

PROJECT DEVELOPMENT PHASE –SPRINT 4

DATE	13 NOV 2022
TEAM ID	PNT2022TMID29717
PROJECT TITLE	INDUSTRY SPECIFIC INTELLIGENT FIRE MANAGEMENT SYSTEM

Sprint 4

Program:

```
#include <time.h>
```

```
bool exhaust_fan_on = false;
```

```
bool sprinkler_on = false;
```

```
float temperature = 0;
```

```
int gas = 0;
```

```
int flame = 0;
```

```
String flame_status = "";
```

```
String accident_status = "";
```

```
String sprinkler_status = "";
```

```
void setup() {  
    Serial.begin(99900);  
}
```

```
void loop() {
```

```
    //setting a random seed
```

```
    srand(time(0));
```

```
    //initial variable
```

```
    temperature = random(-20,125);
```

```
    gas = random(0,1000);
```

```
    int flamereading = random(200,1024);
```

```
    flame = map(flamereading,0,1024,0,2);
```

```
    //set a flame status
```

```
    switch (flame) {
```

```

case 0:
    flame_status = "No Fire";
    Serial.println("Flame Status : "+flame_status);
    break;
case 1:
    flame_status = "Fire is Detected";
    Serial.println("Flame Status : "+flame_status);
    break;
}

//Gas Detection

if(gas > 100){
    Serial.println("Gas Status : Gas leakage Detected");
}
else{
    exhaust_fan_on = false;
    Serial.println("Gas Status : No Gas leakage Detected");
}

//send the sprinkler status
if(flame){
    sprinkler_status = "working";
    Serial.println("Sprinkler Status : "+sprinkler_status);
}
else{
    sprinkler_status = "not working";
    Serial.println("Sprinkler Status : "+sprinkler_status);
}

//toggle the fan according to gas

if(gas > 100){
    exhaust_fan_on = true;
    Serial.println("Exhaust fan Status : Working");
}
else{
    exhaust_fan_on = false;
    Serial.println("Exhaust fan Status : Not Working");
}

Serial.println("");
Serial.println("");
Serial.println(".....*****.....");
Serial.println("");
Serial.println("");

```

```

    delay(3000);
}

```

Output:

The screenshot shows the WOKWI simulation environment. On the left, the 'sketch.ino' file is open, displaying the initial setup code. The code includes a delay of 3000ms and initializes variables for exhaust fan, sprinkler, temperature, gas, flame, and status strings. The 'setup' function begins serial communication at 999000 baud. The 'loop' function is currently empty. On the right, the 'Simulation' window shows the output of the first state. The status is 'Fire is Detected', gas status is 'Gas leakage Detected', sprinkler status is 'working', and exhaust fan status is 'Working'. The simulation is running at 35% speed.

```

1 #include <time.h>
2
3
4 bool exhaust_fan_on = false;
5 bool sprinkler_on = false;
6
7 float temperature = 0;
8
9 int gas = 0;
10 int flame = 0;
11
12 String flame_status = "";
13 String accident_status = "";
14 String sprinkler_status = "";
15
16
17 void setup() {
18
19     Serial.begin(999000);
20
21 }
22
23 void loop(){
24
25 //setting a random seed
26
27 srand(time(0));
28
29
30 //initial variable

```

Simulation Output:

```

Fire is Detected
Gas Status : Gas leakage Detected
Sprinkler Status : working
Exhaust fan Status : Working

```

Flame Status : Fire is Detected
Gas Status : Gas leakage Detected
Sprinkler Status : working
Exhaust fan Status : Working

The screenshot shows the WOKWI simulation environment. On the left, the 'sketch.ino' file is open, displaying the code for the second state. The code initializes variables for temperature, gas, flame, and status strings. The 'setup' function sets a flame status based on a random value. The 'loop' function checks for gas detection and updates the status strings. On the right, the 'Simulation' window shows the output of the second state. The status is 'No Fire', gas status is 'Gas leakage Detected', sprinkler status is 'not working', and exhaust fan status is 'Working'. The simulation is running at 36% speed.

```

28
29 //initial variable
30
31 temperature = random(-20,125);
32 gas = random(0,1000);
33 int flamereading = random(200,1024);
34 flame = map(flamereading,0,1024,0,2);
35
36
37 //set a flame status
38 switch (flame) {
39
40
41
42 case 0:
43     flame_status = "No Fire";
44     Serial.println("Flame Status : "+flame_status);
45     break;
46 case 1:
47     flame_status = "Fire is Detected";
48     Serial.println("Flame Status : "+flame_status);
49     break;
50 }
51
52
53 //Gas Detection
54
55
56 if(gas > 100){
57     Serial.println("Gas Status : Gas leakage Detected");

```

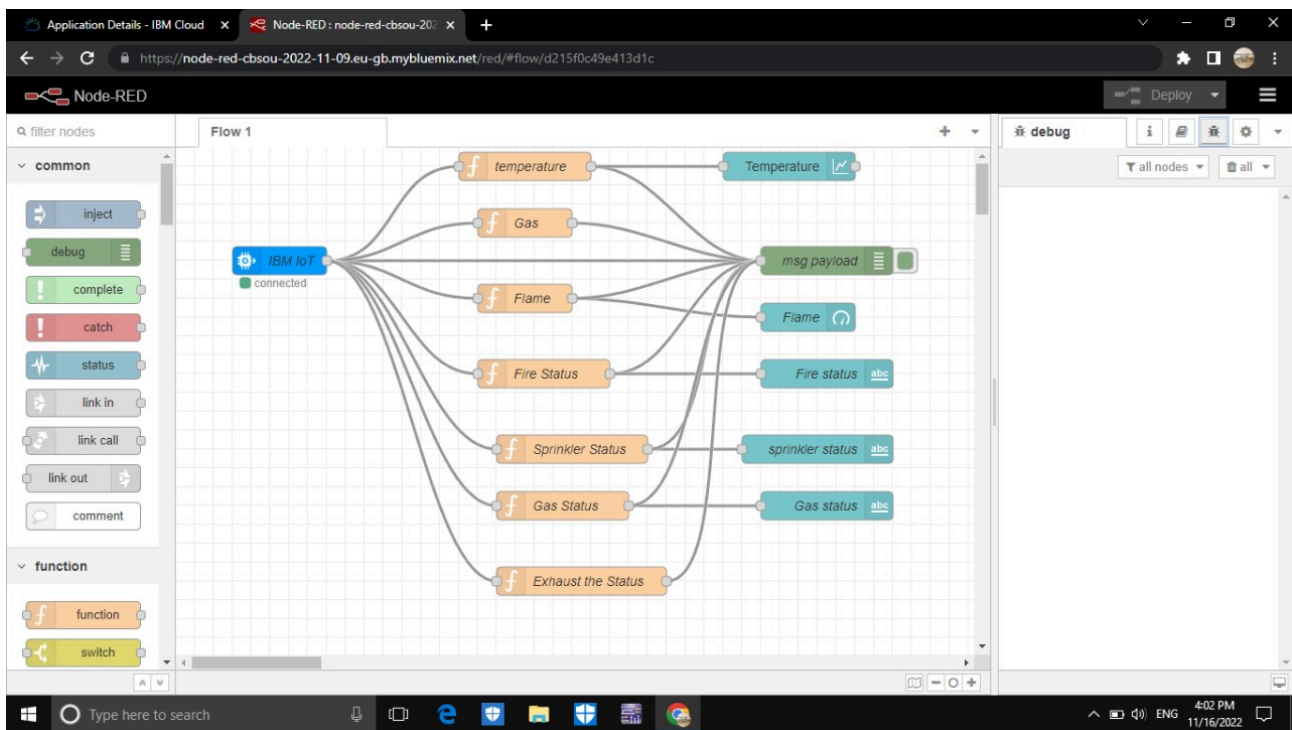
Simulation Output:

```

Flame Status : Fire is Detected
Gas Status : Gas leakage Detected
Sprinkler Status : working
Exhaust fan Status : Working

```

Flame Status : No Fire
Gas Status : Gas leakage Detected
Sprinkler Status : not working
Exhaust fan Status : Working



IBM WATSON IOT PLATFORM

The screenshot shows the IBM Watson IoT Platform dashboard. The main view displays a list of devices. The first device, '1234', is 'Disconnected' and has a type of 'nandy'. Below the device list, there is a table showing recent events:

Event	Value
eventnandhu	{"gas":18,"temp":92,"flame":86}
eventnandhu	{"gas":42,"temp":116,"flame":64}

An overlay window titled 'Device Type: nandy' is open, showing the configuration for the 'eventnandhu' event type. The 'Event type name' is 'eventnandhu'. The 'Schedule' is set to 'Every Minute'. The 'Payload' is configured with a JSON structure:

```

{
  "gas": random(10, 100),
  "temp": random(10, 125),
  "flame": random(50, 100)
}

```

The bottom status bar shows the time as 11:24 PM on 11/18/2022.

NODE RED OUTPUT

