

### Sprint - 3

Date	17 November 2022
Team ID	PNT2022TMID29259
Project Name	Project - Industry-specific intelligent fire management system

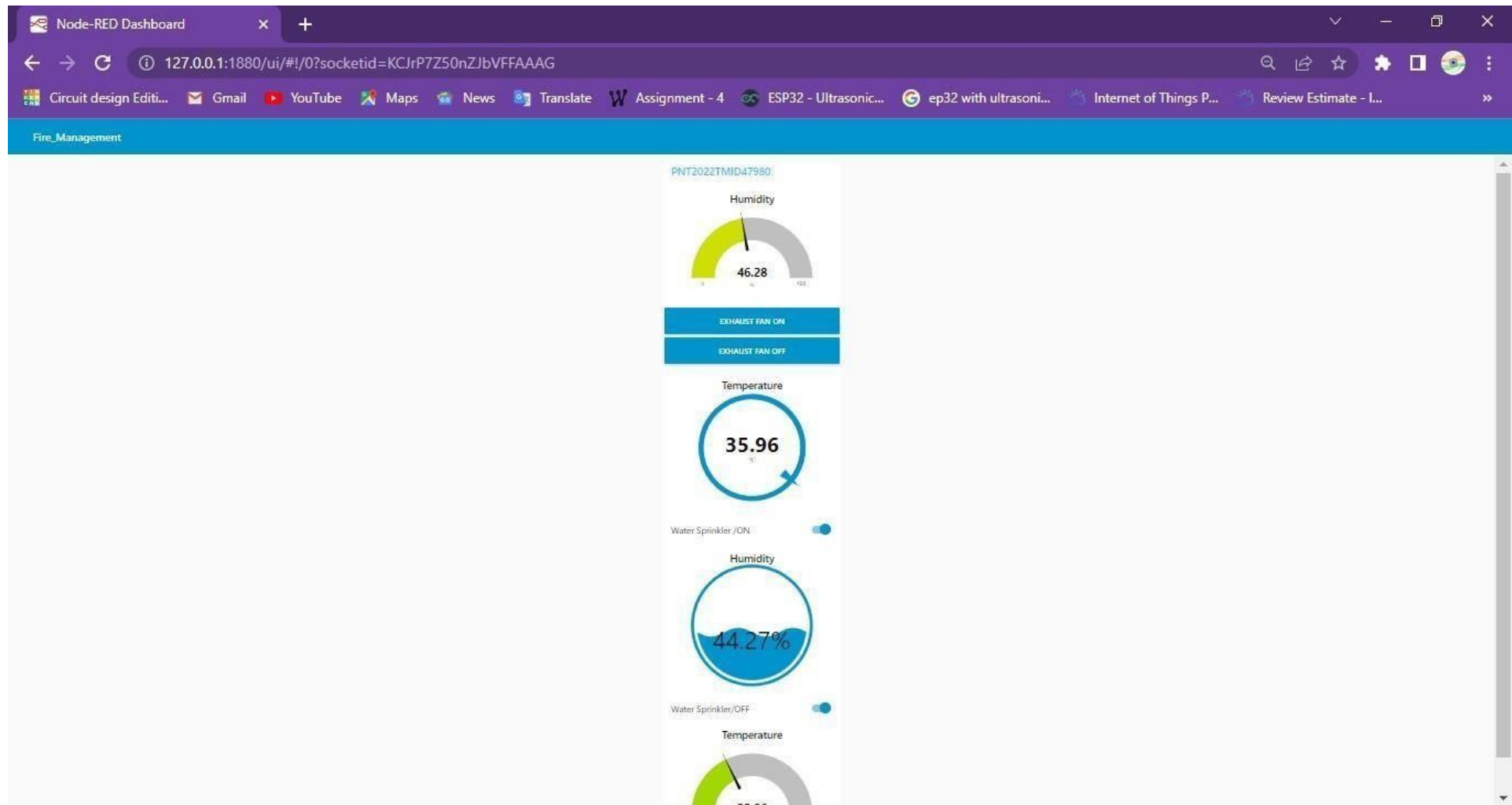
US-1 Configure the connection security and create API keys that are used in the Node-RED service for accessing the IBM IoT Platform.

US-2 Create a Node-RED service.

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

**LINK:** <https://wokwi.com/projects/348816699284259410>





**TRANSFERRING DATA FROM NODE-RED INTO WEB UI**

DESKTOP:



## CLOUDANT

The screenshot shows the Cloudant web interface. On the left is a sidebar with navigation icons and a 'Log Out' button. The main area displays a database named 'db'. At the top right, there are controls for 'Document ID', 'Options', and document format (JSON). Below this, there are tabs for 'Table', 'Metadata', and 'JSON', with 'Table' selected. A 'Create Document' button is on the right. The main content is a table of documents with columns 'id', 'key', and 'value'. The table lists 15 documents, each with a unique ID and a key, and a value containing a revision string. At the bottom right, it shows 'Showing document 1 - 20.' and 'Documents per page: 20'.

id	key	value
657846f21e0cb8ead462fd89321d28...	657846f21e0cb8ead462fd89321d28...	{ "rev": "1-1c9683229f242d4133b7f..." }
657846f21e0cb8ead462fd89321dd3...	657846f21e0cb8ead462fd89321dd3...	{ "rev": "1-8aeee9d453a632f539ee9c..." }
657846f21e0cb8ead462fd8932201e...	657846f21e0cb8ead462fd8932201e...	{ "rev": "1-7b6df30912cf9fde43ca8b..." }
657846f21e0cb8ead462fd8932203d...	657846f21e0cb8ead462fd8932203d...	{ "rev": "1-a9bec25d7f94ccc71ce692..." }
70ea2e4bb2a9c635be3ce2603a25a...	70ea2e4bb2a9c635be3ce2603a25a...	{ "rev": "1-b567b4cce122c31e1666fc..." }
70ea2e4bb2a9c635be3ce2603a268...	70ea2e4bb2a9c635be3ce2603a268...	{ "rev": "1-217497b95c16c3d228800..." }
70ea2e4bb2a9c635be3ce2603a272...	70ea2e4bb2a9c635be3ce2603a272...	{ "rev": "1-a01738b27517a2bb4b93b..." }
70ea2e4bb2a9c635be3ce2603a273...	70ea2e4bb2a9c635be3ce2603a273...	{ "rev": "1-13230a9f364a021a02422..." }
7170def319e06e12e85b74c728897...	7170def319e06e12e85b74c728897...	{ "rev": "1-4bdfcbf4dbbf888784fc24d..." }
7170def319e06e12e85b74c7288b7...	7170def319e06e12e85b74c7288b7...	{ "rev": "1-5b1a46d23a6c259bd5b97..." }
7170def319e06e12e85b74c7288c2...	7170def319e06e12e85b74c7288c2...	{ "rev": "1-782ab5b4e98aed22641a1..." }

## 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(" °C");
  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);
}
else
{
    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.

## The API key has been added.

Authentication tokens are non-recoverable. If you misplace this token, you will need to re-register the API key to generate a new authentication token.

### Generated Details

API Key a-4aqwut-gahbbnkql5 

Authentication Token dtAhr+HB3E-xIpbAgZ 



Make a note of the generated authentication token. Lost authentication tokens cannot be recovered. If you lose the token, you must reregister the API to generate a new token.

### API Key Information

Description -

Role Standard Application

Expires Never

## US-2 Create a Node-RED service

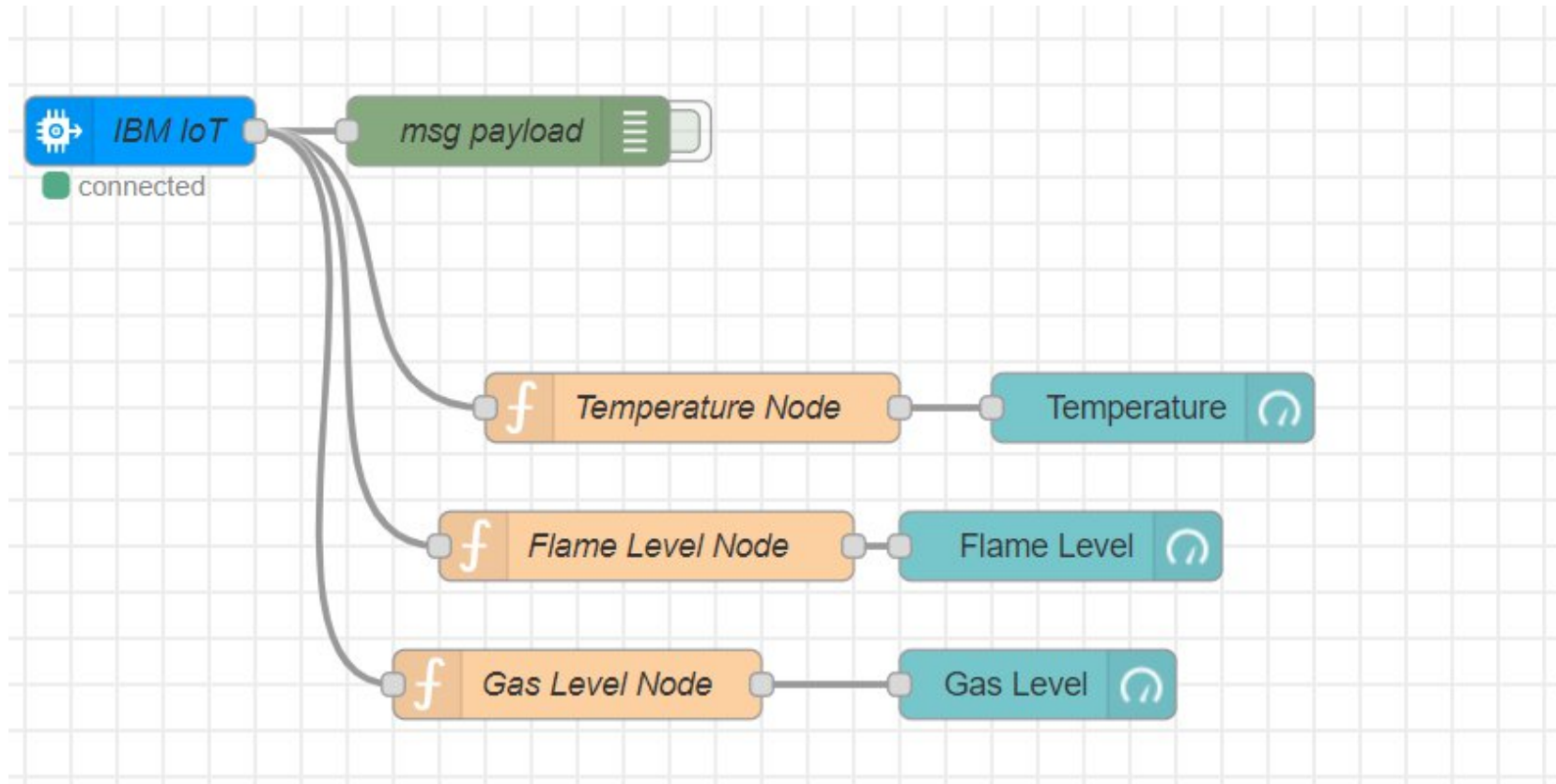


fig1 - Monitoring the sensor values - Temperature, Flame Level, Gas Level. These values are randomly generated by IBM WATSON IOT PLATFORM.

```
11/3/2022, 9:04:47 AM  node: msg payload
iot-2/type/B11M3EDeviceType/id/B11M3EDeviceID/evt/event_1/fmt/json : msg.payload : Object
  ▶ { Temperature: 1, Flame_Level: 62, Gas_Level: 38 }

11/3/2022, 9:04:50 AM  node: msg payload
iot-2/type/B11M3EDeviceType/id/B11M3EDeviceID/evt/event_1/fmt/json : msg.payload : Object
  ▶ { Temperature: 1, Flame_Level: 78, Gas_Level: 11 }

11/3/2022, 9:04:53 AM  node: msg payload
iot-2/type/B11M3EDeviceType/id/B11M3EDeviceID/evt/event_1/fmt/json : msg.payload : Object
  ▶ { Temperature: 99, Flame_Level: 36, Gas_Level: 55 }

11/3/2022, 9:04:56 AM  node: msg payload
iot-2/type/B11M3EDeviceType/id/B11M3EDeviceID/evt/event_1/fmt/json : msg.payload : Object
  ▶ { Temperature: 71, Flame_Level: 24, Gas_Level: 46 }

11/3/2022, 9:05:00 AM  node: msg payload
iot-2/type/B11M3EDeviceType/id/B11M3EDeviceID/evt/event_1/fmt/json : msg.payload : Object
  ▶ { Temperature: 38, Flame_Level: 92, Gas_Level: 63 }

11/3/2022, 9:05:03 AM  node: msg payload
iot-2/type/B11M3EDeviceType/id/B11M3EDeviceID/evt/event_1/fmt/json : msg.payload : Object
  ▶ { Temperature: 74, Flame_Level: 98, Gas_Level: 84 }

11/3/2022, 9:05:06 AM  node: msg payload
iot-2/type/B11M3EDeviceType/id/B11M3EDeviceID/evt/event_1/fmt/json : msg.payload : Object
  ▶ { Temperature: 87, Flame_Level: 81, Gas_Level: 44 }
```

Fig 2 - Temperature, Flame\_Level, Gas\_Level values displayed in deploy tab in node-red

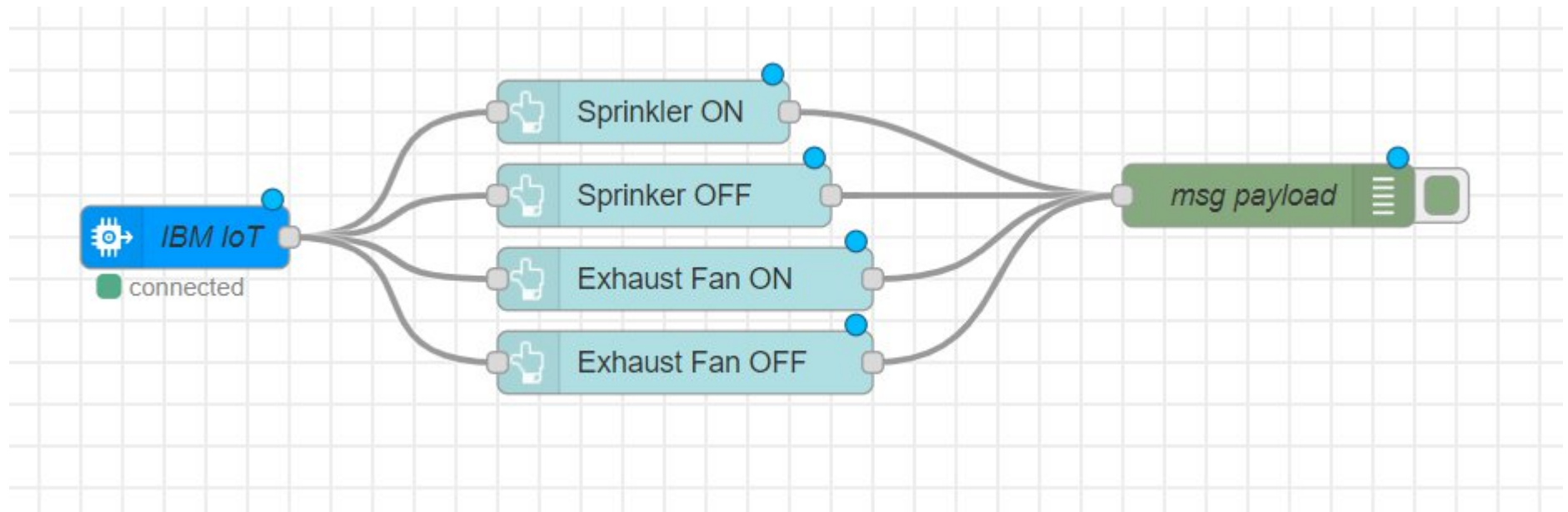




fig 3 - Control buttons (Sprinkler ON, Sprinkler OFF, Exhaust Fan ON, Exhaust Fan OFF)

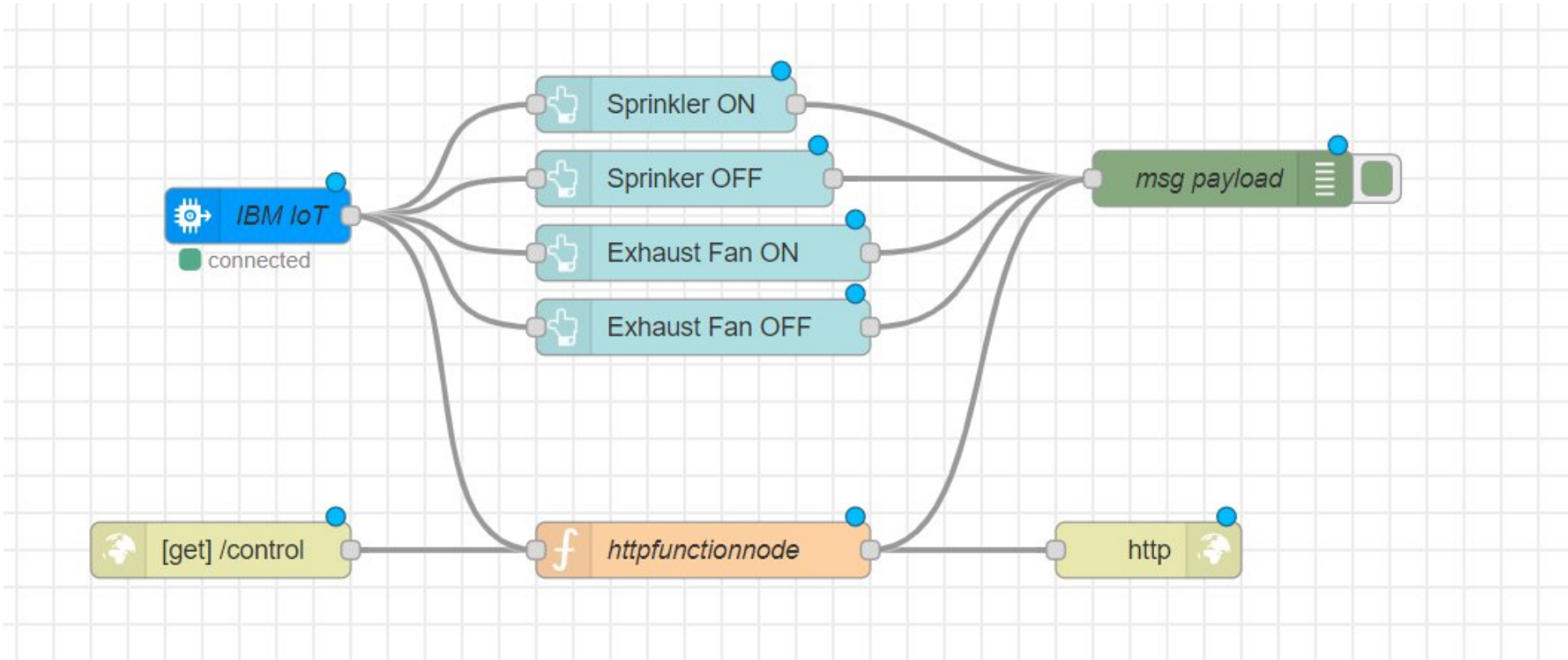


Fig 4 - Using HTTP in and HTTP response in network option, <http://127.0.0.1:1880/#flow/f74f1b96473dc208/control> will display the control options

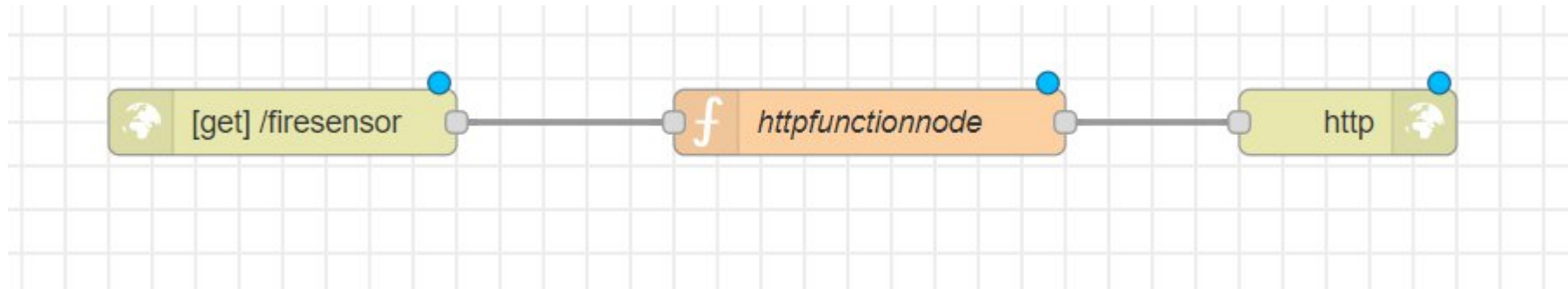


Fig 5 - Using HTTP in and HTTP response in network option, <http://127.0.0.1:1880/#flow/f74f1b96473dc208/firesensor> will display the sensor values like Temperature, Gas\_Level and Flame\_Level from the IBM WATSON IOT PLATFORM.

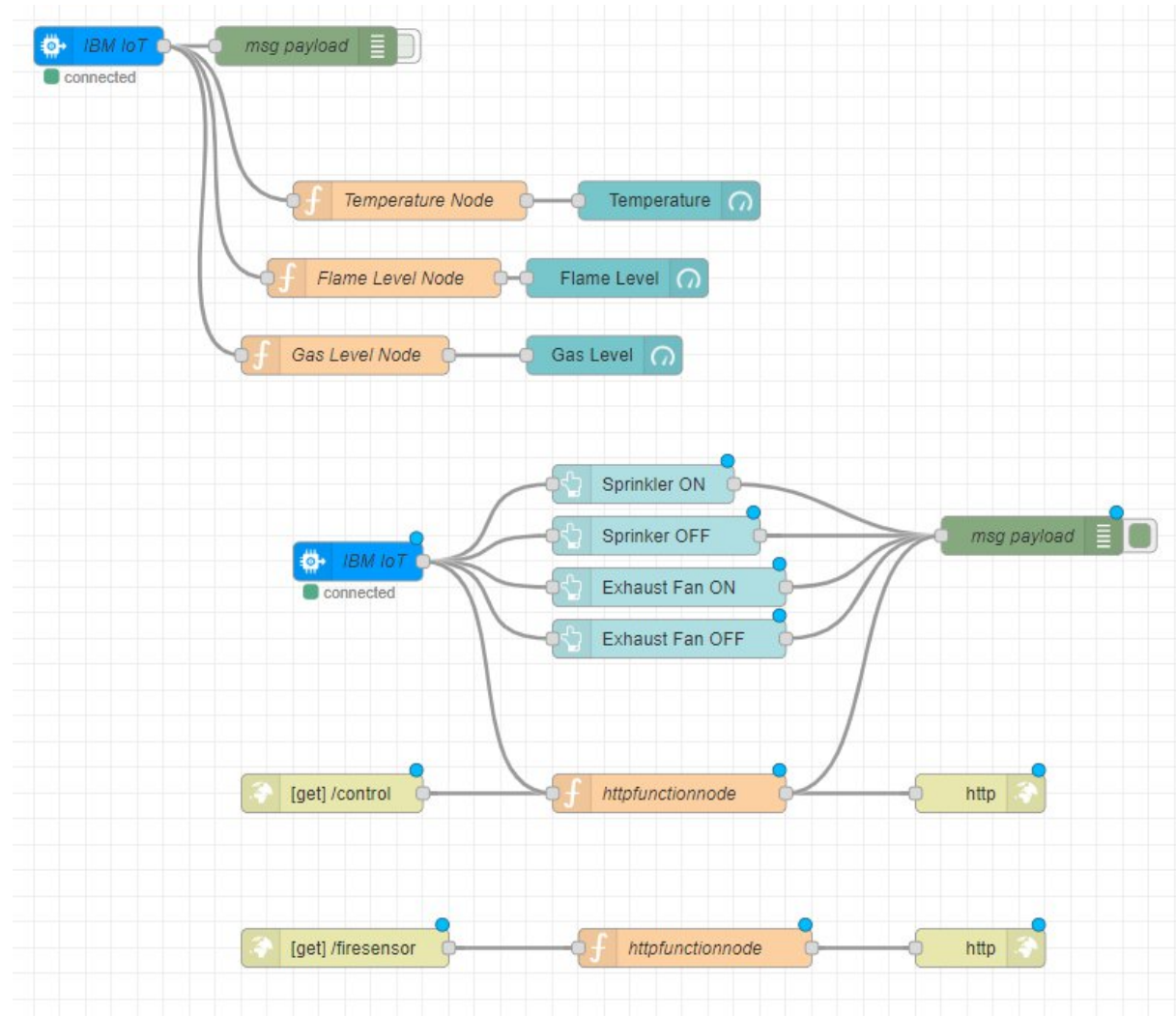


Fig 6 - Entire Node-Red connection for our project

Edit ibmiot in node

Delete

Cancel

Done

⚙ Properties

⚙

📄

🖨

🔑 Authentication

API Key

▼

🔑 API Key

a6cb71b59d73b36b

▼

✎

⚙ Input Type

Device Event

▼

🔑 Device Type

☐ All or

B11M3EDeviceType

👤 Device Id

☐ All or

B11M3EDeviceID

📋 Event

☒ All or

+

📄 Format

☐ All or

json

🌟 QoS

0

▼

🔑 Name

IBM IoT

🔑 Service

registered

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 Temperature Node

Setup On Start **On Message** On Stop

```
1 msg.payload = msg.payload.Temperature
2 global.set('t', msg.payload)
3 return msg;
```

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.

**Edit gauge node**

Delete

Cancel

Done

⚙️ Properties

⚙️

📄

🖨️

📊 Group

[Control] Industry specific intelligent fire

✎

📏 Size

auto

☰ Type

Gauge

▼

🏷️ Label

Temperature

🏷️ Value format

{{value}}

🏷️ Units

C

Range

min

0

max

10

Colour gradient

Sectors

0

...

optional

...

optional

...

10

🏷️ Name

Fig 9 - Properties of Temperature Gauge.

Edit gauge node

Delete

Cancel

Done

⚙ Properties

⚙

📄

🔗

📊 Group

[Control] Industry specific intelligent fire ▾

✎

📏 Size

auto

☰ Type

Gauge ▾

🔗 Label

Flame Level

🔗 Value format

{{value}}

🔗 Units

units

Range

min 0

max 10

Colour gradient

Sectors

0

...

optional

...

optional

...

10

🔖 Name



Fig 9 - Properties of Flame\_Level Gauge.

Edit gauge node

Delete

Cancel

Done

⚙ Properties

⚙

📄

🖨

📊 Group

[Control] Industry specific intelligent fire ▾

✎

📏 Size

auto

☰ Type

Gauge ▾

🔍 Label

Gas Level

🔍 Value format

{{value}}

🔍 Units

units

Range

min 0

max 10

Colour gradient

Sectors

0

...

optional

...

optional

...

10

🏷 Name

Fig 9 - Properties of Gas\_Level Gauge.

Edit ibmiot in node

Delete

Cancel

Done

Properties

Authentication

API Key

API Key

a6cb71b59d73b36b

Input Type

Device Command

Device Type

All or

B11M3EDeviceType

Device Id

All or

B11M3EDeviceID

Command

All or

onoff

Format

All or

String

QoS

0

Name

IBM IoT

Service

registered

Fig 9 - Properties of IBM IOT Node.

Edit button node

Delete

Cancel

Done

⚙ Properties

⚙

📄

🖼

📊 Group

[Control] Industry specific intelligent fi

▼

✎

📏 Size

auto

🖼 Icon

optional icon

🏷 Label

Sprinkler ON

📘 Tooltip

optional tooltip

🔥 Color

optional text/icon color

🔥 Background

optional background color

📧 When clicked, send:

Payload

▼ {} {"command": "SprinklerON"}

...

Topic

▼ msg. topic

➔ If msg arrives on input, emulate a button click:

☐

Fig 10 - Properties of Sprinkler ON button node.

**Edit http in node**

Delete

Cancel

Done

**Properties**

Method

GET

▼

URL

/control

Name

Name

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

**Edit function node**

Delete

Cancel

Done

**Properties**

Name

httpfunctionnode

Setup

On Start

**On Message**

On Stop

1 msg.payload = msg.payload.command

2 return msg;

Fig 11 - Properties of Control HTTP Function Node.

**Edit function node**

Delete

Cancel

Done

**Properties**

Name

httpfunctionnode

Setup

On Start

**On Message**

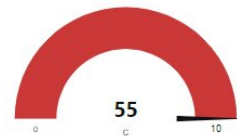
On Stop

```
1 msg.payload={ "Temperature":global.get('t'),
2               "Flame_Level":global.get('f'),
3               "Gas_Level":global.get('g')}
4 return msg;
```

## Control

### Industry specific intelligent fire management system

#### Temperature



SPRINKLER ON

EXHAUST FAN ON

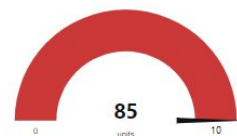
SPRINKLER OFF

EXHAUST FAN OFF

#### Flame Level



#### Gas Level



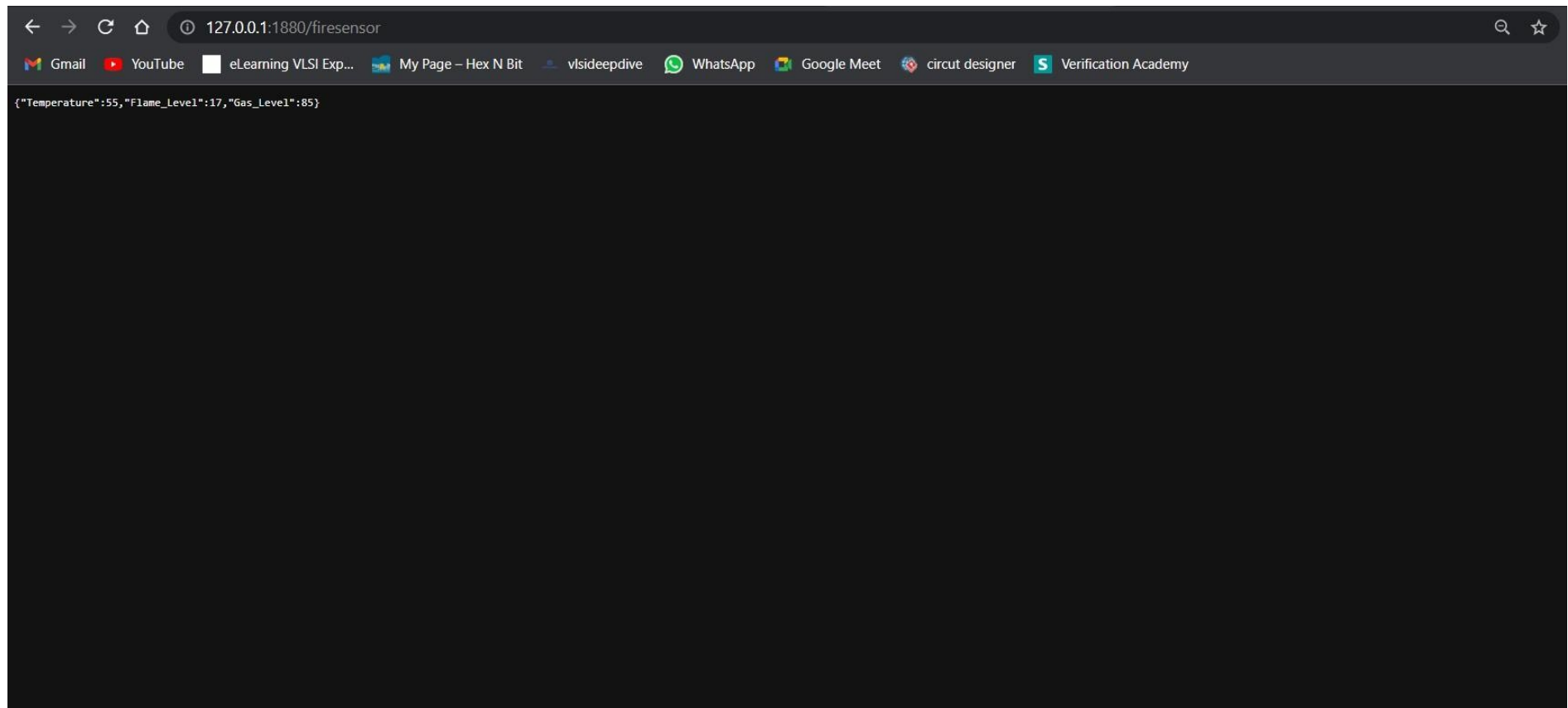




Fig 12 - Properties of Monitor HTTP Function Node

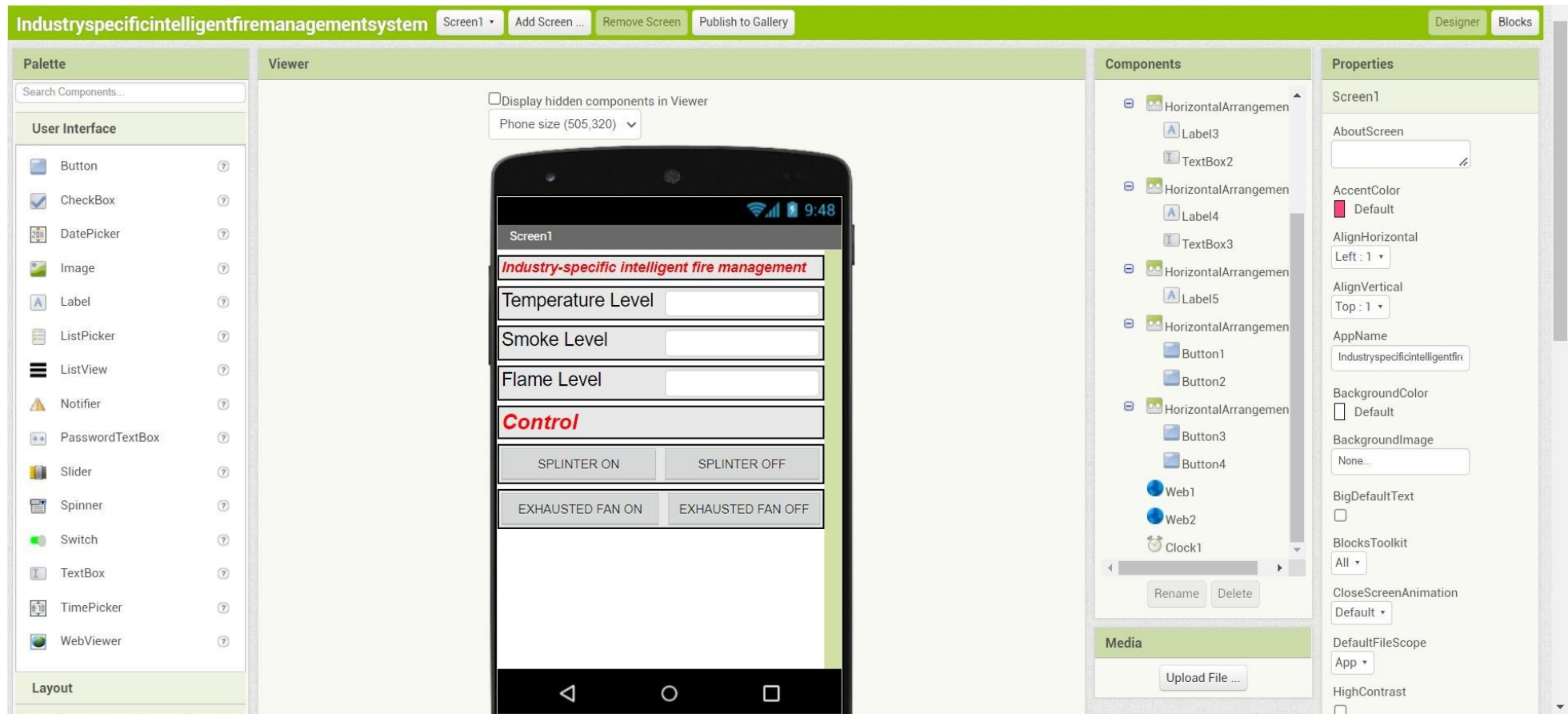


Fig 13 - Front-end APP for our project, to display the Temperature Level, Smoke Level and Flame Level with control buttons like Sprinkler ON and OFF and Exhaust Fan ON and OFF