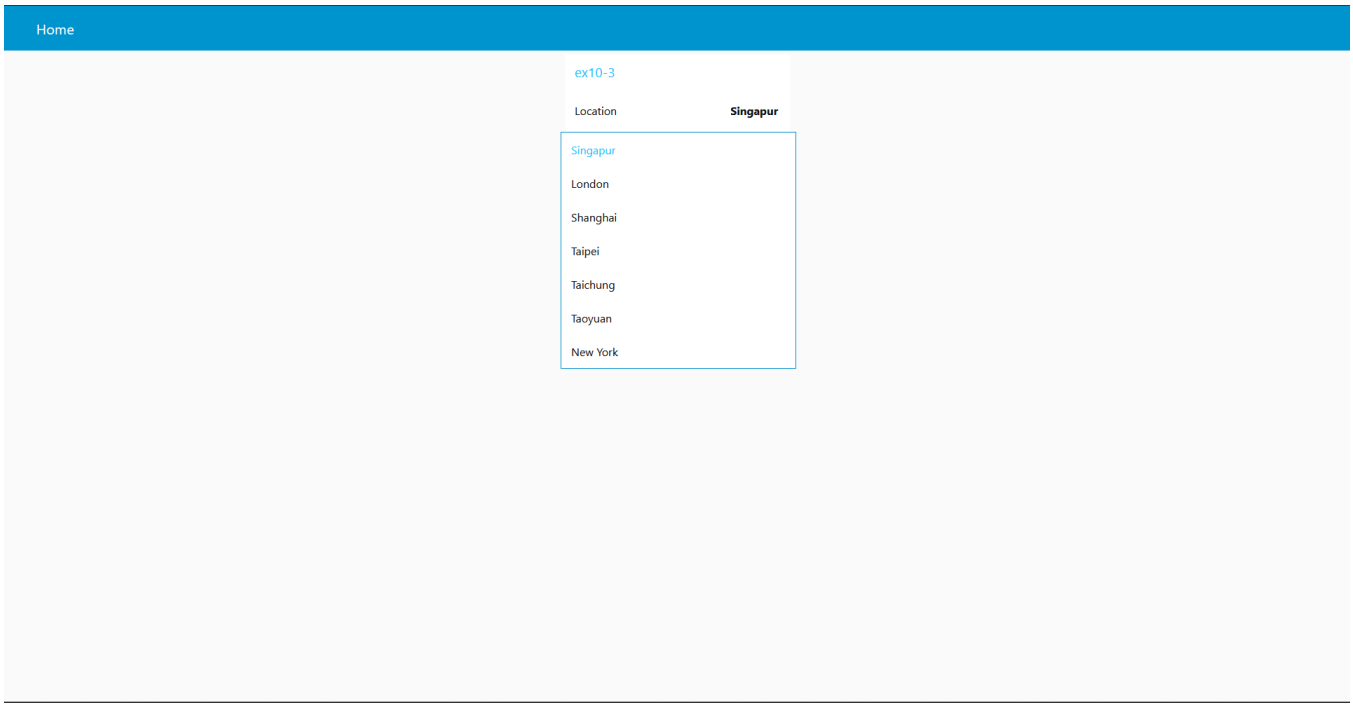
Exercise 10-3 Show the weather of the selected city on the dashboard.

Based on the dropdown menu in exercise 10-2, with auto-generated request link based on the option selected to request and process data accordingly.

Note: node "http request" is in its default settings.
Note: node "dropdown" is manually filled with options. It has the exact same settings in exercise 10-2.

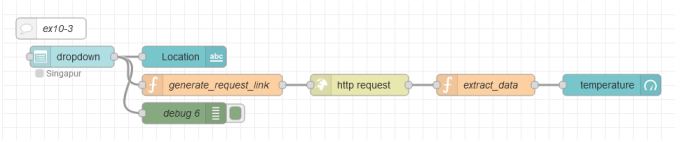


(a) dashboard

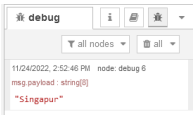


(b) dashboard options

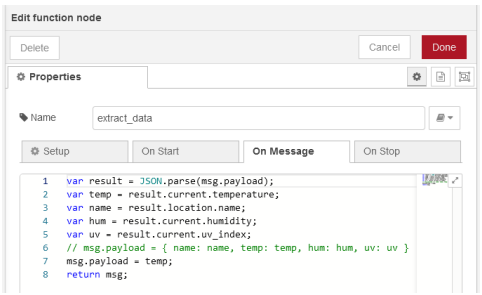
Figure 4: Exercise 10-3-1.



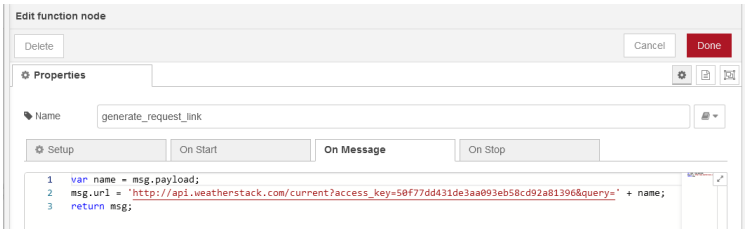
(a) flow



(b) debug

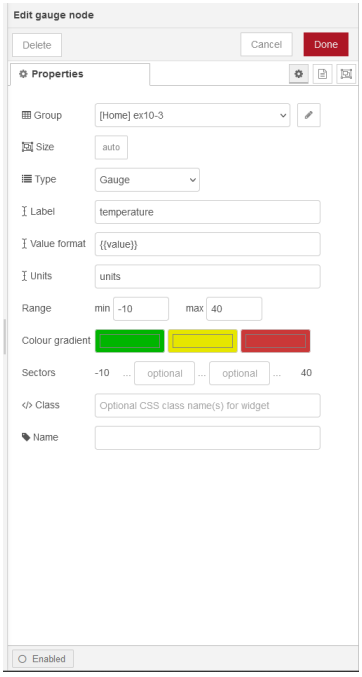


(c) node: extract data



(d) node: generate request link

(e) node: location



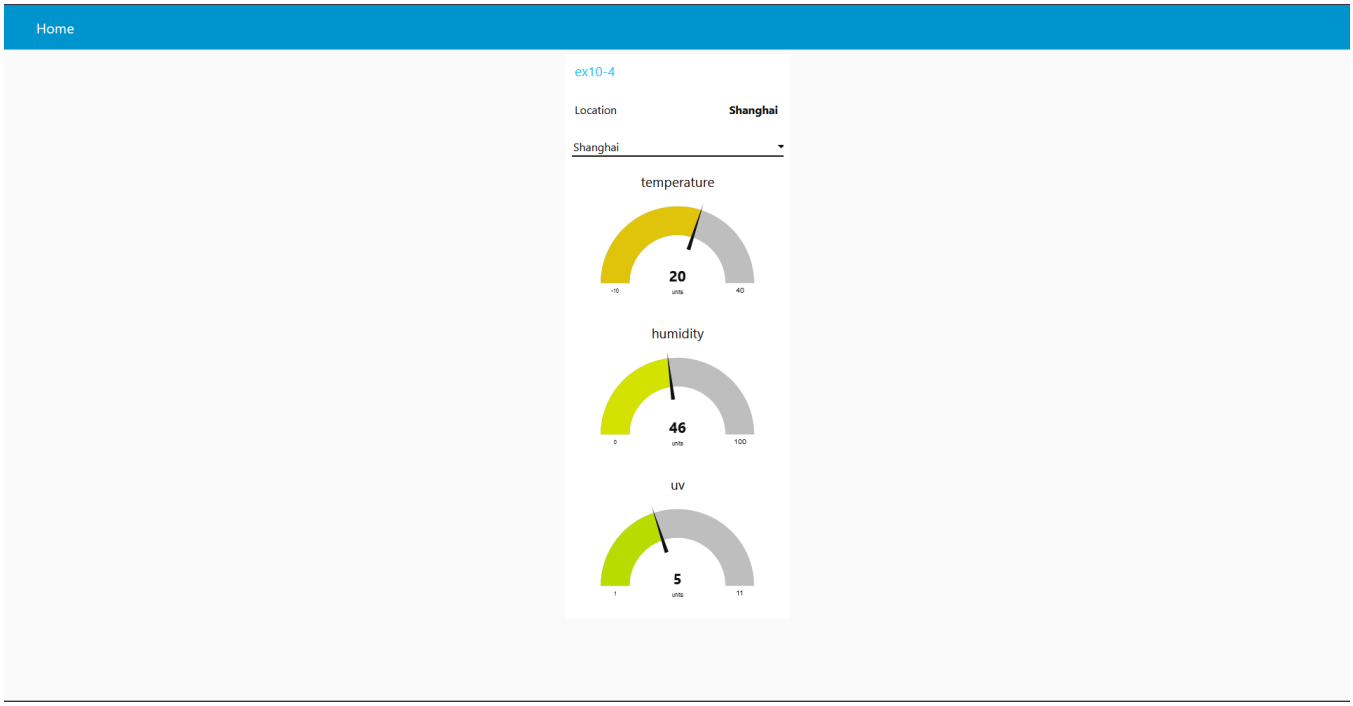
(f) node: temperature

Figure 5: Exercise 10-3-2.

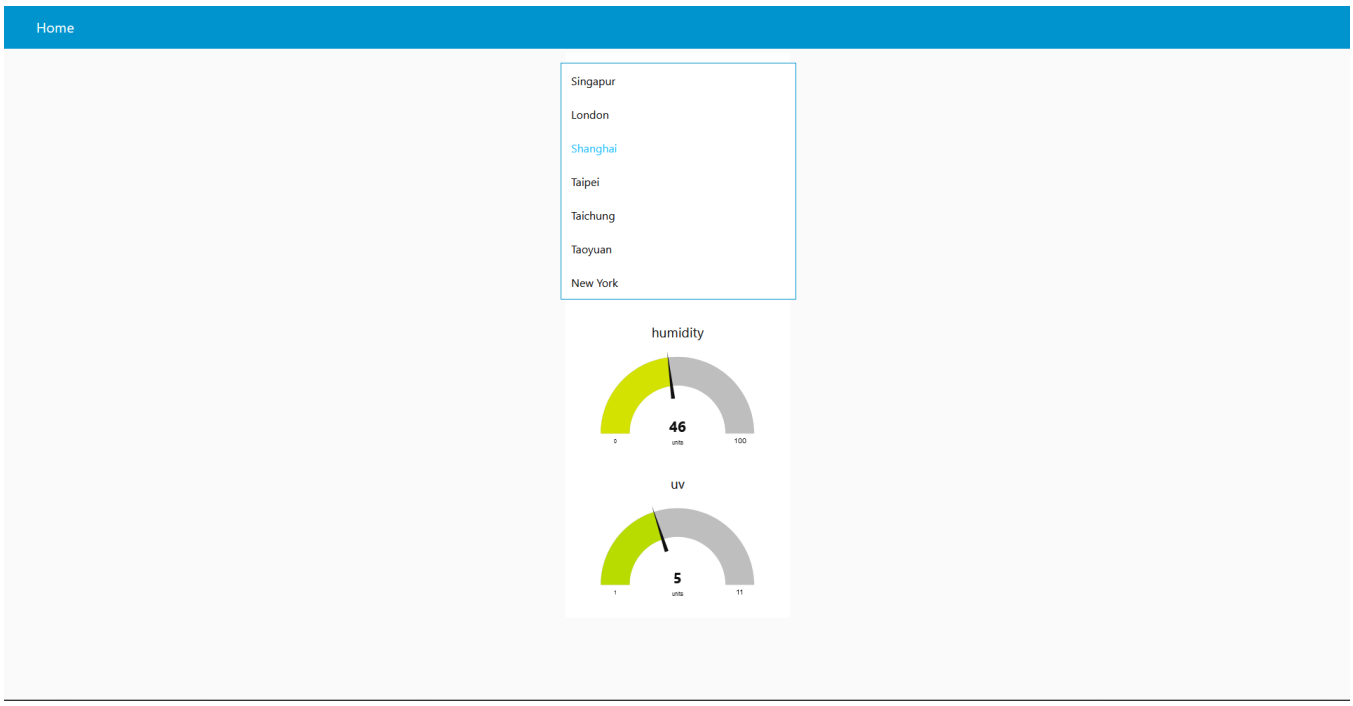
Exercise 10-4 Show the UV index and temperature of the selected city on the dashboard.

Based on exercise 10-3, instead of only displaying temperature, humidity and UV data are also displayed with different gauges.

Note: node "http request" is in its default settings.
Note: node "dropdown" is manually filled with options. It has the exact same settings in exercise 10-2.

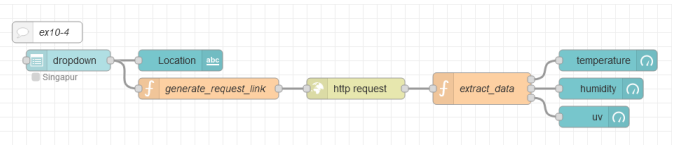


(a) dashboard

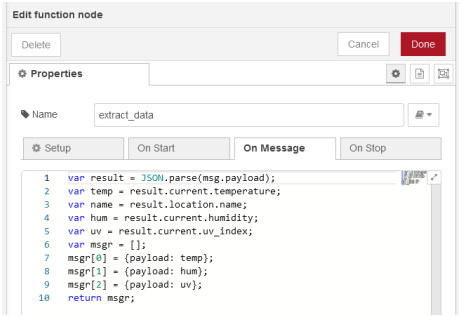


(b) dashboard options

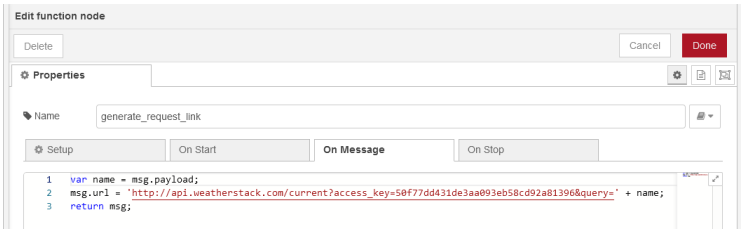
Figure 6: Exercise 10-4-1.



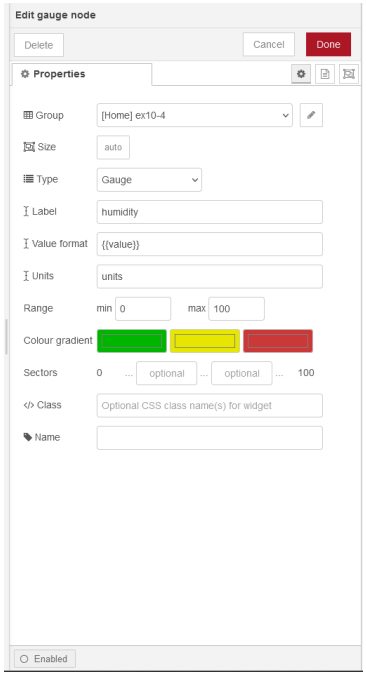
(a) flow



(b) node: extract data

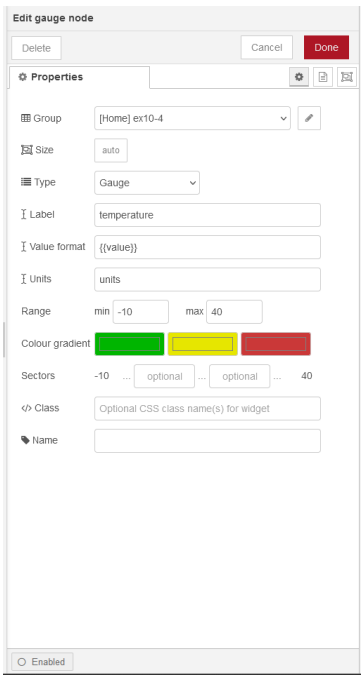


(c) node: generate request link



(d) node: humidity

(e) node: location



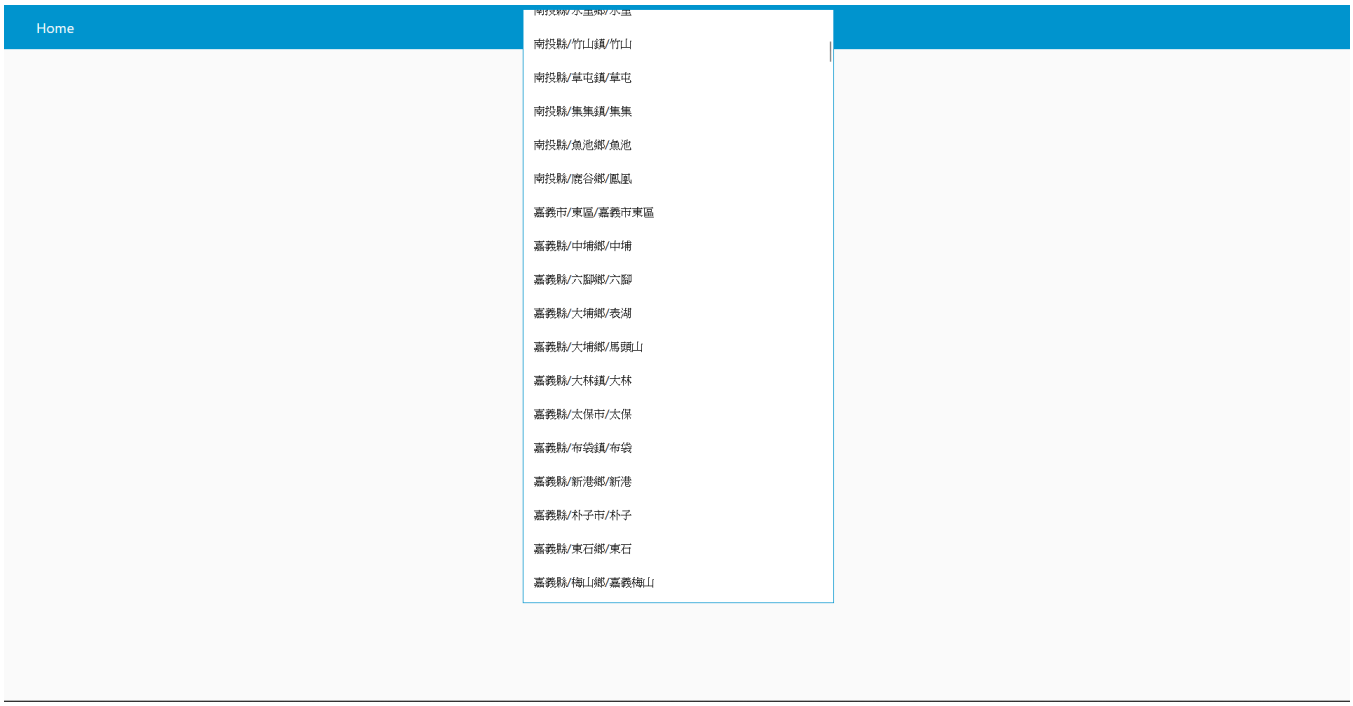
(f) node: temperature

(g) node: uv

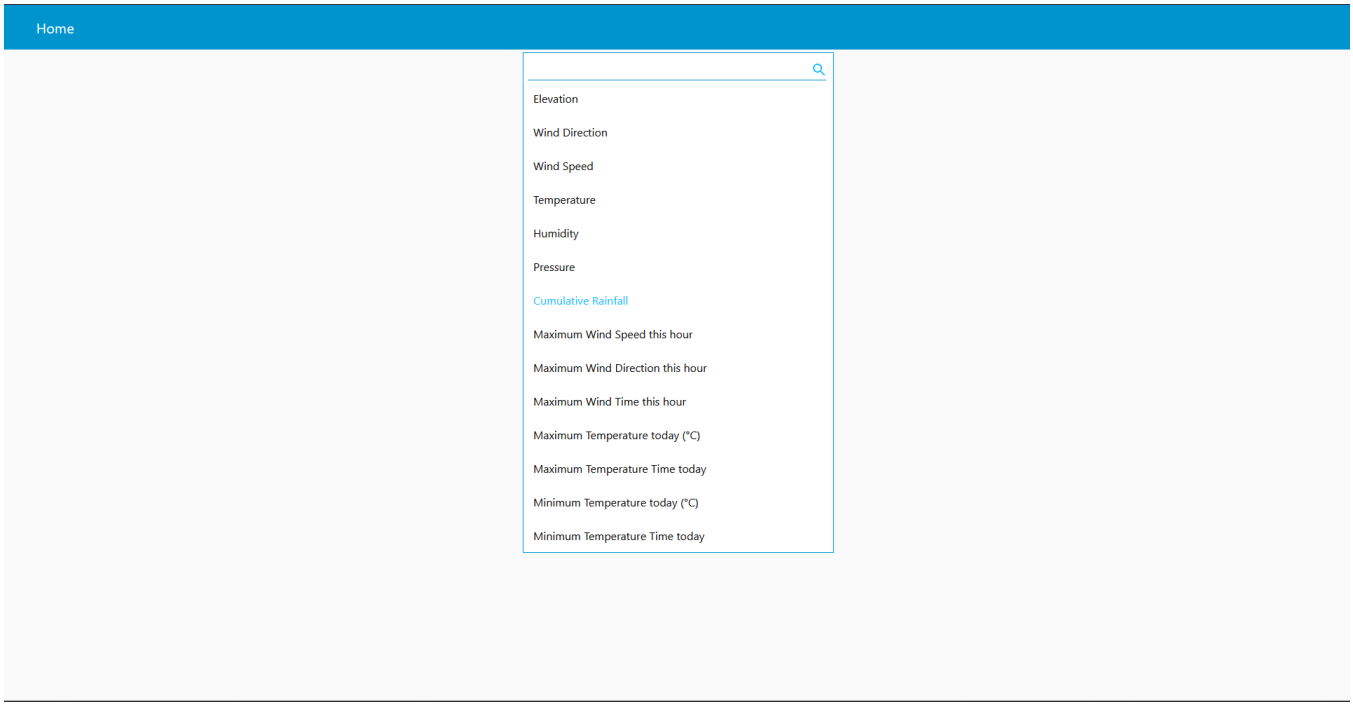
Figure 7: Exercise 10-4-2.

Exercise 10-5 Try another weather dataset to establish a weather station.

Features:
1. Options of "location" (456) and "WeatherElement" (14) are retrieved from return data, which is always going to be up to date.
2. Retrieve data is identified by "stationId" rather than "location" since there are duplicates elements in "location".
2. Provide current time.
3. Provide data retrieval time.



(a) dashboard Location option

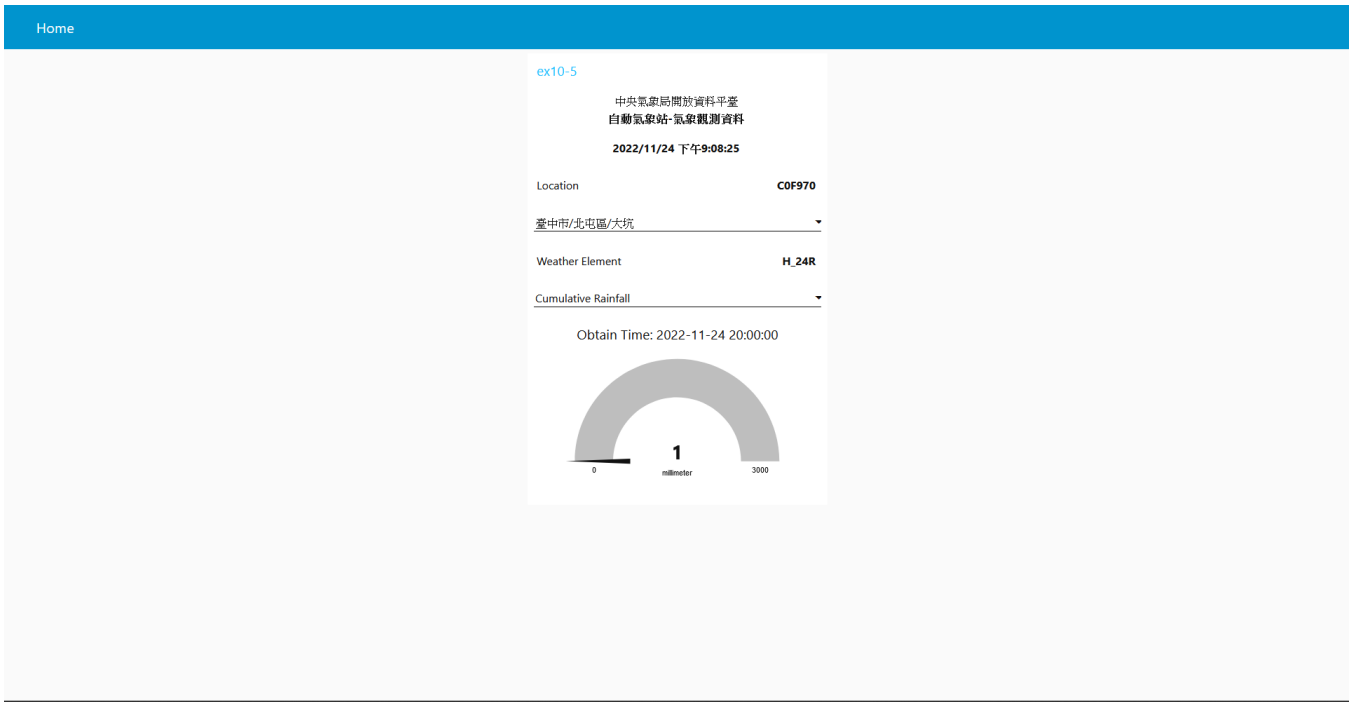


(b) dashboard Weather Element option

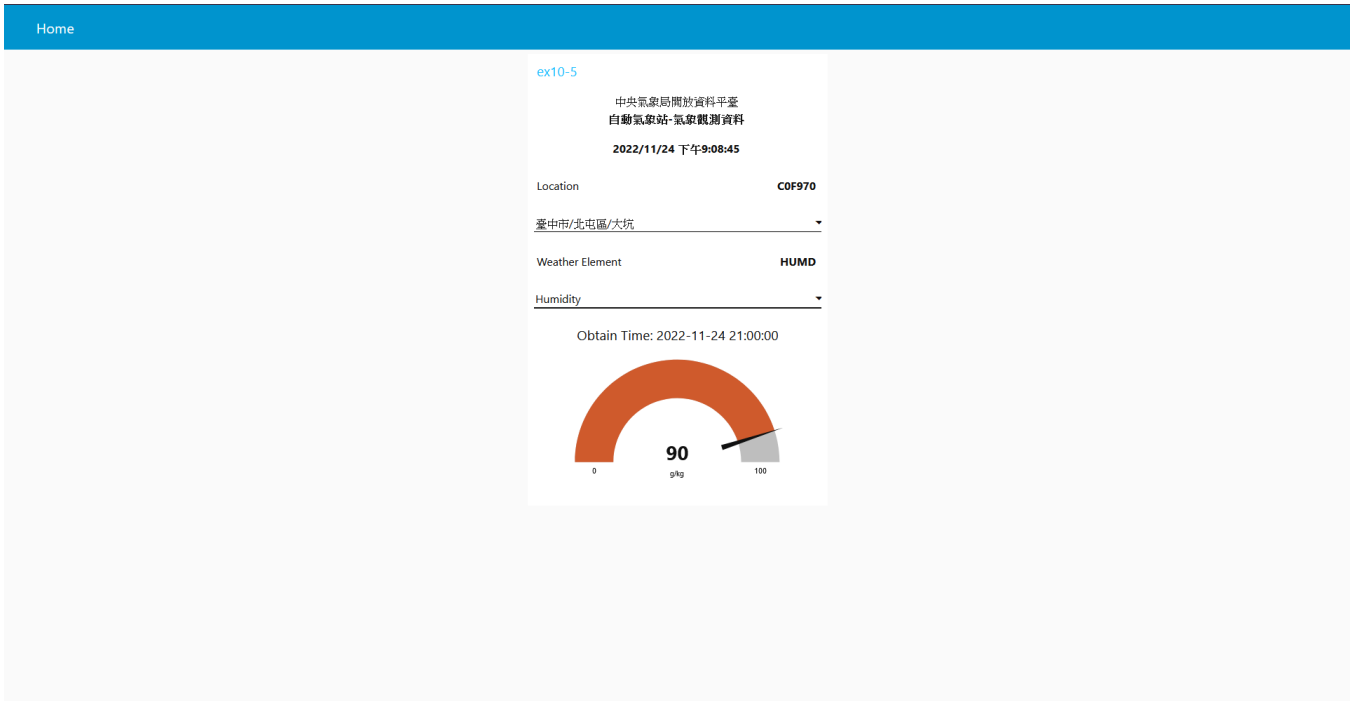
Figure 8: Exercise 10-5-1.

More Features:

4. After selecting "location" and 'WeatherElement", the gauge below shows the retrieved data.
5. Auto retrieve data every 10 seconds.
6. Provide different data ranges on gauge accordingly.
7. Provide different data units on gauge accordingly.
8. If data is unavailable, a warning will be issued and displayed on the dashboard as well.



(a) dashboard 大坑測站當天連續降雨

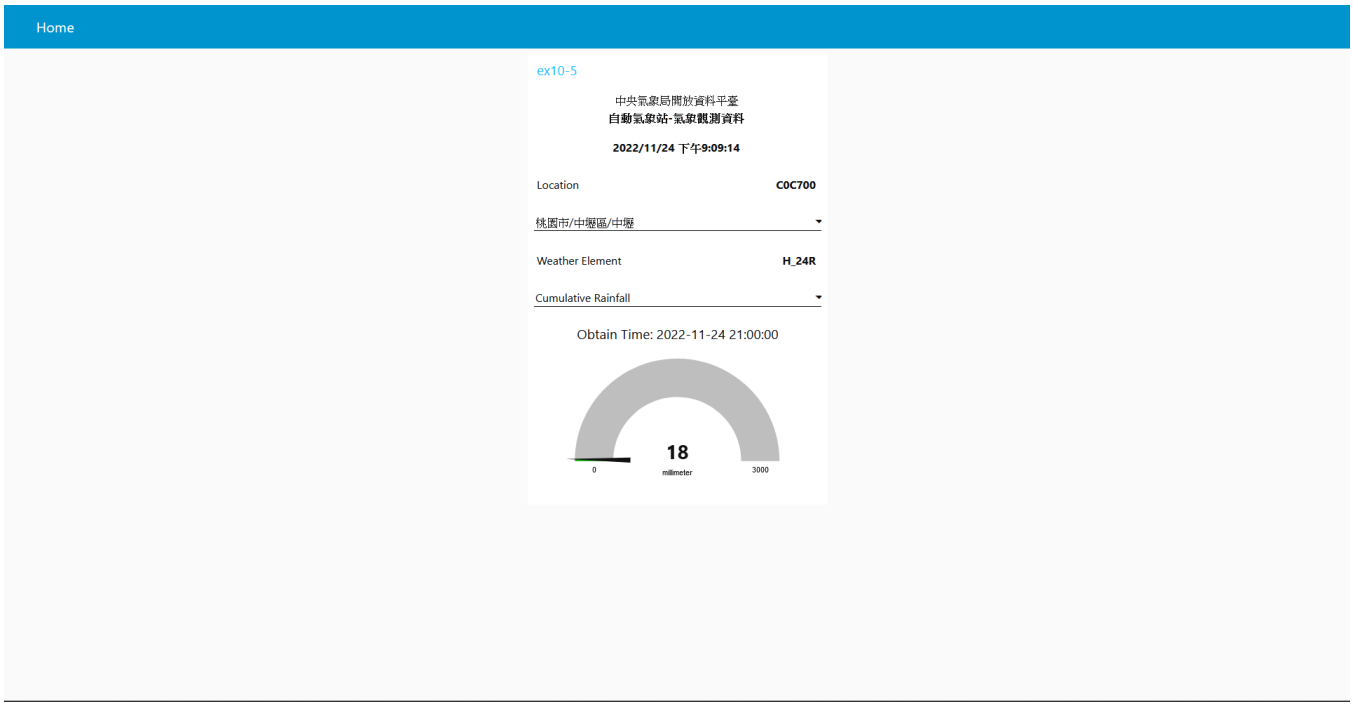


(b) dashboard 大坑測站空氣濕度

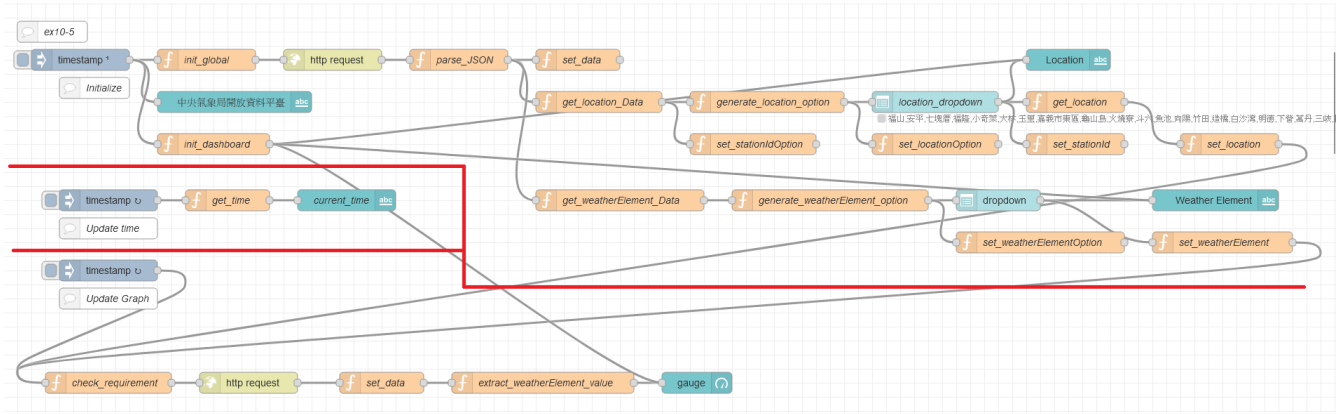
Figure 9: Exercise 10-5-2.

Data Source:

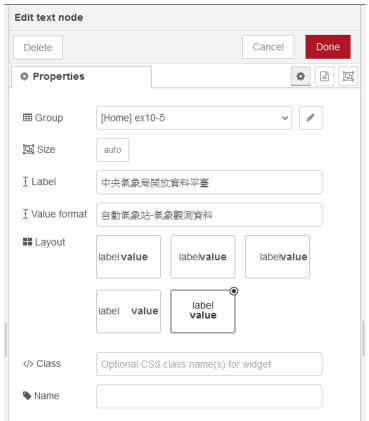
1. 氣象資料開放平台/氣候
2. 中央氣象局開放資料平臺之資料擷取API
3. 自動氣象站資料集說明檔



(a) dashboard 中壢測站當天連續降雨



(b) flow



(c) node: title text

Figure 10: Exercise 10-5.

Note.

1. Node "timestamp" in the "Initialize" section only activates only once after deployment. Trigger the initialization process of both the dashboard and this flow.
2. Node "timestamp" in the "Update time" section loops with 0.5-second intervals. Update the time displayed on the dashboard.
3. Node "timestamp" in the "Update Graph" section loops with 10-second intervals. Trigger the "update graph" process if both options on the dashboard are selected.
4. All "http request" nodes are in their default settings.
5. All "dropdown" nodes are in their default settings.

Codes in Function nodes.

Function - "init_global"

```
var GlobalTemp = {};
GlobalTemp.api = 'https://opendata.cwb.gov.tw/api/v1/rest/datastore/O-A0001-001?
    Authorization=CWB-8523DB1B-D613-401B-849E-BD0A19FAB9E6';
GlobalTemp.data = '';
GlobalTemp.location = ''; //deprecated
GlobalTemp.stationId = '';
GlobalTemp.weatherElement = '';
GlobalTemp.locationOption = ''; //deprecated
GlobalTemp.weatherElementOption = '';
GlobalTemp.stationIdOption = '';
global.set("temp", GlobalTemp);

msg.url = GlobalTemp.api;
return msg;
```

Listing 1: node: init_global

Function - "init_dashboard"

```
msg = {};
msg.unit = 'N/A';
msg.label = 'N/A';
msg.value = 0;
msg.payload = 'N/A';
return msg;
```

Listing 2: node: init_dashboard

Function - "parse_JSON"

```
msg.payload = JSON.parse(msg.payload)
return msg;
```

Listing 3: node: parse_JSON

Function - "set_data"

```
var GlobalTemp = global.get('temp');
GlobalTemp.data = msg.payload;
global.set('temp', GlobalTemp);
return msg;
```

Listing 4: node: set_data

Function - "get_location_Data"

```
var location_data = msg.payload.records.location;
var data_length = location_data.length;
var location_name = [];

for (let index = 0; index < data_length; index++) {
    var temp_locationName = location_data[index].locationName;
    var temp_cityName = location_data[index].parameter[0].parameterValue;
    var temp_townName = location_data[index].parameter[2].parameterValue;
    var temp_stationId = location_data[index].stationId;
    location_name[index] = {
        city: temp_cityName,
        town: temp_townName,
        location: temp_locationName,
        id: temp_stationId
        };
}

msg.payload = location_name;
return msg;
```

Listing 5: node: get_location_Data

Function - "generate_location_option"

```
var data = msg.payload;
var option = [];
var location = [];

function compare(a, b) {
    var keya = Object.keys(a)[0];
    var keyb = Object.keys(b)[0];
    // node.warn(keya);
    // node.warn(keya);
    if ( keya < keyb ){
        return -1;
    } else if (keya > keyb ) {
        return 1;
    } else {
        return 0;
    }
}

for (let index = 0; index < data.length; index++) {
    var temp_id = data[index].id;
```

```
    var temp_location = data[index].location;
    var temp_name = data[index].city + '/' + data[index].town + '/'+ data[index
    ].location;
    var temp_obj = {};
    temp_obj[temp_name] = temp_id;
    option[index] = temp_obj;
    location[index] = temp_location;
}

option = option.sort(compare);

msg.options = option;
msg.payload = location;
// node.warn(msg.options);
return msg;
```

Listing 6: node: generate_location_option

Function - "set_stationIdOption"

```
var data = msg.payload;
var option = [];

for (let index = 0; index < data.length; index++) {
    var temp = data[index];
    option[index] = temp.id;
}

var GlobalTemp = global.get('temp');
GlobalTemp.stationIdOption = option;
global.set('temp', GlobalTemp);

return msg;
```

Listing 7: node: set_stationIdOption

Function - "set_locationOption"

```
var GlobalTemp = global.get('temp');
GlobalTemp.locationOption = msg.payload;
global.set('temp', GlobalTemp);
return msg;
```

Listing 8: node: set_locationOption

Function - "get_location"

```
var GlobalTemp = global.get('temp');
var locationOption = GlobalTemp.locationOption;
var stationIdOption = GlobalTemp.stationIdOption;

node.warn(msg.payload);

for (let index = 0; index < stationIdOption.length; index++) {
    var temp = stationIdOption[index];
    if (temp === msg.payload) {
```

```
        GlobalTemp.location = locationOption[index];
    }
}

global.set('temp', GlobalTemp);

msg.payload = GlobalTemp.location;
return msg;
```

Listing 9: node: get_location

Function - "set_stationId"

```
var GlobalTemp = global.get('temp');
GlobalTemp.stationId = msg.payload;
global.set('temp', GlobalTemp);
```

Listing 10: node: set_stationId

Function - "set_location"

```
var GlobalTemp = global.get('temp');
GlobalTemp.location = msg.payload;
node.warn(msg.payload);
global.set('temp', GlobalTemp);
return msg;
```

Listing 11: node: set_location

Function - "get_weatherElement_Data"

```
var weatherElement_data = msg.payload.records.location[0].weatherElement;
var weatherElementLength = weatherElement_data.length;
var weatherElementName = [];

for (let index = 0; index < weatherElementLength; index++) {
    weatherElementName[index] = weatherElement_data[index].elementName;
}

msg.payload = weatherElementName;
return msg;
```

Listing 12: node: get_weatherElement_Data

Function - "generate_weatherElement_option"

```
var data = msg.payload;
var option = [];
var weatherElement = [];

for (let index = 0; index < data.length; index++) {
    var temp = data[index];
    var temp_obj = {};

    if (temp === 'ELEV') {
        var temp_name = 'Elevation';
    }else if (temp === 'WDIR') {
        var temp_name = 'Wind Direction';
```

```
    }else if (temp === 'WDSD') {
        var temp_name = 'Wind Speed';
    }else if (temp === 'TEMP') {
        var temp_name = 'Temperature';
    }else if (temp === 'HUMD') {
        var temp_name = 'Humidity';
    }else if (temp === 'PRES') {
        var temp_name = 'Pressure';
    }else if (temp === 'H_24R') {
        var temp_name = 'Cumulative Rainfall';
    }else if (temp === 'H_FX') {
        var temp_name = 'Maximum Wind Speed this hour';
    }else if (temp === 'H_XD') {
        var temp_name = 'Maximum Wind Direction this hour';
    }else if (temp === 'H_FXT') {
        var temp_name = 'Maximum Wind Time this hour';
    }else if (temp === 'D_TX') {
        var temp_name = 'Maximum Temperature today (°C)';
    }else if (temp === 'D_TXT') {
        var temp_name = 'Maximum Temperature Time today';
    }else if (temp === 'D_TN') {
        var temp_name = 'Minimum Temperature today (°C)';
    }else if (temp === 'D_TNT') {
        var temp_name = 'Minimum Temperature Time today';
    }

    temp_obj[temp_name] = temp;
    option[index] = temp_obj;
    weatherElement[index] = temp;
}

msg.options = option;
msg.payload = weatherElement;
// node.warn(msg.options);
return msg;
```

Listing 13: node: generate_weatherElement_option

Function - "set_weatherElementOption"

```
var GlobalTemp = global.get('temp');
GlobalTemp.weatherElementOption = msg.payload;
global.set('temp', GlobalTemp);
return msg;
```

Listing 14: node: set_weatherElementOption

Function - "set_weatherElement"

```
var GlobalTemp = global.get('temp');
GlobalTemp.weatherElement = msg.payload;
global.set('temp', GlobalTemp);
return msg;
```

Listing 15: node: set_weatherElement

Function - "get_time"

```
var timestamp = msg.payload;
var date = new Date(timestamp);
var timestring = date.toLocaleDateString() + ' ' + date.toLocaleTimeString();
msg.payload = timestring;
return msg;
```

Listing 16: node: get_time

Function - "check_requirement"

```
var GlobalTemp = global.get('temp');
var location = GlobalTemp.location;
var weatherElement = GlobalTemp.weatherElement;
var locationOption = GlobalTemp.locationOption;
var weatherElementOption = GlobalTemp.weatherElementOption;
var api = GlobalTemp.api;

if (locationOption.includes(location)) {
    var condition1 = 1;
} else {
    var condition1 = 0;
}

if (weatherElementOption.includes(weatherElement)) {
    var condition2 = 1;
} else {
    var condition2 = 0;
}

// node.warn(condition1);
// node.warn(condition2);

if (condition1 === 1) {
    // node.warn("condition1 satisfied");
    if (condition2 === 1) {
        // node.warn("condition2 satisfied")
        var msg = {};
        msg.url = api;
        // node.warn(msg);
        return msg;
    }
}
```

Listing 17: node: check_requirement

Function - "set_data"

```
msg.payload = JSON.parse(msg.payload)
var GlobalTemp = global.get('temp');
GlobalTemp.data = msg.payload;
global.set('temp', GlobalTemp);
return msg;
```

Listing 18: node: set_data

Function - "extract_weatherElement_value"

```
var location_data = msg.payload.records.location;
var GlobalTemp = global.get('temp');
var location = GlobalTemp.location;
var stationId = GlobalTemp.stationId;
var weatherElement = GlobalTemp.weatherElement;
var GlobalRange = global.get('range');
var msg = {};

for (let i = 0; i < location_data.length; i++) {
    var temp = location_data[i].stationId;
    if (temp === stationId) {
        // node.warn(temp)
        var time = location_data[i].time.obsTime;
        // node.warn(time);
        var weatherElement_data = location_data[i].weatherElement;
        for (let j = 0; j < weatherElement_data.length; j++) {
            var temp = weatherElement_data[j].elementName;
            if (temp === weatherElement) {
                // node.warn(temp)
                msg.payload = parseInt(weatherElement_data[j].elementValue);
                msg.label = time;
                if (temp === 'ELEV') {
                    msg.ui_control = { min: 0, max: 3952 };
                    msg.unit = 'meter';
                } else if (temp === 'WDIR' || temp === 'H_XD') {
                    msg.ui_control = { min: 0, max: 360 };
                    msg.unit = 'angle';
                } else if (temp === 'TEMP' || temp === 'D_TX' || temp === 'D_TN'
) {
                    msg.ui_control = { min: -20, max: 45 };
                    msg.unit = 'degree (°C)';
                } else if (temp === 'HUMD') {
                    msg.ui_control = { min: 0, max: 100 };
                    msg.payload = parseFloat(weatherElement_data[j].elementValue
);
                    msg.payload = msg.payload * 100;
                    msg.unit = 'g/kg';
                } else if (temp === 'PRES') {
                    msg.ui_control = { min: 950, max: 1050 };
                    msg.unit = 'hpa';
                } else if (temp === 'H_24R') {
                    msg.ui_control = { min: 0, max: 3000 };
                    msg.unit = 'millimeter'
                } else if (temp === 'H_FX') {
                    msg.ui_control = { min: 0, max: 61.3 };
                    msg.unit = 'm/s'
                } else {
                    msg.ui_control = { min: 'N/A', max: 'N/A' };
                }
                if (msg.payload === -99) {
```

```
                    msg.payload = 'Data Currently Unavailable';
                    var warning = stationId + '/' + location + '/' +
    weatherElement + ' is unavailable currently.';
                    node.warn(warning);
                }
                return msg;
            }
        }
    }
}
```

Listing 19: node: extract_weatherElement_value