CO326 Project: Smart Building
Group C- Safety

Introduction

The use of different IoT devices for automation in buildings has become very popular in recent years. Fire detection and avoidance of fire accidents is one of the necessary and important applications of home automation using IoT. Traditional fire alarm system requires a huge installation cost and labor.

With the development of Internet of things technology, the application of IoT technology to fire alarms can accurately locate the fire point and has the characteristics of real-time monitoring, fire tracking, online upgrades, and convenient maintenance. The proposed IOT based fire alarm system basically detects fire at an early stage, generates an automatic alarm and notify the remote user or fire control station about the fire outbreak. This also tries to extinguish the fire.

The use of a Microcontroller, scada, MQTT is proposed to sense the surroundings for occurrence of fire with the help of a temperature, fire and gas sensor. MQTT is used to collect the data collected by sensor nodes to gateway and upload it to servers to solve the indoor location problems of fire locations and evacuation plans. The development of a fire alert system is built based on an integrated system. The fire is detected at an early stage and the system generates an alarm and sends SMS or call alerts to mobile numbers stored in the database.

Simultaneously, a water sprayer producing device is switched on for the control of fire. The real time sensing readings, current time updates, smoke status, the zones where fire has been detected can be seen through the scada system in real time. Actuators and other stuff like doors opening can also be controlled manually. This prototype system can help users to improve their safety standards with immediate response by preventing accidents. This will eventually allow both the lives and the properties from the disaster. The functions of each module and its implementation is described in detail.

Sensors to read data and send to MQTT Server

Smoke inhalation is the major factor for most of the fire deaths rather than burns. If smoke can be detected earlier, the victim will be able to have enough time to escape the building. Therefore, we use the **MQ2 sensor** as a smoke detector which can provide early detection of fire and alert the victim. Then to observe the temperature changes **TDA** sensors are used.Whenever a person sees a fire , he pulls the fire alarm that is the **pull station** to activate the fire alarm system.

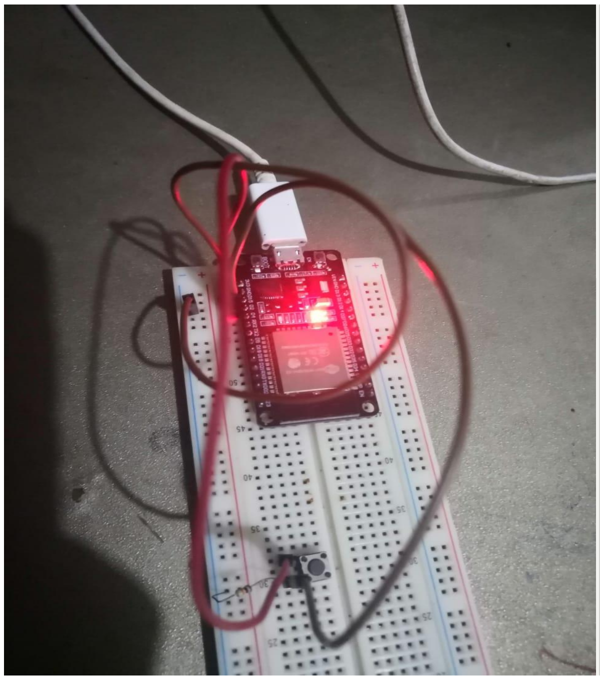
These three sensors publish their sensor readings to the node-red server via mqtt.We didn't use real sensors MQ2 and TDA in this project. Instead we use already simulated sensor readings. Those readings (given in model.csv) will be published every second to the relevant topics. And we use Push Button (instead of pull station) to activate the fire alarm system in our project.

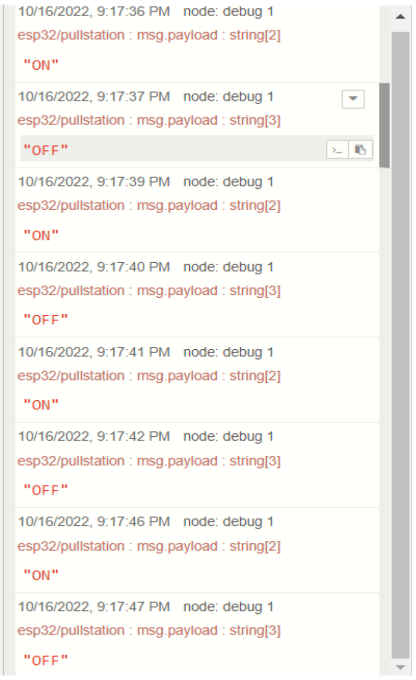
Given in the below table are topics that readings should be published to and data format,

Controller	Data Format	Example
MQ2 gas sensor	Time: Time of sensor reading Gas: Gas sensor reading in integer	{"time":"2022-04-101T45:56:12.658Z", "Gas": 45}
Pull station	State: Status of pull station	"State": 0(Off) / 1(On)
Temperature Sensor	Time: Time of sensor reading temp: Temperature in Celsius	{"time":"2022-04-101T45:56:12.658Z", "temp":"28.4"}

Push Button as Pull station

As mentioned above, we used a push button to activate the alarm system based on the status. Here we have used micropython as a programming language with an esp32 microcontroller. ESP32 publishes status of the push button via mqtt to the node-red server.



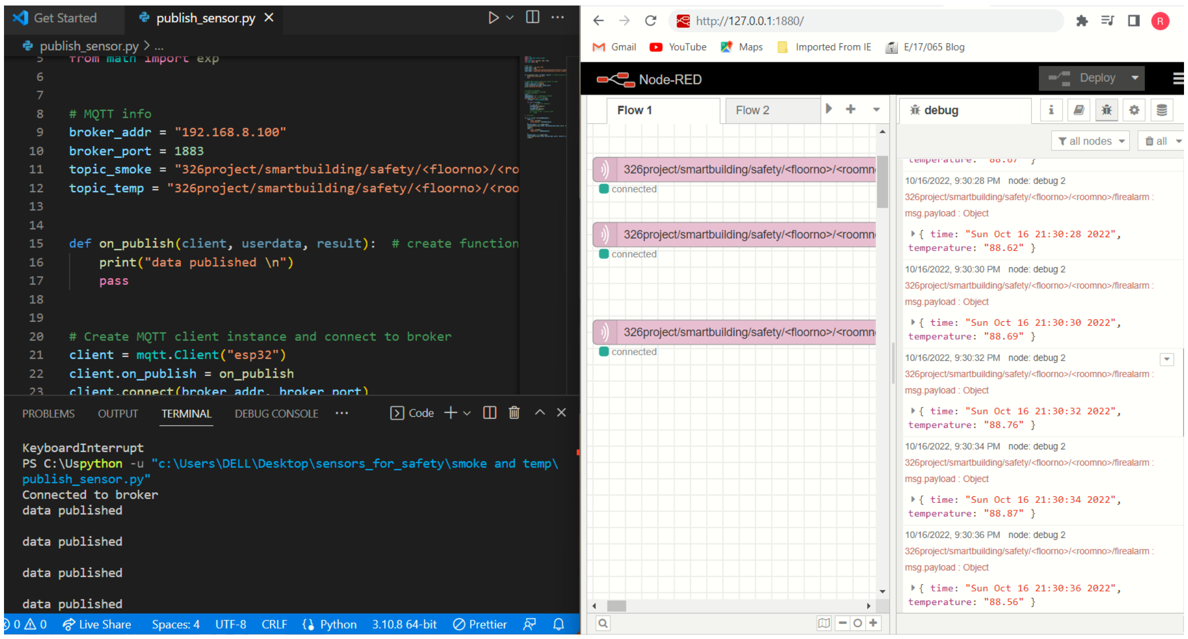


```
10/16/2022, 9:17:36 PM   node: debug 1
esp32/pullstation : msg.payload : string[2]
"ON"
10/16/2022, 9:17:37 PM   node: debug 1
esp32/pullstation : msg.payload : string[3]
"OFF"
10/16/2022, 9:17:39 PM   node: debug 1
esp32/pullstation : msg.payload : string[2]
"ON"
10/16/2022, 9:17:40 PM   node: debug 1
esp32/pullstation : msg.payload : string[3]
"OFF"
10/16/2022, 9:17:41 PM   node: debug 1
esp32/pullstation : msg.payload : string[2]
"ON"
10/16/2022, 9:17:42 PM   node: debug 1
esp32/pullstation : msg.payload : string[3]
"OFF"
10/16/2022, 9:17:46 PM   node: debug 1
esp32/pullstation : msg.payload : string[2]
"ON"
10/16/2022, 9:17:47 PM   node: debug 1
esp32/pullstation : msg.payload : string[3]
"OFF"
```

Circuit and how status of the push button is published can be seen from above given figures,whenever it activates.

MQ2 Sensor and Fire alarm detector

Here we didn't use real hardware components, but we used simulated sensor readings recorded in a csv file. It will be uploaded to our python scripts every seconds.If a real hardware exists, then it monitors the concentration of smoke and temperature for each second. That is the basic idea, we bring from the simulated csv file.From below given figures that can be understood.



```python
from math import exp


# MQTT info
broker_addr = "192.168.8.100"
broker_port = 1883
topic_smoke = "326project/smartbuilding/safety/<floorno>/<ro
topic_temp = "326project/smartbuilding/safety/<floorno>/<roo


def on_publish(client, userdata, result):  # create function
    print("data published \n")
    pass


# Create MQTT client instance and connect to broker
client = mqtt.Client("esp32")
client.on_publish = on_publish
client.connect(broker_addr, broker_port)
```

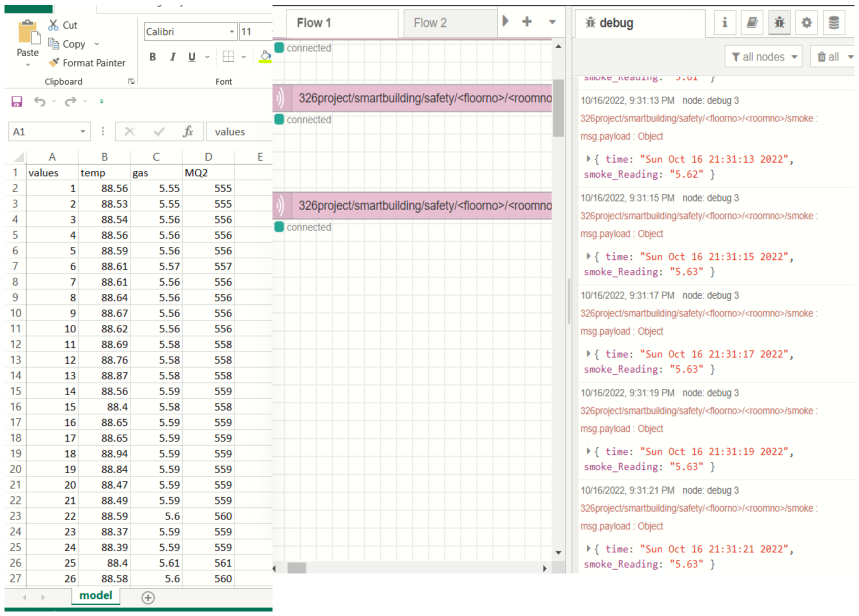
```
KeyboardInterrupt
PS C:\Uspython -u "c:\Users\DELL\Desktop\sensors_for_safety\smoke and temp\
publish_sensor.py"
Connected to broker
data published

data published

data published

data published
```

```
10/16/2022, 9:30:28 PM   node: debug 2
326project/smartbuilding/safety/<floorno>/<roomno>/firealarm :
msg.payload : Object
▸{ time: "Sun Oct 16 21:30:28 2022",
temperature: "88.62" }
10/16/2022, 9:30:30 PM   node: debug 2
326project/smartbuilding/safety/<floorno>/<roomno>/firealarm :
msg.payload : Object
▸{ time: "Sun Oct 16 21:30:30 2022",
temperature: "88.69" }
10/16/2022, 9:30:32 PM   node: debug 2
326project/smartbuilding/safety/<floorno>/<roomno>/firealarm :
msg.payload : Object
▸{ time: "Sun Oct 16 21:30:32 2022",
temperature: "88.76" }
10/16/2022, 9:30:34 PM   node: debug 2
326project/smartbuilding/safety/<floorno>/<roomno>/firealarm :
msg.payload : Object
▸{ time: "Sun Oct 16 21:30:34 2022",
temperature: "88.87" }
10/16/2022, 9:30:36 PM   node: debug 2
326project/smartbuilding/safety/<floorno>/<roomno>/firealarm :
msg.payload : Object
▸{ time: "Sun Oct 16 21:30:36 2022",
temperature: "88.56" }
```

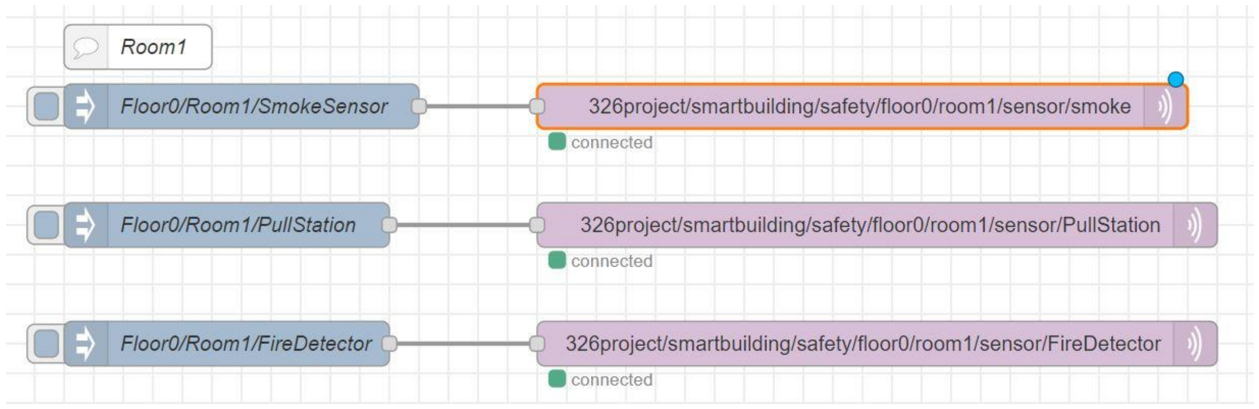


	A	B	C	D
1	values	temp	gas	MQ2
2	1	88.56	5.55	555
3	2	88.53	5.55	555
4	3	88.54	5.56	556
5	4	88.56	5.56	556
6	5	88.59	5.56	556
7	6	88.61	5.57	557
8	7	88.61	5.56	556
9	8	88.64	5.56	556
10	9	88.67	5.56	556
11	10	88.62	5.56	556
12	11	88.69	5.58	558
13	12	88.76	5.58	558
14	13	88.87	5.58	558
15	14	88.56	5.59	559
16	15	88.4	5.58	558
17	16	88.65	5.59	559
18	17	88.65	5.59	559
19	18	88.94	5.59	559
20	19	88.84	5.59	559
21	20	88.47	5.59	559
22	21	88.49	5.59	559
23	22	88.59	5.6	560
24	23	88.37	5.59	559
25	24	88.39	5.59	559
26	25	88.4	5.61	561
27	26	88.58	5.6	560

```
10/16/2022, 9:31:13 PM   node: debug 3
326project/smartbuilding/safety/<floorno>/<roomno>/smoke :
msg.payload : Object
▸{ time: "Sun Oct 16 21:31:13 2022",
smoke_Reading: "5.62" }
10/16/2022, 9:31:15 PM   node: debug 3
326project/smartbuilding/safety/<floorno>/<roomno>/smoke :
msg.payload : Object
▸{ time: "Sun Oct 16 21:31:15 2022",
smoke_Reading: "5.63" }
10/16/2022, 9:31:17 PM   node: debug 3
326project/smartbuilding/safety/<floorno>/<roomno>/smoke :
msg.payload : Object
▸{ time: "Sun Oct 16 21:31:17 2022",
smoke_Reading: "5.63" }
10/16/2022, 9:31:19 PM   node: debug 3
326project/smartbuilding/safety/<floorno>/<roomno>/smoke :
msg.payload : Object
▸{ time: "Sun Oct 16 21:31:19 2022",
smoke_Reading: "5.63" }
10/16/2022, 9:31:21 PM   node: debug 3
326project/smartbuilding/safety/<floorno>/<roomno>/smoke :
msg.payload : Object
▸{ time: "Sun Oct 16 21:31:21 2022",
smoke_Reading: "5.63" }
```

Node-Red implementation for sensors

Sensor readings which are published to MQTT topics corresponding to the floors, and rooms shown below.


Room1
Floor0/Room1/SmokeSensor
326project/smartbuilding/safety/floor0/room1/sensor/smoke
connected
Floor0/Room1/PullStation
326project/smartbuilding/safety/floor0/room1/sensor/PullStation
connected
Floor0/Room1/FireDetector
326project/smartbuilding/safety/floor0/room1/sensor/FireDetector
connected

Read data from MQTT Server and take decisions and control actuators

The actuators should be subscribed to the topics which give input to them, in order for the data to be exchanged. The MQTT broker will publish the controlled data by processing, and the actuators corresponding to them will get that data. Each controller unit will be able to get data from the MQTT broker since it has subscribed to certain topics.
Therefore, for the safety purpose we use the **Fire Sprinkler System** which is an active fire protection method, consisting of a water supply system, providing adequate pressure to spray water into the room if fire has been detected. Then to alert the people in the building, **Fire Alarms** are used. Whenever fire detects, warn people in the building that there may be a fire and to evacuate.
To evacuate optimally in the correct shortest distance path, use the **Lightning System** to lead them.

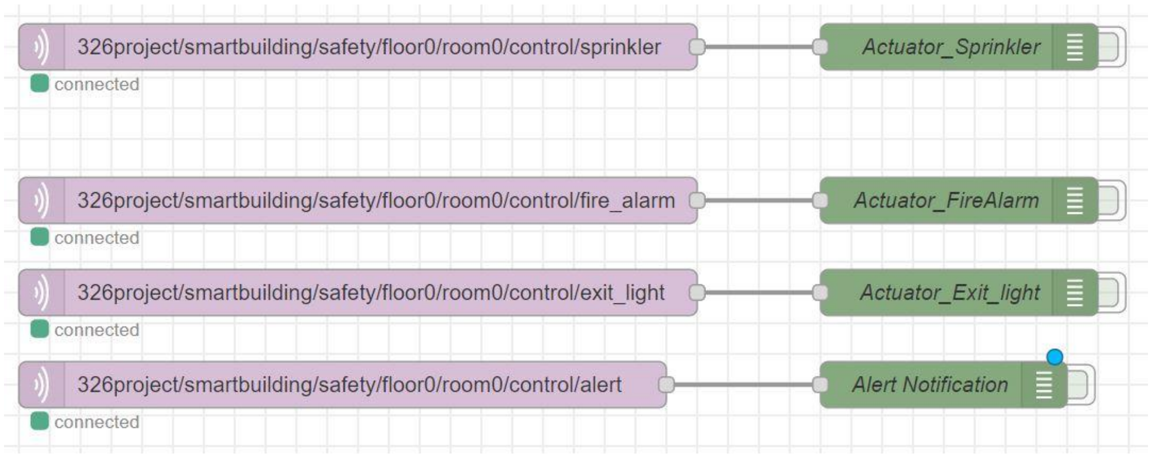
Lightning System as LED

The SCADA system determines whether a fire has occurred and takes decisions on the best path to take based on the location of the fire and the distance to the nearest exit. This information is published in a standerdized format to the relevant topic to which the node controller that controls the actuators is subscribed. Controller unit in each room picks up the information and lights up the correct path to exit.

Piezo buzzer as Fire Alarm

Similarly the fire alarm is started when the relevant topic state is updated to 1. The SCADA system determines a fire when sensor outputs reach a certain threshold. Then the topic is signaledwhich will be reflected by the alarm**.**

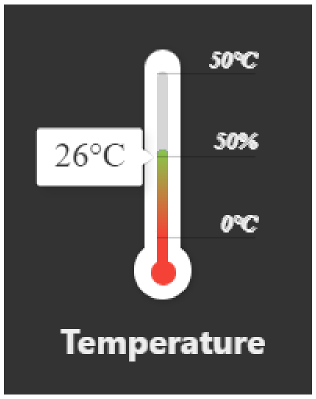
Node-Red implementation for actuators


326project/smartbuilding/safety/floor0/room0/control/sprinkler
connected
Actuator_Sprinkler
326project/smartbuilding/safety/floor0/room0/control/fire_alarm
connected
Actuator_FireAlarm
326project/smartbuilding/safety/floor0/room0/control/exit_light
connected
Actuator_Exit_light
326project/smartbuilding/safety/floor0/room0/control/alert
connected
Alert Notification

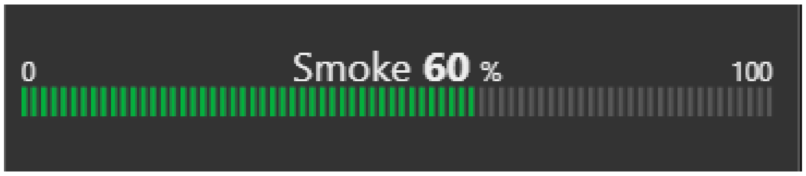
Read Data from MQTT Server and Display System Status on SCADA

A SCADA system is a combination of hardware and software that enables the automation of industrial processes by capturing Operational Technology data. Our SCADA system connects the sensors that we implemented and can monitor in real time.
The node red dashboard SCADA displays the sensor/actuator statuses as well as the fire related environmental conditions of each room in each floor.

- Sensor/Actuator Statuses
    - Temperature



- Smoke level

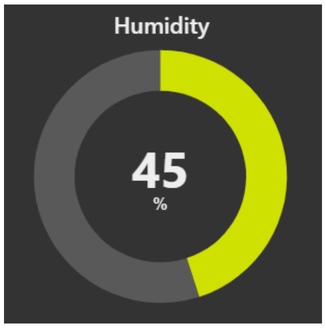


- Sprinkler status

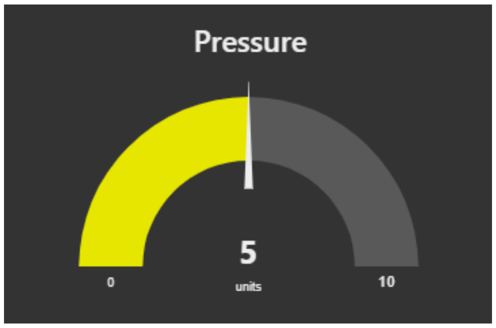
- Fire Alarm status

- Pull Station status

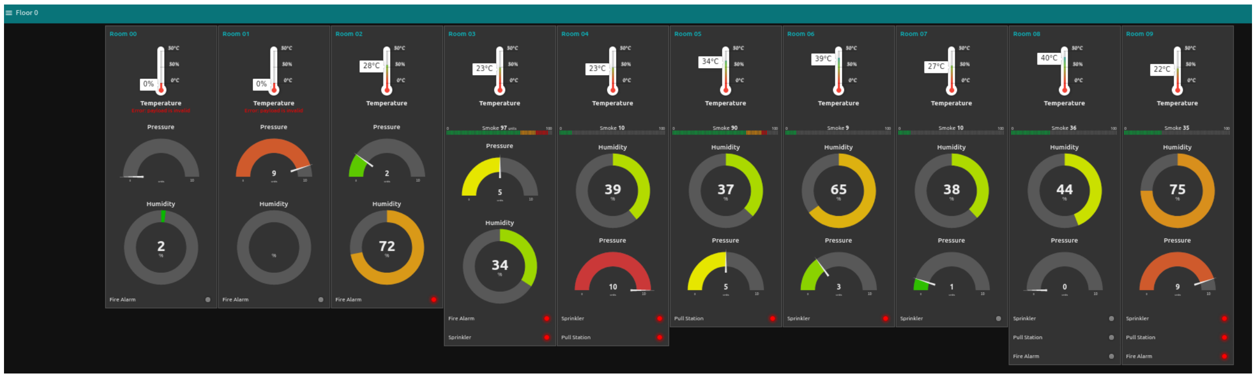
- Environmental Conditions Related to Fire
  - Humidity



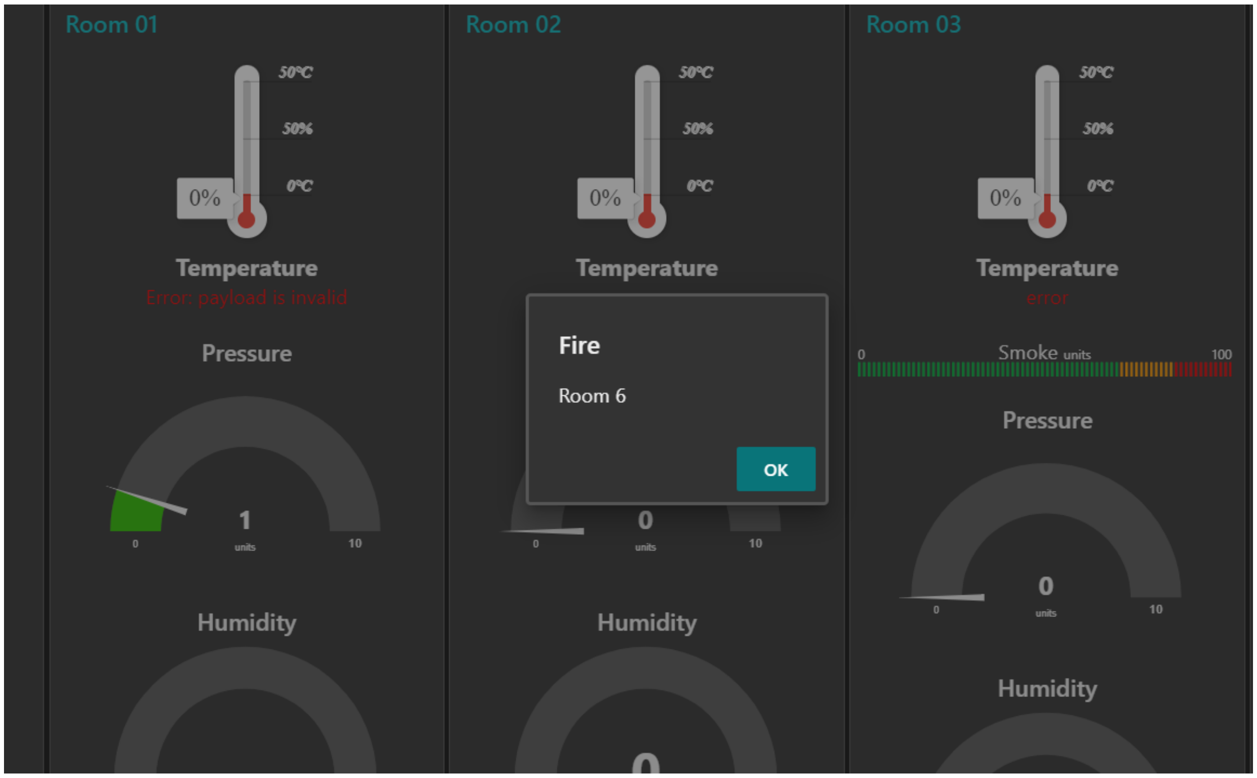
  - Pressure



- The Overall SCADA Dashboard for one room

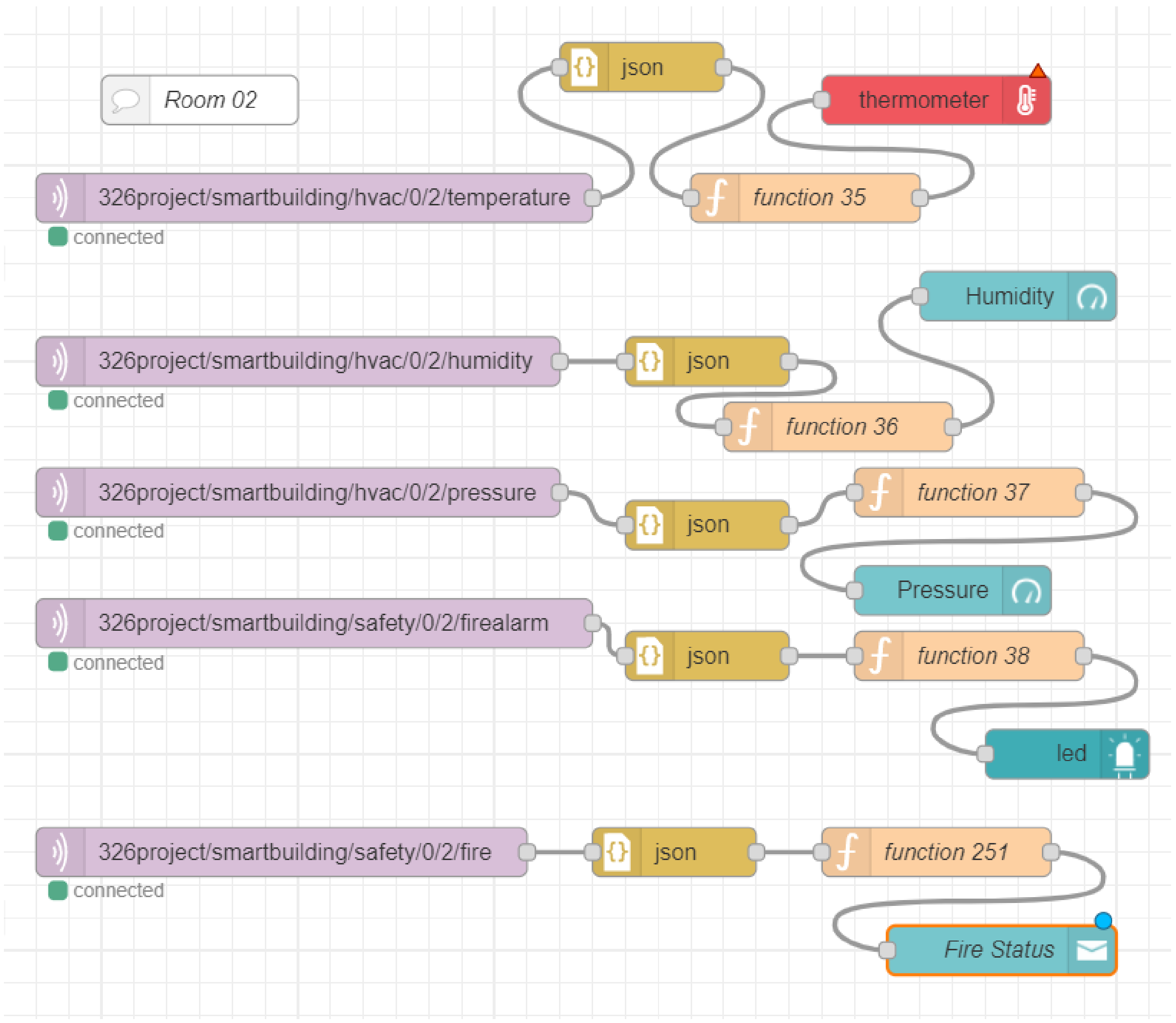


Additionally, if any of the fire detectors detect fire, a notification pops-up overlaying the entire SCADA dashboard indicating where the fire is.



Considering a single room,

	MQTT Topic	**Publisher**
Temperature	326project/smartbuilding/hvac/<floor_no>/<room_no>/temperature	HVAC
Pressure	326project/smartbuilding/hvac/<floor_no>/<room_no>/pressure	HVAC
Humidity	326project/smartbuilding/hvac/<floor_no>/<room_no>/humidity	HVAC
Smoke	326project/smartbuilding/safety/<floor_no>/<room_no>/smoke	Safety
Sprinkler Status	326project/smartbuilding/safety/<floor_no>/<room_no>/sprinkler	Safety
Fire Alarm Status	326project/smartbuilding/hvac/<floor_no>/<room_no>/firealarm	Safety
Pull Station Status	326project/smartbuilding/hvac/<floor_no>/<room_no>/pullstation	Safety
Fire detector	326project/smartbuilding/safety/<floor_no>/<room_no>/fire	Safety

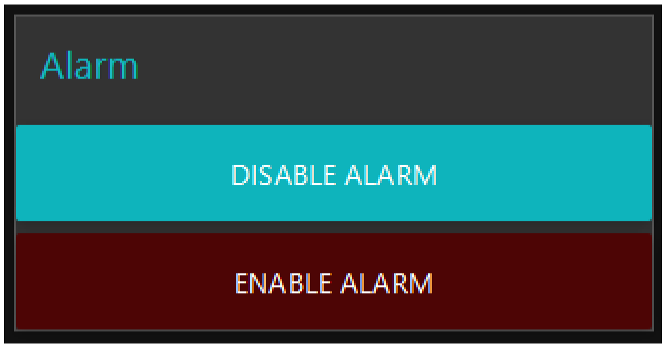

Room 02
json
thermometer
326project/smartbuilding/hvac/0/2/temperature
connected
function 35
Humidity
326project/smartbuilding/hvac/0/2/humidity
connected
json
function 36
326project/smartbuilding/hvac/0/2/pressure
connected
function 37
json
Pressure
326project/smartbuilding/safety/0/2/firealarm
connected
json
function 38
led
326project/smartbuilding/safety/0/2/fire
connected
json
function 251
Fire Status

NodeRed implementation of the SCADA for Room 2 of Floor 0

Inputs from the SCADA are sent to the MQTT server

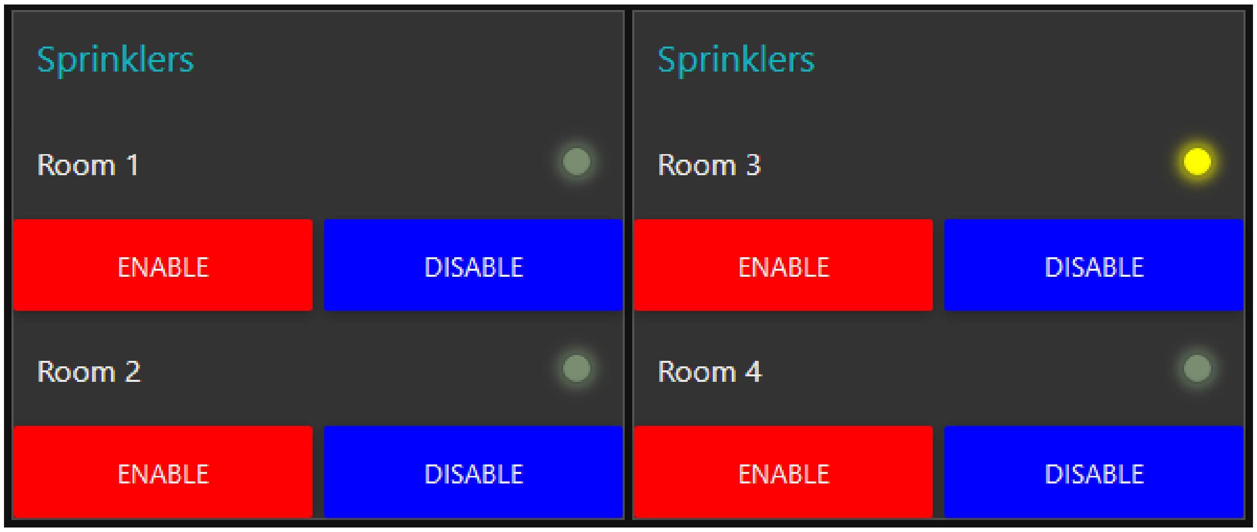
The details below in this section explain the controls for one floor and the same applies to other floors as well.

The SCADA will have controls to trigger alarms on and off. This is analogous to a pulsation inside SCADA. Alarms are for a whole floor, so controlling different alarm modules individually is not facilitated.



SCADA dashboard allows controlling the sprinklers in each room. And it shows whether the sprinkler is enabled or not with a yellow light.

SCADA publishes the payload to the suitable topic(eg: 326project/smartbuilding/safety/0/1/sprinkler) and the microcontroller which has subscribed to the particular topic will enable or disable the sprinkler.

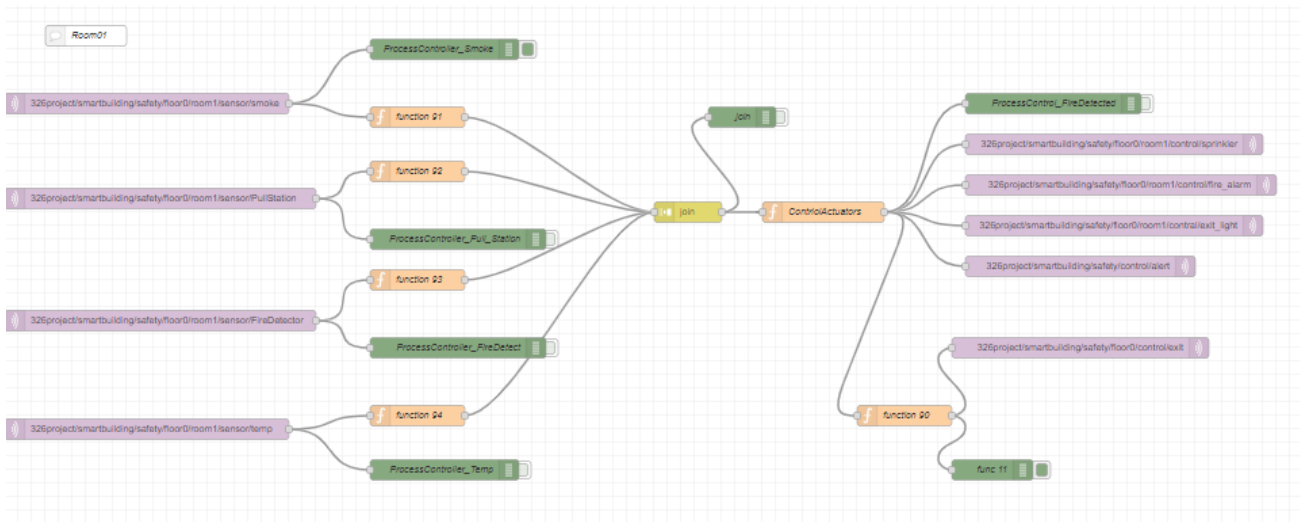


The yellow notification lights are sensitive to both SCADA dashboard buttons and actual physical controls.
Similar segments are implemented for each room on a floor.

Process controller with operating and optimizing process and algorithms

Activating actuators on fire

To determine if a room appears to be on fire, the temperature and smoke sensor outputs are used. If the temperature and smoke level go above the threshold value or the pull station is activated, ‘fire’ message and the room number will be published to the database, and actuators will start to work. We used an algorithm to determine the threshold.


Room01
ProcessController_Smoke
326project/smartbuilding/safety/floor0/room1/sensor/smoke
function 91
function 92
326project/smartbuilding/safety/floor0/room1/sensor/PullStation
ProcessController_Pull_Station
function 93
326project/smartbuilding/safety/floor0/room1/sensor/FireDetector
ProcessController_FireDetect
function 94
326project/smartbuilding/safety/floor0/room1/sensor/temp
ProcessController_Temp
join
join
ControlActuators
ProcessControl_FireDetected
326project/smartbuilding/safety/floor0/room1/control/sprinkler
326project/smartbuilding/safety/floor0/room1/control/fire_alarm
326project/smartbuilding/safety/floor0/room1/control/exit_light
326project/smartbuilding/safety/control/alert
326project/smartbuilding/safety/floor0/control/exit
function 90
func 11

Fire evacuation plan

Properly designed evacuation strategies are key to ensuring that building occupants and responsible persons feel confident and prepared to safely exit a building in the event of an emergency. They save lives when carried out correctly.

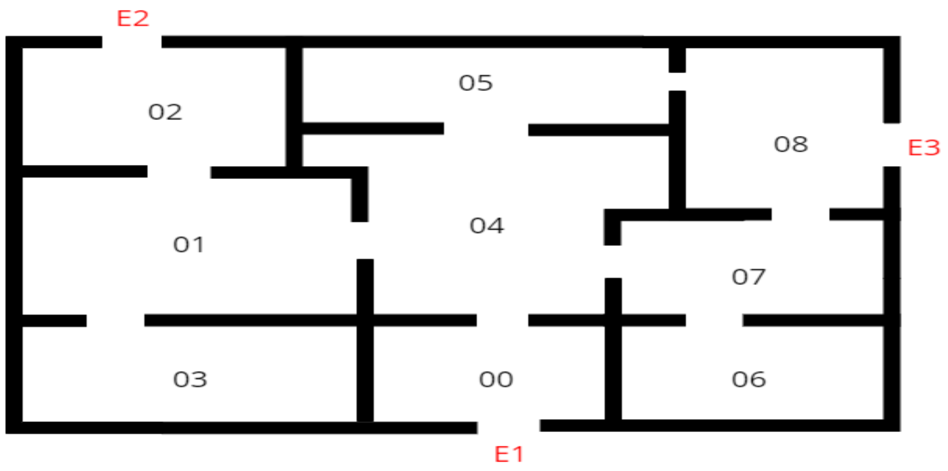
Horizontal Phased Evacuation :

- Horizontal, phased evacuation is the method of moving people away from the area of danger to a safer place on the same floor.
- The occupants of each floor should be directed to the nearest emergency exit.
- We created a function based on Dijkstra's algorithm that can be used to find the closest emergency exit from each zone and output the paths to the closest exit. These routes will be chosen based on the location of the fire.

Steps : First, all relevant data from the database will be imported.

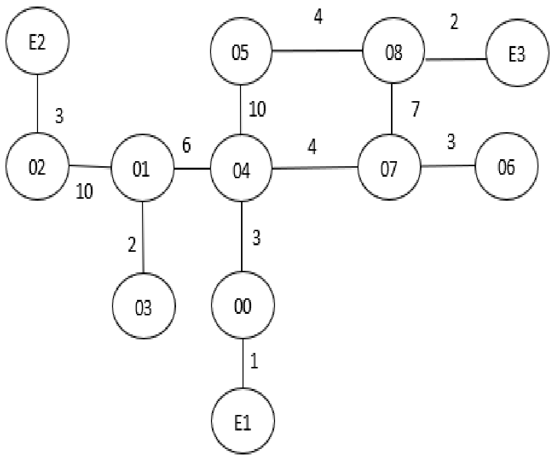
    - Status of each room (Whether or not a room is affected by fire)
    - Space between rooms

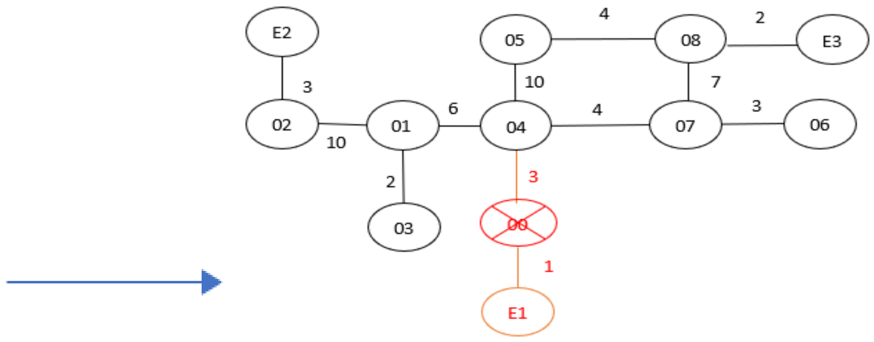
- Then, using the state of each node, a rule-based algorithm will generate a list of nodes that are not affected by fire in order to avoid visiting dangerous areas.
- A new graph based on the chosen nodes will be generated after the edges connected to the unsafe nodes are removed.



Sample state :

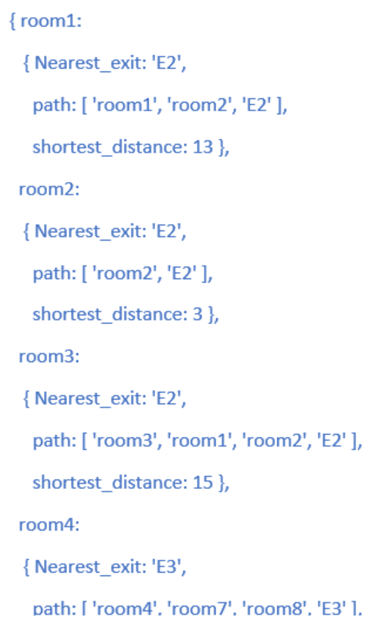
Zone	State
00	True
01	False
02	False
03	False
04	False
05	False
06	False
07	False
08	False





- Then dijkstra's algorithm will be applied to the new graph to determine the shortest path between each safe node and all emergency exits, and the closest exit will be chosen.

- The function's output will include the distance, the route that must be taken, and the closest exit for each node.

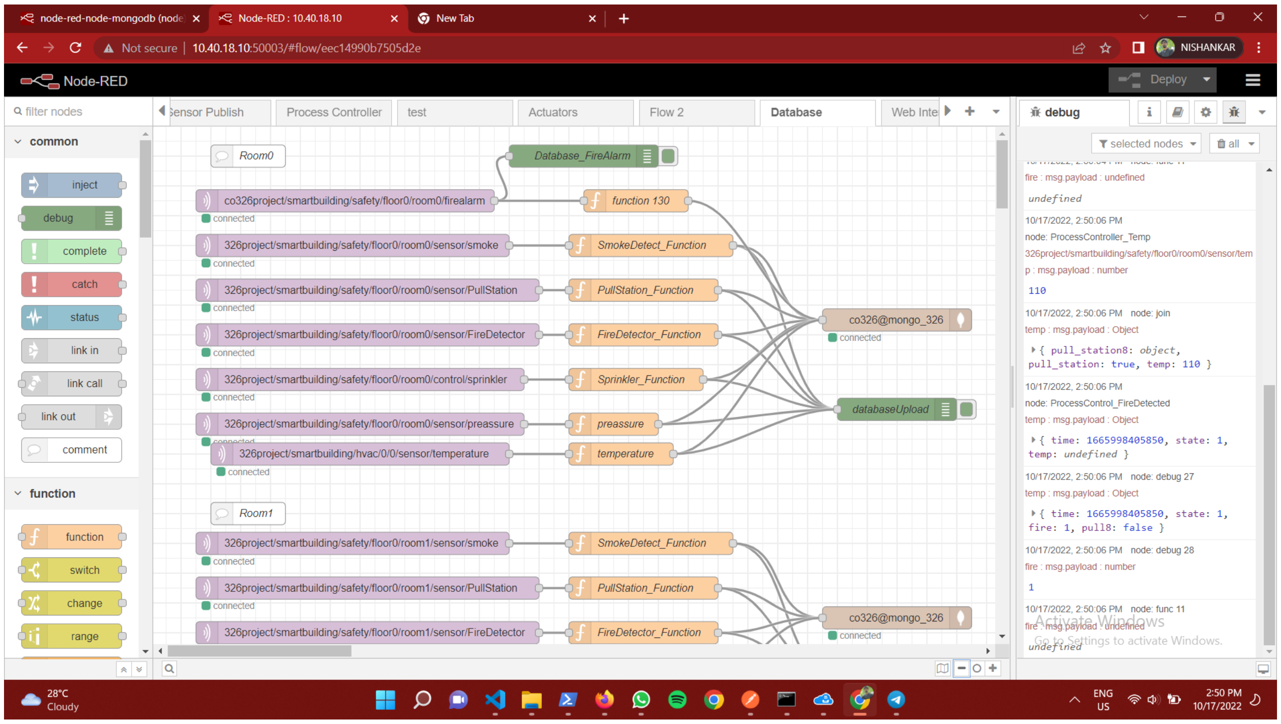


```
{ room1:
   { Nearest_exit: 'E2',
     path: [ 'room1', 'room2', 'E2' ],
     shortest_distance: 13 },
  room2:
   { Nearest_exit: 'E2',
     path: [ 'room2', 'E2' ],
     shortest_distance: 3 },
  room3:
   { Nearest_exit: 'E2',
     path: [ 'room3', 'room1', 'room2', 'E2' ],
     shortest_distance: 15 },
  room4:
   { Nearest_exit: 'E3',
     path: [ 'room4', 'room7', 'room8', 'E3' ],
```

MQTT Data, Commands and events should be stored in the database

Storing data

Every sensor data and the output from the process control (Actuator states) will be stored in the database for future analytics, to determine the escape route when a fire is detected and showcasing the values on a web interface.


node-red-node-mongodb (node
Node-RED : 10.40.18.10
New Tab
Not secure | 10.40.18.10:50003/#flow/eec14990b7505d2e
NISHANKAR
Node-RED
Deploy
filter nodes
Sensor Publish
Process Controller
test
Actuators
Flow 2
Database
Web Inte
debug
selected nodes
all
common
inject
debug
complete
catch
status
link in
link call
link out
comment
function
function
switch
change
range
Room0
Database_FireAlarm
326project/smartbuilding/safety/floor0/room0/firealarm
function 130
326project/smartbuilding/safety/floor0/room0/sensor/smoke
SmokeDetect_Function
326project/smartbuilding/safety/floor0/room0/sensor/PullStation
PullStation_Function
326project/smartbuilding/safety/floor0/room0/sensor/FireDetector
FireDetector_Function
co326@mongo_326
326project/smartbuilding/safety/floor0/room0/control/sprinkler
Sprinkler_Function
databaseUpload
326project/smartbuilding/safety/floor0/room0/sensor/preassure
preassure
326project/smartbuilding/hvac/0/0/sensor/temperature
temperature
connected
Room1
326project/smartbuilding/safety/floor0/room1/sensor/smoke
SmokeDetect_Function
326project/smartbuilding/safety/floor0/room1/sensor/PullStation
PullStation_Function
326project/smartbuilding/safety/floor0/room1/sensor/FireDetector
FireDetector_Function
co326@mongo_326
Activate Windows
Go to Settings to activate Windows.
28°C
Cloudy
ENG
US
2:50 PM
10/17/2022

```
fire : msg.payload : undefined
undefined

10/17/2022, 2:50:06 PM
node: ProcessController_Temp
326project/smartbuilding/safety/floor0/room0/sensor/temp : msg.payload : number
110

10/17/2022, 2:50:06 PM   node: join
temp : msg.payload : Object
{ pull_station8: object, pull_station: true, temp: 110 }

10/17/2022, 2:50:06 PM
node: ProcessControl_FireDetected
temp : msg.payload : Object
{ time: 1665998405850, state: 1, temp: undefined }

10/17/2022, 2:50:06 PM   node: debug 27
temp : msg.payload : Object
{ time: 1665998405850, state: 1, fire: 1, pull8: false }

10/17/2022, 2:50:06 PM   node: debug 28
fire : msg.payload : number
1

10/17/2022, 2:50:06 PM   node: func 11
fire : msg.payload : undefined
undefined
```

This sensor data for every room (totally 7 rooms) is saved in the database. The database we used to maintain the data is mongodb because it's no SQL so no need to create any models and best suitable for real time data management.

It's an example of the database function which is used to store the smoke detection data.
In the database for sensor every value from all the rooms will be saved; they can be identified separately by the msg.topic which will indicate the floor no and the room number.

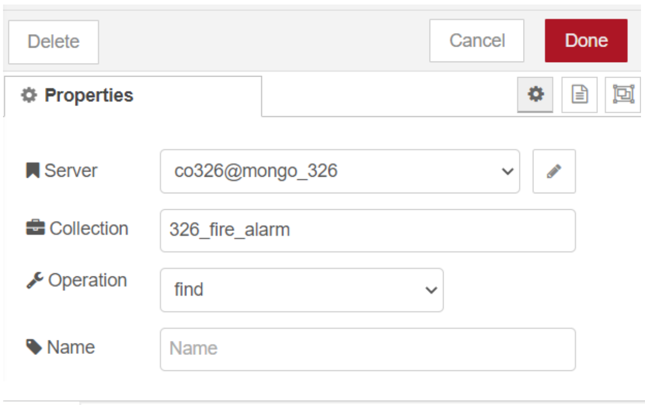
```
msg.collection ='326_smoke_detect'
msg.payload={
    "time": Date.now(),
    "data": msg.payload
}
return msg;
```

<u>Retrieving data</u>

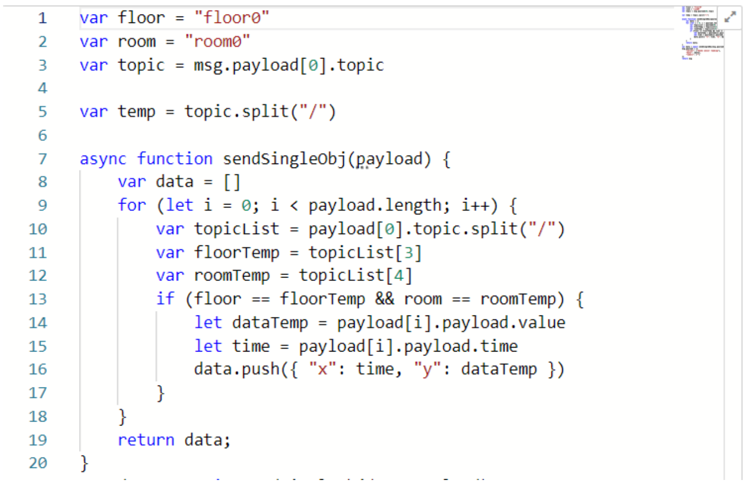
Need to retrieve the data from the database to visualize them on the web interface and to find the safest and shortest path for the people to evacuate the building when a fire is detected.

Fire detection will save on the database with its room number and floor number so when retrieving the data we can analyze it to find the path.

Retrieving data from the database for web interface:



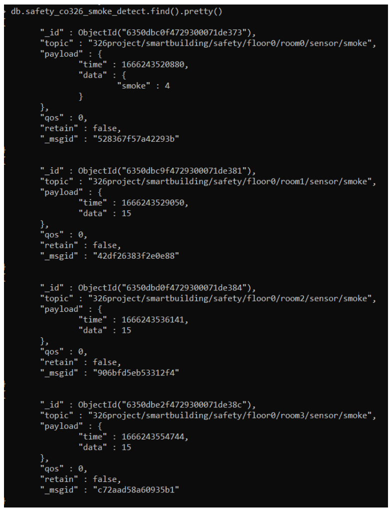
Server: co326@mongo_326
Collection: 326_fire_alarm
Operation: find
Name: Name



```
var floor = "floor0"
var room = "room0"
var topic = msg.payload[0].topic

var temp = topic.split("/")

async function sendSingleObj(payload) {
    var data = []
    for (let i = 0; i < payload.length; i++) {
        var topicList = payload[0].topic.split("/")
        var floorTemp = topicList[3]
        var roomTemp = topicList[4]
        if (floor == floorTemp && room == roomTemp) {
            let dataTemp = payload[i].payload.value
            let time = payload[i].payload.time
            data.push({ "x": time, "y": dataTemp })
        }
    }
    return data;
}
```



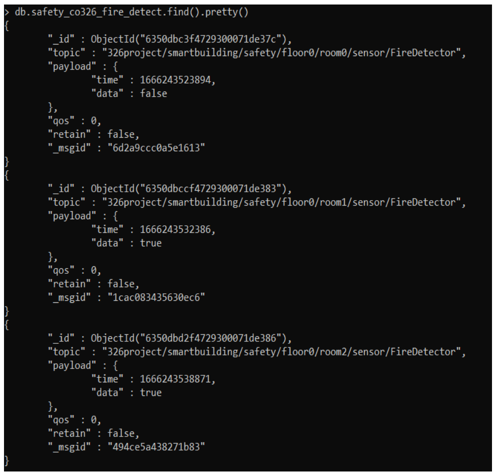
```
db.safety_co326_smoke_detect.find().pretty()

        "_id" : ObjectId("6350dbc0f4729300071de373"),
        "topic" : "326project/smartbuilding/safety/floor0/room0/sensor/smoke",
        "payload" : {
                "time" : 1666243520880,
                "data" : {
                        "smoke" : 4
                }
        },
        "qos" : 0,
        "retain" : false,
        "_msgid" : "528367f57a42293b"

        "_id" : ObjectId("6350dbc9f4729300071de381"),
        "topic" : "326project/smartbuilding/safety/floor0/room1/sensor/smoke",
        "payload" : {
                "time" : 1666243529050,
                "data" : 15
        },
        "qos" : 0,
        "retain" : false,
        "_msgid" : "42df26383f2e0e88"

        "_id" : ObjectId("6350dbd0f4729300071de384"),
        "topic" : "326project/smartbuilding/safety/floor0/room2/sensor/smoke",
        "payload" : {
                "time" : 1666243536141,
                "data" : 15
        },
        "qos" : 0,
        "retain" : false,
        "_msgid" : "906bfd5eb53312f4"

        "_id" : ObjectId("6350dbe2f4729300071de38c"),
        "topic" : "326project/smartbuilding/safety/floor0/room3/sensor/smoke",
        "payload" : {
                "time" : 1666243554744,
                "data" : 15
        },
        "qos" : 0,
        "retain" : false,
        "_msgid" : "c72aad58a60935b1"
```



```
> db.safety_co326_fire_detect.find().pretty()
{
        "_id" : ObjectId("6350dbc3f4729300071de37c"),
        "topic" : "326project/smartbuilding/safety/floor0/room0/sensor/FireDetector",
        "payload" : {
                "time" : 1666243523894,
                "data" : false
        },
        "qos" : 0,
        "retain" : false,
        "_msgid" : "6d2a9ccc0a5e1613"
}
{
        "_id" : ObjectId("6350dbccf4729300071de383"),
        "topic" : "326project/smartbuilding/safety/floor0/room1/sensor/FireDetector",
        "payload" : {
                "time" : 1666243532386,
                "data" : true
        },
        "qos" : 0,
        "retain" : false,
        "_msgid" : "1cac083435630ec6"
}
{
        "_id" : ObjectId("6350dbd2f4729300071de386"),
        "topic" : "326project/smartbuilding/safety/floor0/room2/sensor/FireDetector",
        "payload" : {
                "time" : 1666243538871,
                "data" : true
        },
        "qos" : 0,
        "retain" : false,
        "_msgid" : "494ce5a438271b83"
}
```

Web interface or SCADA pages should display the data in the database

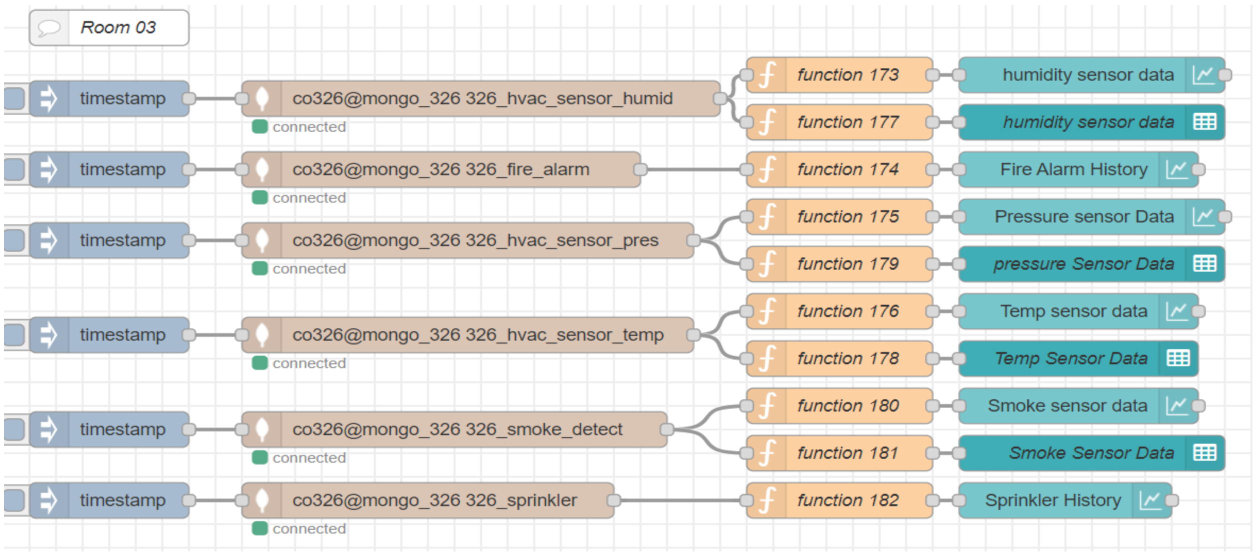
The web interface shows the past data on the MongoDB database. it is located in the collections named

- 326_fire_detect
- 326_smoke_detect
- 326_sprinkler
- 326_fire_Alarm
- 326_pull_station
- 326_voltage
- 326_current

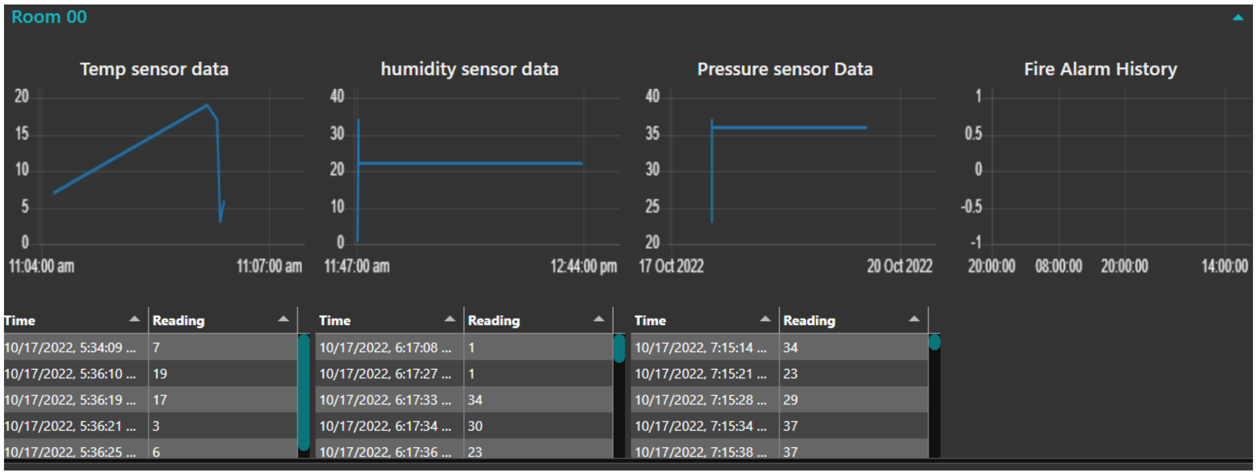
SCADA interface has a tab for web interface it fetches data from the MongoDB database and shows the data in the format of the line chart. The chart shows the sensor reading value wrt generated time.

Every Room have separate section these section shows

- Temperature sensor data
- Pressure sensor data
- Humidity sensor data
- Smoke sensor data
- status of the
  - Alarm
  - Pull station
  - Sprinkler



Node-red development for Room0 in floor0



Time	Reading
10/17/2022, 5:34:09 ...	7
10/17/2022, 5:36:10 ...	19
10/17/2022, 5:36:19 ...	17
10/17/2022, 5:36:21 ...	3
10/17/2022, 5:36:25 ...	6

Time	Reading
10/17/2022, 6:17:08 ...	1
10/17/2022, 6:17:27 ...	1
10/17/2022, 6:17:33 ...	34
10/17/2022, 6:17:34 ...	30
10/17/2022, 6:17:36 ...	23

Time	Reading
10/17/2022, 7:15:14 ...	34
10/17/2022, 7:15:21 ...	23
10/17/2022, 7:15:28 ...	29
10/17/2022, 7:15:34 ...	37
10/17/2022, 7:15:38 ...	37

Web interface out for Room0 in floor0

Created separate section for every room in floors. Every room shows their corresponding data.

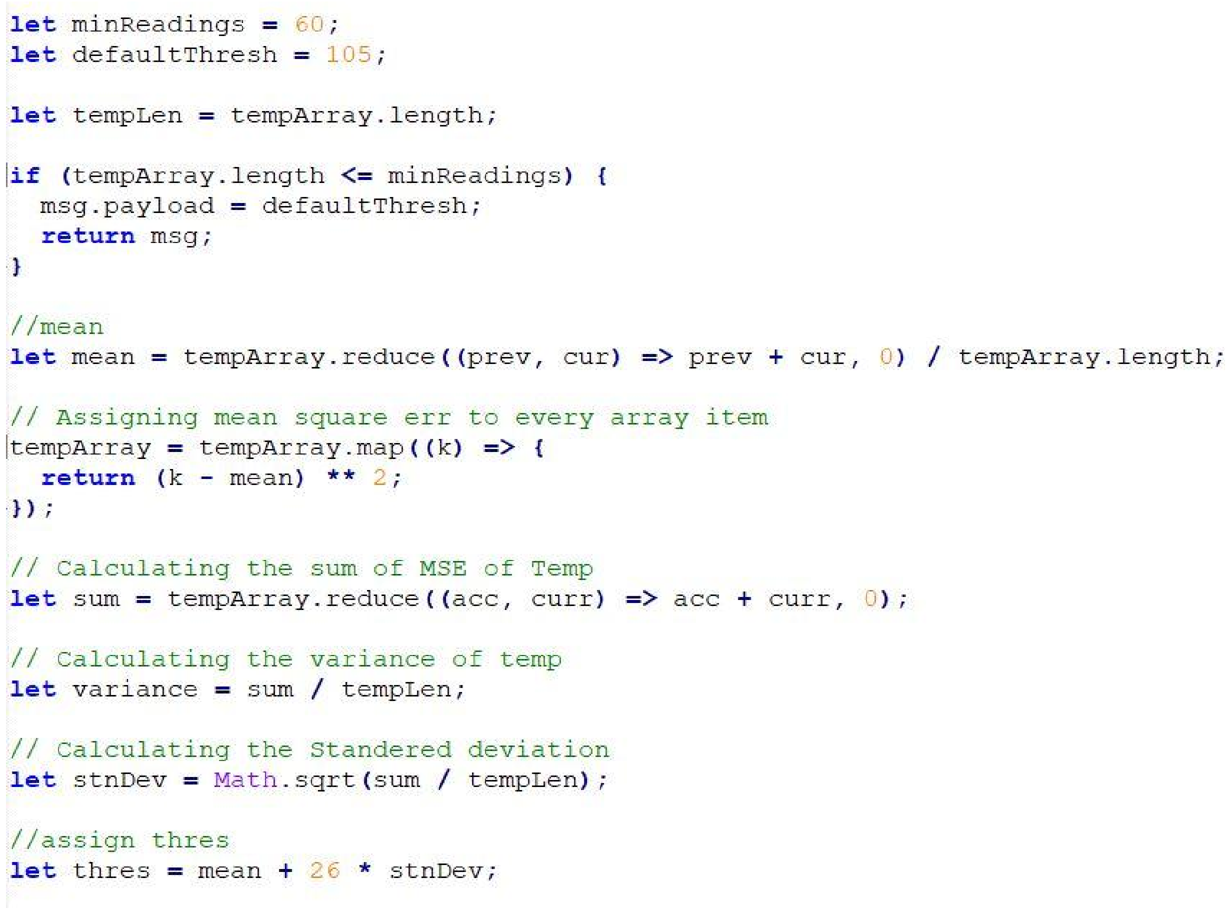
-

Data analytics: Prediction, Optimization, Correlation

Find Threshold value of temperature

There are two different types of finding threshold value, a static and a rate of rise detector. A fixed temperature threshold is designed to detect at a set temperature. Fixed temperature detectors find fire when the room temperature is higher than the set point. Static datapoint thresholds which are assigned manually, and rate of rise detector are calculated by anomaly detection algorithms and continuously trained by a datapoint's recent historical values.
A rate of rise heat detector responds like a standard, fixed-temperature heat detector with slowly developing fires. When fires build, quickly the rate-of-rise compensation detects when their set point is reached and detects fire and will send a signal to the control if the rise of temperature is nearly 15 degrees F (8.3 C) or more per minute. Users should verify that the space where the heat detector will be installed does not have naturally rapid temperature rises that exceed the detector’s trip point.
As the static threshold value is less accurate, we use the rate of rise detector, and so that we calculate the mean, standard deviation of recent historical values. From that, the sudden rise of temperature in a short time period matches according to the number of times standard deviation. So, each room’s threshold value will be calculated. Improvement and adjustment of this algorithm is needed to get the best accuracy result proper to local or regional conditions. As per the kaggle temperature data set, It is detected as the number of times of standard deviation

is 26. As per that the below algorithm will detect the threshold temperature of each room. For any other concern, the default static threshold value is published, it will be used when enough data can not be taken in the recent past.



```
let minReadings = 60;
let defaultThresh = 105;

let tempLen = tempArray.length;

if (tempArray.length <= minReadings) {
  msg.payload = defaultThresh;
  return msg;
}

//mean
let mean = tempArray.reduce((prev, cur) => prev + cur, 0) / tempArray.length;

// Assigning mean square err to every array item
tempArray = tempArray.map((k) => {
  return (k - mean) ** 2;
});

// Calculating the sum of MSE of Temp
let sum = tempArray.reduce((acc, curr) => acc + curr, 0);

// Calculating the variance of temp
let variance = sum / tempLen;

// Calculating the Standered deviation
let stnDev = Math.sqrt(sum / tempLen);

//assign thres
let thres = mean + 26 * stnDev;
```

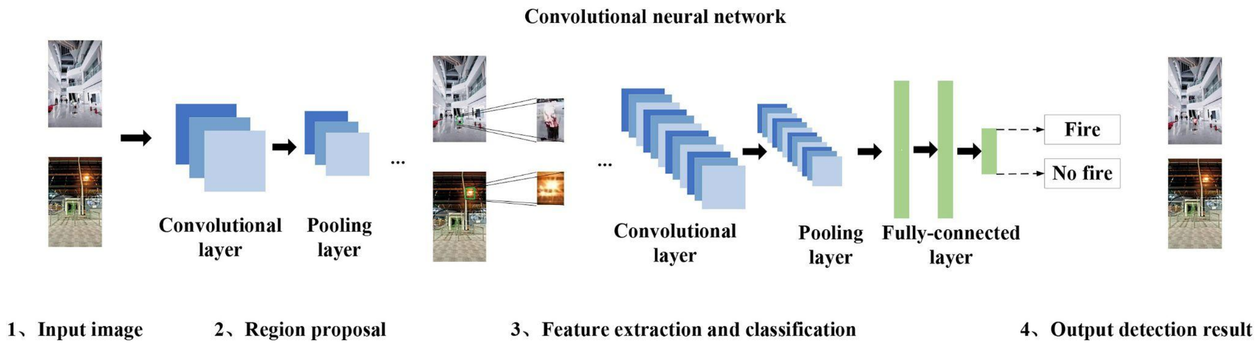
.

Prediction

As a new fire detection technology, image fire detection has recently played a crucial role in reducing fire losses by alarming users early through early fire detection. Image fire detection is based on an algorithmic analysis of images. However, there is a lower accuracy, delayed detection, and a large amount of computation in common detection algorithms, including manually and machine automatically extracting image features.

With rapid economic development, the increasing scale and complexity of constructions has introduced great challenges in fire control. Therefore, early fire detection and alarm with high sensitivity and accuracy is essential to reduce fire losses. However, traditional fire detection technologies, like smoke and heat detectors, are not suitable for large spaces, complex buildings, or spaces with many disturbances. Due to the limitations of above detection technologies, missed detections, false alarms, detection delays and other problems often occur, making it even more difficult to achieve early fire warnings. So we came to image based fire detection techniques. The technique has many advantages such as early fire detection, high accuracy, flexible system installation, and the capability to effectively detect fires in large spaces and complex building structures. It processes image data from a camera by algorithms to determine the presence of a fire or fire risk in images. Therefore, the detection algorithm is the core of this technology, directly determining the performance of the image fire detector.

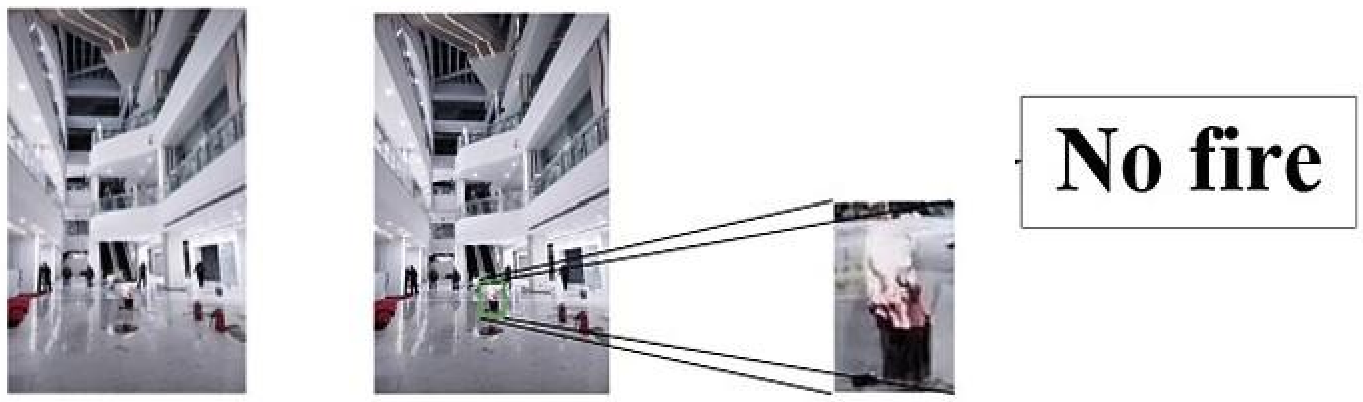
Convolutional neural network(CNN) **architectures**
According to the principle of object detection algorithms, the flow of image fire detection algorithms based on convolutional neural networks is designed. It will help to predict the huge fire before the disaster.The detection CNN has functions of region proposals, feature extraction and classification. Firstly, The CNN takes an image as input and outputs region proposals by convolution, pooling, etc. Secondly, the region-based object detection CNN decides the presence or absence of fire in proposal regions through convolutional layers, pooling layers, fully-connected layers, etc.



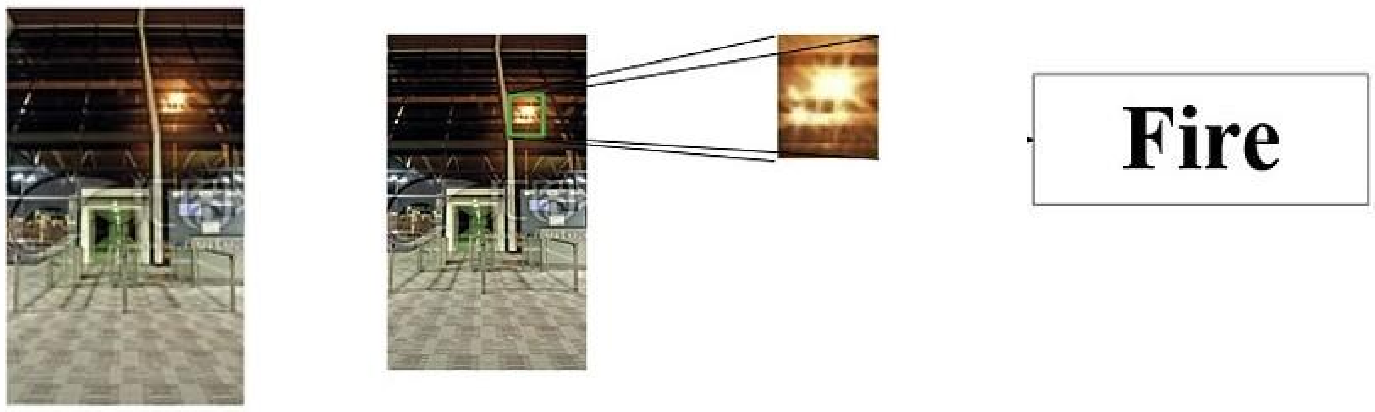
A new special convolutional neural network was developed to detect fire regions using the existing YOLOv3 algorithm (a real-time object detection algorithm that identifies specific objects in videos, live feeds, or images.). We selected the YOLOv3 network to improve and use it for the successful detection and warning of fire disasters through real time cameras.

By modifying the algorithm, we can record the results of a rapid and high-precision detection of fire, during both day and night, irrespective of the shape and size. Another advantage is that the algorithm is capable of detecting fires that are 1 m long and 0.3 m wide at a distance of 50 m.

Input: : Region selector: Output



Input: : Region selector: Output



References

https://www.alarmgrid.com/faq/at-what-temperature-do-heat-alarms-trigger#:~:text=Heat%20Detectors%20react%20to%20the.panel%20and%20trigger%20an%20alarm.

https://realpars.com/fire-alarm-system/

https://www.firesafe.org.uk/fire-emergency-evacuation-plan-or-fire-procedure/

https://www.kaggle.com/datasets/javi2270784/gas-sensor-array-temperature-modulation

https://www.sciencedirect.com/topics/engineering/convolutional-layer