

## BUILD A WEB APPLICATION USING NODE-RED SERVICE

### Build a web application using Node - Red

|              |  |
|--------------|--|
| Team ID      | PNT2022TMID21246                                   |
| Project Name | Smart Farmer – IoT based smart farming application |

**Team Leader:** LAVANYA R

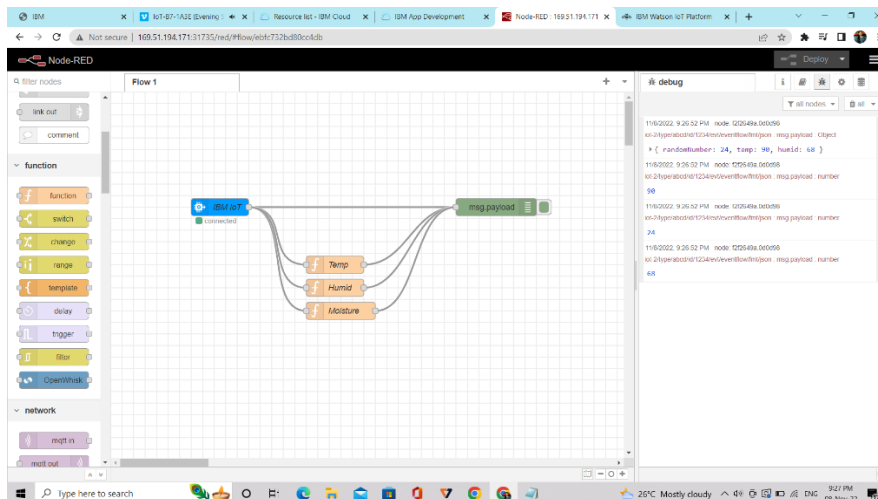
**Team Member:** ABIRAMIPRIYA J

**Team Member:** KALAISELVI S

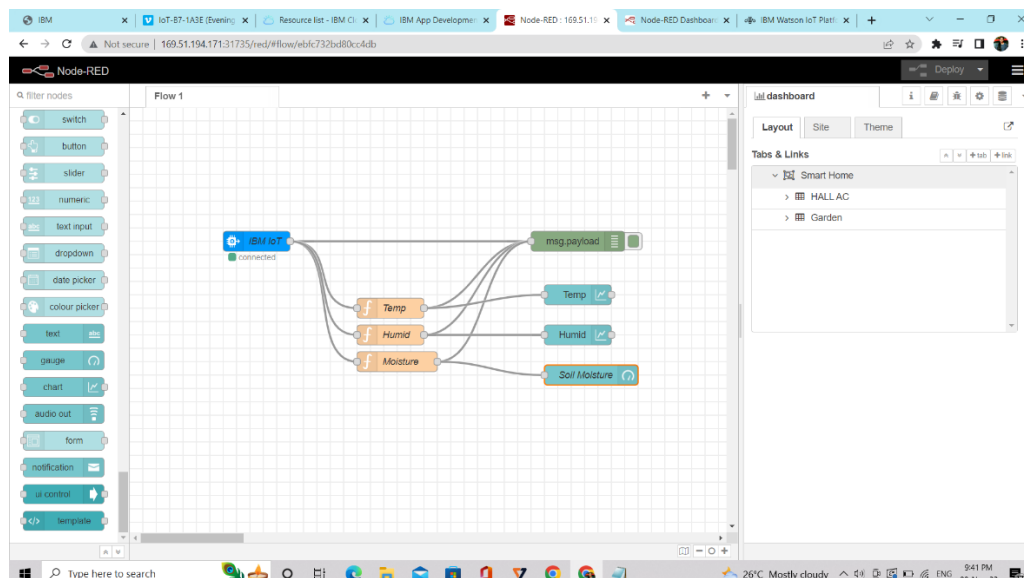
**Team Member:** KAVIYA D

### CREATION OF SMART HOME WEB APPLICATION USING NODE RED:

**INSERTING THREE FUNCTION NODES IN NODE RED AND CONNECTING WITH THE DEVICE IN THE IBM WATSON:**

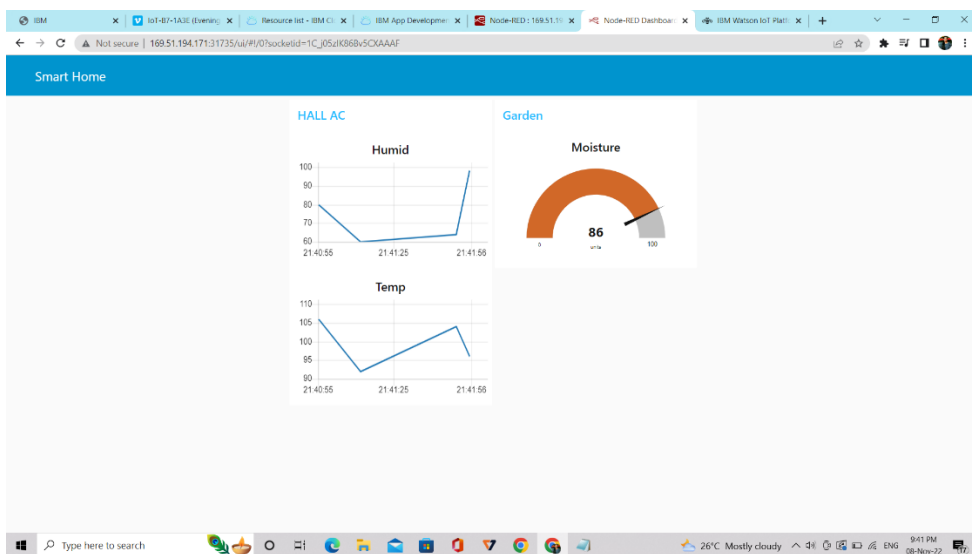
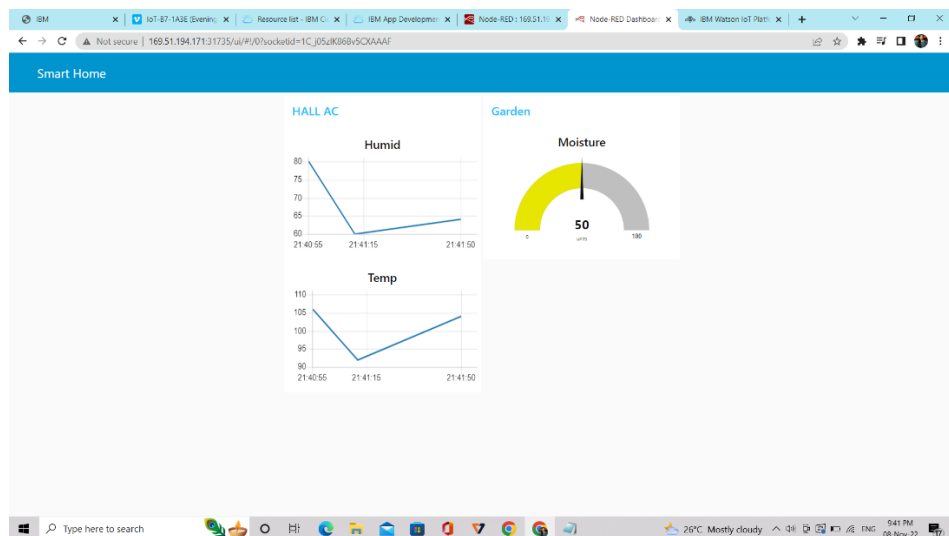
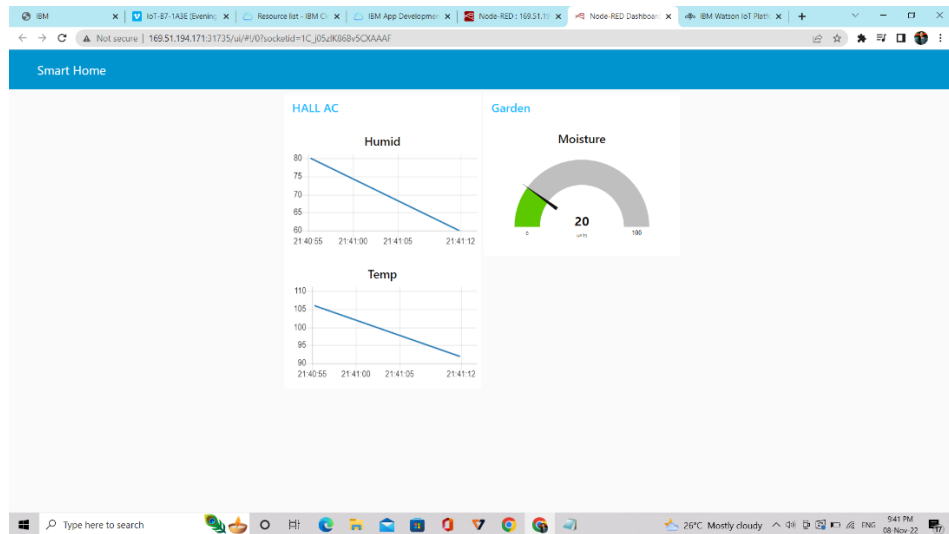


**CONNECT NODES FOR VISUALIZING THE RESULTS IN THE DASHBOARD:**

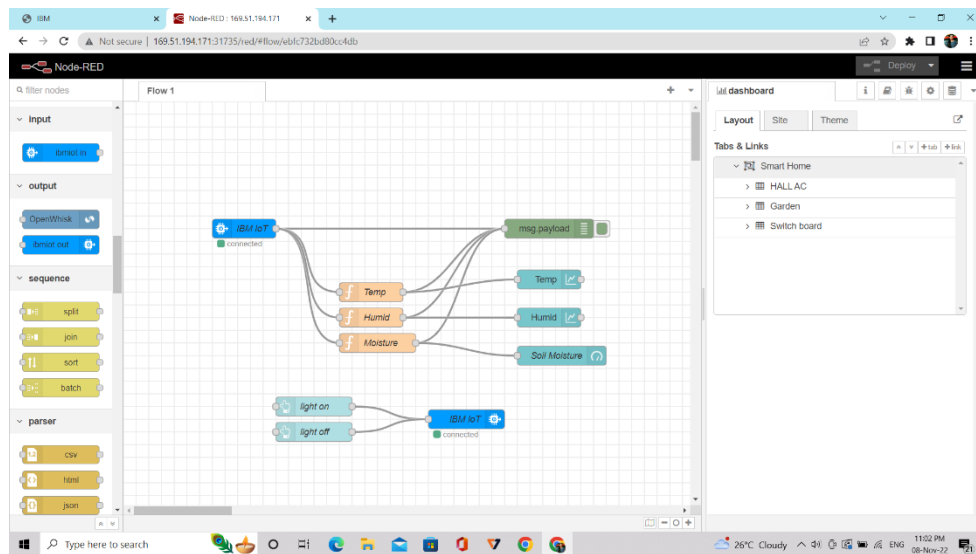


**CLICK ON DEPLOY IN NODE RED AND OPEN NODE RED DASHBOARD:**

**THE TEMPERATURE, HUMIDITY AND MOISTURE KEEPS ON CHANGING**



## INSERTING TWO SWITCHES OFF AND ON:



## PUBLISH AND SUBSCRIBE PYTHON CODE FOR TEMPERATURE, HUMIDITY AND MOISTURE AND ITS OUTPUT IN THE PYTHON IDLE:

```

def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data["command"])
    status=cmd.data["command"]
    if status=="lighton":
        print("led is on")
    elif status=="lightoff":
        print("led is off")
    else:
        print("please send proper command")

deviceOptions={"org":organization, "type":deviceType, "id":deviceId, "auth-id":
deviceClientId, "device_client":(deviceOptions)}

except Exception as e:
    print("Caught exception connecting device: %s" % str(e))
    sys.exit()

deviceCli.connect()

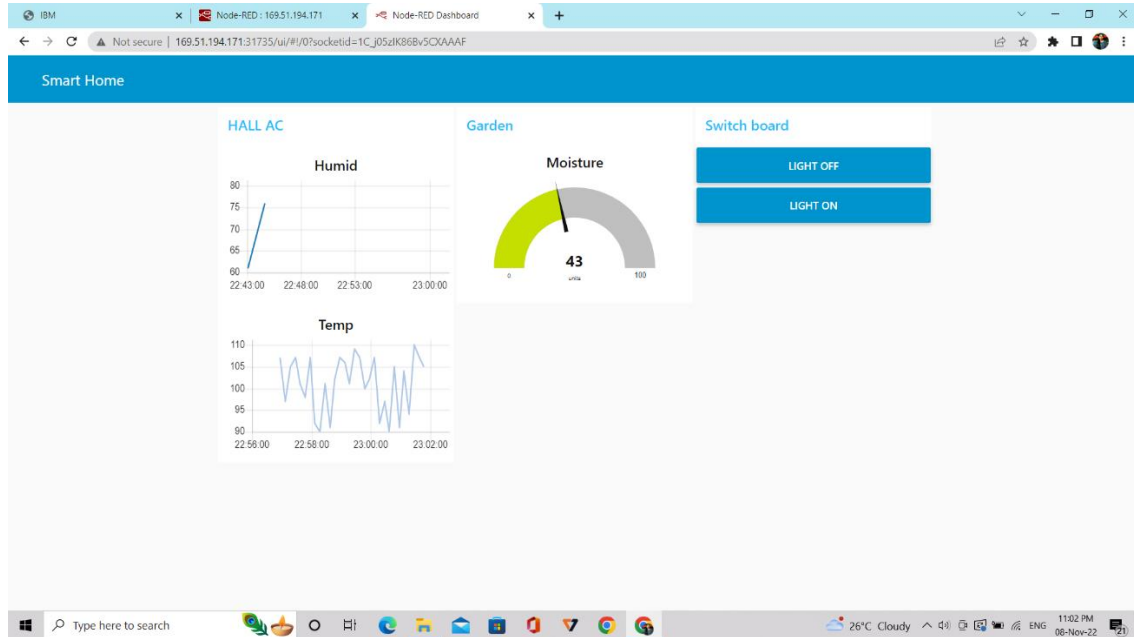
while True:
    temp=random.randint(90,110)
    humid=random.randint(60,105)
    data={"temp":temp, "Humid":humid}
    #print data
    def myCommandCallback():
        print("Published Temperature=%s C %s %s" % (temp, "Humidity=%s" % humid, "1"))
    success=deviceCli.publishEvent("IoTDevice", "json", data, qos=0, on_publish=myCommandCallback)
    if not success:
        print("Not connected to IoT")
    time.sleep(10)

deviceCli.commandCallback=myCommandCallback
deviceCli.disconnect()
    
```

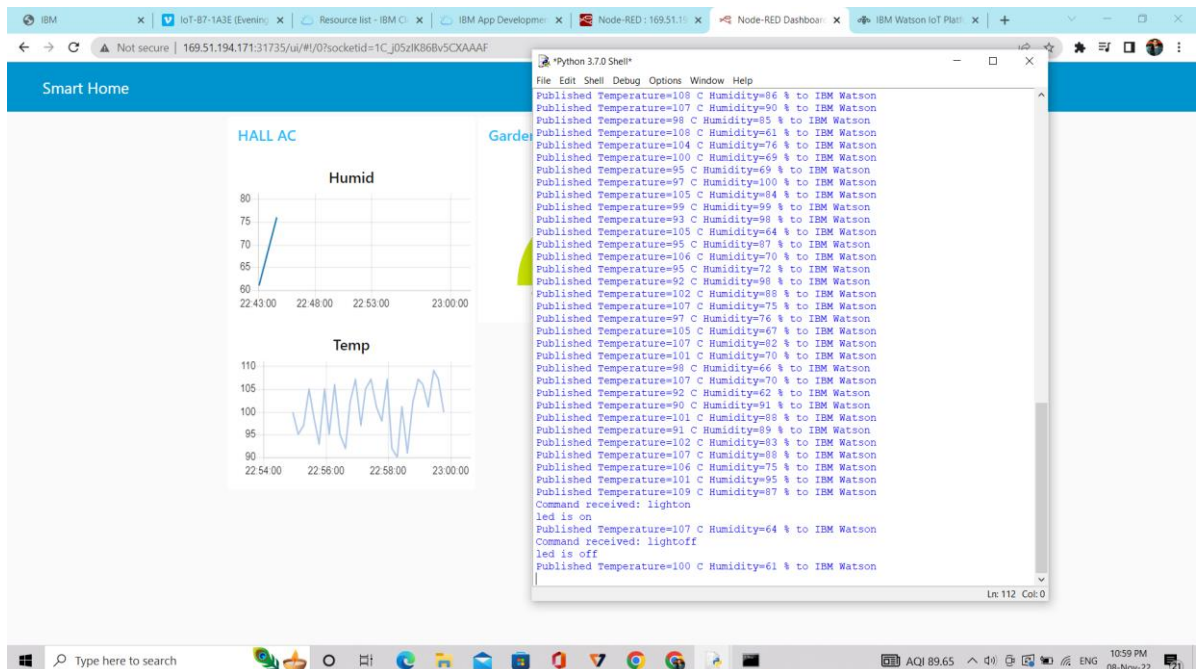
## OUTPUT IN THE IDLE AND DEVICE EVENTS:

| Event      | Value                   |
|------------|-------------------------|
| IoT Sensor | ["temp":92,"Humid":92]  |
| IoT Sensor | ["temp":93,"Humid":81]  |
| IoT Sensor | ["temp":94,"Humid":62]  |
| IoT Sensor | ["temp":92,"Humid":100] |
| IoT Sensor | ["temp":105,"Humid":92] |

## NODE RED DASHBOARD:



## OUTPUT AFTER THE LED IS MADE ON AND OFF:



## OUTPUT OF THE CODE IN IBM WATSON DEVICE EVENTS:

The screenshot displays the IBM Watson IoT Platform interface. The top navigation bar includes tabs for 'Browse', 'Action', 'Device Types', and 'Interfaces'. A search bar labeled 'Search by Device ID' is present. The main content area shows a table of devices, with one device (ID: 1234) selected. Below the device list, a detailed view for device 1234 is shown, including tabs for 'Identity', 'Device Information', 'Recent Events', 'State', and 'Logs'. The 'Recent Events' tab is active, displaying a list of events with columns for Event, Value, Format, and Last Received.

| Event     | Value                   | Format | Last Received     |
|-----------|-------------------------|--------|-------------------|
| IoTSensor | {"temp":91,"Humid":71}  | json   | a few seconds ago |
| IoTSensor | {"temp":93,"Humid":81}  | json   | a few seconds ago |
| IoTSensor | {"temp":94,"Humid":62}  | json   | a few seconds ago |
| IoTSensor | {"temp":92,"Humid":100} | json   | a few seconds ago |
| IoTSensor | {"temp":105,"Humid":92} | json   | a few seconds ago |