Project Development Phase

SMART WASTE MANAGEMENT SYSTEM FOR METROPOLITAN CITIES

TEAM ID: PNT2022TMID39310

TEAM MEMBERS

ROLE	TEAM MEMBERS NAME	ROLL NO
TEAM LEADER	RUBESH .S	(422619104035)
TEAM MEMBER 1	ABIRAMI .S	(422619104002)
TEAM MEMBER 2	HARITHA .S	(422619104016)
TEAM MEMBER 3	TAMIZHSELVAN .D	(422619104044)

Project Development - Delivery Of Sprint-4

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-4	Develop A Web Application Using Node-RED Service	USN- 8	Create Node Red Flow To Get Data From Device	10	High	S.Rubesh S.Haritha
Sprint-4	Develop A Web Application Using Node-RED Service	USN- 9	UI Nodes Installation	10	Medium	S.Abirami D.Tamizhselvan

Delivery

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

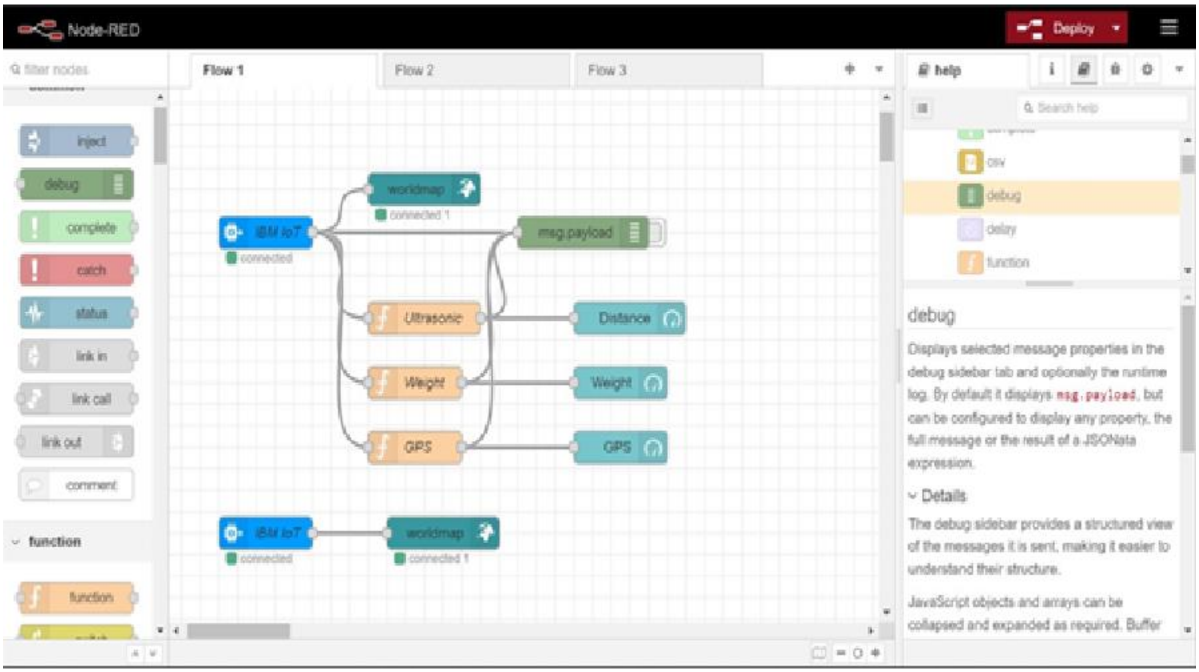
Create Node Red Flow To Get Data From Device

Task Assigned: S.Rubesh,S.Haritha

Task Started On: 14 Nov 2022

Task Completion Date: 15 Nov 2022

Create Node Red Flow To Get Data From Device



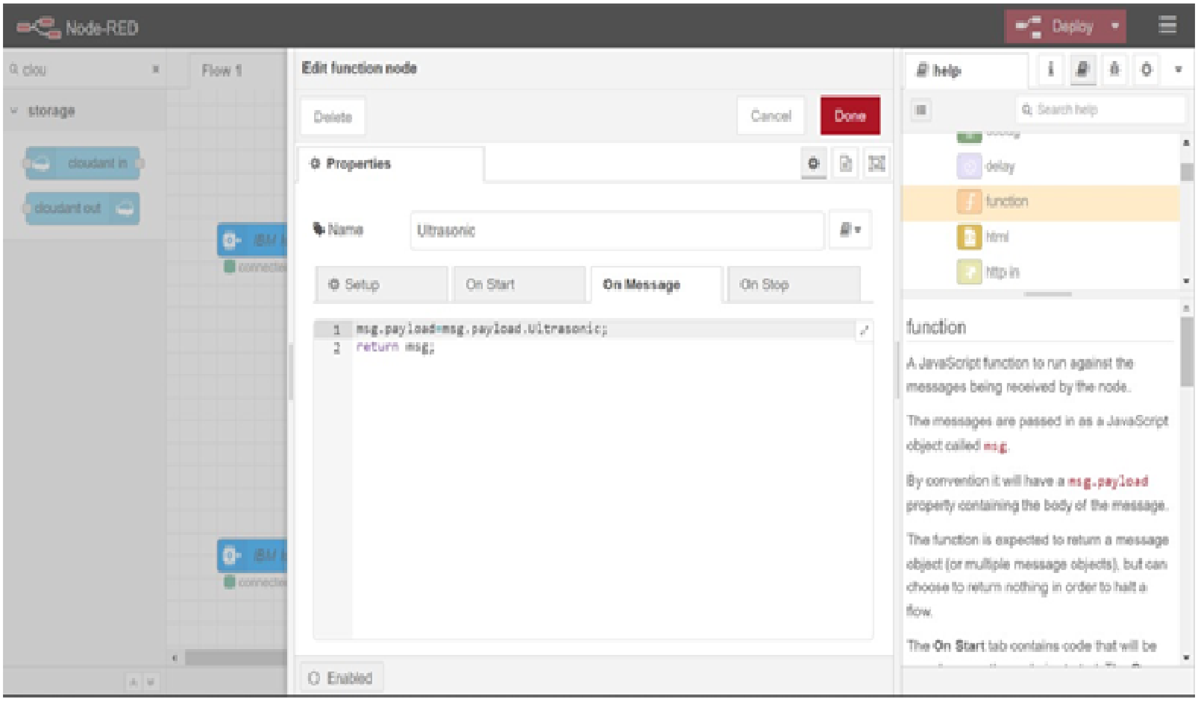
debug

Displays selected message properties in the debug sidebar tab and optionally the runtime log. By default it displays `msg.payload`, but can be configured to display any property, the full message or the result of a JSONata expression.

Details

The debug sidebar provides a structured view of the messages it is sent, making it easier to understand their structure.

JavaScript objects and arrays can be collapsed and expanded as required. Buffer



Name: Ultrasonic

```
msg.payload=msg.payload.Ultrasonic;
return msg;
```

function

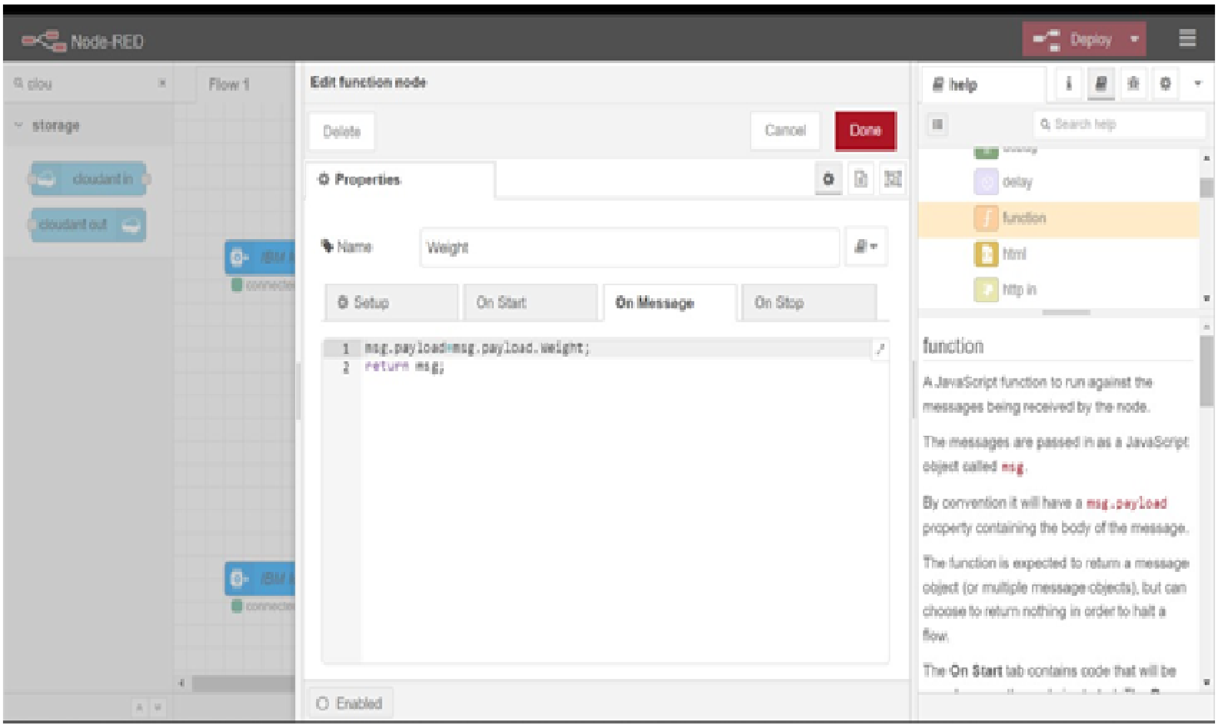
A JavaScript function to run against the messages being received by the node.

The messages are passed in as a JavaScript object called `msg`.

By convention it will have a `msg.payload` property containing the body of the message.

The function is expected to return a message object (or multiple message objects), but can choose to return nothing in order to halt a flow.

The **On Start** tab contains code that will be



Edit function node

Name: Weight

```
msg.payload=msg.payload.weight;
return msg;
```

function

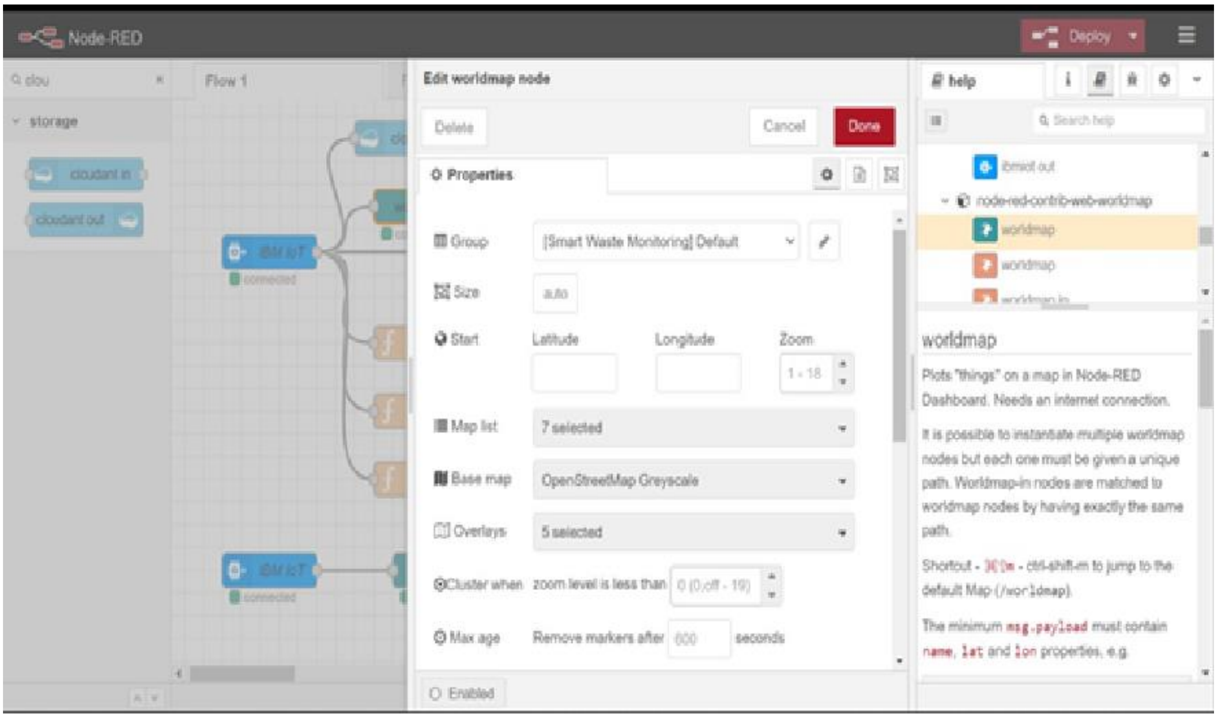
A JavaScript function to run against the messages being received by the node.

The messages are passed in as a JavaScript object called `msg`.

By convention it will have a `msg.payload` property containing the body of the message.

The function is expected to return a message object (or multiple message objects), but can choose to return nothing in order to halt a flow.

The **On Start** tab contains code that will be



Edit worldmap node

Group: [Smart Waste Monitoring] Default

Size: auto

Start: Latitude Longitude Zoom 1 - 18

Map list: 7 selected

Base map: OpenStreetMap Greyscale

Overlays: 5 selected

Cluster when zoom level is less than 0 (0,off - 19)

Max age: Remove markers after 600 seconds

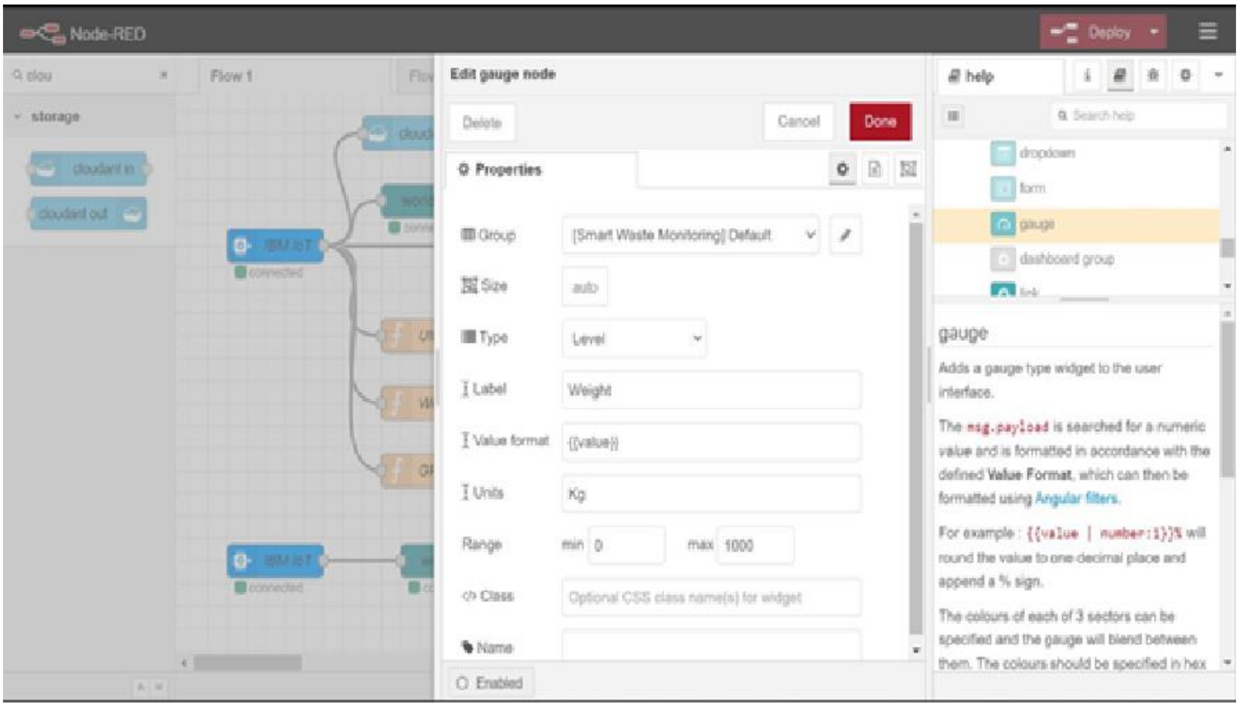
worldmap

Plots "things" on a map in Node-RED Dashboard. Needs an internet connection.

It is possible to instantiate multiple worldmap nodes but each one must be given a unique path. Worldmap-in nodes are matched to worldmap nodes by having exactly the same path.

Shortcut - ⌘⇧m - ctrl-shift-m to jump to the default Map (`/worldmap`).

The minimum `msg.payload` must contain `name`, `lat` and `lon` properties, e.g.



Group [Smart Waste Monitoring] Default

Size auto

Type Level

Label Weight

Value format {{value}}

Units Kg

Range min 0 max 1000

Class Optional CSS class name(s) for widget

Name

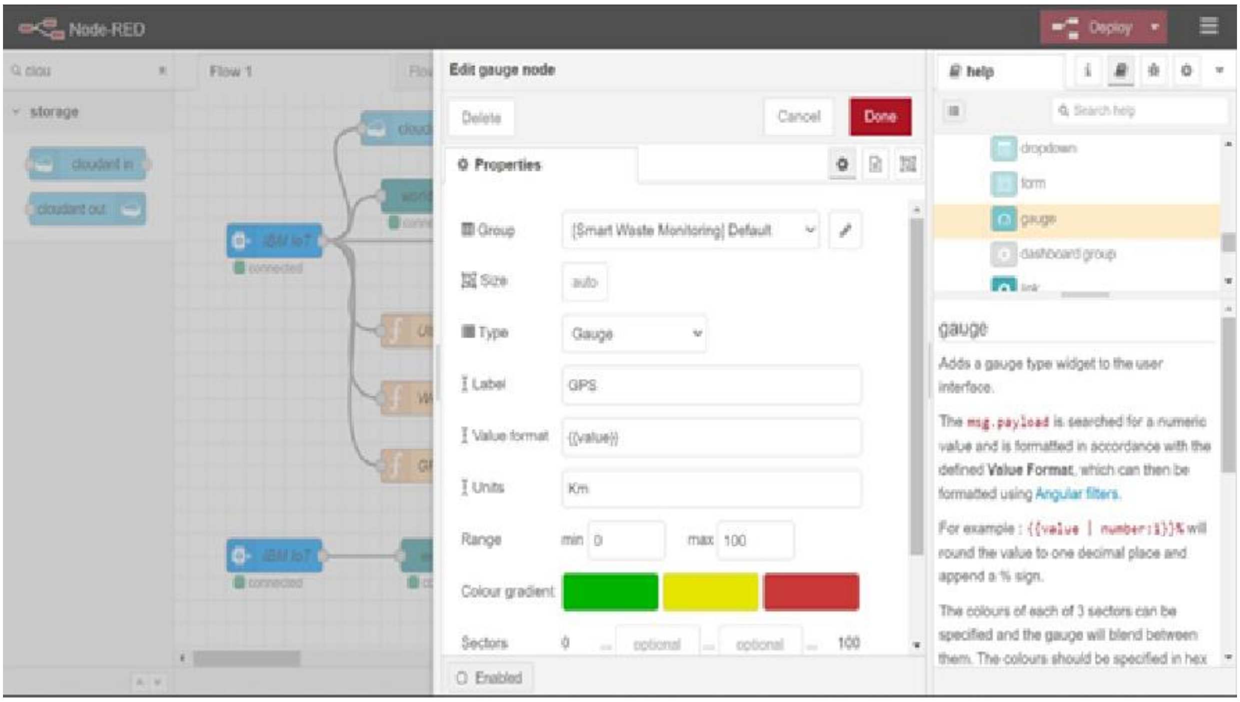
gauge

Adds a gauge type widget to the user interface.

The msg.payload is searched for a numeric value and is formatted in accordance with the defined **Value Format**, which can then be formatted using Angular filters.

For example : `{{value | number:1}}%` will round the value to one decimal place and append a % sign.

The colours of each of 3 sectors can be specified and the gauge will blend between them. The colours should be specified in hex



Group [Smart Waste Monitoring] Default

Size auto

Type Gauge

Label GPS

Value format {{value}}

Units Km

Range min 0 max 100

Colour gradient 🟩 🟨 🟥

Sectors 0 ... optional ... optional ... 100

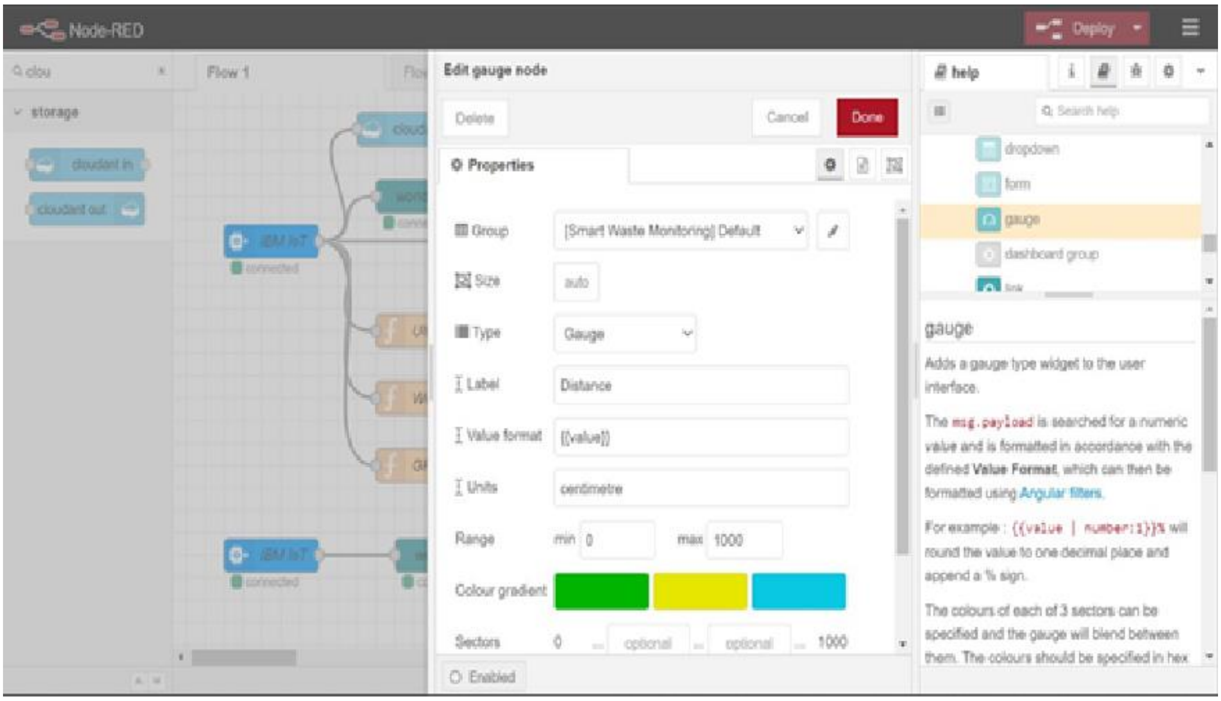
gauge

Adds a gauge type widget to the user interface.

The msg.payload is searched for a numeric value and is formatted in accordance with the defined **Value Format**, which can then be formatted using Angular filters.

For example : `{{value | number:1}}%` will round the value to one decimal place and append a % sign.

The colours of each of 3 sectors can be specified and the gauge will blend between them. The colours should be specified in hex



Edit gauge node

Delete Cancel Done

Properties

Group: [Smart Waste Monitoring] Default

Size: auto

Type: Gauge

Label: Distance

Value format: {{value}}

Units: centimetre

Range: min 0 max 1000

Colour gradient: 🟩 🟨 🟦

Sectors: 0 ... optional ... optional ... 1000

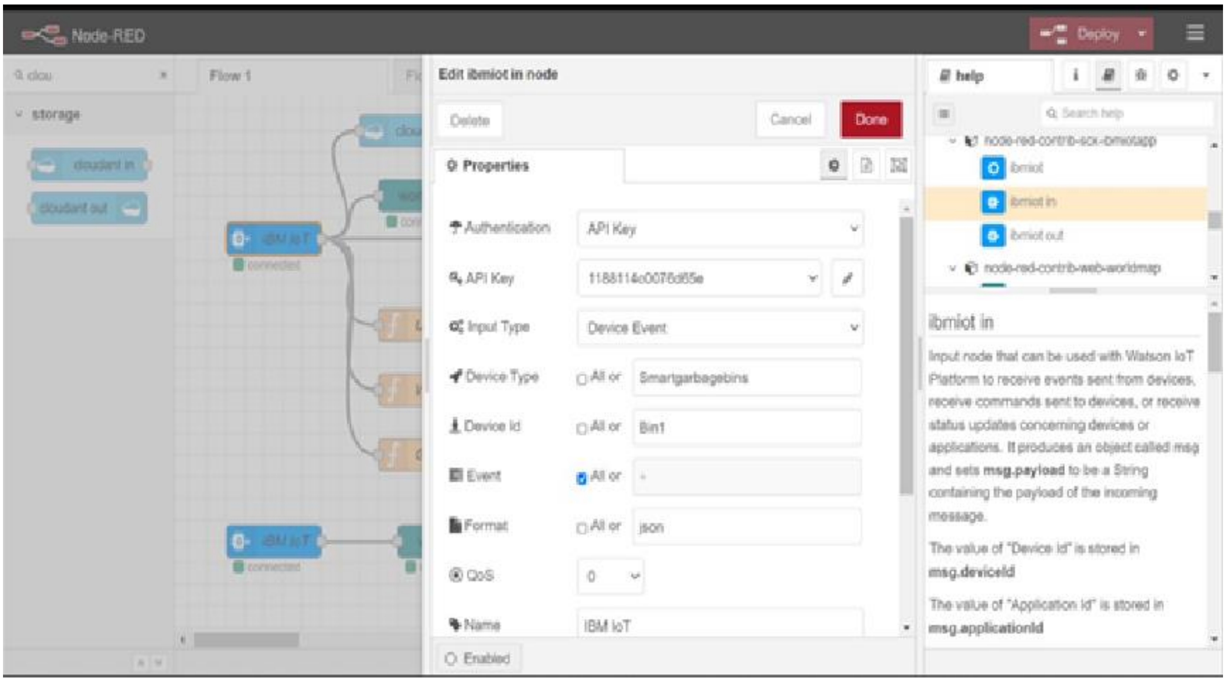
gauge

Adds a gauge type widget to the user interface.

The msg.payload is searched for a numeric value and is formatted in accordance with the defined **Value Format**, which can then be formatted using Angular filters.

For example : {{value | number:1}}% will round the value to one decimal place and append a % sign.

The colours of each of 3 sectors can be specified and the gauge will blend between them. The colours should be specified in hex



Edit ibmiot in node

Delete Cancel Done

Properties

Authentication: API Key

API Key: 1188114c0076d65e

Input Type: Device Event

Device Type: □ All or Smartgarbagebins

Device Id: □ All or Bin1

Event: ☑ All or +

Format: □ All or json

QoS: 0

Name: IBM IoT

ibmiot in

Input node that can be used with Watson IoT Platform to receive events sent from devices, receive commands sent to devices, or receive status updates concerning devices or applications. It produces an object called msg and sets **msg.payload** to be a String containing the payload of the incoming message.

The value of "Device Id" is stored in **msg.deviceId**

The value of "Application Id" is stored in **msg.applicationId**

CODE:

```
import time
 import sys
 import ibmiotf.application
import ibmiotf.device
import random
import sys
#Provide your IBM Watson Device Credentials
organization = "a7mbs7"
deviceType ="Smartgarbagebins"
deviceId = "Bin1"
authMethod = "token"
authToken = "Sakthi@2001"
# Initialize GPIO
def myCommandCallback(cmd):
print("Command received: %s" % cmd.data['command'])
status=cmd.data['command']
if status =="lighton":
    print("led in on")
    else :
        print ("led is off") try:
    deviceOptions = {"org": organization, "type": deviceType, "id": deviceId,
"authmethod":authMethod, "auth-token": authToken}
    deviceCli = ibmiotf.device.Client(deviceOptions)
#.............................................
except Exception as e:
    print("Caught exception connecting device: %s" % str(e))
    sys.exit()
#Connect and send a datapoint "hello" with value "world" into the cloud as an event oftype "greeting" 10 times
deviceCli.connect()
 while True:
    #Get Sensor Data from DHT11
time.sleep(5)
    Ultrasonic=random.randint(0,80)
Weight=random.randint(0,100)
lat = round(random.uniform(11.03, 11.50), 6)
long = round(random.uniform(76.80, 76.90), 6)
GPS = str(lat) + str(',') + str(long)
 myData = {'Ultrasonic' : Ultrasonic, 'Weight'
Weight , 'GPS' : GPS}
  #print data     def
myOnPublishCallback():
        print ("Published Ultrasonic = %s Cm" %Ultrasonic, "Weight:%s kg " %Weight,
"GPS: %s"%GPS)
success = deviceCli.publishEvent("IoTSensor", "json", data=myData, qos=0,
on_publish=myOnPublishCallback)
 if not success:
            print("Not connected to IoTF")
            time.sleep(1)
            deviceCli.commandCallback =
myCommandCallback
```

```
        # Disconnect the device and application from
the cloud deviceCli.disconnect()
```

RESULT:

Thus the Node Red Flow is successfully created and Got Data From the Device.