

TEAM ID	PNT2022TMID04587
PROJECT NAME	Industry - specific intelligent fire management system

## **WOWKI SIMULATION**

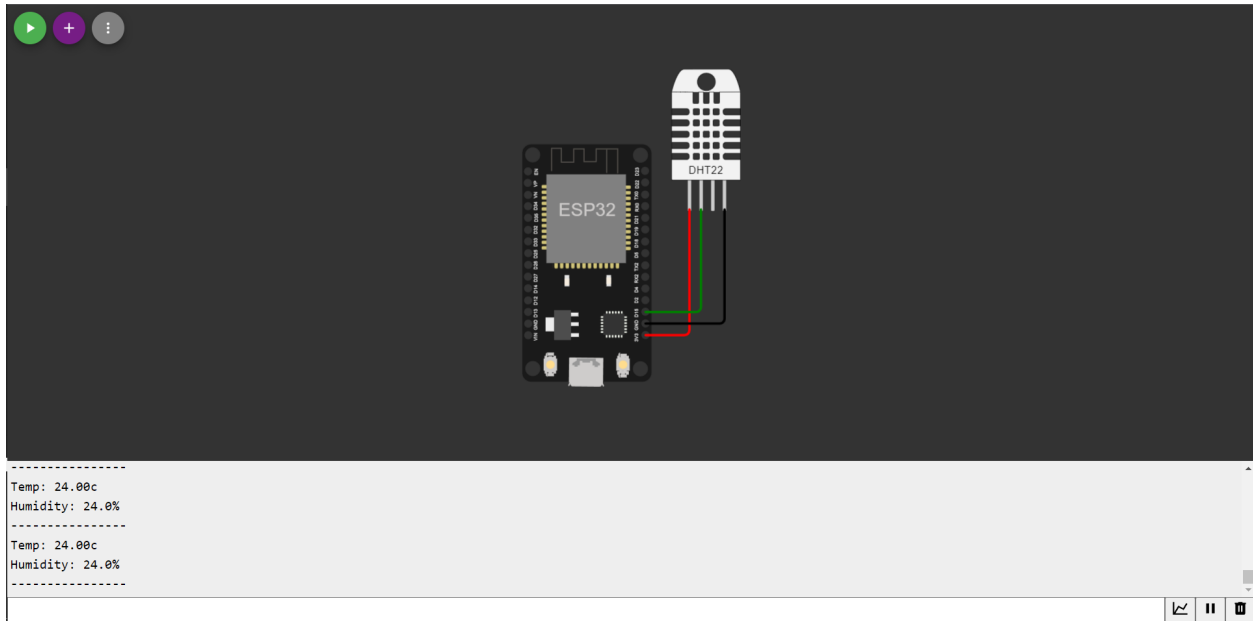
Using the wowki online simulator we have simulated the hardware part of the project. The circuit consists of NodeMCU and sensors to monitor parameters like temperature, humidity, gas and to detect flame and take necessary actions like turning on sprinkler and exhaust fan is done accordingly.

## **CODE:**

```
#include "DHTesp.h"
const int DHT_PIN = 15;
DHTesp dhtSensor;

void setup() {
    // put your setup code here, to run once:
    Serial.begin(115200);
    Serial.println("Hello, ESP32!");
    dhtSensor.setup(DHT_PIN,DHTesp::DHT22);
}

void loop() {
    TempAndHumidity data = dhtSensor.getTempAndHumidity();
    Serial.println("Temp: " + String(data.temperature,2)+"c");
    Serial.println("Humidity: " + String(data.humidity,1)+"%");
    Serial.println("-----");
    delay(100); // this speeds up the simulation
}
```



## **PYTHON**

The python code is used to randomly generate temperature, humidity, gas and flame values and it can be pushed to the IBM cloud.

### **CODE:**

```
import time
import sys
import ibmiotf.device
import ibmiotf.application
import random

organization = "fvg3cq"
deviceType = "NodeMCU"
deviceId = "12345"
authMethod = "token"
authToken = "12345678"

def myCommandCallback (cmd):
    print("Command received: %s" % cmd.data['command'])
    status = cmd.data['command']
    if status == "sprinkleron":
        print("sprinkler is on")
```

```

elif status == "sprinkleroff":
    print("sprinkler is off")
elif status=="fanon"):
    print("fan is on")
elif status=="fanoff"):
    print("fan is off")
else:
    print("Please send proper command")

```

```

print(cmd)

```

```

"""def myCommandCallback2(cmd):
    print("Command received: %s" % cmd.data['command'])
    status = cmd.data['command']
    if status == "fanon":
        print("fan is on")
    else:
        print("fan is off")
    print(cmd)"""

```

```

try:
    deviceOptions = {"org": organization, "type": deviceType, "id":
deviceId, "auth-method": 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 of
type "greeting" 10 times
deviceCli.connect()

```

```

while True:
    #Get Sensor Data from DHT11
    temp=random.randint(0,70)
    gas=random.randint(0,100)
    flame=random.randint(0,1)
    data = { 'temp' : temp, 'gas': gas, 'flame': flame }
    #print data

```

```

def myOnPublishCallback():
    print ("Published Temperature = %s C" % temp, "Gas = %s %" % gas, "flame = %s %" % flame,"to IBM Watson")
    success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0,
on_publish=myOnPublishCallback)
    if not success:
        print("Not connected to IoT")
    time.sleep(1)
    deviceCli.commandCallback = myCommandCallback
    #deviceCli.commandCallback2 = myCommandCallback2
# Disconnect the device and application from the cloud
deviceCli.disconnect()

```

```

Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
Python 3.7.4 (tags/v3.7.4:09359112e, Jul 8 2019, 19:29:22) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: E:\IBM\IBM.py =====
2022-11-18 22:35:09,106 ibmiotf.device.Client INFO Connected successfully: d:inbee2:NodeMCU:12345
Published Temperature = 31 C Gas = 48 % flame = 0 % to IBM Watson
Published Temperature = 67 C Gas = 47 % flame = 1 % to IBM Watson
Published Temperature = 69 C Gas = 23 % flame = 1 % to IBM Watson
Published Temperature = 9 C Gas = 24 % flame = 1 % to IBM Watson
Published Temperature = 7 C Gas = 40 % flame = 1 % to IBM Watson
Published Temperature = 3 C Gas = 11 % flame = 0 % to IBM Watson
Published Temperature = 37 C Gas = 35 % flame = 1 % to IBM Watson
Published Temperature = 11 C Gas = 52 % flame = 0 % to IBM Watson
Published Temperature = 29 C Gas = 52 % flame = 0 % to IBM Watson
Published Temperature = 12 C Gas = 1 % flame = 1 % to IBM Watson
Published Temperature = 40 C Gas = 16 % flame = 0 % to IBM Watson
Published Temperature = 56 C Gas = 92 % flame = 0 % to IBM Watson
Published Temperature = 30 C Gas = 54 % flame = 1 % to IBM Watson
Published Temperature = 64 C Gas = 42 % flame = 0 % to IBM Watson
Published Temperature = 51 C Gas = 20 % flame = 0 % to IBM Watson
Published Temperature = 45 C Gas = 21 % flame = 0 % to IBM Watson
Published Temperature = 65 C Gas = 95 % flame = 1 % to IBM Watson

```

**IBM CLOUD**

Data from the wowki simulator or python random generator code is updated to the IBM cloud.

IBM Watson IoT Platform

charansuryab.19ece@kongu.edu

ID: fvg3cq

Browse

Action

Device Types

Interfaces

Add Device

Q

Search by Device ID

Device Simulator

<input type="checkbox"/>	Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location
<input checked="" type="checkbox"/>	12345	Connected	NodeMCU	Device	16 Nov 2022 23:11	

Identity

Device Information

Recent Events

State

Logs

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

Event	Value	Format	Last Received
IoTSensor	{"temp":40,"gas":22,"flame":1}	json	a few seconds ago
IoTSensor	{"temp":43,"gas":74,"flame":1}	json	a few seconds ago
IoTSensor	{"temp":18,"gas":49,"flame":1}	json	a few seconds ago
IoTSensor	{"temp":65,"gas":42,"flame":1}	json	a few seconds ago
IoTSensor	{"temp":51,"gas":91,"flame":0}	json	a few seconds ago

## NODE-RED SERVICE

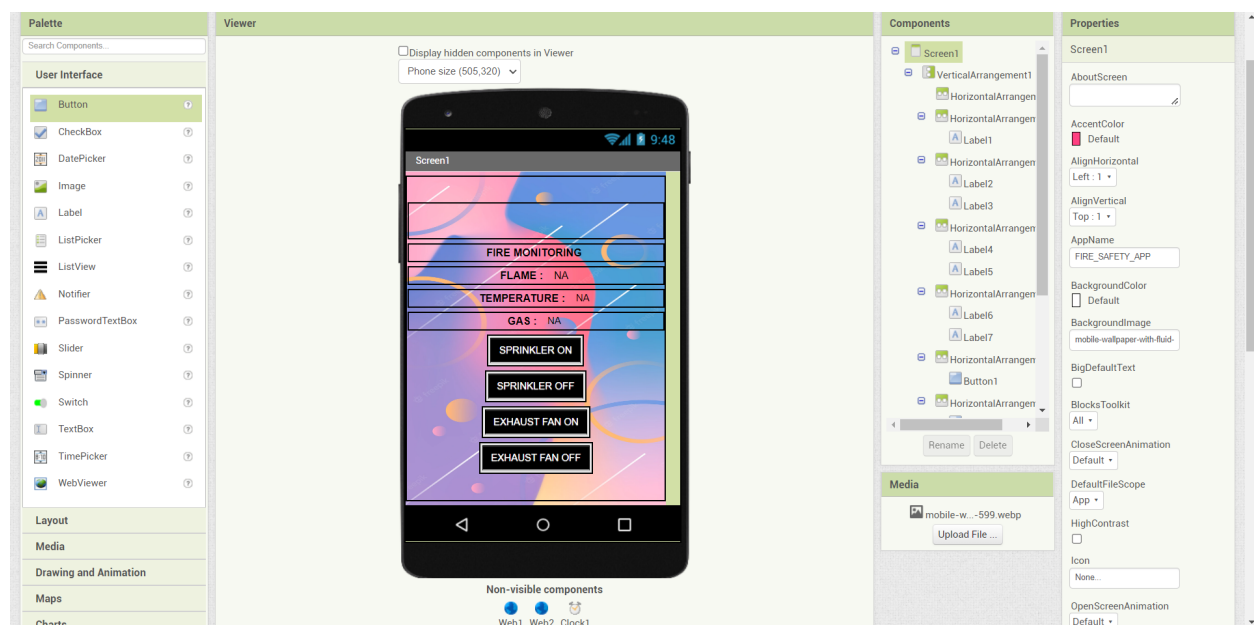
Node-Red service is used to create a dashboard to have a visual image of the various parameters.

## NODE-RED DASHBOARD



## MIT APP INVENTOR

The MIT app inventor is used to create an mobile application with which we can monitor the parameters like temperature, flame, gas and take precautionary action using the controls provided in it.



MIT APP INVENTOR

Projects • Connect • Build • Settings • Help • My Projects View Trash Guide Report an Issue English • charansuriya2002@gmail.com •

FIRE\_SAFETY\_APP Screen1 • Add Screen ... Remove Screen Publish to Gallery Designer Blocks

Blocks

- Built-in
  - Control
  - Logic
  - Math
  - Text
  - Lists
  - Dictionaries
  - Colors
  - Variables
  - Procedures
- Screen1
  - VerticalArrangement1
    - HorizontalArrangement1
      - Label1
      - Label2

Media

- mobile-w...-599.webp

Viewer

when Web1 .GotText

do

set Label3 .Text to look up in pairs key flame

pairs call Web1 .JsonTextDecode jsonText get responseContent

notFound not found

set Label5 .Text to look up in pairs key temp

pairs call Web1 .JsonTextDecode jsonText get responseContent

notFound not found

set Label7 .Text to look up in pairs key gas

pairs call Web1 .JsonTextDecode jsonText get responseContent

notFound not found

Show Warnings

MIT APP INVENTOR

Projects • Connect • Build • Settings • Help • My Projects View Trash Guide Report an Issue English • charansuriya2002@gmail.com •

FIRE\_SAFETY\_APP Screen1 • Add Screen ... Remove Screen Publish to Gallery Designer Blocks

Blocks

- Built-in
  - Control
  - Logic
  - Math
  - Text
  - Lists
  - Dictionaries
  - Colors
  - Variables
  - Procedures
- Screen1
  - VerticalArrangement1
    - HorizontalArrangement1
      - Label1
      - Label2

Media

- mobile-w...-599.webp

Viewer

when Clock1 .Timer

do

set Web1 .Uri to http://159.122.179.91:31976/sensor

call Web1 .Get

when Button1 .Click

do

set Web2 .Uri to http://159.122.179.91:31976/command?command=spr...

call Web2 .Get

when Button2 .Click

do

set Web2 .Uri to http://159.122.179.91:31976/command?command=spr...

call Web2 .Get

when Button3 .Click

do

set Web2 .Uri to http://159.122.179.91:31976/command?command=fanon

call Web2 .Get

when Button4 .Click

do

set Web2 .Uri to http://159.122.179.91:31976/command?command=fanoff

call Web2 .Get

Show Warnings

4:23 PM | 5.1KB/s



Screen1

### FIRE MONITORING

FLAME : 1

TEMPERATURE : 21

GAS : 81

SPRINKLER ON

SPRINKLER OFF

EXHAUST FAN ON

EXHAUST FAN OFF