

PROJECT DEVELOPMENT PHASE

DELIVERY OF SPRINT-3

Date	7 November 2022
Team Id	PNT2022TMID01159
Project Name	Hazardous area monitoring for industrial power plants using IOT.

Contributors:

 **Narmadha.M**

 **Nivetha.R**

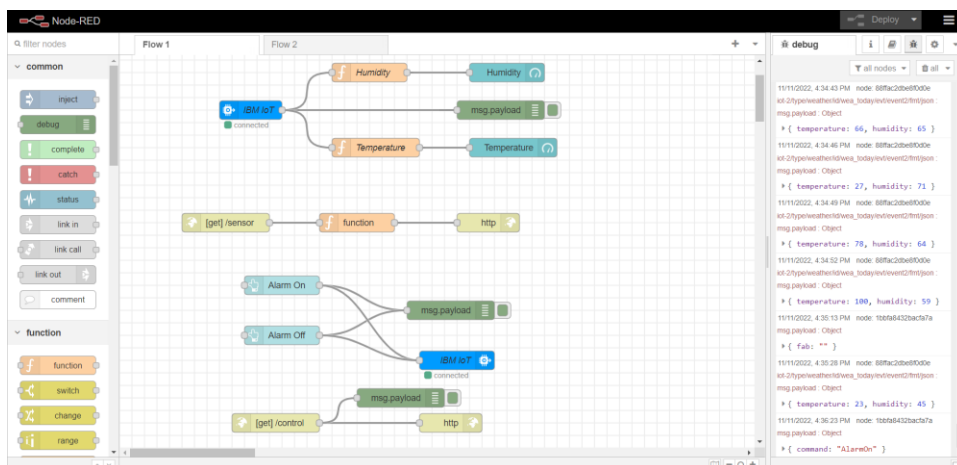
 **Nithyasri.A**

 **Shalini.S**

SPRINT 3: MIT Application Inventor

* Building an application for our project using MIT application, designing the model and testing the application.

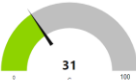
STEP 1: Connecting required nodes in the Node-red platform.



☰ weather detecting

Detecting

Temperature



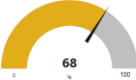
A semi-circular gauge with a green segment on the left and a grey segment on the right. The needle points to the number 31. Below the gauge, '°C' is written.

31
°C

ALARM ON

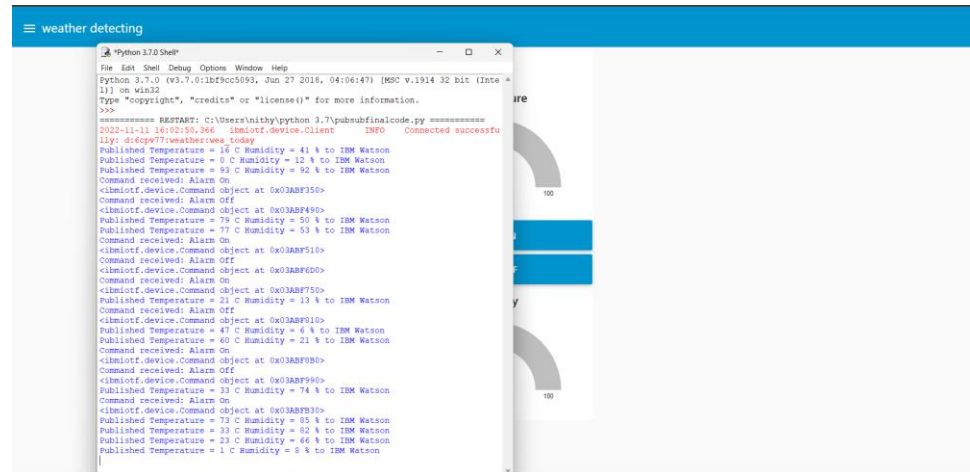
ALARM OFF

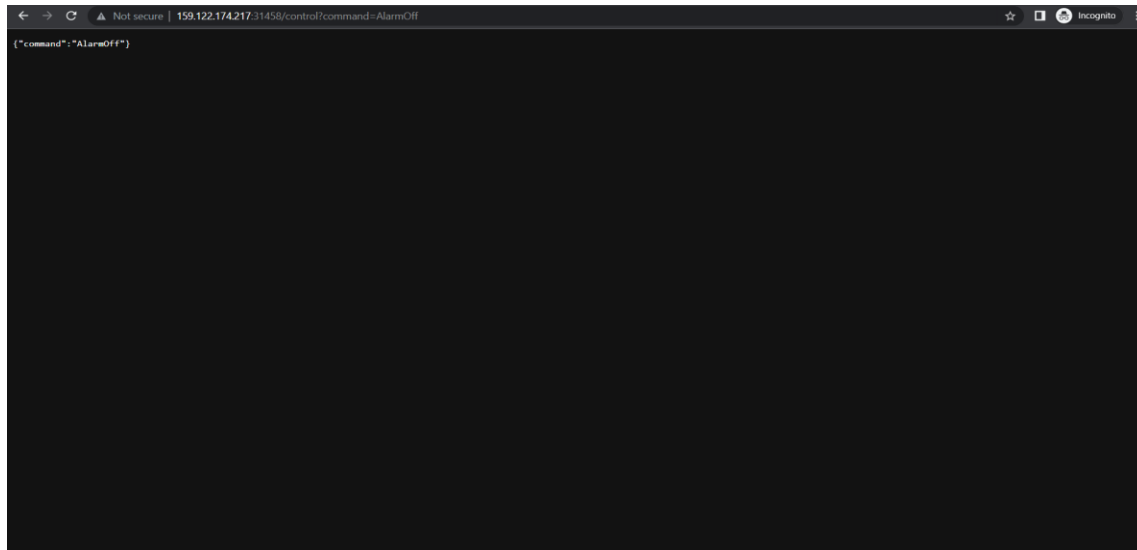
Humidity



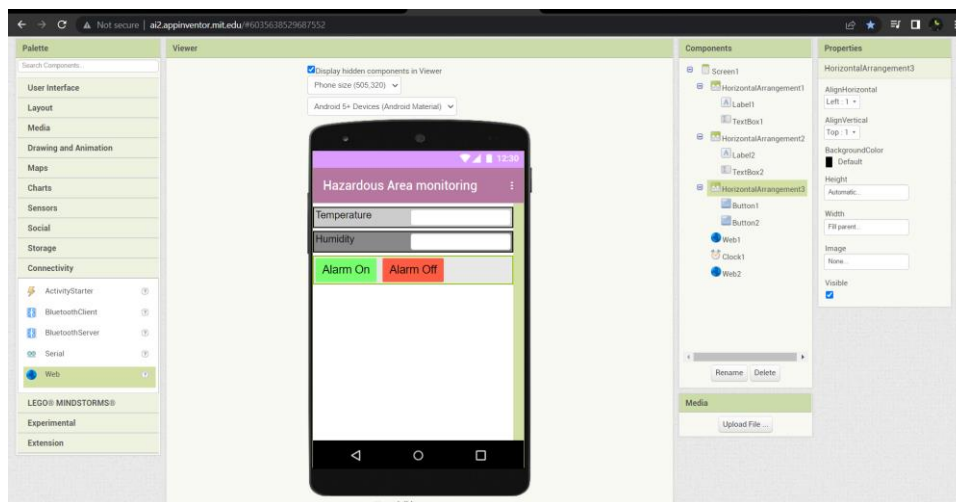
A semi-circular gauge with an orange segment on the left and a grey segment on the right. The needle points to the number 68. Below the gauge, '%' is written.

68
%

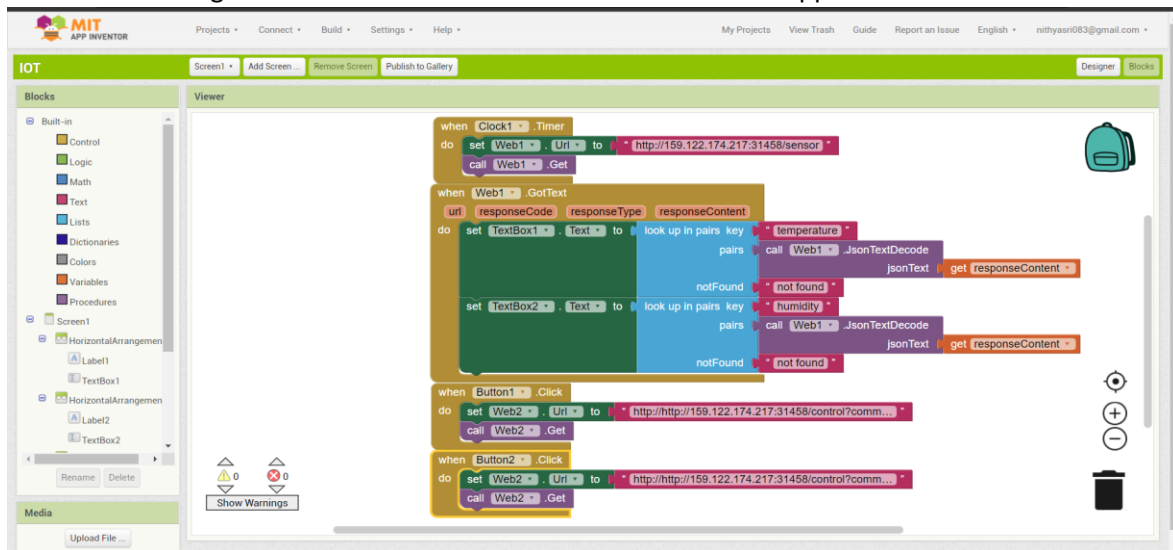




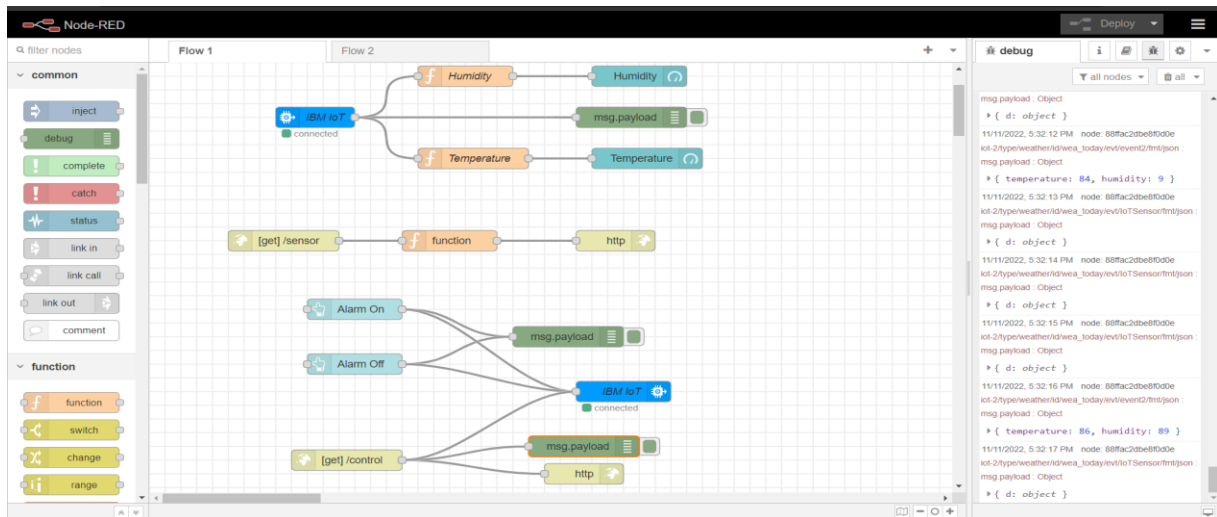
STEP 3: Connecting with the MIT Application Inventor to display temperature, humidity and alarm condition.

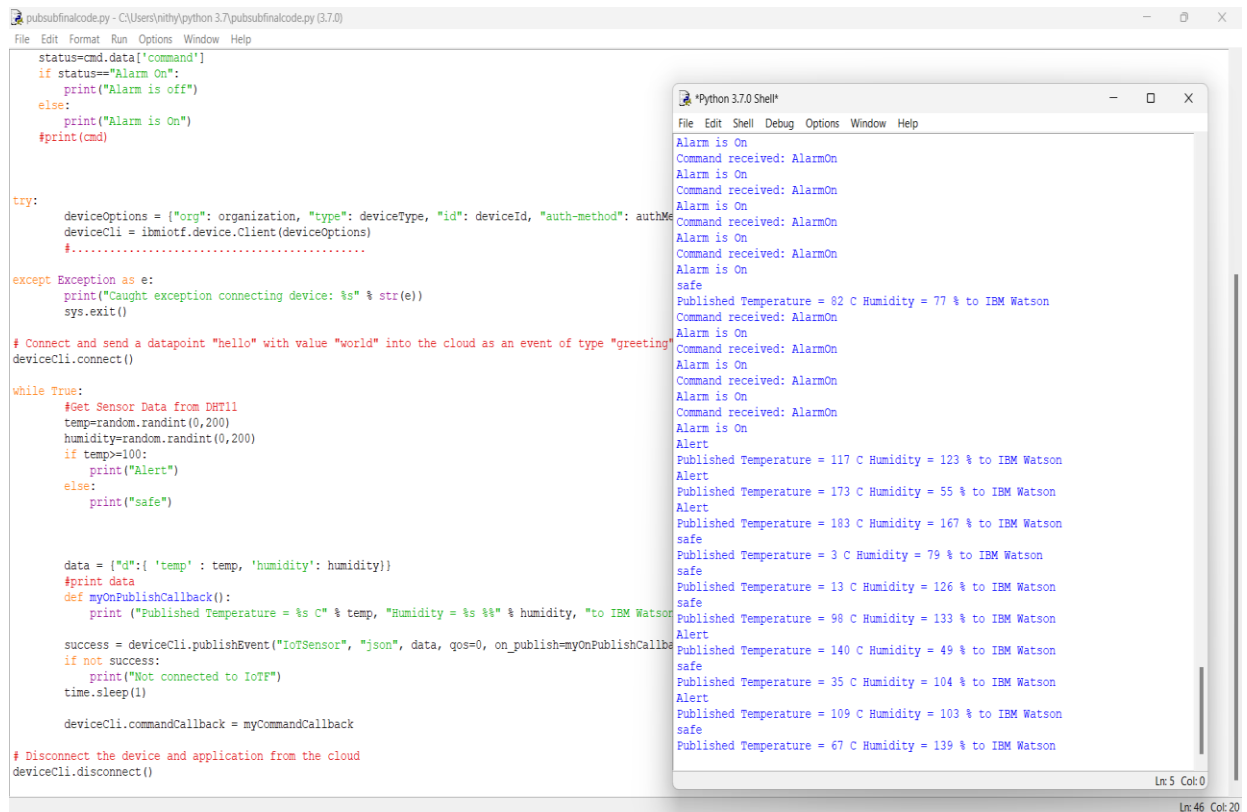


STEP-4: Attaching web link with the connected blocks in the MIT application inventor



STEP-5: Detecting high temperature and displaying "ALERT" message in the MIT application.





The image shows a Python script in a text editor and its execution output in a terminal window. The script, named `pubsubfinalcode.py`, is located at `C:\Users\nithy\python 3.7\pubsubfinalcode.py (3.7.0)`. It uses the `ibmiotf` library to connect to a cloud service and publish sensor data.

```
status=cmd.data['command']
if status=="Alarm On":
    print("Alarm is off")
else:
    print("Alarm is On")
#print(cmd)

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

while True:
    #Get Sensor Data from DHT11
    temp=random.randint(0,200)
    humidity=random.randint(0,200)
    if temp>=100:
        print("Alert")
    else:
        print("safe")

    data = {"d":{"temp": temp, 'humidity': humidity}}
    #print data
    def myOnPublishCallback():
        print ("Published Temperature = %s C" % temp, "Humidity = %s %" % humidity, "to IBM Watson")

    success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0, on_publish=myOnPublishCallba
    if not success:
        print("Not connected to IoT")
        time.sleep(1)

    deviceCli.commandCallback = myCommandCallback

# Disconnect the device and application from the cloud
deviceCli.disconnect()
```

The terminal window, titled `Python 3.7.0 Shell`, shows the output of the script. It displays a sequence of "Alarm is On" and "Command received: AlarmOn" messages, followed by "safe" and "Alert" status updates. The script then publishes temperature and humidity data to IBM Watson, with messages like "Published Temperature = 82 C Humidity = 77 % to IBM Watson". The output continues with various temperature and humidity readings, alternating between "Alert" and "safe" status.

STEP 6: Downloading apk file and building mobile application using python script for sensing temperature for hazardous area monitoring conditions in industrial areas.

4:57



 VoD 4G LTE1 37  VoD 4G LTE2 

Hazardous Area monitoring



Temperature

23

Humidity

45

Alarm On

Alarm Off

