

# **INDUSTRY-SPECIFIC INTELLIGENT FIRE MANAGEMENT SYSTEM**

**Team ID - PNT2022TMID45189**

**Mentor: S.Kavitha**

**Team Leader – Selvaganapathi M**

**Team Member 1 – Aravind T**

**Team Member 2 –Laiyancemary A**

**Team Member 3 – Sathish I**

# Agenda

- ❖PROJECT OBJECTIVE
  - ❖PROBLEM STATEMENT
  - ❖PROPOSED SOLUTIONS
  - ❖TECHNICAL ARCHITECTURE
  - ❖WORKING PROCESS OF THE PROJECT
  - ❖ADVANTAGES
  - ❖FUTURE SCOPE
-

# PROJECT OBJECTIVE

A fire alarm system warns people when smoke, fire, carbon monoxide or other fire-related or general notification emergencies are detected. These alarms may be activated automatically from smoke detectors and heat detectors or may also be activated via manual fire alarm activation devices such as manual call points or pull stations. Alarms can be either motorized bells or wall mountable sounders or horns.



# PROBLEM STATEMENT

- ❖ We need to design a fire alarm system that all family members can use in single-family residences.
- ❖ It must be able to detect fires at all locations, residents must be able to activate it from convenient locations themselves, and it must alert residents in all portions of the house.

# PROPOSED SOLUTIONS



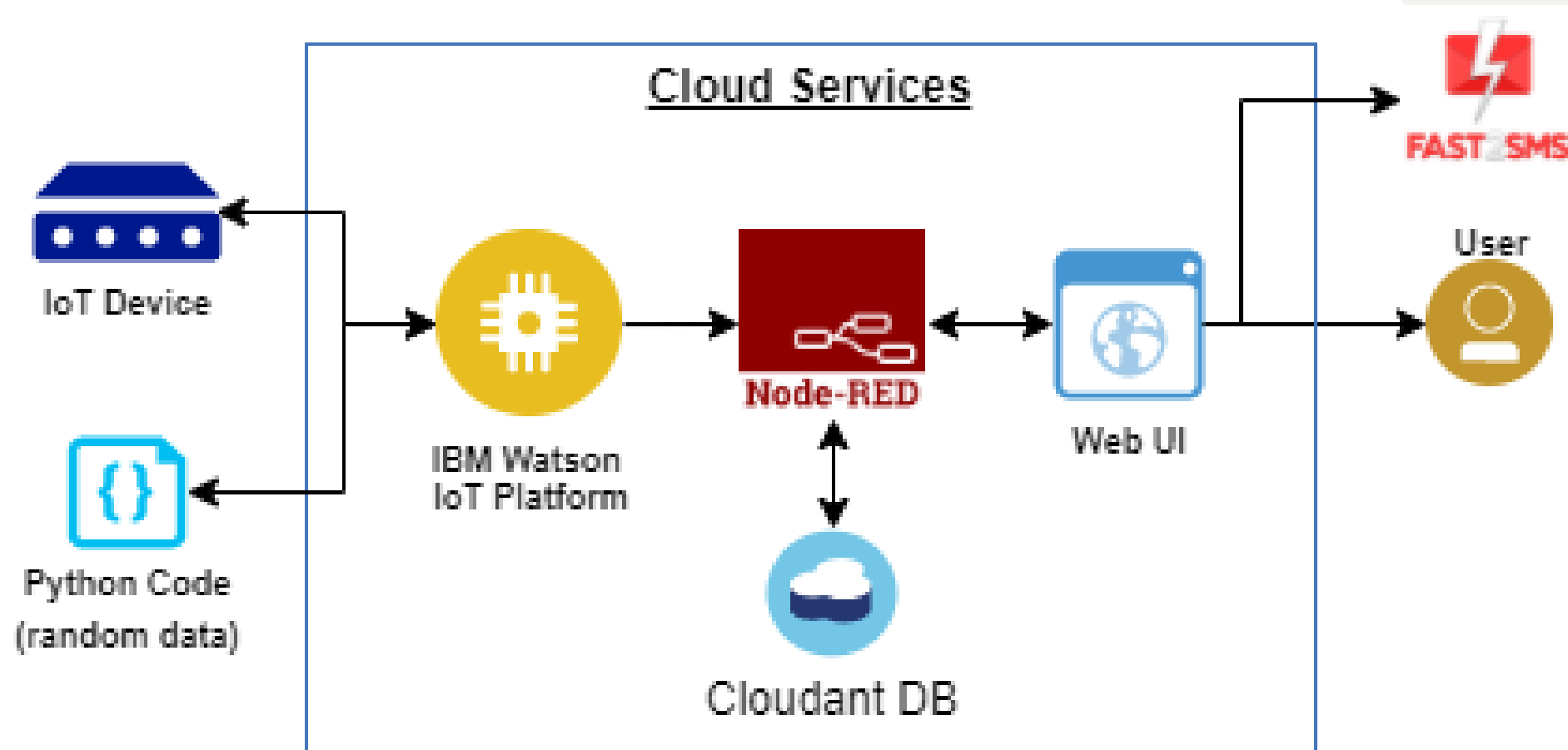
- ❖ The task of a fire-fighting system is to early detect and minimise the consequences of a fire, and thus protect people and property. Simple fire-fighting systems consist of a fire and smoke detector, a control panel and fixed fire-fighting systems, e.g. a system of pipes filled with an extinguishing agent and provided with outlet nozzles.
- ❖ Fire-fighting systems may be divided into four main types, depending on the applied extinguishing agent: water, water mist, foam and gas extinguishing systems.



# FUNCTIONAL REQUIREMENT

FUNCTIONAL REQUIREMENT (EPIC)	SUB REQUIREMENT (STORY / SUB – TASK)
IOT devices	Sensor and Wi-Fi module
Software	Web UI, Node-RED, IBM WATSON, MIT APP

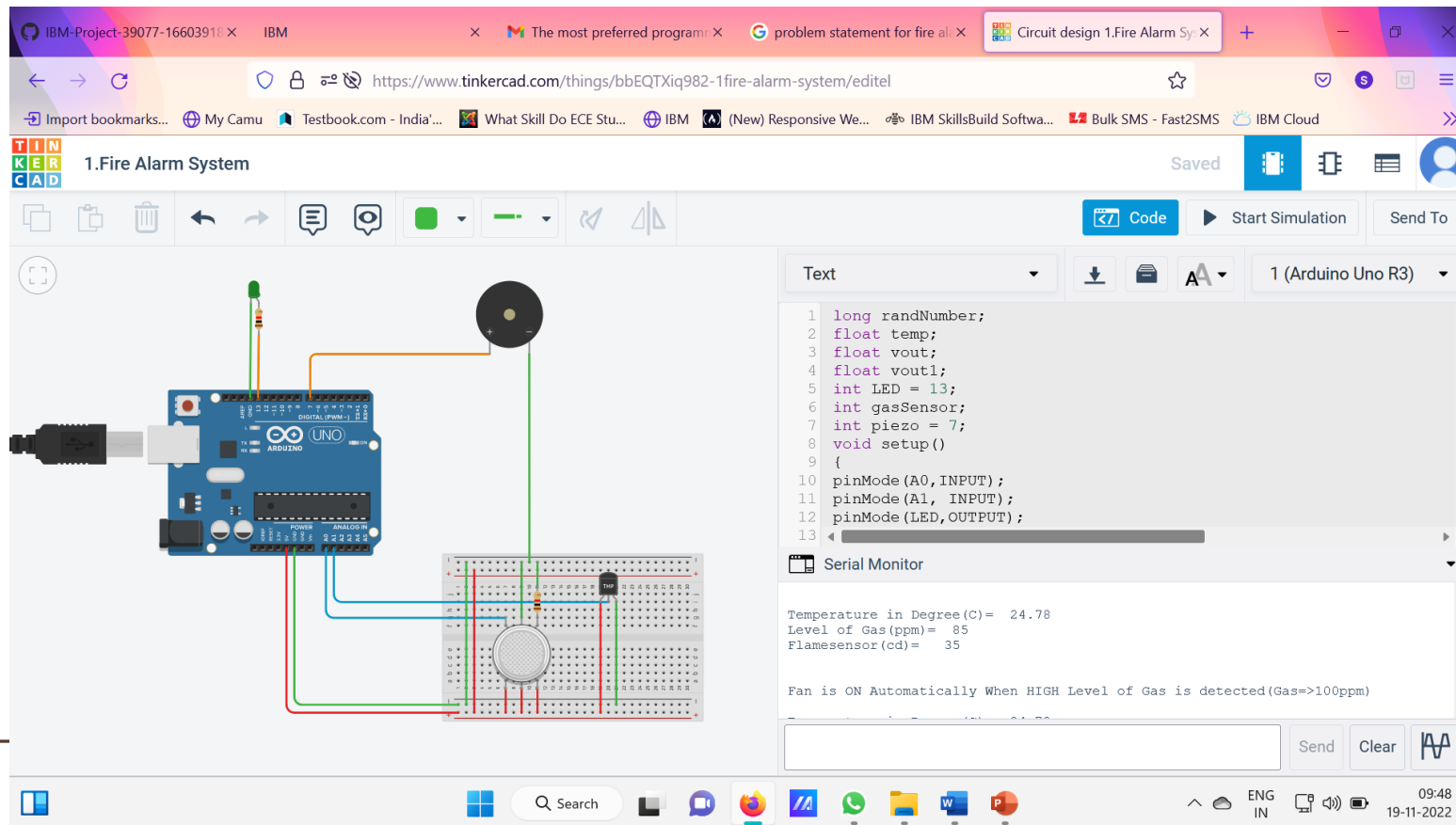
# TECHNICAL ARCHITECTURE



# WORKING PROCESS OF THE PROJECT

## Step 1

- ❖ Simulation Creation (Connect sensor Arduino with Python code)
- ❖ Using the Tinkercard Software create and simulate the model Smart Farmer – IOT Enabled Smart Farming Application





## Step 2

- Software (Create device in the IoT Watson platform, workflow for IoT scenarios using Node-Red)

The screenshot displays the IBM Watson IoT Platform dashboard. The main page is titled 'Browse Devices' and includes a sidebar with navigation icons. The 'Simulations' overlay is active, showing '1/50 Simulations Running' and a 'New Simulation' button. Below this, a table lists the device details for '12'.

Device ID	Status	Device Type	Class ID
12	Disconnected	abcd	Device

Items per page: 50 | 1-1 of 1 item

# IOT using Node - red

The screenshot displays the Node-RED web interface in a browser. The browser's address bar shows the URL `159.122.179.199:31683/red/#flow/8838c2728e6f08db`. The Node-RED interface includes a left sidebar with node categories: 'common' (inject, debug, complete, catch, status, link in, link call, link out, comment) and 'function' (function, switch, change, range, template, delay, trigger). The main workspace, titled 'Flow 1', contains a flow diagram. The flow starts with an 'IBM IoT' node (connected) that branches into three function nodes: 'temperature', 'gas', and 'flame'. Each function node is connected to a corresponding output node: 'Temperature', 'Gas', and 'Flame'. A 'mydb' node is also connected to the 'temperature' function node. Below this, a 'switch' node is connected to an 'http request' node, which is then connected to a 'msg.payload' node. At the bottom, a '[get] /data' node is connected to a 'webpage' node, which is connected to an 'http' node. Another 'IBM IoT' node (connected) is connected to a 'msg.payload' node. A '[get] /command' node is connected to an 'http' node. The right sidebar shows a 'debug' console with a list of nodes and a log of messages. The messages are JSON objects containing temperature, gas, and flame data. The system tray at the bottom shows the Windows taskbar with various icons and the date/time '13:33 19-11-2022'.

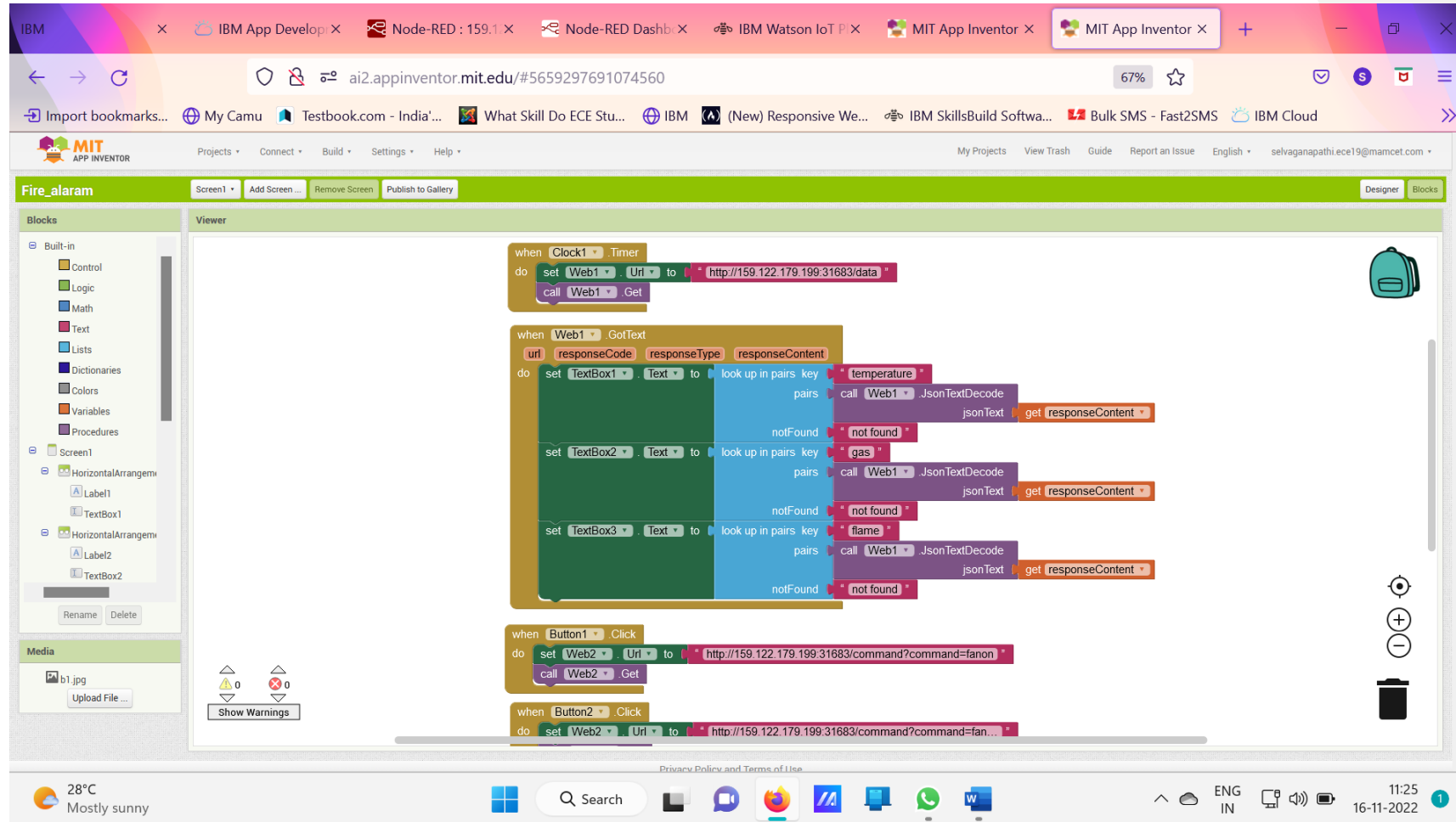
```
graph LR
    IoT1[IBM IoT] --> temp_f[temperature]
    IoT1 --> gas_f[gas]
    IoT1 --> flame_f[flame]
    temp_f --> Temp[Temperature]
    gas_f --> Gas[Gas]
    flame_f --> Flame[Flame]
    mydb[mydb] --> temp_f
    switch[switch] --> http_req[http request]
    http_req --> msg_payload1[msg.payload]
    get_data["[get] /data"] --> webpage[webpage]
    webpage --> http1[http]
    fan_on[fan on] --> IoT2[IBM IoT]
    fan_off[fan off] --> IoT2
    sprinkler_on[sprinkler on] --> IoT2
    sprinkler_off[sprinkler off] --> IoT2
    get_command["[get] /command"] --> http2[http]
    IoT2 --> msg_payload2[msg.payload]
```

debug console output:

```
{ "temperature": 39, "gas": 37, "flame": 93 }
19/11/2022, 1:32:45 pm node: 294f8f5fea9fc2f2
iot-2/type/abcd/id/12/ev/Eventflow/fmt/json :
msg.payload : number
39
19/11/2022, 1:32:46 pm node: 294f8f5fea9fc2f2
iot-2/type/abcd/id/12/ev/Eventflow/fmt/json :
msg.payload : number
37
19/11/2022, 1:32:47 pm node: 294f8f5fea9fc2f2
iot-2/type/abcd/id/12/ev/Eventflow/fmt/json :
msg.payload : number
93
19/11/2022, 1:33:44 pm node: 294f8f5fea9fc2f2
iot-2/type/abcd/id/12/ev/Eventflow/fmt/json :
msg.payload : Object
{ "temperature": 46, "gas": 69, "flame": 75 }
19/11/2022, 1:33:44 pm node: 294f8f5fea9fc2f2
iot-2/type/abcd/id/12/ev/Eventflow/fmt/json :
msg.payload : number
46
19/11/2022, 1:33:44 pm node: 294f8f5fea9fc2f2
iot-2/type/abcd/id/12/ev/Eventflow/fmt/json :
msg.payload : number
69
19/11/2022, 1:33:44 pm node: 294f8f5fea9fc2f2
iot-2/type/abcd/id/12/ev/Eventflow/fmt/json :
msg.payload : number
75
```

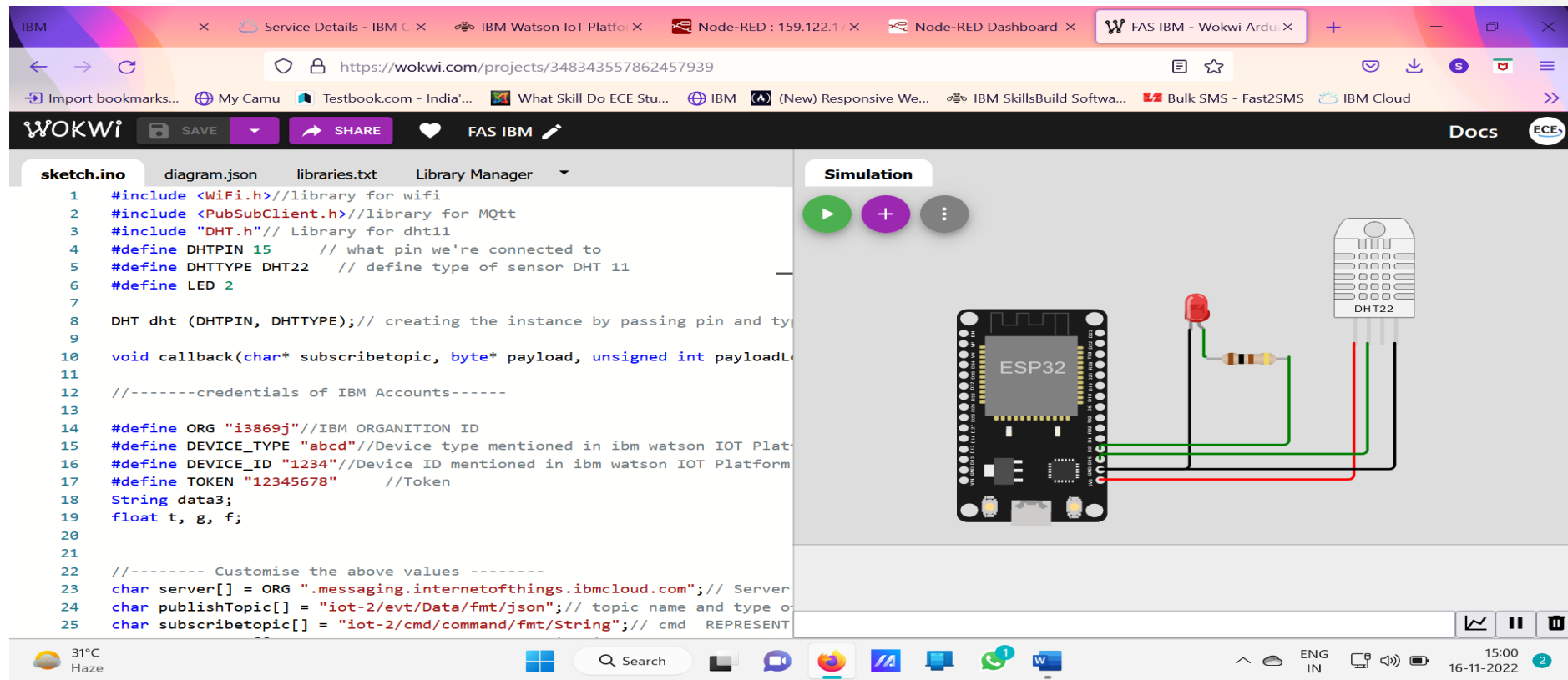
# Step 3

- MIT App Inventor, Dashboard (Application for your project using MIT App, Design the model and test the App)

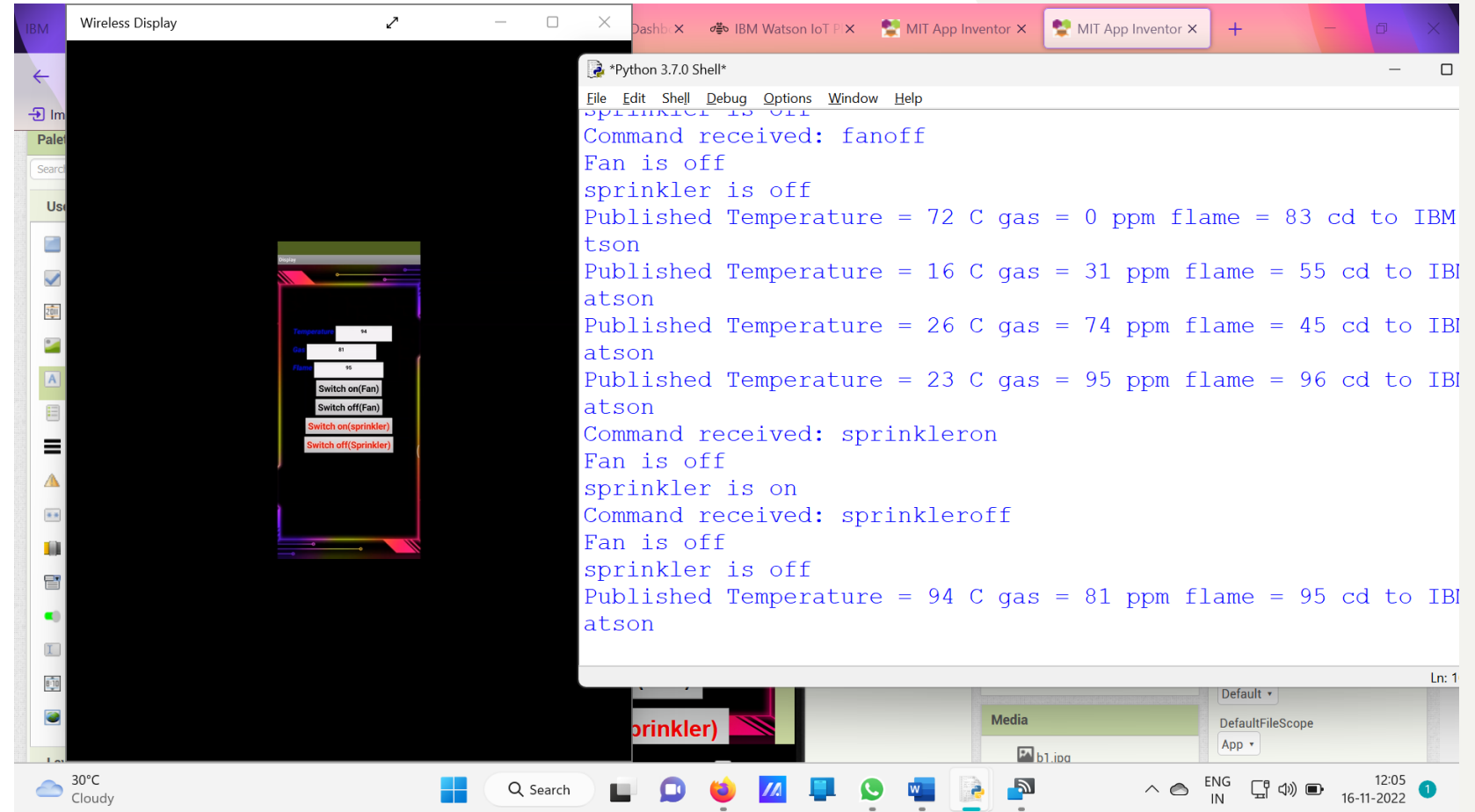


# Step 4

- Web UI (to make the user interact with the software) / Run a simulation using the wokwi online platform



# Mobile Application Output Using MIT invetor



# ADVANTAGES

- ❖ Active monitoring for gas leakage and fire breakout.
  - ❖ Automatic alerting of admin as well as fire authorities using SMS.
  - ❖ Automatically turning on/off sprinkler as well as exhaust fan.
  - ❖ Authentication is required to turn on/off of sprinkler and exhaust fan as well as sending SMS alert manually.
  - ❖ It automatically detect false fire breakout reducing unnecessary panic by using flow sensors we can confirm that the sprinkler system is working as it intended, All device status can be shown in a dashboard
  - ❖ Users can see the dashboard using a web application
-

# FUTURE SCOPE

Fire detection technologies have been slow to evolve compared to rapidly advancing smart devices. Understandably, global companies focus their efforts on developing high-return products, **Sensor-Assisted Fire Fighting, High-Pressure Water Mist, Drones, Fireballs**, these are the upcoming technologies in the fire alarm system.



# REFERENCE

- ✓ <https://www.tinkercad.com/things/bbEQTXiq982>
- ✓ <https://wokwi.com/projects/348343557862457939>
- ✓ <https://github.com/IBM-EPBL/IBM-Project-44952-1660727584>



The background features a minimalist design with organic, flowing shapes in muted red and sage green. A thin, light-colored line drawing of leaves is positioned in the upper left corner. The text "THANK YOU ...," is centered in a dark brown, serif font.

**THANK YOU ...,**