Sprint - 3

Date	04 November 2022
Team ID	PNT2022TMID16771
Project Name	Project - Industry-specific intelligent fire management system
Maximum Marks	20 marks

US-1 Configure the connection security and create API keys that are used in the Node-RED service for accessing the IBM IoT Platform.

US-2 Create a Node-RED service.

WEB UI LINK : https://node-red-dashboard059.eu-gb.mybluemix.net/fire

LINK: https://wokwi.com/projects/347685130732569171

Welcome to Project! Delighted to | IBM | Industry - Specific Intelligent Fire

wokwi.com/projects/347685130732569171

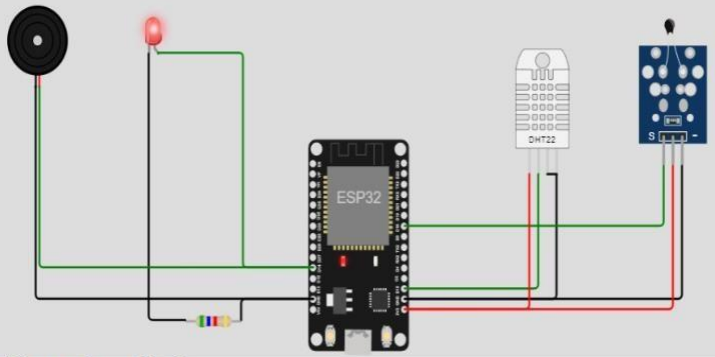
WOKWI SAVE SHARE Docs

sketch.ino | diagram.json | libraries.txt | Library Manager

```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQtt
#include "DHT.h"// Library for dht sensor
#define DHTPIN 15    // what pin we're connected to
#define DHTTYPE DHT22   // define type of sensor DHT 22
#define LED 14

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

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

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

#define ORG "88653s"//IBM ORGANITION ID
#define DEVICE_TYPE "iot_device"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "wokwi_us"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN ")l(u!YYO)NmKr9sk(k"   //Token
String data3;
float h, t;
const float BETA = 3950; // should match the Beta Coefficient of the thermistor


//-------- 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
char subscribetopic[] = "iot-2/cmd/test/fmt/String"; // cmd  REPRESENT command type AND C
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

```

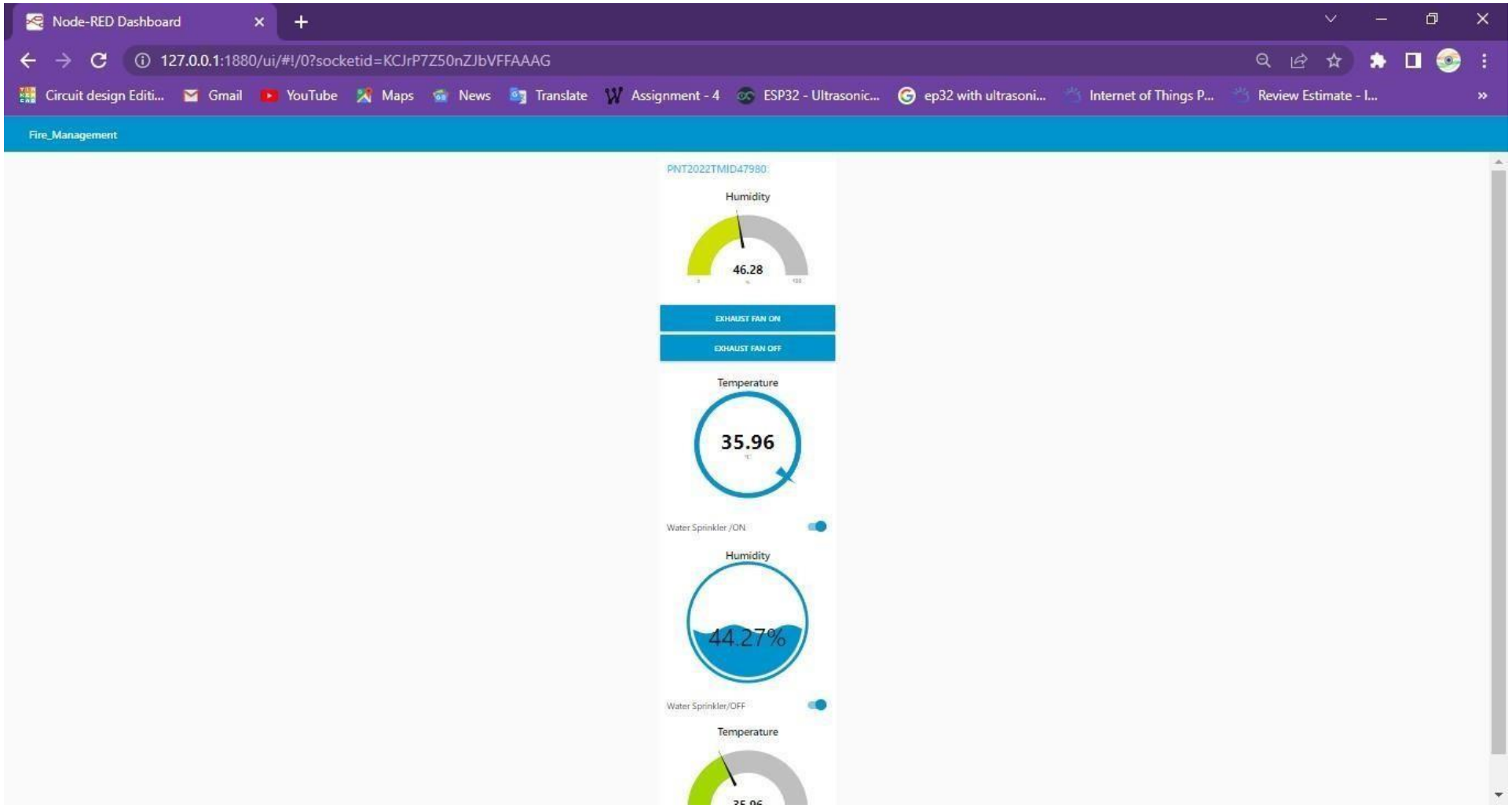
Simulation

00:12.999 98%




```
Alert..!Temperature:36.40
Humidity:46.50
Sending payload: {"Data":{"temperature":36.40,"humidity":46.50}}
Publish ok
If Temperature increased,the alarm and alert light would indicates.
Temperature: 36.40 ℃
Alert..!
```

25°C Mostly clear | Search | ENG IN | 00:05 19-11-2022

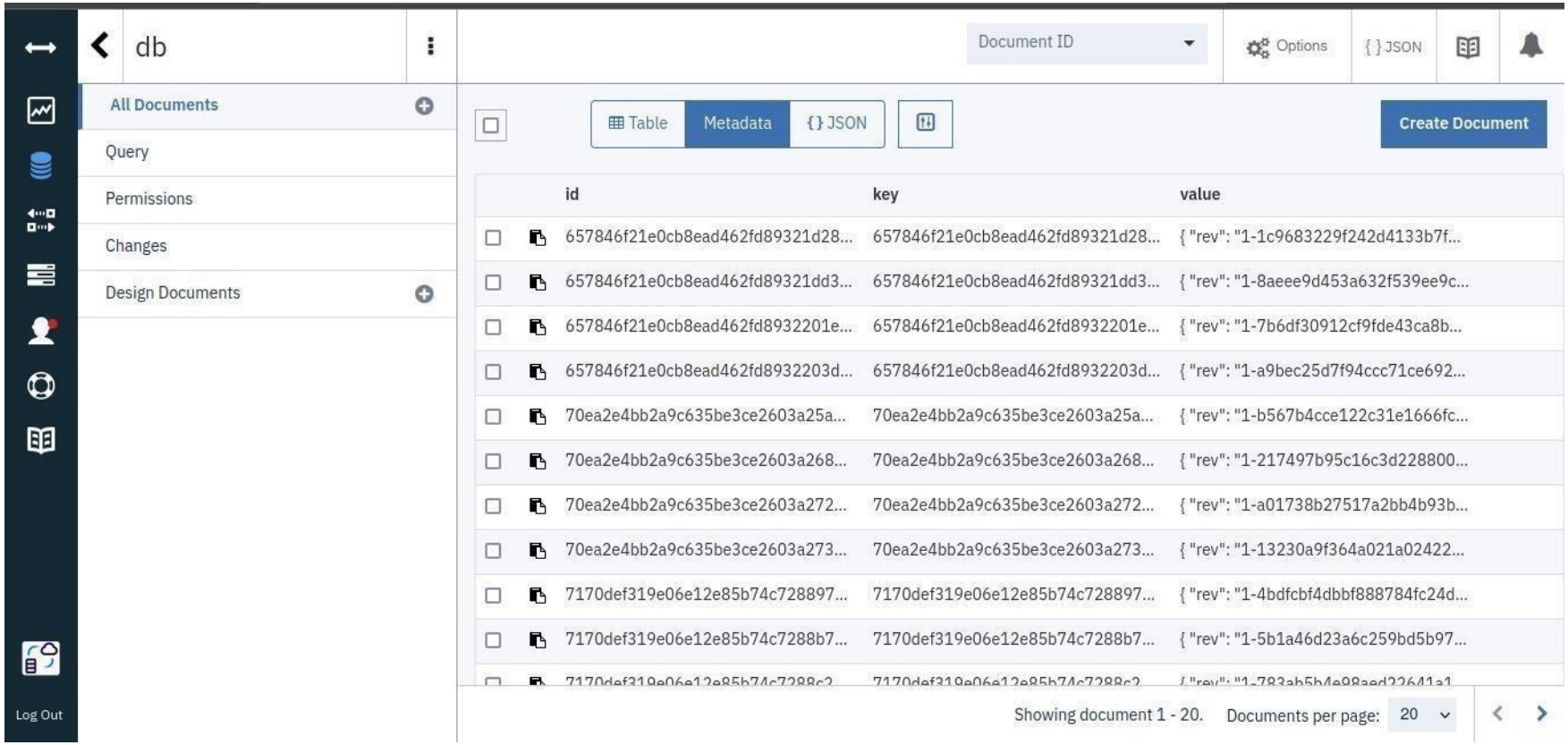


TRANSFERRING DATA FROM NODE-RED INTO WEB UI

DESKTOP:


https://node-red-dashboard059.eu-gb.mybluemix.net/fire
FIRE MANAGEMENT SYSTEM
TEMPERATURE : 56
GAS : 721
FLAME : 299
FIRE STATUS : No Fire
SPRINKLER STATUS: Not Working
GAS STATUS : Gas Leakage is Detected
EXHAUST FAN STATUS : Working

CLOUDANT



WOKWI CODE

```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQtt
#include "DHT.h"// Library for dht sensor
 #define DHTPIN 15    // what pin we're connected to
#define DHTTYPE DHT22 // define type of sensor DHT 22
#define LED 14
```

```
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 "88653s"//IBM ORGANITION ID
#define DEVICE_TYPE "iot_device"//Device type mentioned in ibm watson IOT
Platform
#define DEVICE_ID "wokwi_us"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN ")l(u!YYO)NmKr9sk(k" //Token
String data3; float
h,  t;  const  float
BETA  =  3950;  //
should  match  the
```

```
Beta Coefficient of
the thermistor

//-------- 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/test/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
 void setup() // configureing the ESP32

{
  Serial.begin(115200);
  dht.begin(); delay(10);
  Serial.println();
  wificonnect();
  mqttconnect();
  Serial.begin(9600);
```

```
  analogReadResolution(10
  ); pinMode(18,INPUT);
  pinMode(14,OUTPUT);
  pinMode(12,OUTPUT);
}
 void loop() // Recursive
Function
{
 h = dht.readHumidity(); t =
  dht.readTemperature();
  Serial.print("Temperature:")
  ;
  Serial.println(t);
  Serial.print("Humidity:");
  Serial.println(h);

  PublishData(t, h);
  delay(1000); if
  (!client.loop()) {
    mqttconnect();
  }

//..............Analog Temperature Sensor.....................
 int analogValue = analogRead(18);
  float celsius = 1 / (log(1 /
  (1023. / analogValue - 1)) /
  BETA + 1.0 / 298.15)
```

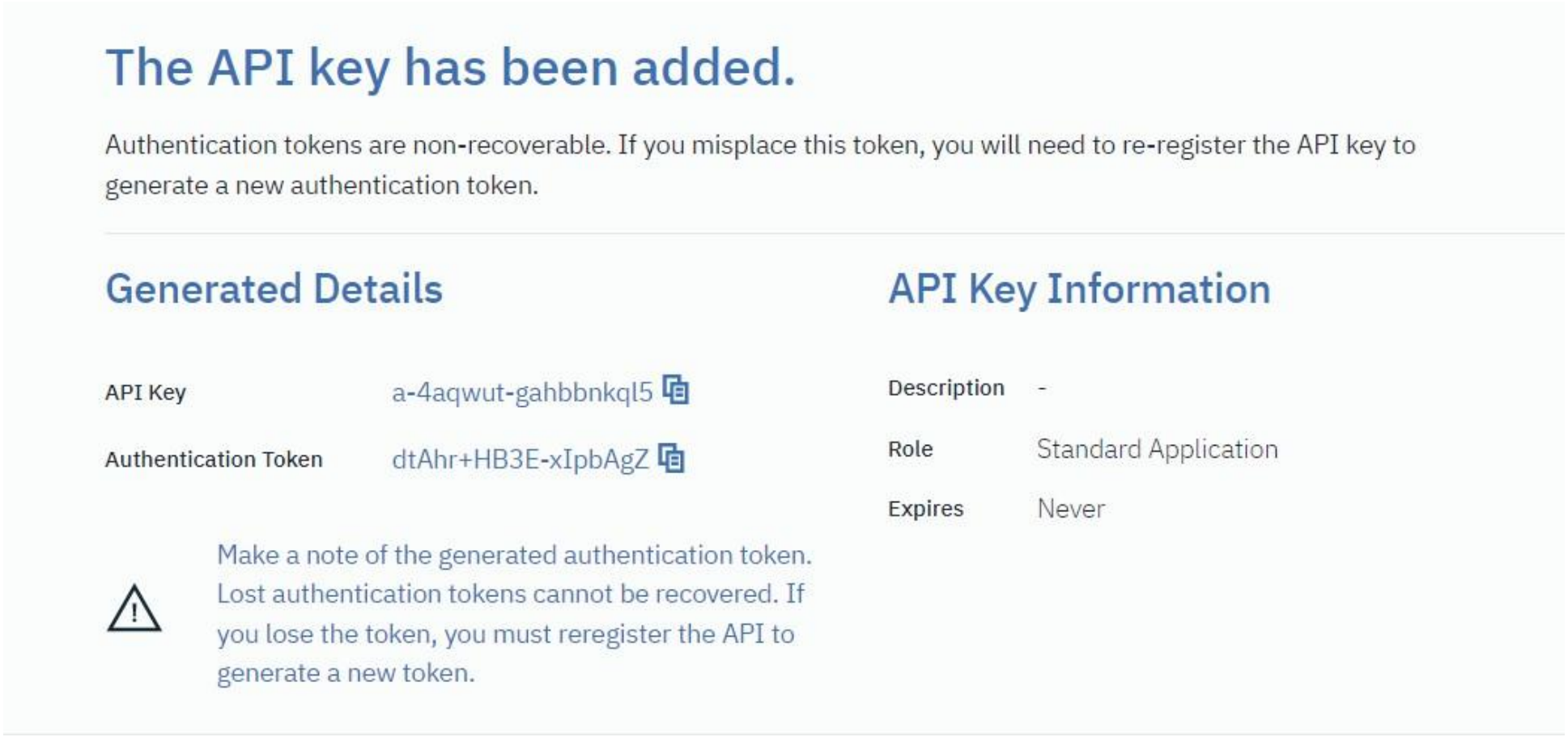
```
+ 36.4;
  Serial.print("Temperature: ");
  Serial.print(celsius);
  Serial.println(" ℃");
  Serial.print("Alert..!");
 if(celsius >= 35)
    digitalWrite(14, HIGH);
  else
    digitalWrite(14, LOW);
  delay(1000);

}

/*.....................................retrieving to
Cloud ..............................*/
 void PublishData(float temp, float humid)
{
  mqttconnect(); //function call for connecting to ibm

  /* creating the String in in form JSon to update the data to ibm cloud
  */

  String payload = "{\"Data\":{\"temperature\":";
  payload += temp;
  payload += "," "\"humidity\":";
  payload += humid; 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
    Serial.println("If Temperature increased,the alarm and alert light would
indicates. ");
  } 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); digitalWrite(LED,HIGH);

}
el
se
{
          Serial.println(data3);
          digitalWrite(LED,LOW);

  } data3="";
}
```

US-1 Configure the connection security and create API keys that are used in the Node-RED service for accessing the IBM IoT Platform.



US-2 Create a Node-RED service

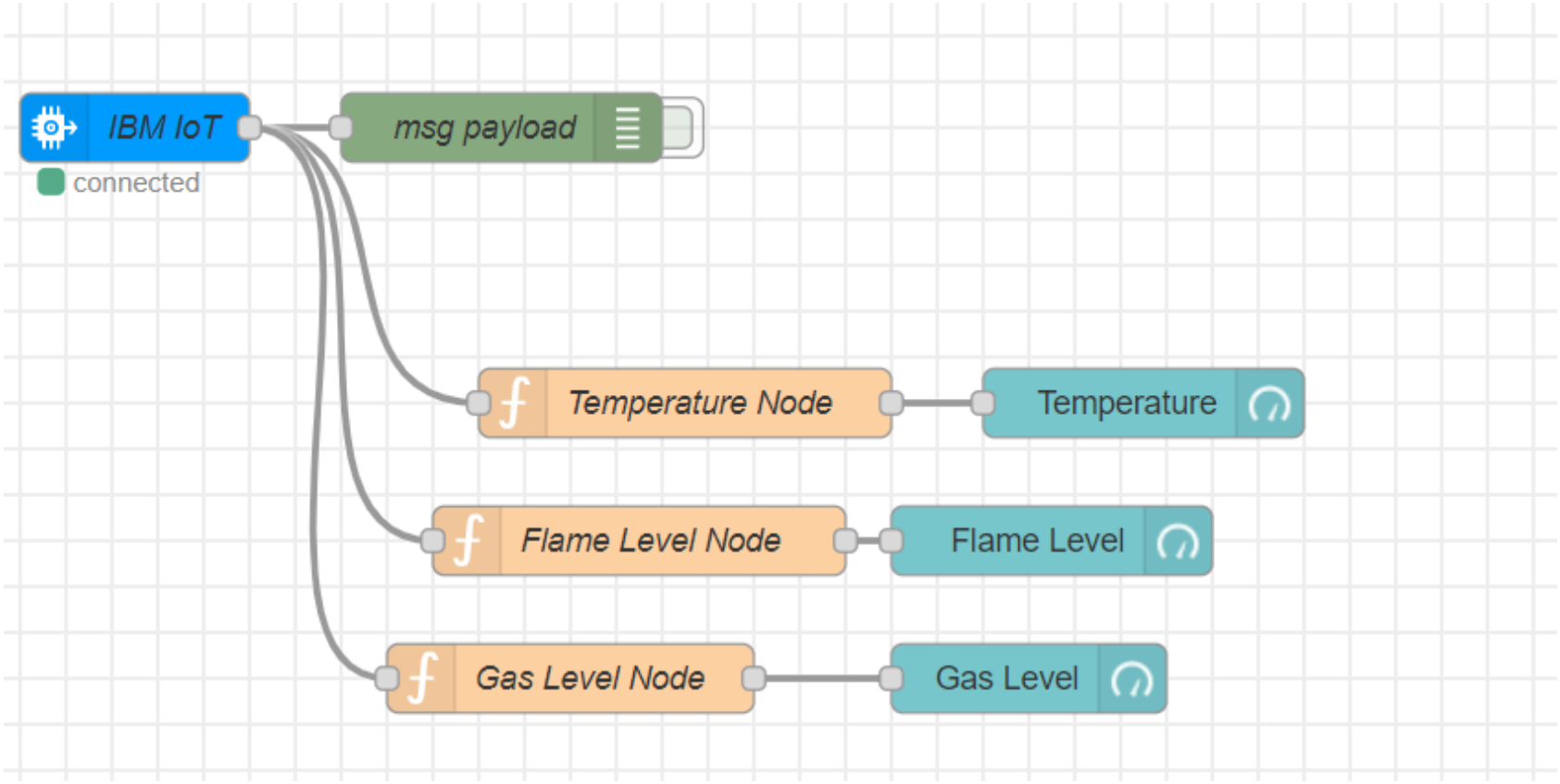


fig1 - Monitoring the sensor values - Temperature, Flame Level, Gas Level. These values are randomly generated by IBM WATSON IOT PLATFORM.

Fig 2 - Temperature, Flame_Level, Gas_Level values displayed in deploy tab in node-red

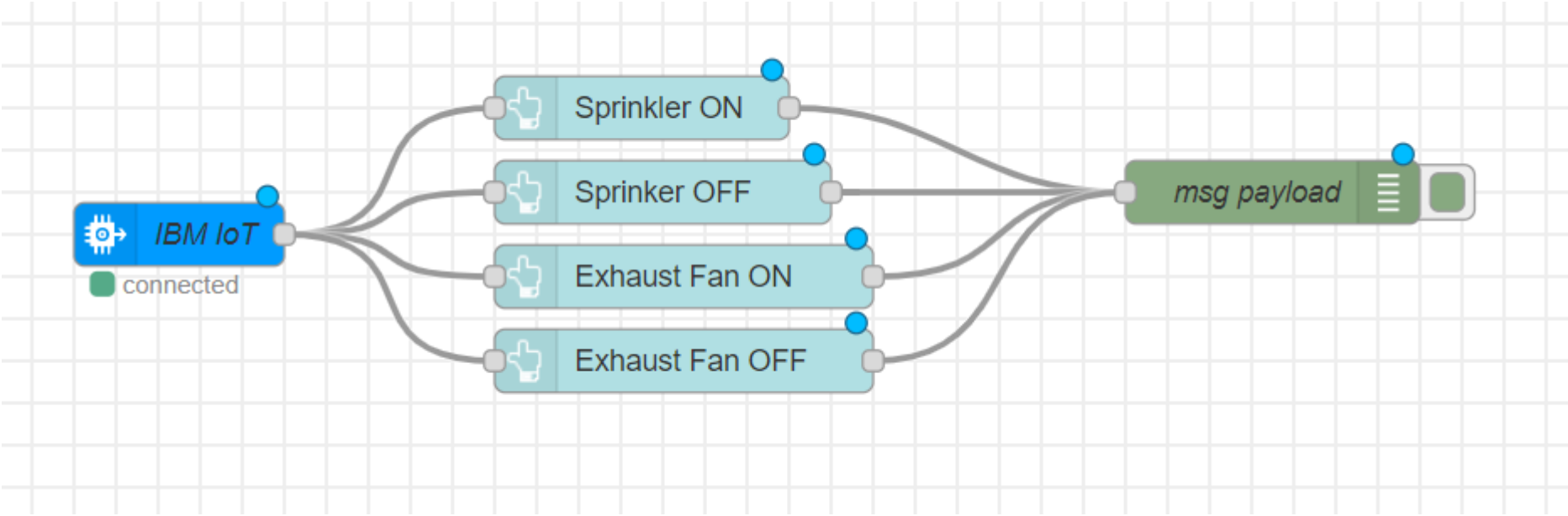


fig 3 - Control buttons (Sprinkler ON, Sprinkler OFF, Exhaust Fan ON, Exhaust Fan OFF)

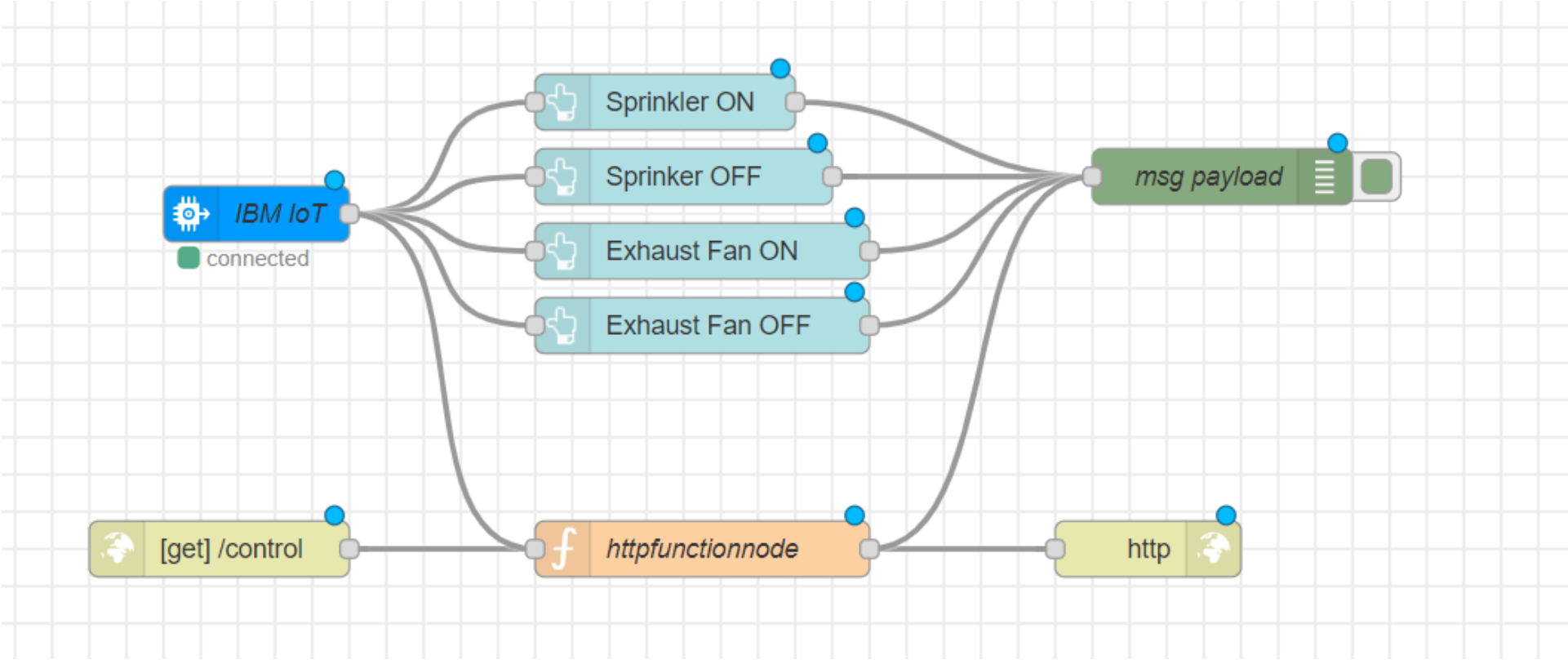


Fig 4 - Using HTTP in and HTTP response in network option, http://127.0.0.1:1880/#flow/f74f1b96473dc208/control will display the control options

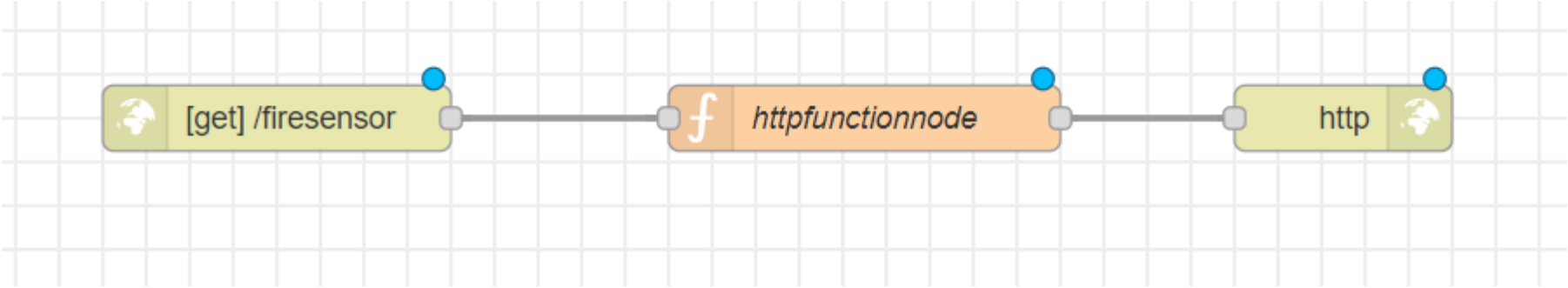


Fig 5 - Using HTTP in and HTTP response in network option, http://127.0.0.1:1880/#flow/f74f1b96473dc208/firesensor will display the sensor values like Temperature, Gas_Level and Flame_Level from the IBM WATSON IOT PLATFORM.

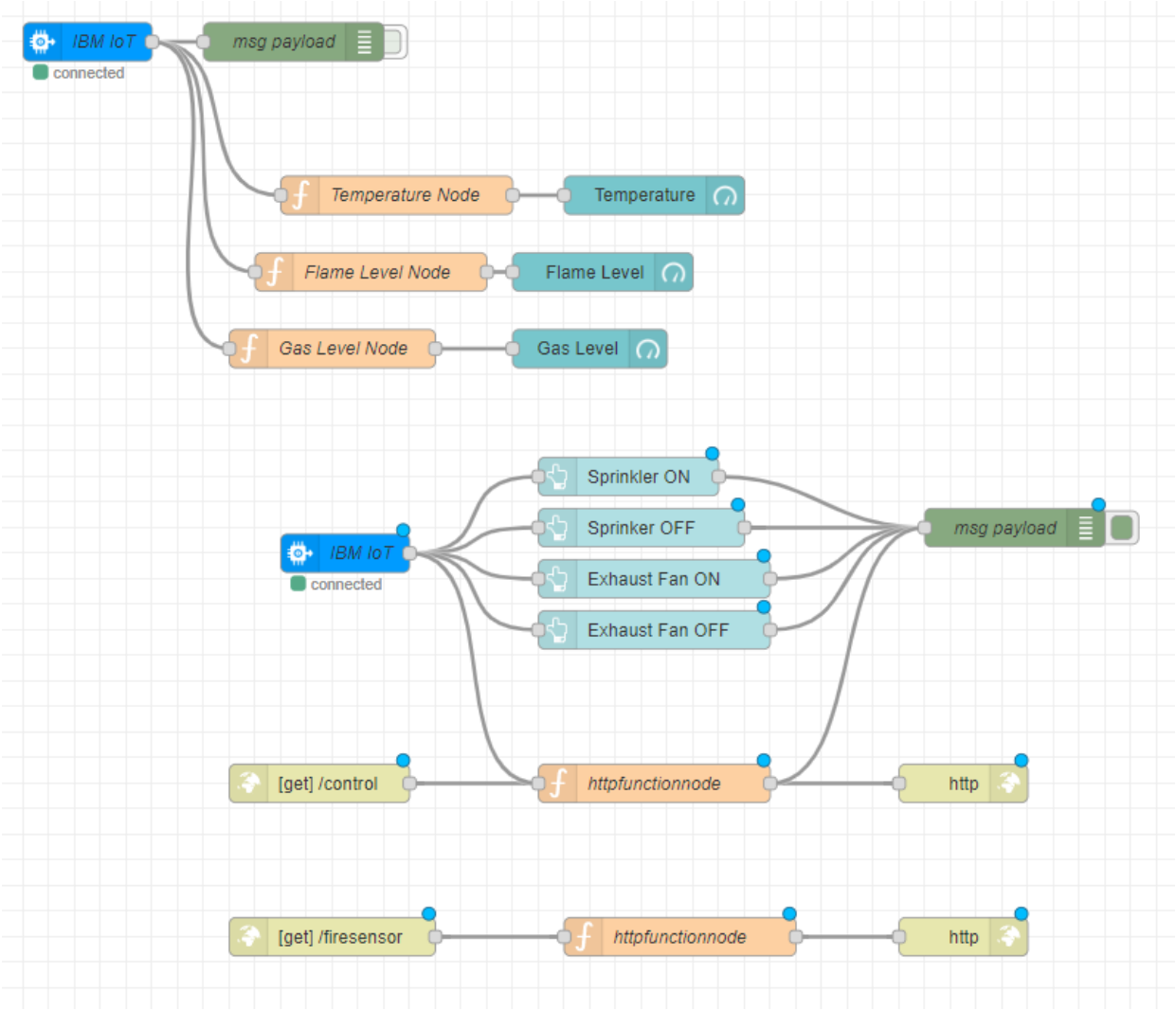


Fig 6 - Entire Node-Red connection for our project

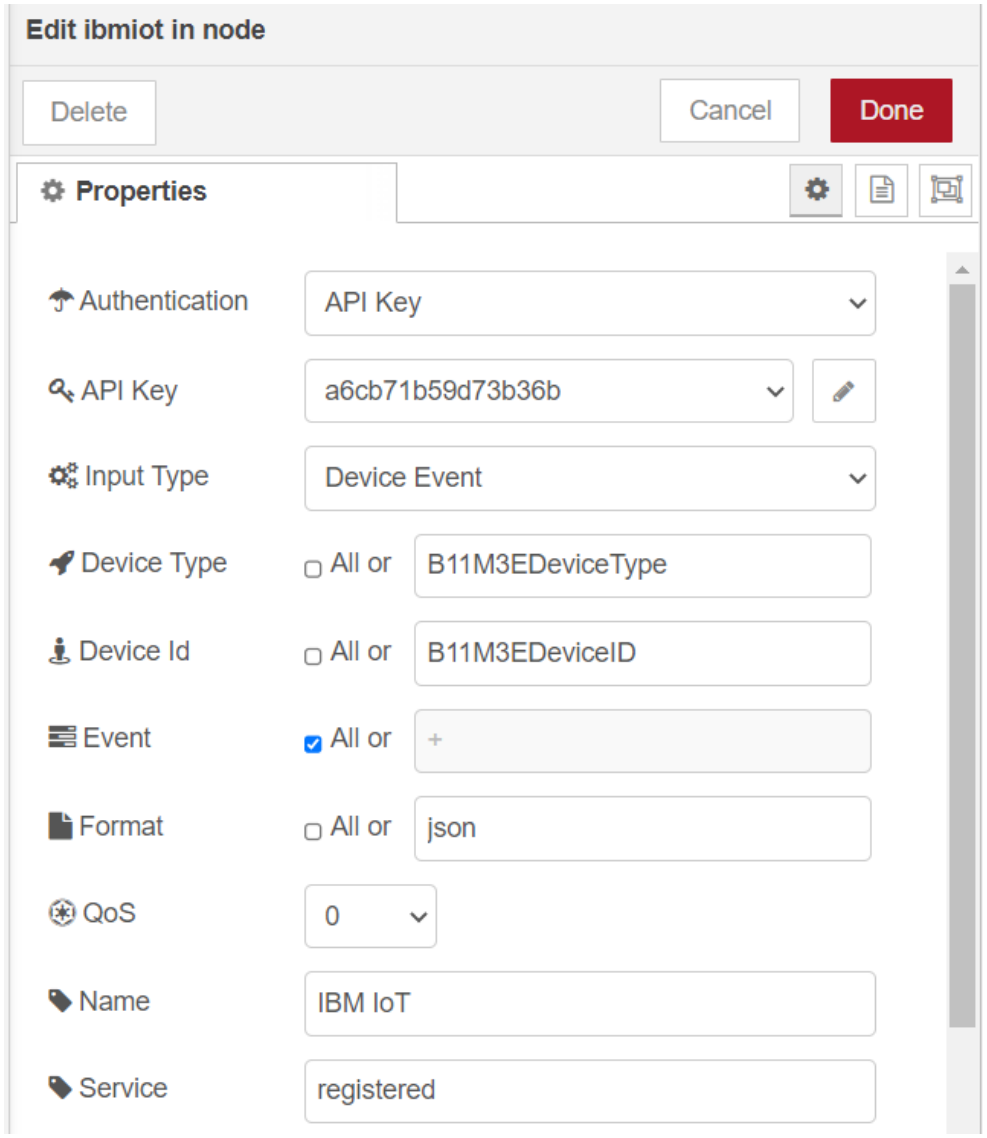
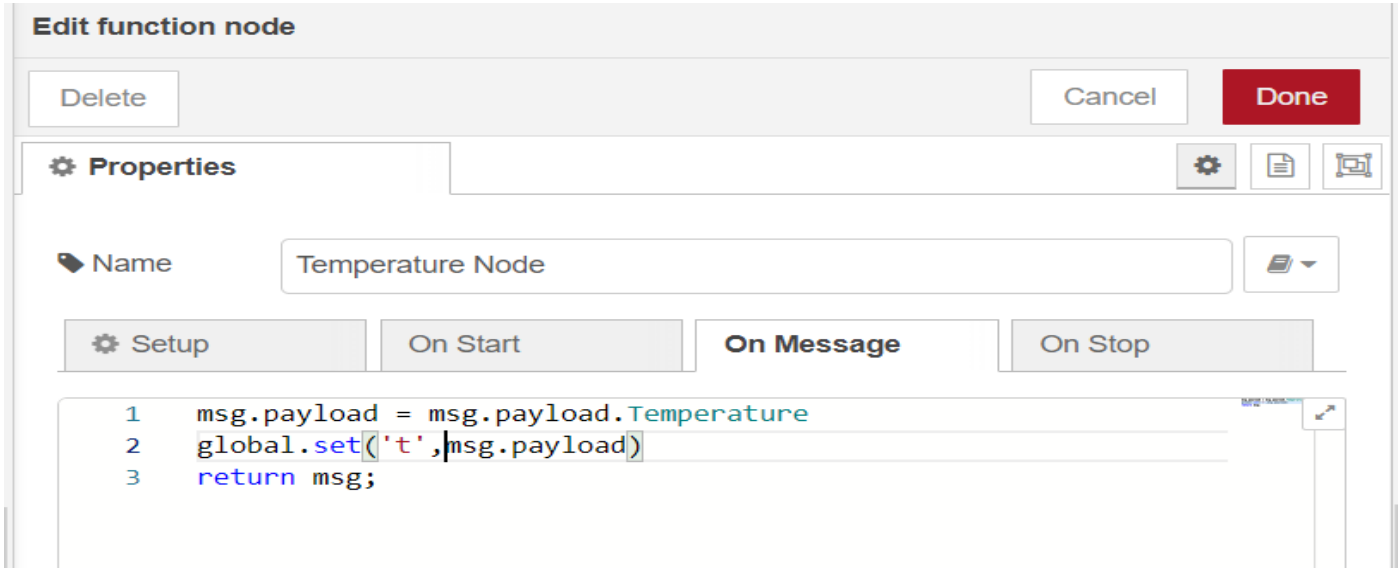
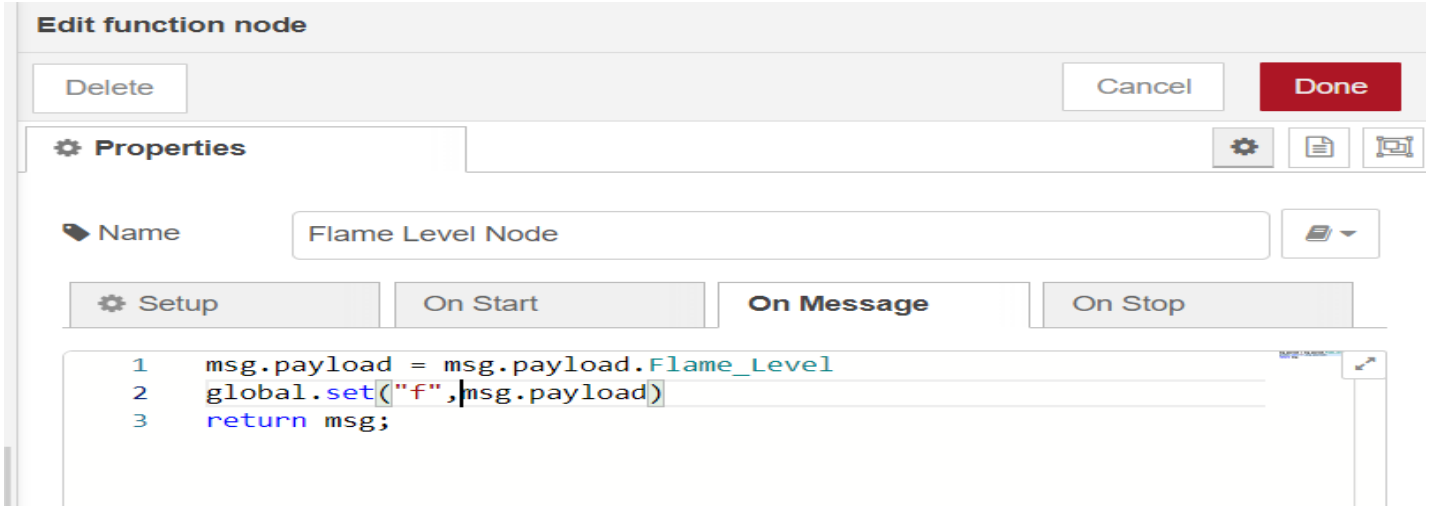


Fig 7 - Properties of IBM IOT are shown. The API key, Device Type, Device ID are taken from IBM IOT WATSON PLATFORM.




Edit function node
Delete
Cancel
Done
Properties
Name
Flame Level Node
Setup
On Start
On Message
On Stop
1 msg.payload = msg.payload.Flame_Level
2 global.set("f",msg.payload)
3 return msg;

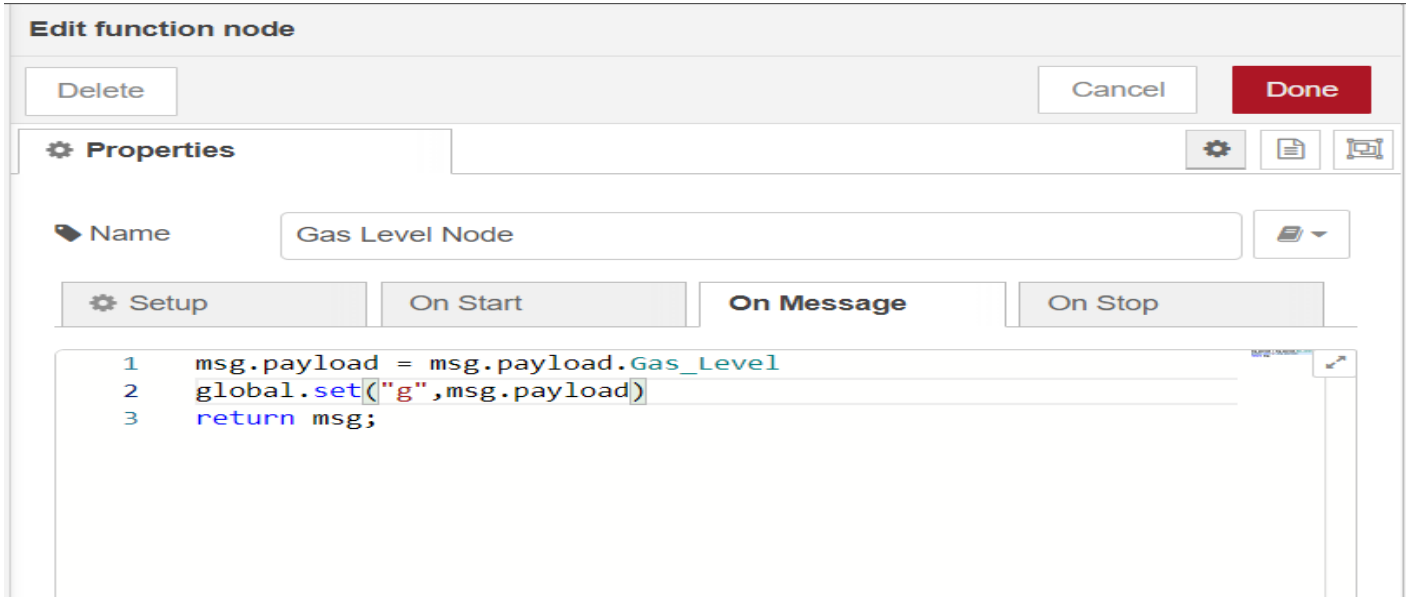

Edit function node
Delete
Cancel
Done
Properties
Name
Gas Level Node
Setup
On Start
On Message
On Stop
1 msg.payload = msg.payload.Gas_Level
2 global.set("g",msg.payload)
3 return msg;

Fig 8 - Properties of Function Node -Temperature Node, Flame_Level Node, Gas_Level Node.

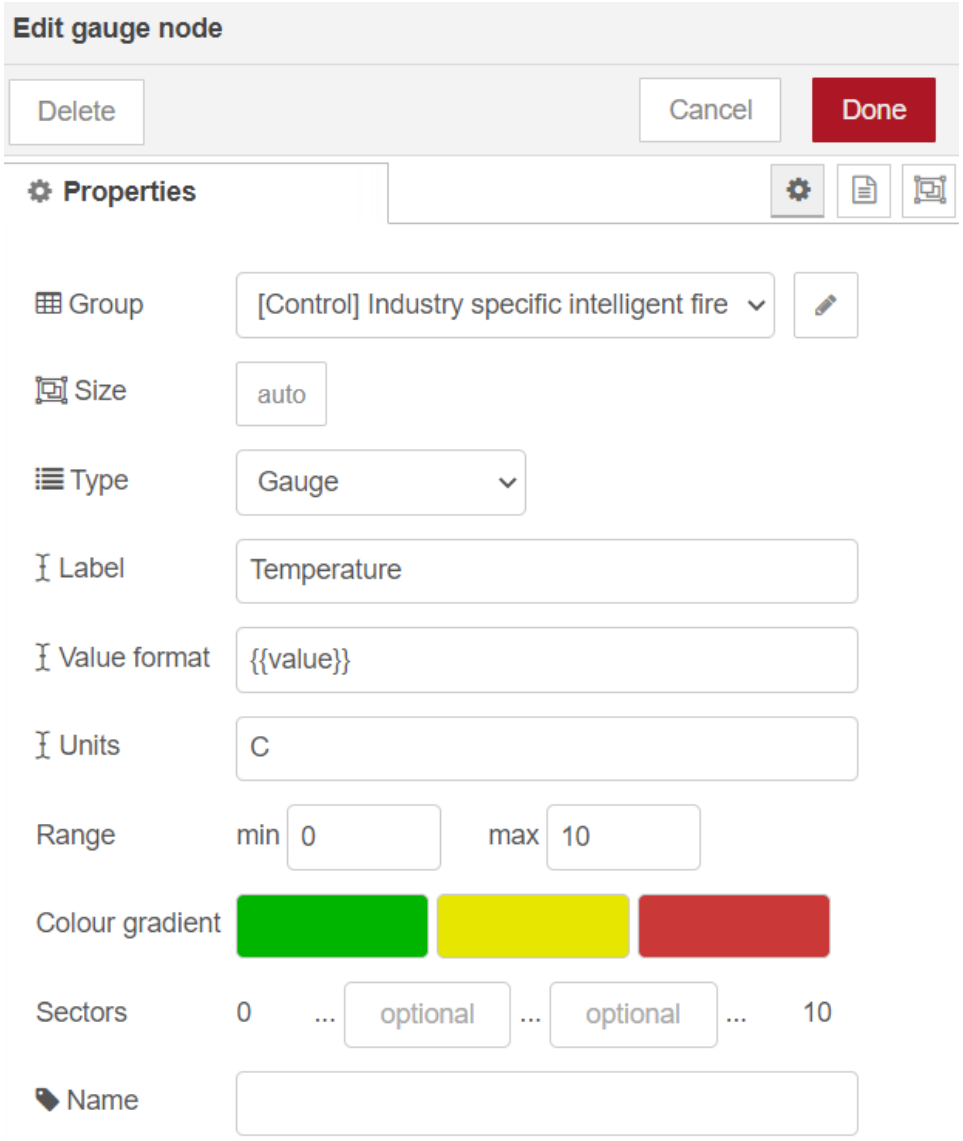


Fig 9 - Properties of Temperature Gauge.

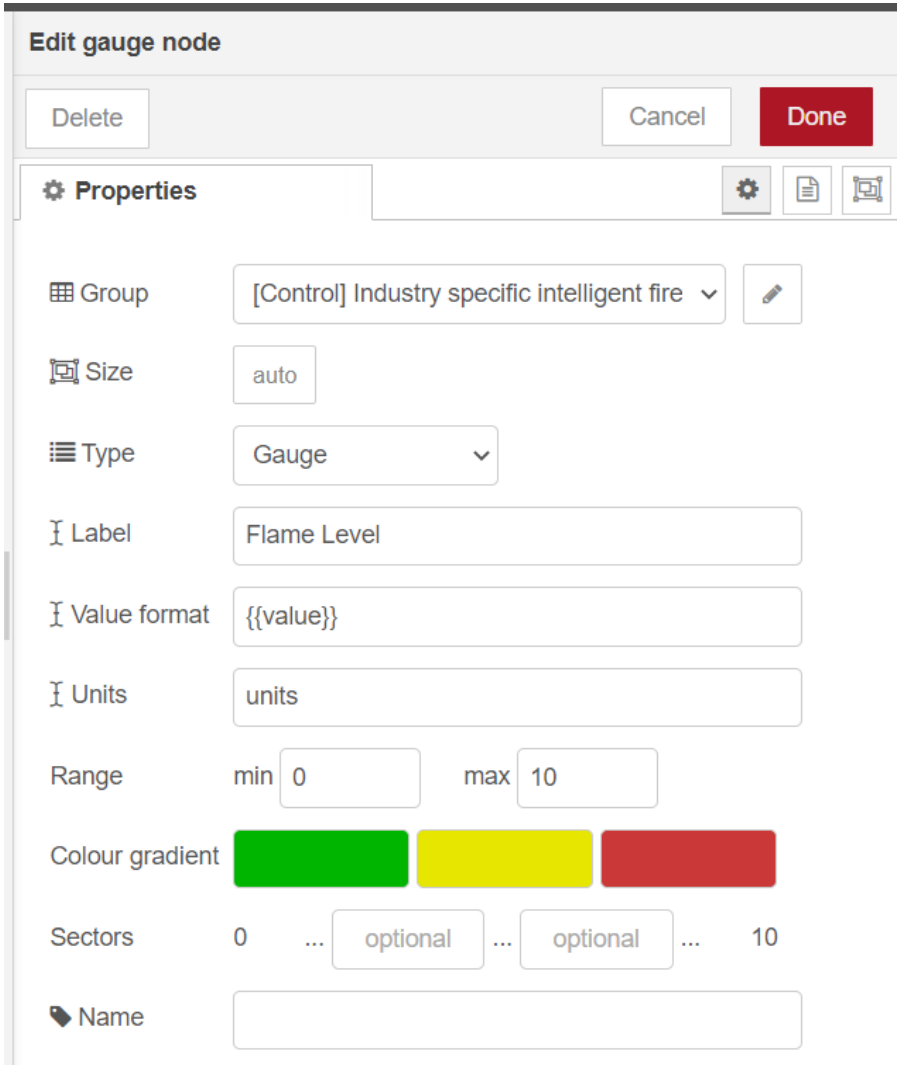


Fig 9 - Properties of Flame_Level Gauge.

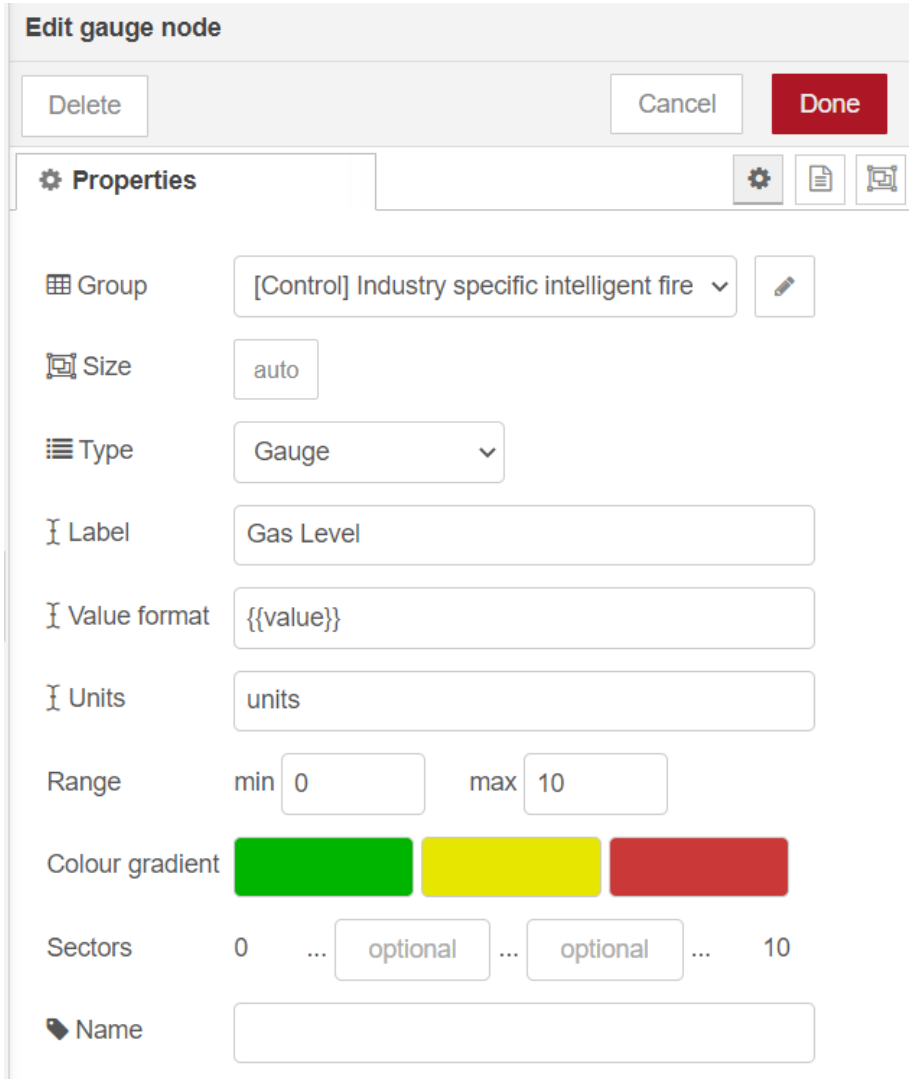


Fig 9 - Properties of Gas_Level Gauge.

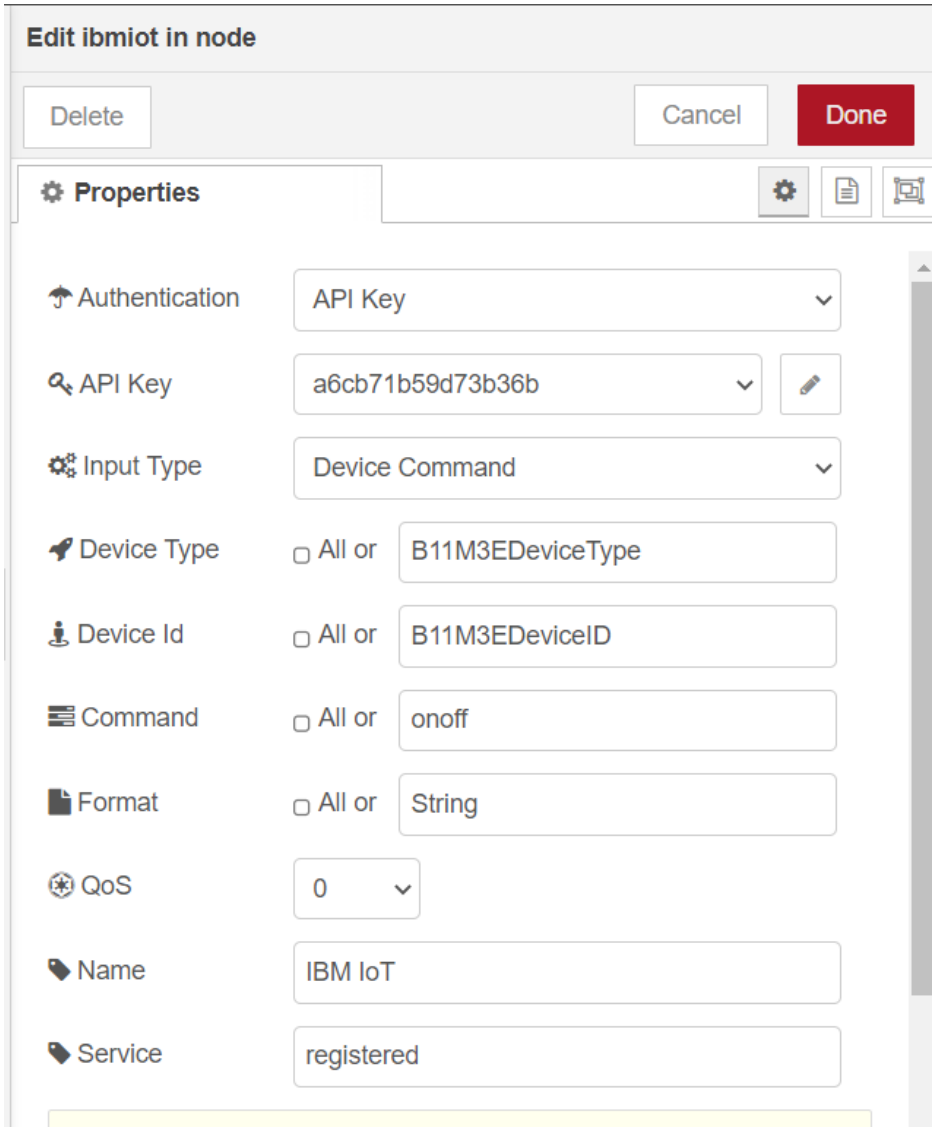


Fig 9 - Properties of IBM IOT Node.

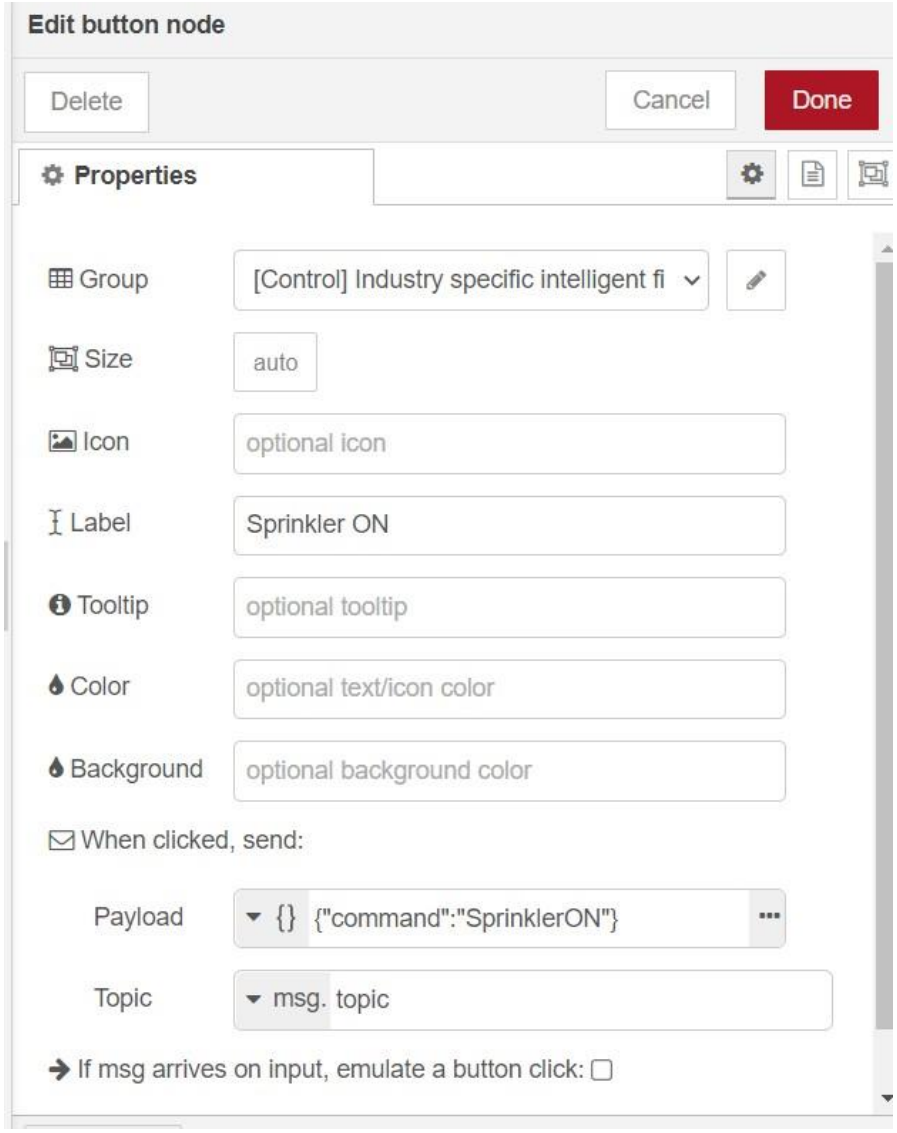


Fig 10 - Properties of Sprinkler ON button node.

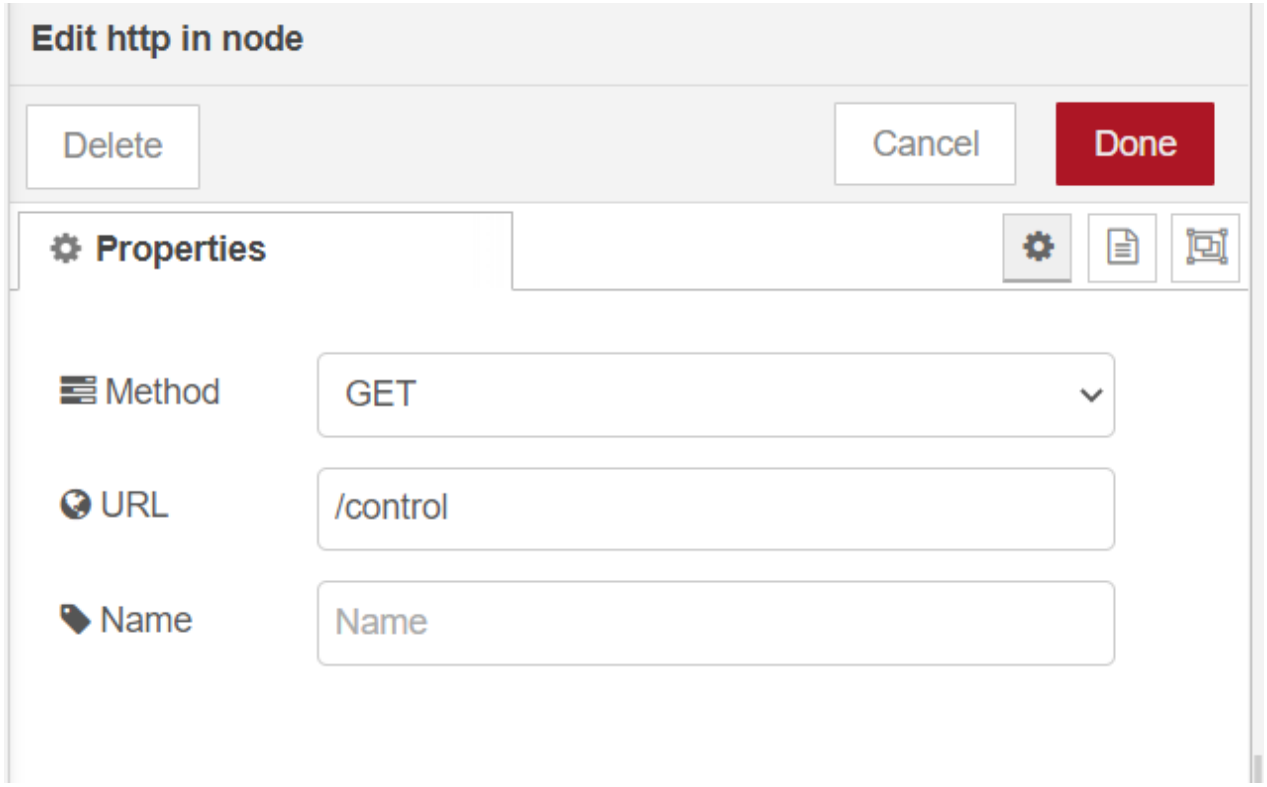


Fig 10 - Properties of HTTP Node with method GET and URL /control,

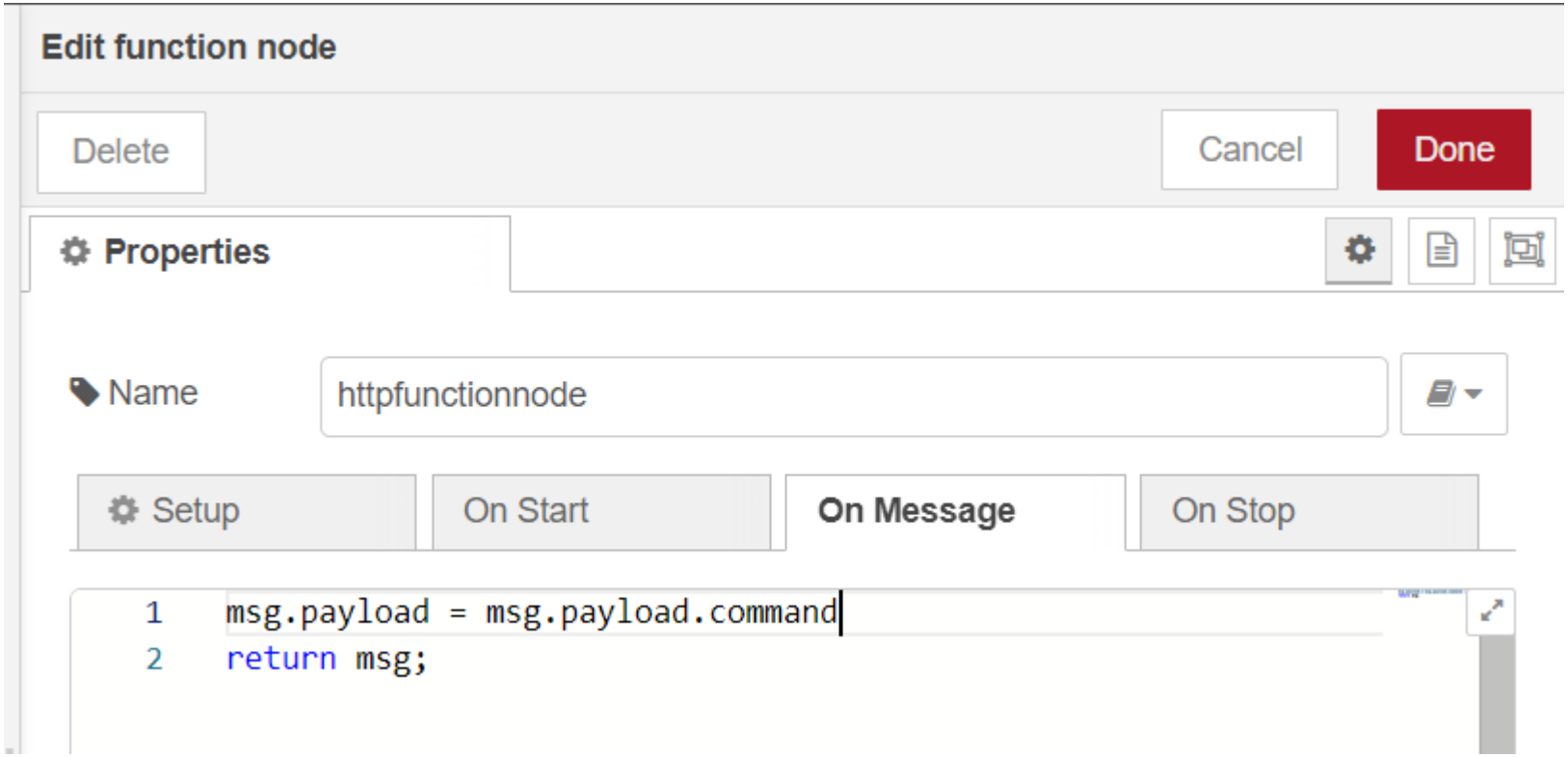
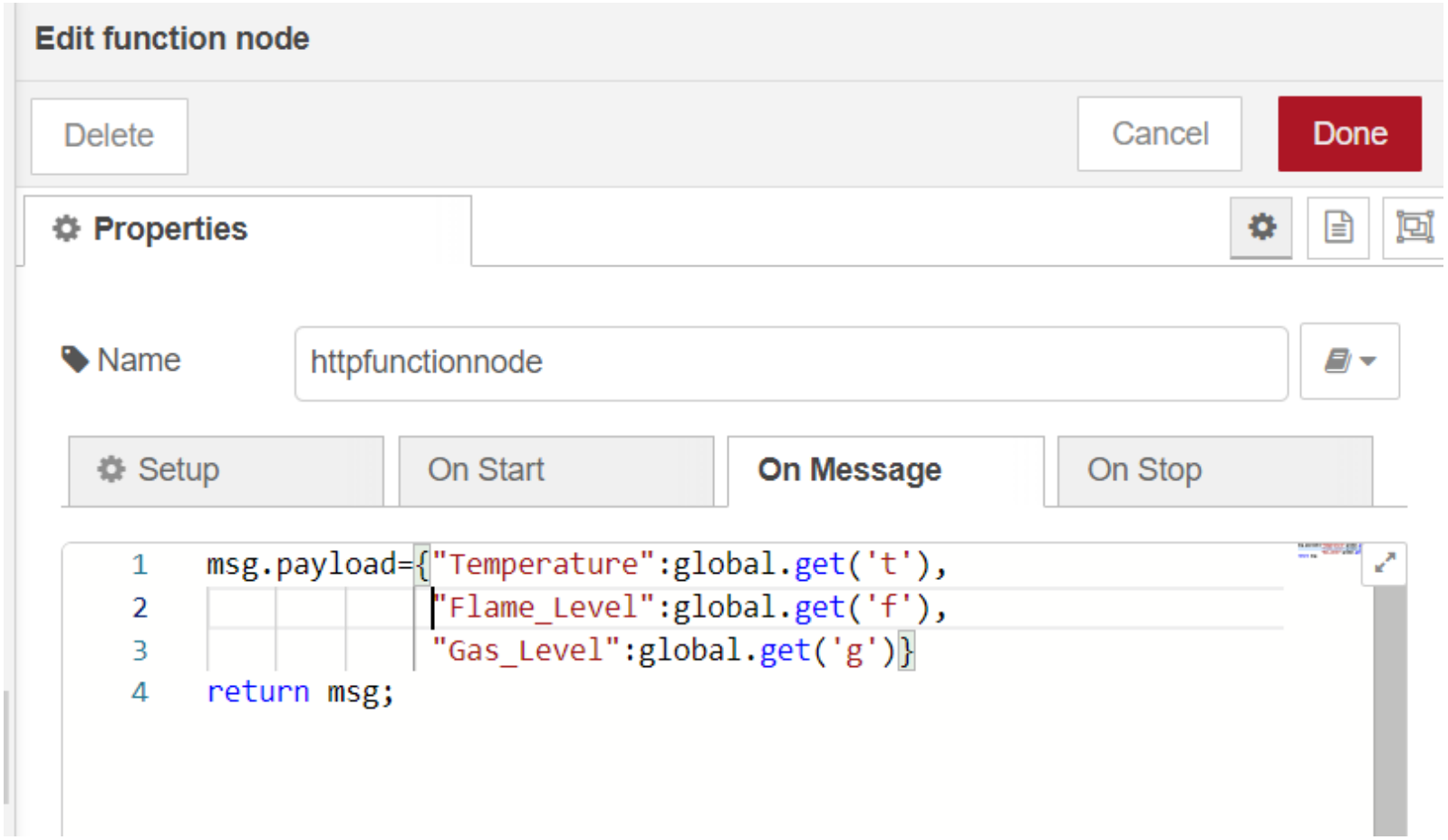
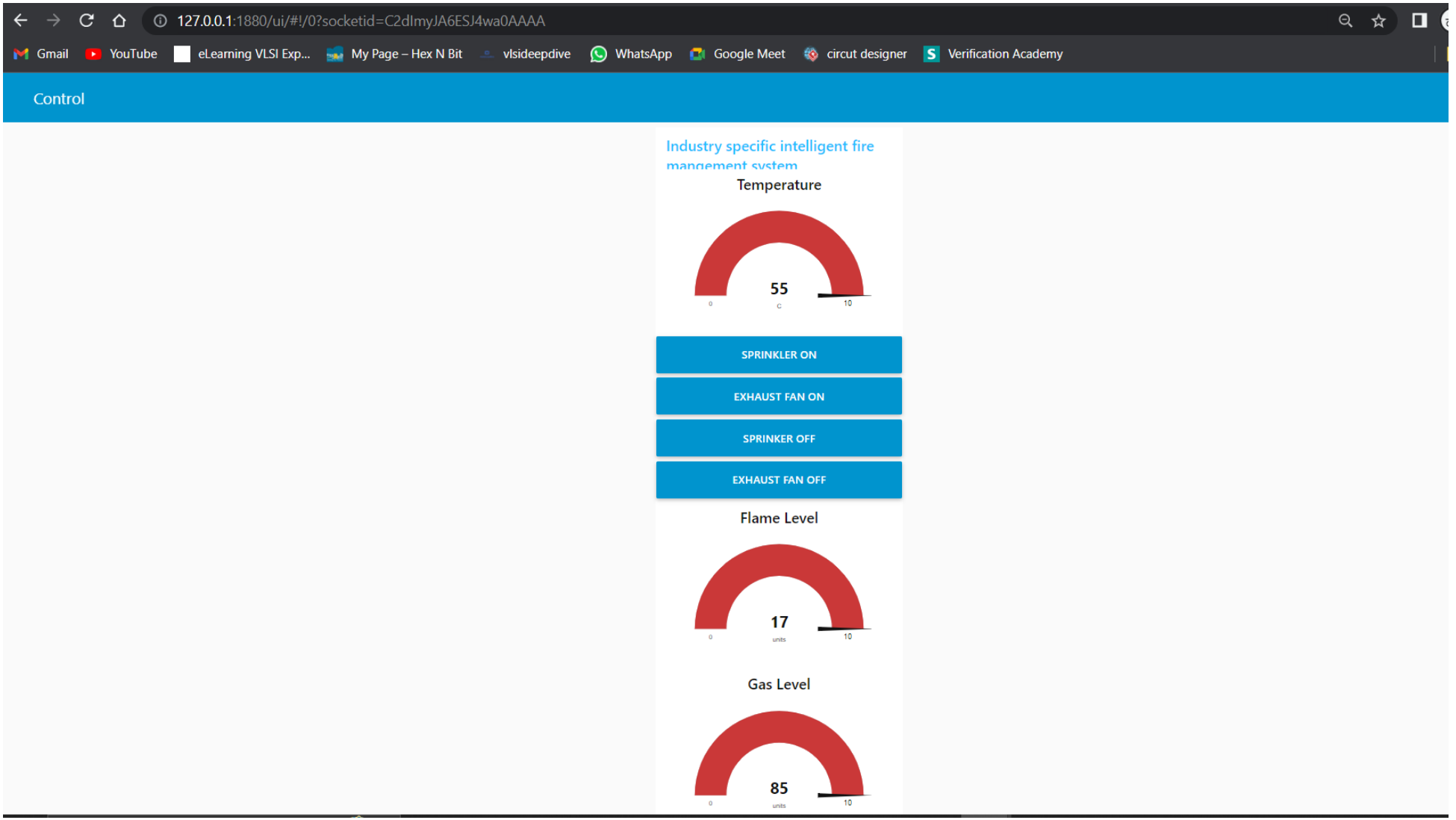


Fig 11 - Properties of Control HTTP Function Node.




127.0.0.1:1880/ui/#!/0?socketid=C2dImyJA6ESJ4wa0AAAA
Gmail
YouTube
eLearning VLSI Exp...
My Page – Hex N Bit
vlsideepdive
WhatsApp
Google Meet
circut designer
Verification Academy
Control
Industry specific intelligent fire mangement system
Temperature
55
C
0
10
SPRINKLER ON
EXHAUST FAN ON
SPRINKER OFF
EXHAUST FAN OFF
Flame Level
17
units
0
10
Gas Level
85
units
0
10

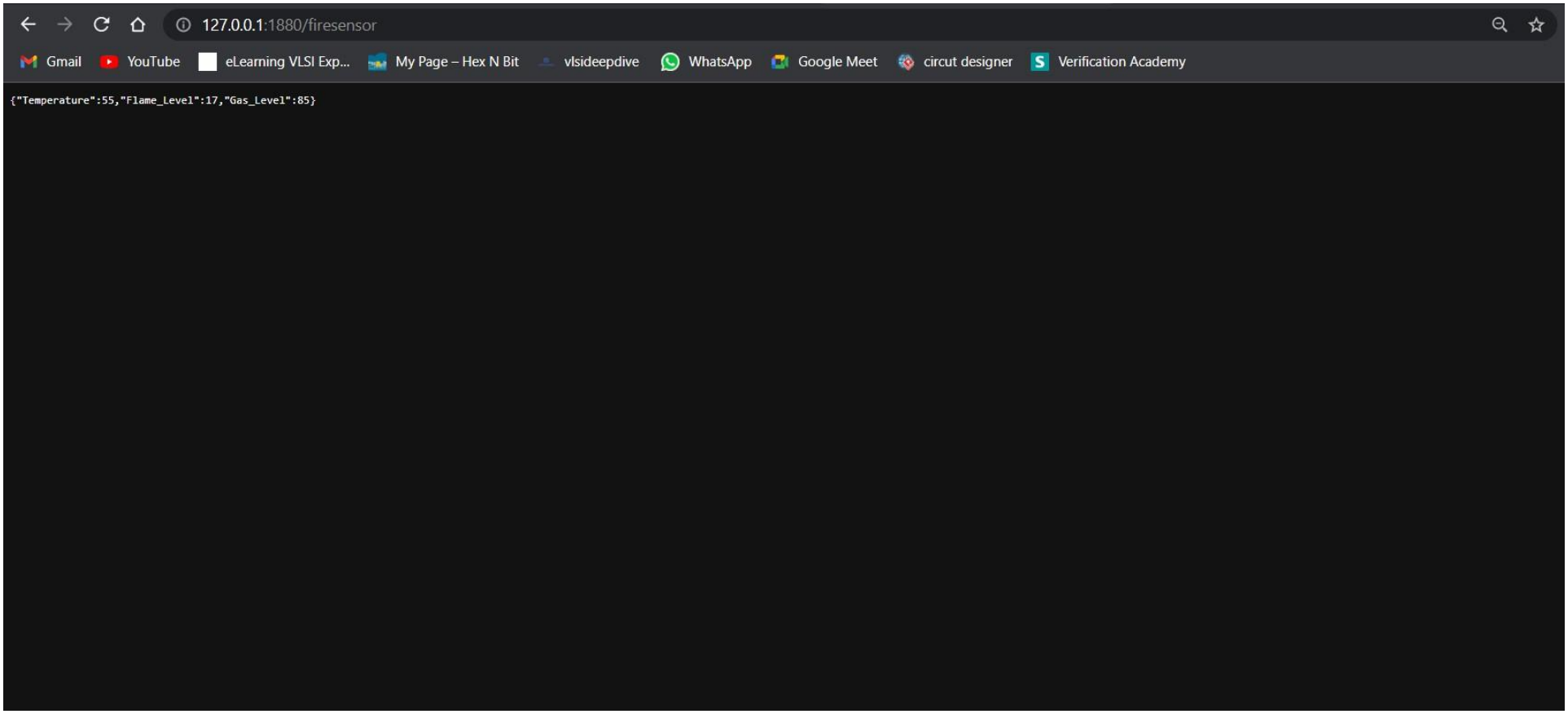

127.0.0.1:1880/firesensor
Gmail
YouTube
eLearning VLSI Exp...
My Page – Hex N Bit
vlsideepdive
WhatsApp
Google Meet
circut designer
Verification Academy
{"Temperature":55,"Flame_Level":17,"Gas_Level":85}

Fig 12 - Properties of Monitor HTTP Function Node

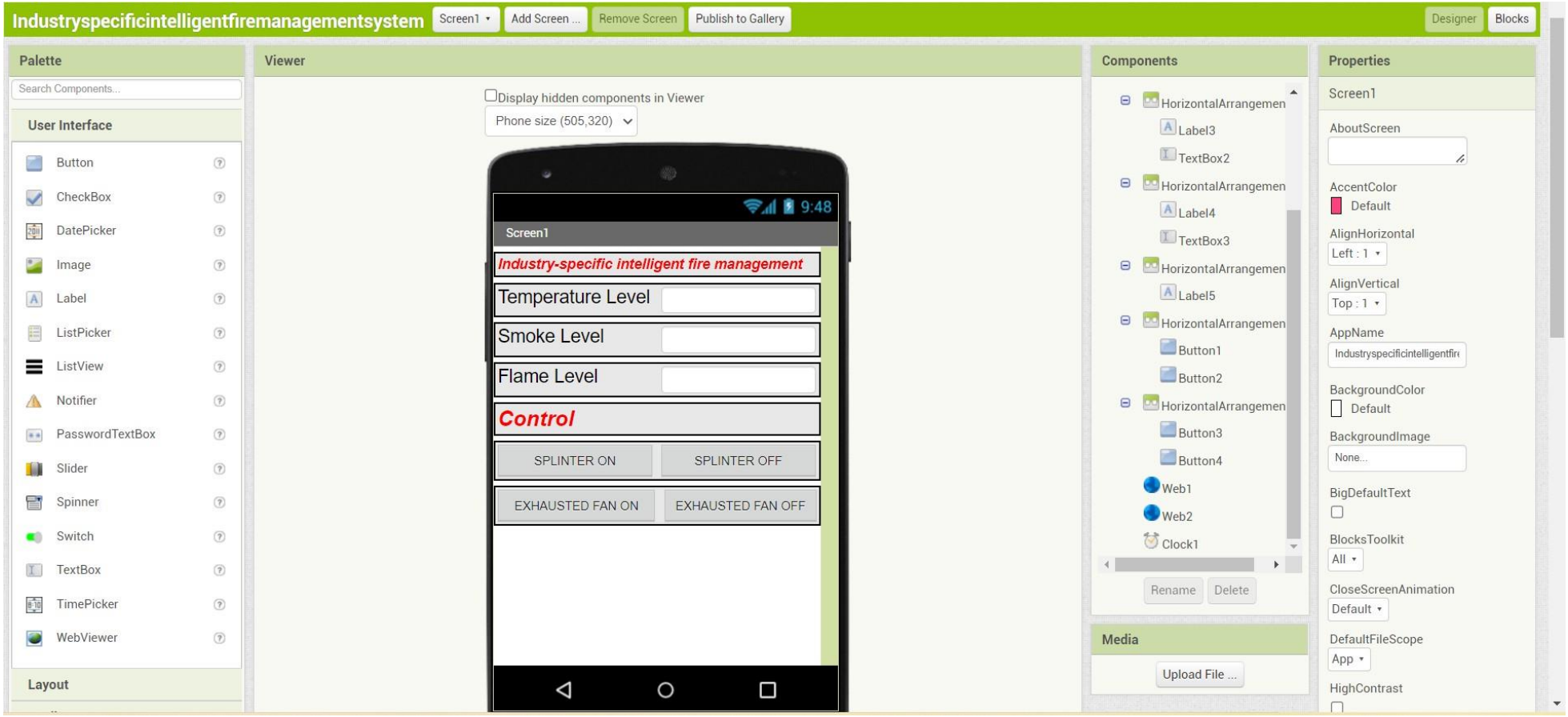


Fig 13 - Front-end APP for our project, to display the Temperature Level, Smoke Level and Flame Level with control buttons like Sprinkler ON and OFF and Exhaust Fan ON and OFF