PROJECT REPORT

IOT BASED SMART CROP PROTECTION SYSTEM FOR AGRICULTURE

TEAM ID: PNT2022TMID21422

TEAM LEAD: SHENBAGA THENDRAL B (917719D090)

TEAM MEMBERS:

HARINI AANANTHI K S

(917719D026)

KEERTHIGA R M

(917719D040)

SNEHA S R

(917719D094)

TABLE OF CONTENTS

1. INTRODUCTION:

1.1 OVERVIEW

This is a smart crop protection project based on Internet Of Things (IoT), that can detect the animals, birds and generate alarm thereby avoiding the destruction of crops by them. Soil moisture, humidity and temperature conditions of agricultural land are measured using Watson IoT services.

IoT is network that connects physical objects or things embedded with electronics, software and sensors through network connectivity that collects and transfers data using cloud for communication. Data is transferred through internet without human to human or human to computer interaction.

No hardware is used in this project. Everything is done virtually using sensors from IBM IoT Simulator, Watson software , Clarifai service and Node RED. The motors and sprinklers in the field can be controlled using the mobile application

- **Project requirements**: Node-RED, IBM Cloud, Clarifai API service IBM Watson IoT, Node.js, IBM Device, IBM IoT Simulator, Python 3.7

- **Project Deliverables**: Application for IoT based Smart Crop Protection System

1.2 PURPOSE :

Crop damage caused by animal attacks is one of the major threats in reducing the crop yield. Due to the expansion of cultivated land into previous wildlife habitat, crop raiding is becoming one of the most conflicts antagonizing human wildlife relationships.

IoT based farming improves the entire agriculture system by monitoring the field in real-time. IoT in agriculture not only saves the time but also reduces the manual involvement in shooing away the animals and birds destructing the crops.

This device also helps the farmer to know about the soil moisture and sprinkle water using motors controller by the mobile application.

1.2.1 SCOPE OF WORK

- Create and configure IBM Cloud Services
    - Create IBM Watson IoT Platform
    - Create a device & configure the IBM IoT Platform
    - Create Node-RED service
    - Create a database in Cloudant DB to store location data
    - Create a cloud object storage service and create a bucket to store the images
- Develop a python script to publish the sensor parameters like Temperature, Humidity, and Soil Moisture to the IBM IoT platform and detect the animals and birds in video streaming using Clarifai.
- Develop a web Application using Node-RED Service.
    - Display the image in the Node-RED web UI and also display the temperature, humidity, and soil moisture levels. Integrate the buttons in the UI to control the Motors.

2. LITERATURE SURVEY:

2.1 EXISTING PROBLEM

- One of the biggest problems farmers face in India is the attack on crops by wild animals in their fields.
- The damage from these attacks significantly and adversely affects the crop yield.
- Attack by wild animals leads to sleepless nights.

2.2 EXISTING SOLUTION

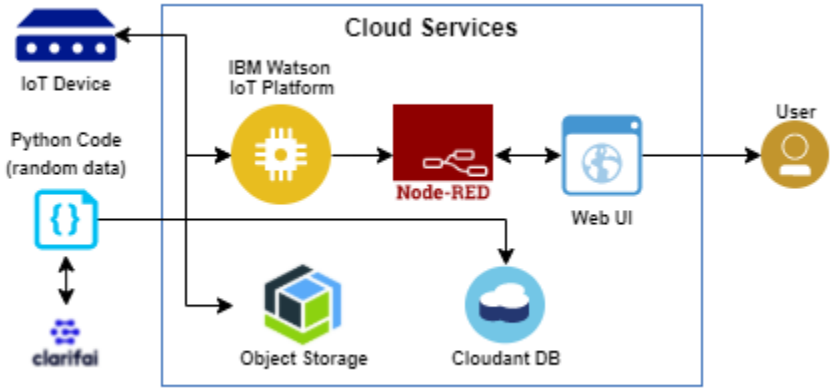
- The current methods used to counter this problem include the use of electrified welded mesh fences (usually 30cm in the ground), chemicals or organic substances and gas cannons.
- Other traditional methods applied by farmers include the use of Hellikites, Balloons, Shot/Gas guns, String & stone, etc. These solutions are often cruel and ineffective. They also require a vast amount of installation and maintenance cost and some of the methods have environmental pollution effect on both humans and animals.
- On the other hand, the chemical products used to prevent these animal attacks have an application cost per hectare and their effectiveness is dependent on weather condition, as rain may cause a dilution effect.

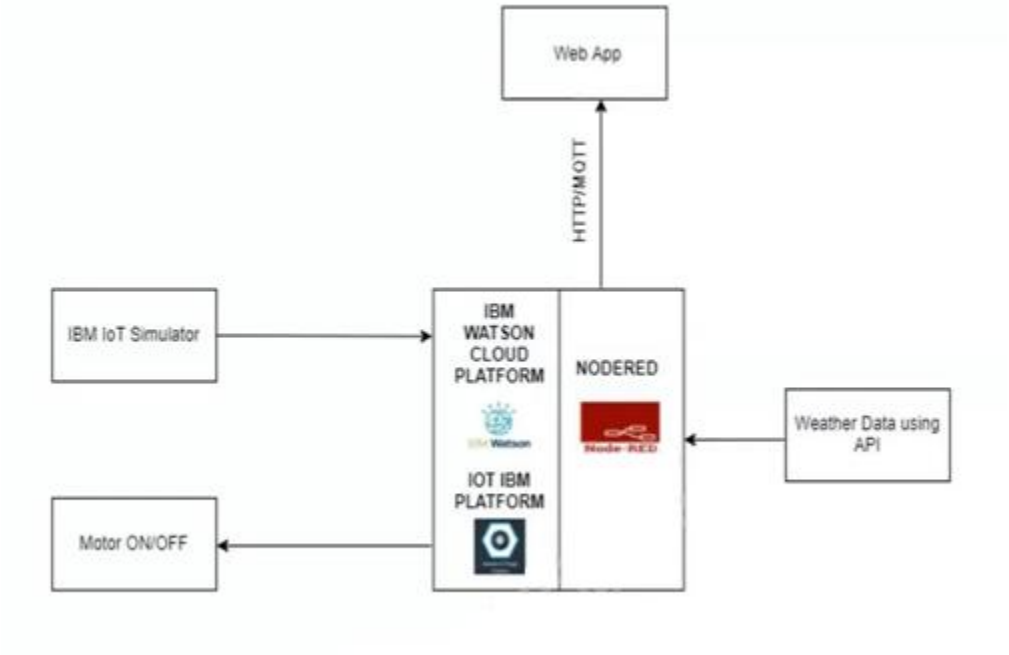
2.3 PROPOSED SOLUTION

- The presence of any animal/bird inside the crop field is detected using the image processing service known as Clarifai.
- These images are captured and loaded to the IBM cloud as objects.
- Based on the continuous keen checks performed and comparison of the captured object with the reference, buzzers /alarms are set to alert the farmers as well as drive away the animals/birds approaching the crop.
- Soil Moisture can be checked by using the sensors that can sense the soil condition and send the moisture content in the soil over the cloud servicesto the web application.
- The supply of water can be controlled from anywhere by controlling the motor state (ON/OFF), using web application.
- Surrounding temperature can also be sensed by the sensors anddisplayed on the application.
- Real time weather conditions can also be known by using different weather APIs from different websites and get displayed on our application and water the crops based on the need.

3. THEORITICAL ANALYSIS:

3.1 BLOCK DIAGRAM





3.2 Hardware / Software Designing

1. Create a device in IBM Cloud.

2. Connect the device to IBM Simulator to get the generated sensor informations.

3. Build Node-RED flow to build a web application to display the weather and sensor informations.

4. The devices can now be controlled using WEB UI.

5. To get the real time weather condition data from open weather we generate pseudo random data for simulation purpose

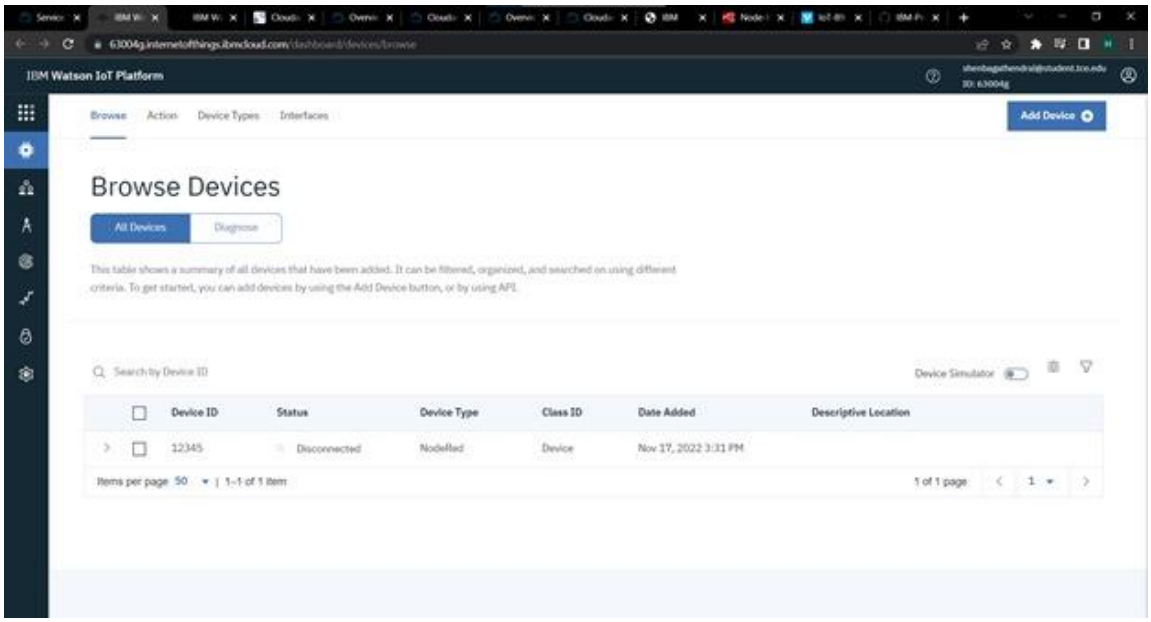
6. It is mapped and integrated in the Node-RED flow.

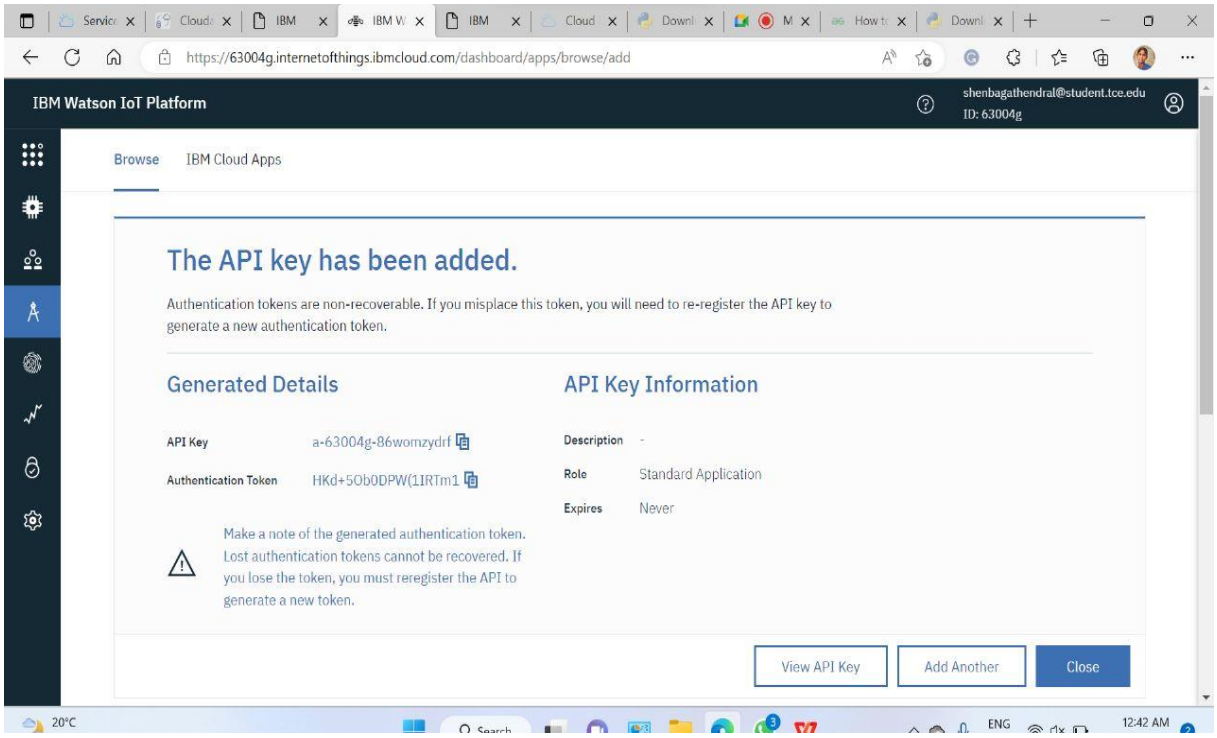
7. Working of the web application to the devices is controlled by python coding.

8. The whole system works integrated

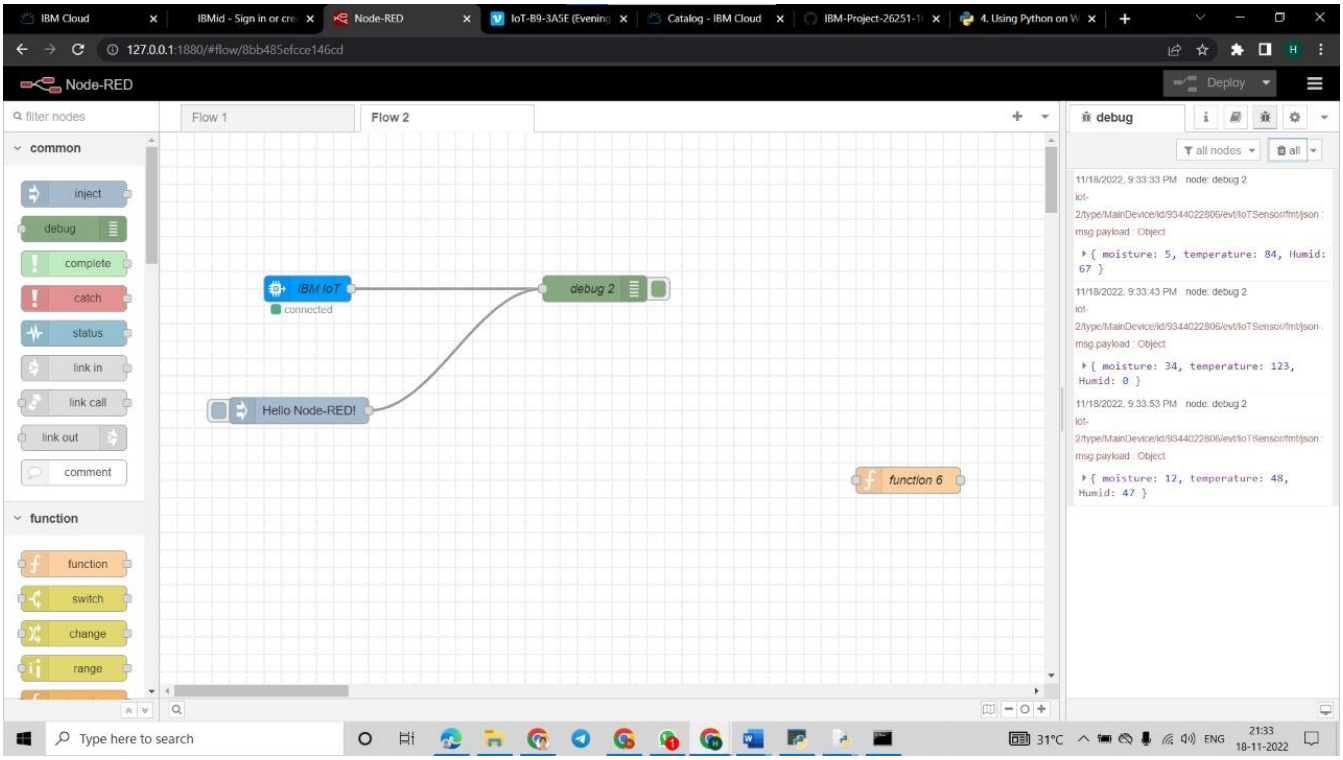
4 EXPERIMENTAL INVESTIGATION:

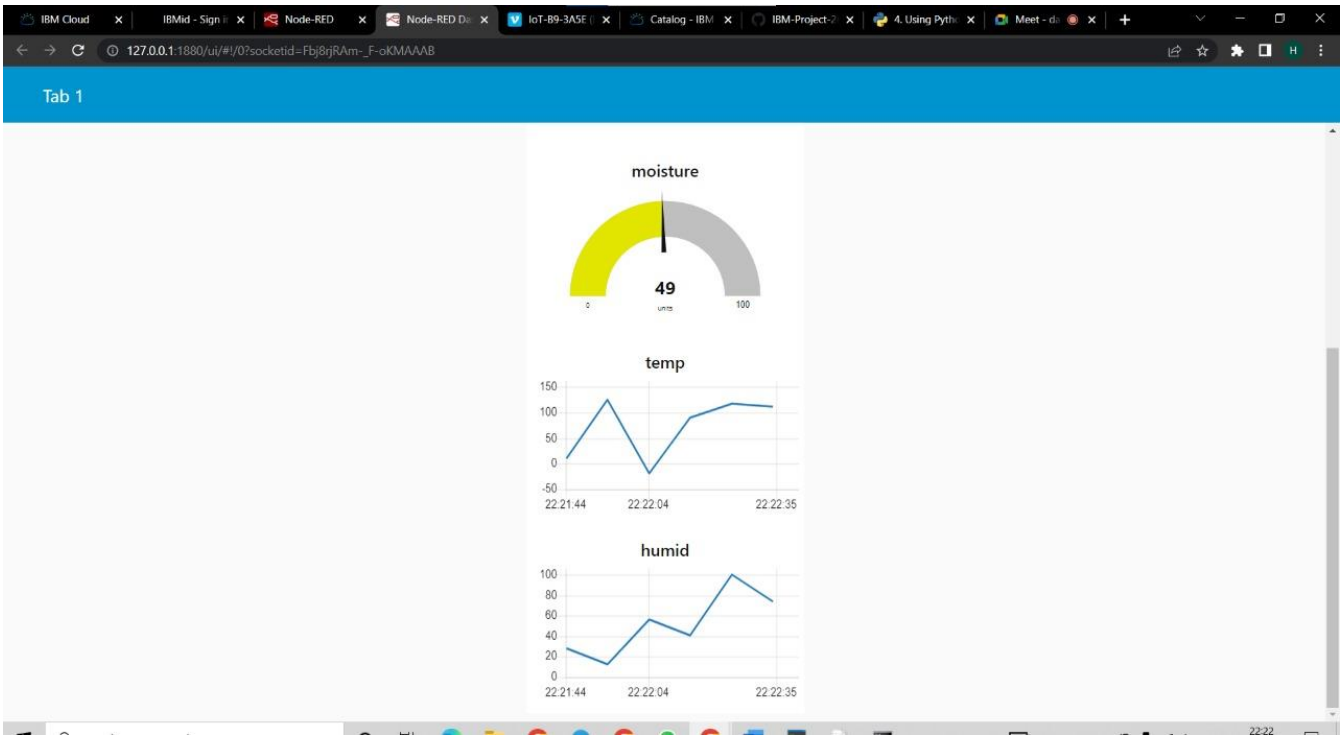
WATSON IOT SIMULATOR:

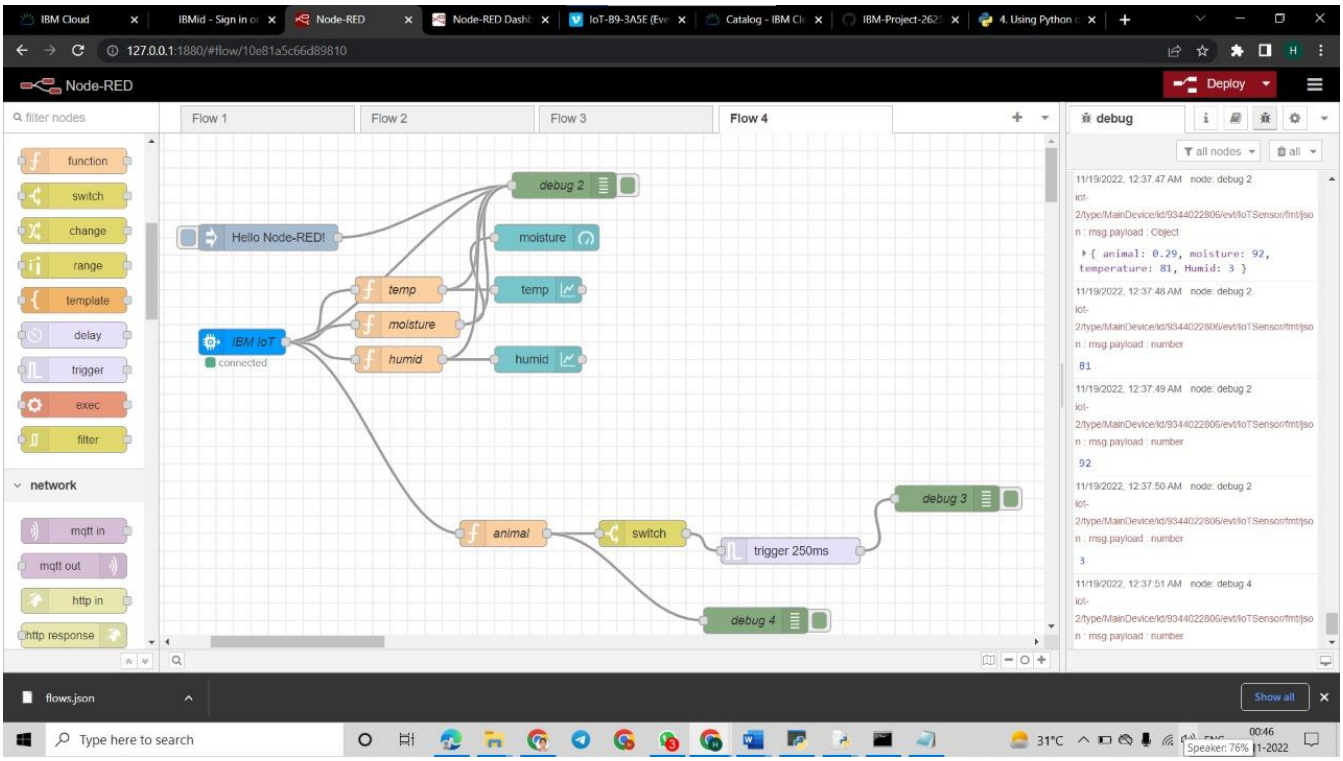




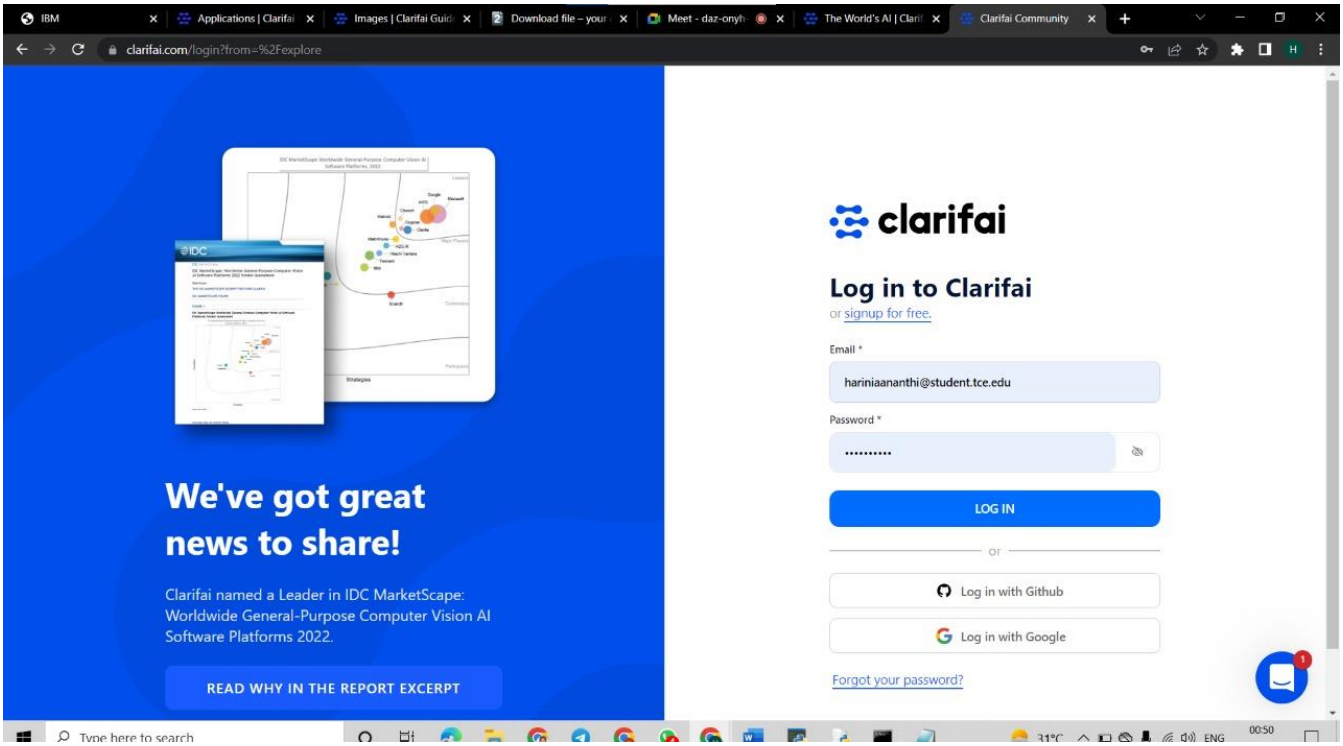
NODE RED / WEB APPLICATION:







CLARIFAI SERVICES:

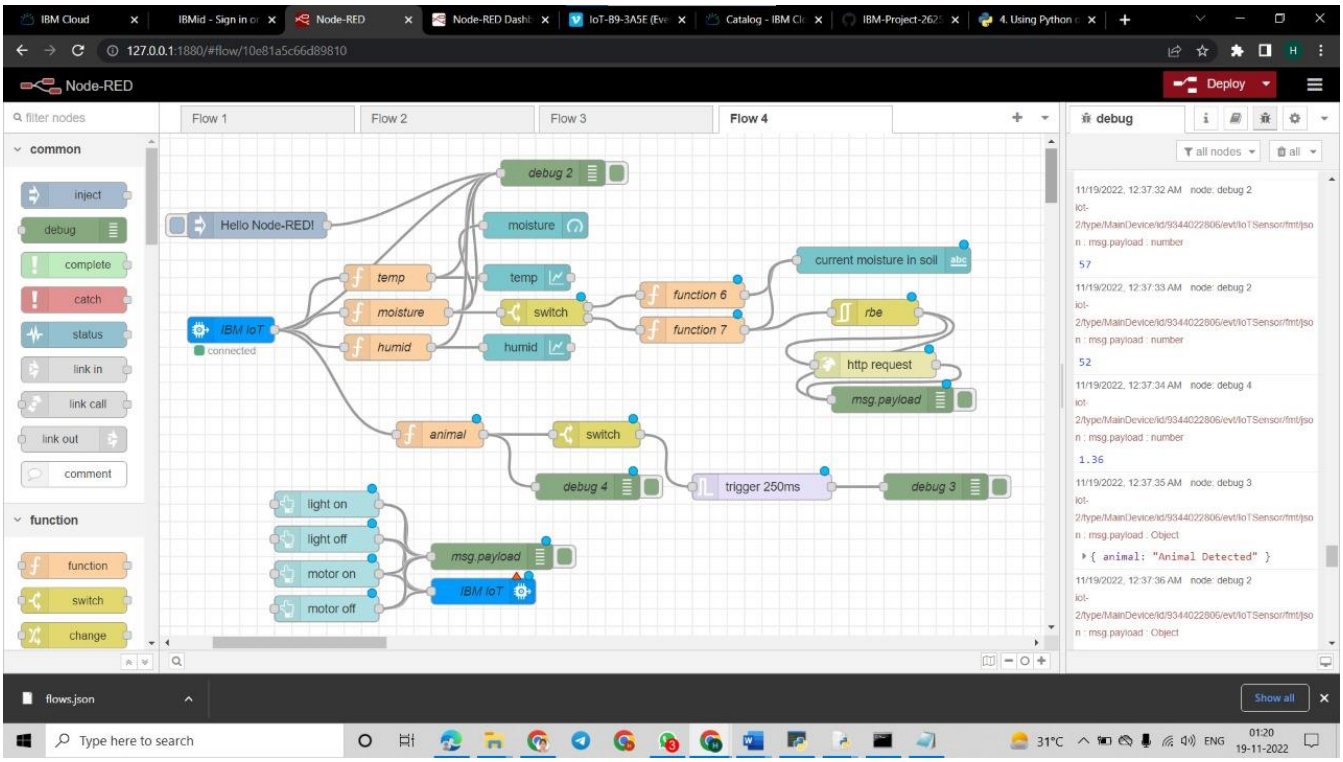


In this project we send the weather data and images of animals from Clarifai through IoT Simulator shown instead of real soil and temperature conditions. Simulator passes the data through IBM Cloud to the web application. The data is displayed on the Dash board. Web Application is built using Node-RED. We have created 2 tabs:

1. IoT Smart Agriculture.
2. Graphical Representation.
3. Node RED Application.
4. Clarifai Services

Web Application is also used to control the devices further like motor, pumps, lights, or any other devices in the agricultural field. In this project the output is passed using python code and the control action is displayed in python code console window as shown in figure.

5 FLOWCHART:



Following are the nodes used in the project in the Web Application:

1. IBM IoT: IN and OUT Nodes.
2. Function Nodes.
3. Gauge Nodes.
4. Chart Nodes.
5. Debug Node.
6. Button Nodes.

Following are the nodes used for the weather condition:

1. Timestamp Node.
2. http request Node
3. Function Nodes.
4. Text Nodes.
5. Debug Node

6 RESULT:

We have successfully built a web based UI and integrated all the services using Node-RED.

7 ADVANTAGES & DISADVANTAGES:

7.1 ADVANTAGES:

- Many difficult challenges can be avoided by making the process automatedand the quality and yield of crops can be maintained.
- All the data like climatic conditions and changes in them, soil or crop conditions and movement of animals/ birds can be easily monitored.
- The process in farming can be controlled using the web applications from anywhere, anytime.
- Risk of crop damage can be lowered to a greater extent.

7.2 DISADVANTAGES:

- Smart Crop Protection requires internet connectivity continuously, but rural parts of the country cannot fulfill this requirement.
- IoT devices are costlier to implement.
- The process in farming can be controlled using the web applications from anywhere, anytime.

8 APPLICATIONS:

- Farming processes can be made more controlled and accurate.
- Live monitoring of all the processes and the conditions on the agricultural field can be monitored.
- Entire flow of the process is under the web application user's control.
- Quality can be maintained.

9 CONCLUSION:

A IoT Web Application is built for smart crop protection system using Clarifai service, Watson IoT platform, Watson simulator, IBM cloud and Node-RED.

10 FUTURE SCOPE:

In future due to more demand of good and more farming in less time, for betterment of the crops ,maintenance of the quantity and quality of crops IoT can be implemented in most of the places.

11 BIBLIOGRAPHY:

- IBM Cloud:

  https://cloud.ibm.com/docs/overview?topic=overview-whatis-platform

- Watson IoT:

  https://www.iotone.com/software/ibm-watson-iot-platform/s62

- Node-RED:

  https://nodered.org/docs/getting-started/windows#3-run-node-red

  -https://www.youtube.com/watch?v=cicTw4SEdxk

- Open weather map:

  https://openweathermap.org/

- Git Hub:

  https://github.com/rachuriharish23/ibmsubscribe

A. SOURCE CODE

MOTOR.PY

```
import time
import sys
import ibmiotf.application  # to install pip install ibmiotf
import ibmiotf.device

# Provide your IBM Watson Device Credentials
organization = "63004g"  # replace the ORG ID
deviceType = "MainDevice" # replace the Device type
deviceId = "9344022806" # replace Device ID
authMethod = "token"
authToken = "a-63004g-86womzydrf"  # Replace the authtoken


def myCommandCallback(cmd):  # function for Callback

    if cmd.data['command'] == 'motoron':
        print("MOTOR ON IS RECEIVED")

    elif cmd.data['command'] == 'motoroff':
        print("MOTOR OFF IS RECEIVED")

    if cmd.command == "setInterval":

        if 'interval' not in cmd.data:
            print("Error - command is missing required information: 'interval'")
        else:
            interval = cmd.data['interval']
    elif cmd.command == "print":
        if 'message' not in cmd.data:
            print("Error - command is missing required information: 'message'")
```

```
        else:
            output = cmd.data['message']
            print(output)


try:
    deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-
method": authMethod,
                     "auth-token": authToken}
    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" 10 times
deviceCli.connect()

while True:
    deviceCli.commandCallback = myCommandCallback

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

SENSOR.PY

```
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random
```

```
#Provide your IBM Watson Device Credentials
organization = "63004g"
deviceType = "MainDevice"
deviceId = "9344022806"
authMethod = "token"
authToken = "9944611970"

# Initialize GPIO
def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data['command'])
    status=cmd.data['command']
    if status == "motoron":
        print ("motor is on")
    elif status == "motoroff":
        print ("motor is off")
    else :
        print ("please send proper command")

try:
        deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-
method": authMethod, "auth-token": authToken}
        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" 10 times
deviceCli.connect()

while True:
        #Get Sensor Data from DHT11
        animal=random.uniform(0.1, 0.99)
        moisture=random.randint(0,110)
        temperature=random.randint(-20,125)
        Humid=random.randint(0,100)

        data = {'animal':animal,'moisture': moisture, 'temperature' : temperature, 'Humid': Humid }
        #print data
        def myOnPublishCallback():
            print ("Published Soil Moisture = %s %%" %moisture,"Temperature = %s C" % temperature, "Humidity = %s %%" % Humid,'animal = %s'%animal, "to IBM Watson")
            if animal>0.98:
                print("Alert")
        success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0, on_publish=myOnPublishCallback)
        if not success:
            print("Not connected to IoTF")
        time.sleep(10)

        deviceCli.commandCallback = myCommandCallback
```

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