Project Development Phase
Sprint-2

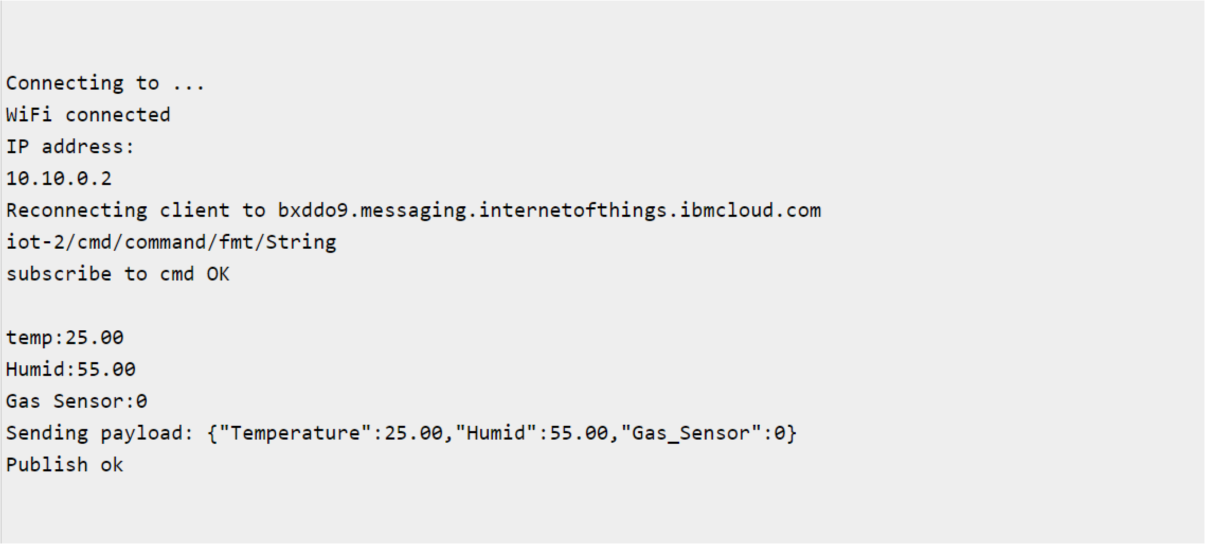
Date	7-November 2022
Team ID	PNT2022TMID35932
Project Name	Hazardous Area Monitoring for Industrial Plant powered by IoT
Maximum Marks	4 Marks

ESP32 Program for Temperature and Dummy Gas sensor simulated in Wokwi:

```
#include "DHT.h"// Library for dht22
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQtt

#define DHTPIN 15     // what pin we're connected to
#define DHTTYPE DHT22   // define type of sensor DHT 11

//GAS SENSOR MQ-02
#define GAS_SENSOR 2



DHT dht (DHTPIN, DHTTYPE);// creating the instance by passing pin and typr of dht connected

void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);

//-------credentials of IBM Accounts------

#define ORG "bxddo9"//IBM ORGANITION ID
#define DEVICE_TYPE "ESP32"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "Assign4"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "45625689713"     //Token
String data3;
float h, t;
int val;


//-------- Customise the above values --------
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";// Server Name
char publishTopic[] = "iot-2/evt/Data/fmt/json";// topic name and type of event perform and
format in which data to be send
char subscribetopic[] = "iot-2/cmd/command/fmt/String";// cmd  REPRESENT command type AND
COMMAND IS TEST OF FORMAT STRING
char authMethod[] = "use-token-auth";// authentication method
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;//client id


//-----------------------------------------
WiFiClient wifiClient; // creating the instance for wificlient
PubSubClient client(server, 1883, callback ,wifiClient); //calling the predefined client id by
passing parameter like server id,portand wificredential

const int DHT_PIN = 15;

void setup() {
  Serial.begin(115200);
  Serial.println();
  wificonnect();
  mqttconnect();
}

void loop() {
  val = digitalRead(GAS_SENSOR);
  //val = random(1) ;
  h = dht.readHumidity();
  t = dht.readTemperature();
  Serial.print("temp:");
  Serial.println(t);
  Serial.print("Humid:");
  Serial.println(h);
  Serial.print("Gas Sensor:");
  Serial.println(val);
  PublishData(t, h, val);
  delay(1000);
  if (!client.loop()) {
    mqttconnect();
  }

  delay(1000);
}

/*.....................................retrieving to Cloud...............................*/

void PublishData(float temp, float humid, int vol) {
  mqttconnect();//function call for connecting to ibm
  /*
     creating the String in in form JSon to update the data to ibm cloud
  */
  String payload = "{\"Temperature\":";
  payload += temp;
  payload += "," "\"Humid\":";
  payload += humid;
  payload += ",""\"Gas_Sensor\":";
  payload += val;
  payload += "}";


  Serial.print("Sending payload: ");
  Serial.println(payload);


  if (client.publish(publishTopic, (char*) payload.c_str())) {
    Serial.println("Publish ok");// if it sucessfully upload data on the cloud then it will
print publish ok in Serial monitor or else it will print publish failed
  } else {
    Serial.println("Publish failed");
  }

}


void mqttconnect() {
  if (!client.connected()) {
    Serial.print("Reconnecting client to ");
    Serial.println(server);
    while (!!!client.connect(clientId, authMethod, token)) {
      Serial.print(".");
      delay(500);
    }
```

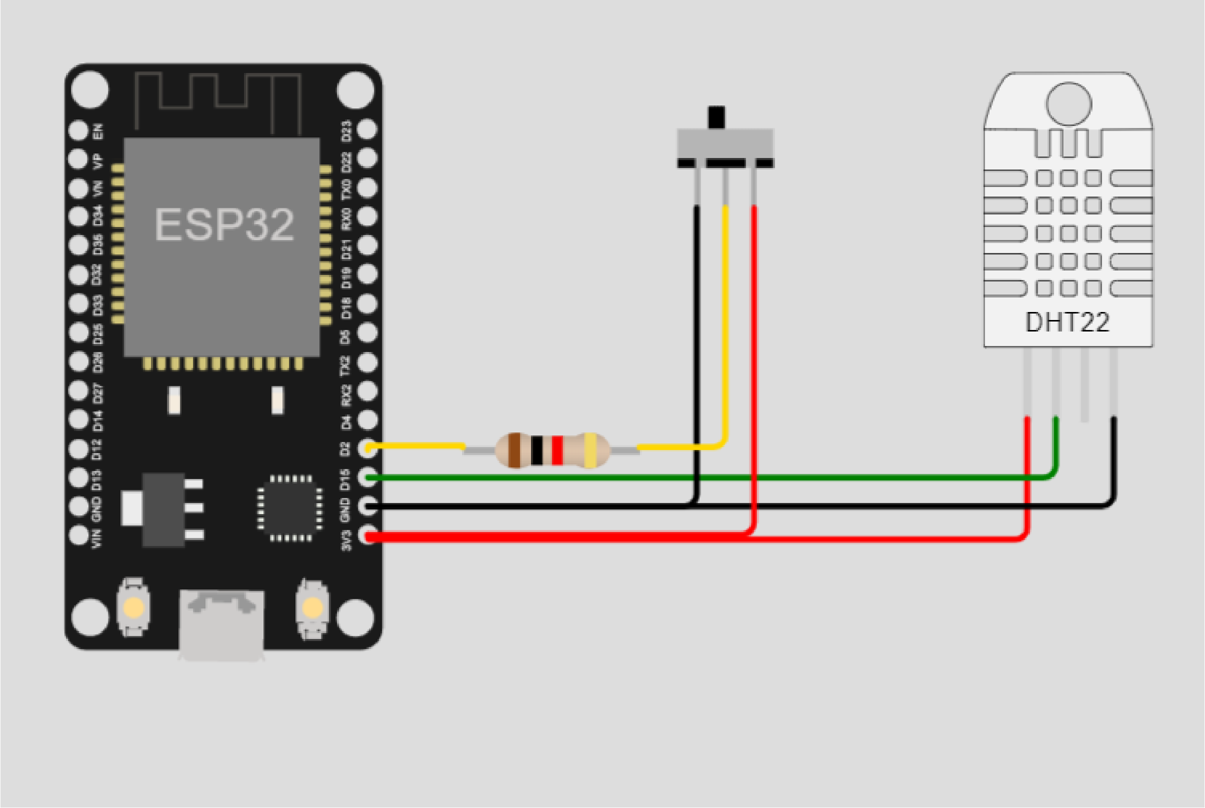
```
     initManagedDevice();
     Serial.println();
  }
}
void wificonnect() //function defination for wificonnect
{
  Serial.println();
  Serial.print("Connecting to ");

  WiFi.begin("Wokwi-GUEST", "", 6);//passing the wifi credentials to establish the connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void initManagedDevice() {
  if (client.subscribe(subscribetopic)) {
    Serial.println((subscribetopic));
    Serial.println("subscribe to cmd OK");
  } else {
    Serial.println("subscribe to cmd FAILED");
  }
}

void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)
{

  Serial.print("callback invoked for topic: ");
  Serial.println(subscribetopic);
  for (int i = 0; i < payloadLength; i++) {
    //Serial.print((char)payload[i]);
    data3 += (char)payload[i];
  }
  Serial.println("data: "+ data3);
  if(data3=="lighton")
  {
Serial.println(data3);

  }
  else
  {
Serial.println(data3);
}
data3="";
}
```

Wokwi Simulation Output:



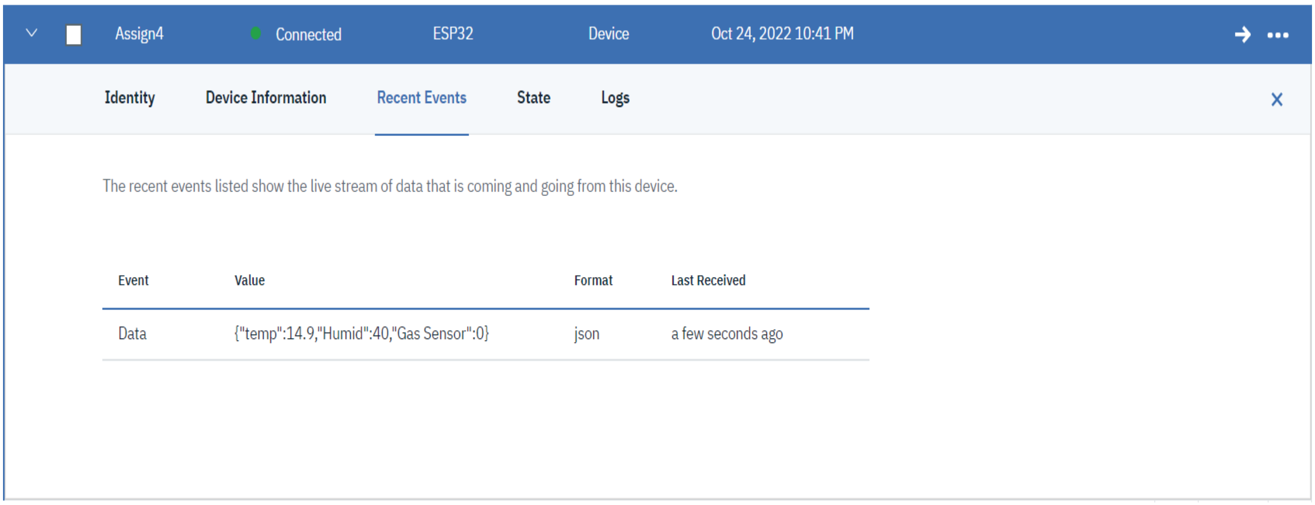
```
Connecting to ...
WiFi connected
IP address:
10.10.0.2
Reconnecting client to bxddo9.messaging.internetofthings.ibmcloud.com
iot-2/cmd/command/fmt/String
subscribe to cmd OK

temp:25.00
Humid:55.00
Gas Sensor:0
Sending payload: {"Temperature":25.00,"Humid":55.00,"Gas_Sensor":0}
Publish ok
```

Wokwi Circuit Diagram:



The Switch acts as dummy input for Gas Sensor.

IBM Watson Dashboard view:

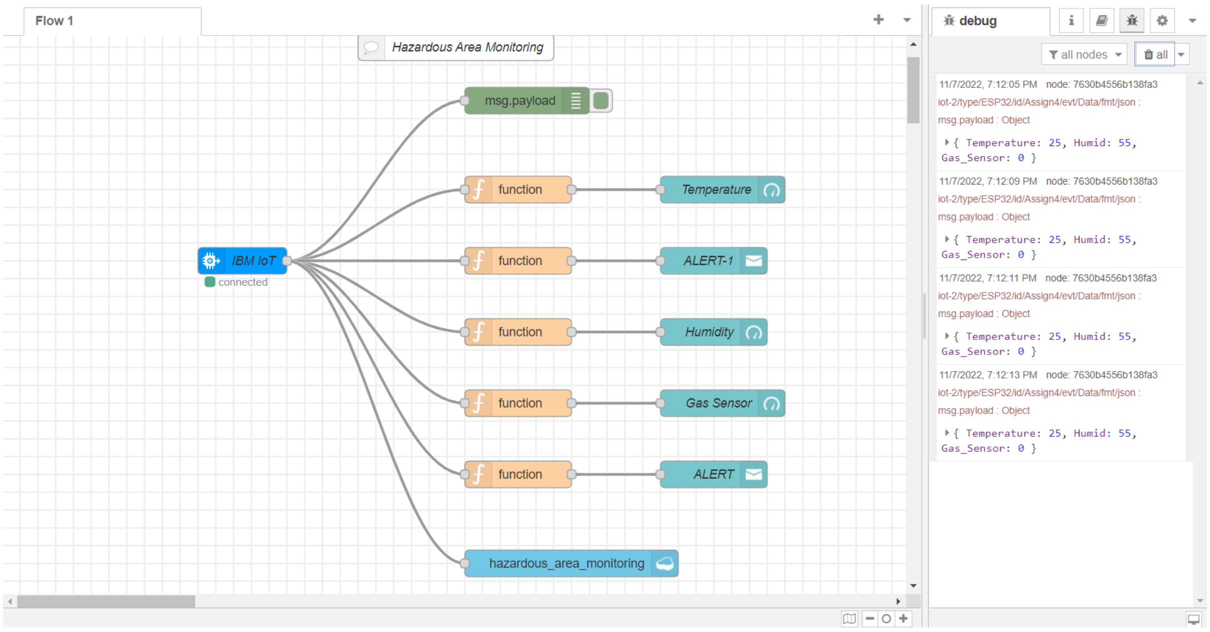


Assign4 Connected ESP32 Device Oct 24, 2022 10:41 PM

The recent events listed show the live stream of data that is coming and going from this device.

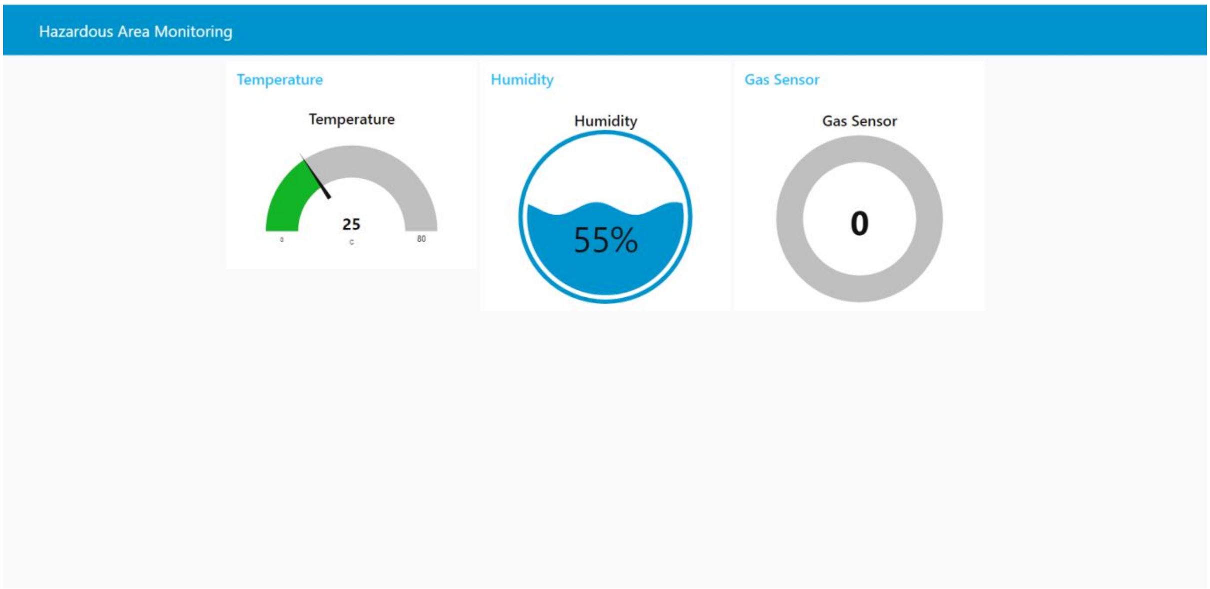
Event	Value	Format	Last Received
Data	{"temp":14.9,"Humid":40,"Gas Sensor":0}	json	a few seconds ago

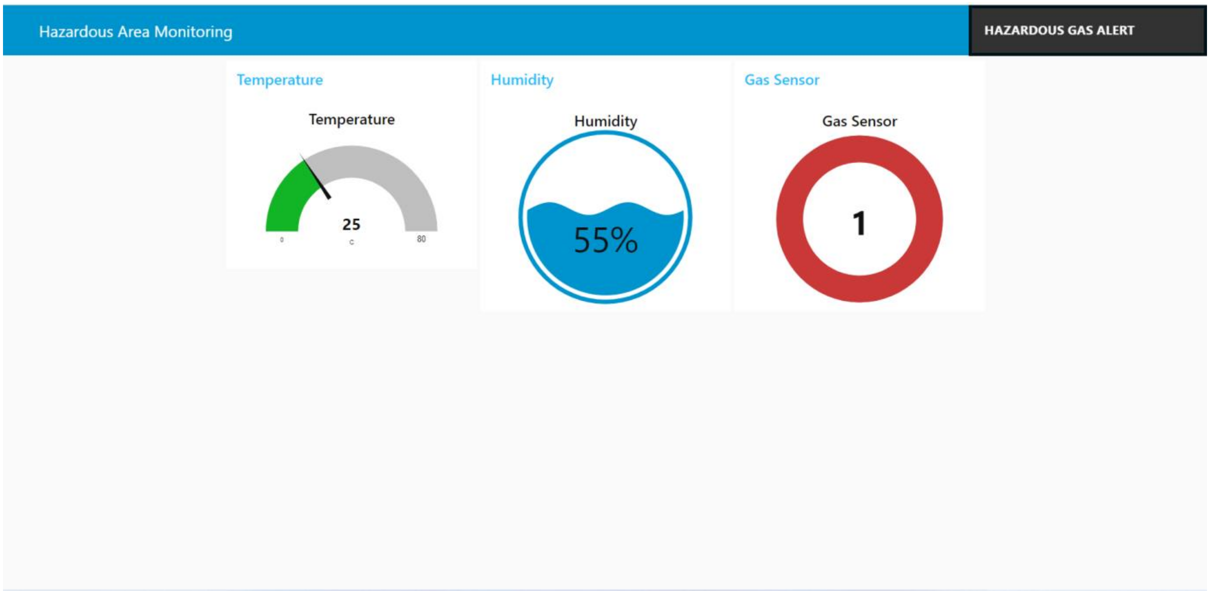
Node Red:

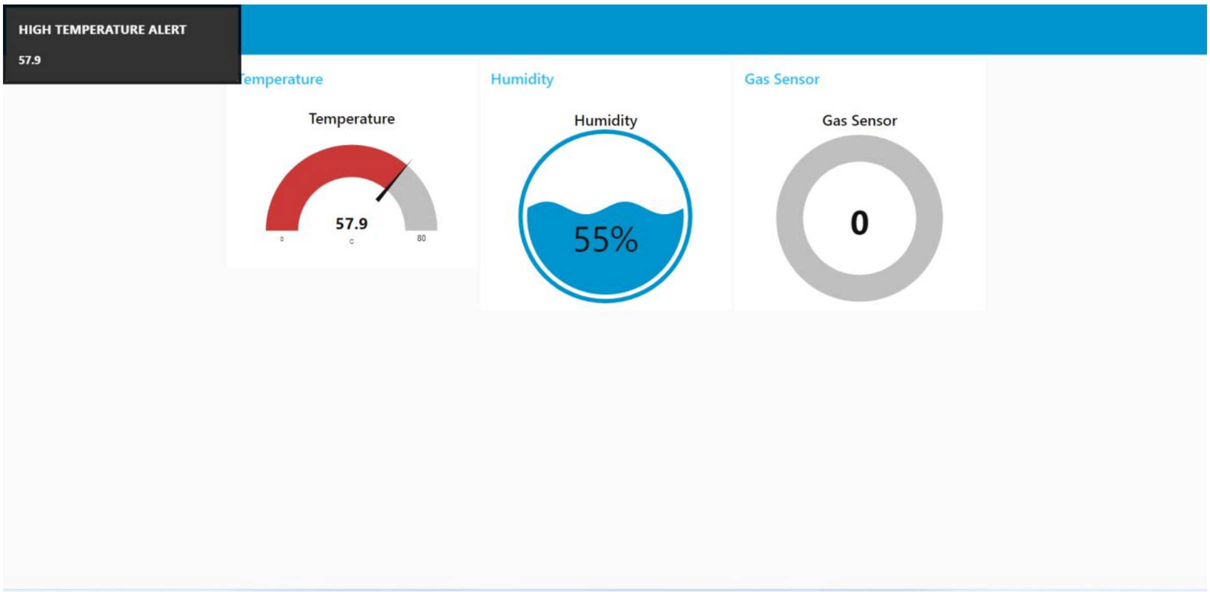


```
11/7/2022, 7:12:05 PM   node: 7630b4556b138fa3
iot-2/type/ESP32/id/Assign4/evt/Data/fmt/json :
msg.payload : Object
 ▸{ Temperature: 25, Humid: 55,
Gas_Sensor: 0 }
11/7/2022, 7:12:09 PM   node: 7630b4556b138fa3
iot-2/type/ESP32/id/Assign4/evt/Data/fmt/json :
msg.payload : Object
 ▸{ Temperature: 25, Humid: 55,
Gas_Sensor: 0 }
11/7/2022, 7:12:11 PM   node: 7630b4556b138fa3
iot-2/type/ESP32/id/Assign4/evt/Data/fmt/json :
msg.payload : Object
 ▸{ Temperature: 25, Humid: 55,
Gas_Sensor: 0 }
11/7/2022, 7:12:13 PM   node: 7630b4556b138fa3
iot-2/type/ESP32/id/Assign4/evt/Data/fmt/json :
msg.payload : Object
 ▸{ Temperature: 25, Humid: 55,
Gas_Sensor: 0 }
```

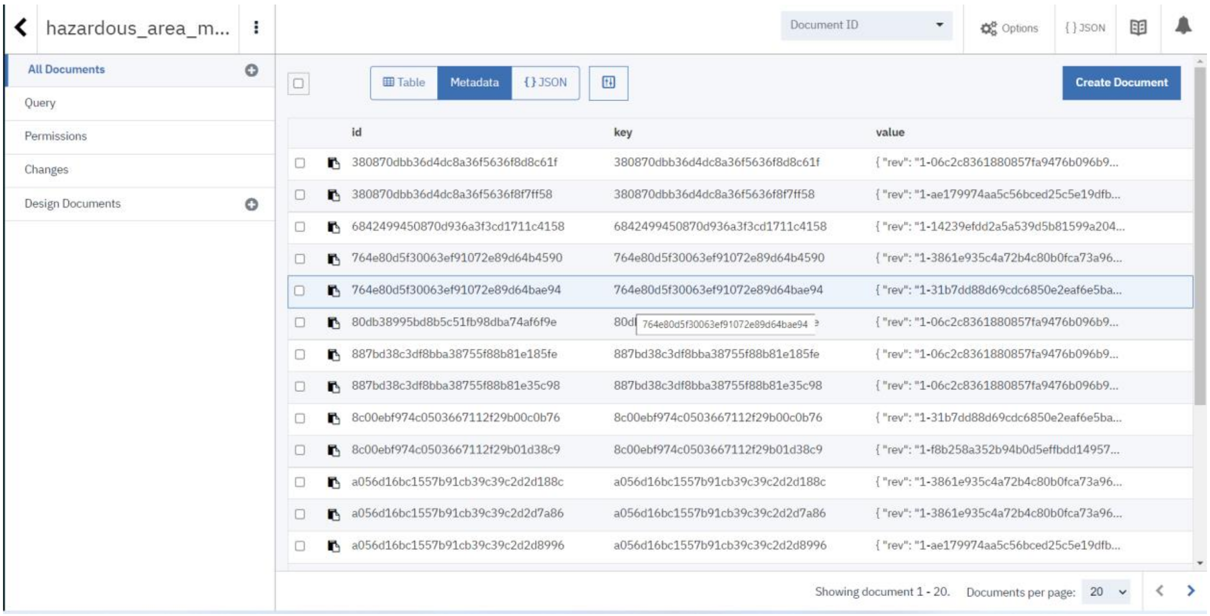
Node Red WEB UI:







CLOUDANT DATABASE:



	id	key	value
□	380870dbb36d4dc8a36f5636f8d8c61f	380870dbb36d4dc8a36f5636f8d8c61f	{ "rev": "1-06c2c8361880857fa9476b096b9...
□	380870dbb36d4dc8a36f5636f8f7ff58	380870dbb36d4dc8a36f5636f8f7ff58	{ "rev": "1-ae179974aa5c56bced25c5e19dfb...
□	6842499450870d936a3f3cd1711c4158	6842499450870d936a3f3cd1711c4158	{ "rev": "1-14239efdd2a5a539d5b81599a204...
□	764e80d5f30063ef91072e89d64b4590	764e80d5f30063ef91072e89d64b4590	{ "rev": "1-3861e935c4a72b4c80b0fca73a96...
□	764e80d5f30063ef91072e89d64bae94	764e80d5f30063ef91072e89d64bae94	{ "rev": "1-31b7dd88d69cdc6850e2eaf6e5ba...
□	80db38995bd8b5c51fb98dba74af6f9e	80d 764e80d5f30063ef91072e89d64bae94 e	{ "rev": "1-06c2c8361880857fa9476b096b9...
□	887bd38c3df8bba38755f88b81e185fe	887bd38c3df8bba38755f88b81e185fe	{ "rev": "1-06c2c8361880857fa9476b096b9...
□	887bd38c3df8bba38755f88b81e35c98	887bd38c3df8bba38755f88b81e35c98	{ "rev": "1-06c2c8361880857fa9476b096b9...
□	8c00ebf974c0503667112f29b00c0b76	8c00ebf974c0503667112f29b00c0b76	{ "rev": "1-31b7dd88d69cdc6850e2eaf6e5ba...
□	8c00ebf974c0503667112f29b01d38c9	8c00ebf974c0503667112f29b01d38c9	{ "rev": "1-f8b258a352b94b0d5effbdd14957...
□	a056d16bc1557b91cb39c39c2d2d188c	a056d16bc1557b91cb39c39c2d2d188c	{ "rev": "1-3861e935c4a72b4c80b0fca73a96...
□	a056d16bc1557b91cb39c39c2d2d7a86	a056d16bc1557b91cb39c39c2d2d7a86	{ "rev": "1-3861e935c4a72b4c80b0fca73a96...
□	a056d16bc1557b91cb39c39c2d2d8996	a056d16bc1557b91cb39c39c2d2d8996	{ "rev": "1-ae179974aa5c56bced25c5e19dfb...

Showing document 1 - 20. Documents per page: 20