Sprint 4

IoT based Smart crop protection system for Agriculture

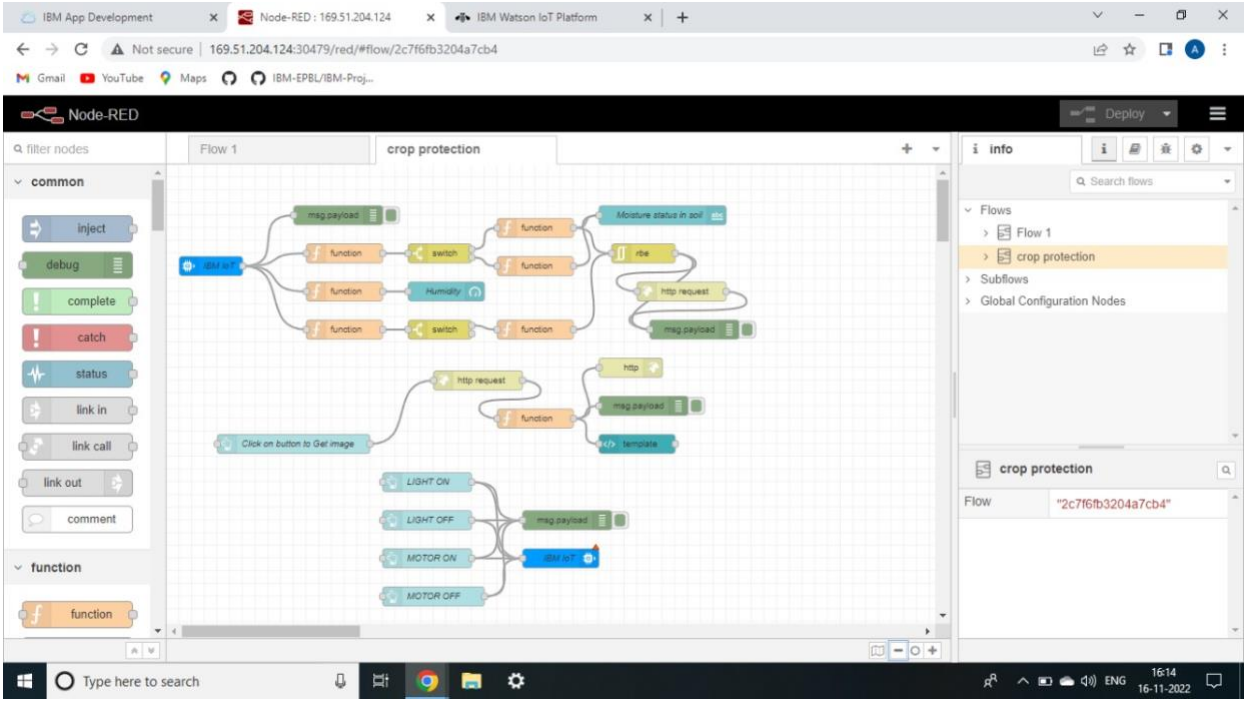
Team ID:PNT2022TMID45173

Sprint-4

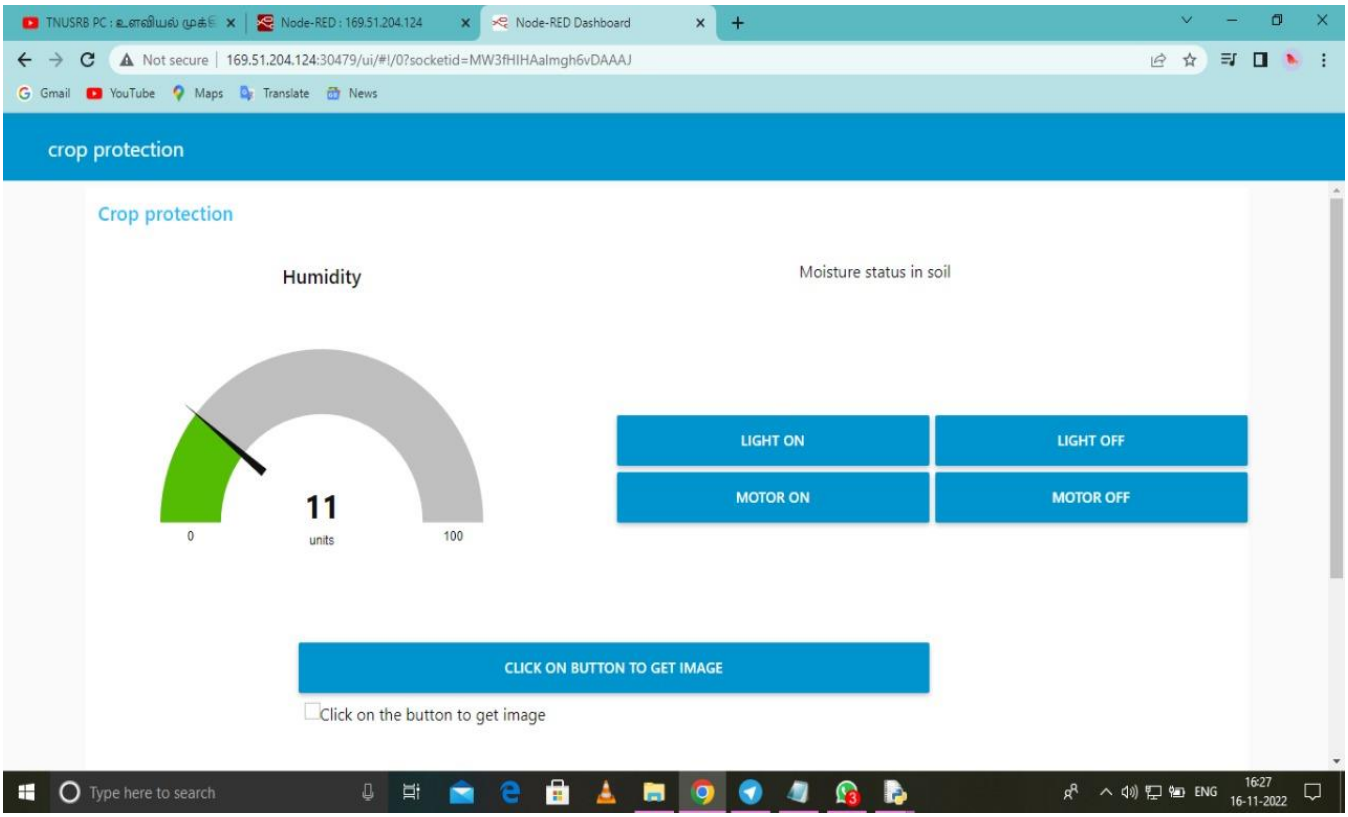
Web UI (to make the user interact with the software) / Run a simulation using the wokwi online platform

PROCESS

Using Node-Red for the Web UI process



Output in Node-RED Dashboard:



Program :

python ibm.1.py - C:\Users\ELCOT\AppData\Local\Programs\Python\Python37\python ibm.1.py (3.7.0)

File Edit Format Run Options Window Help

```
import cv2
import numpy as np
import wiotp.sdk.device
import playsound
import random
import time
import datetime
import ibm_boto3
from ibm_botocore.client import config, ClientError
#CloudantDB
from cloudant.client import Cloudant
from cloudant.error import CloudantException
from cloudant.result import Result, ResultByKey
from clarifai_grpc.channel.clarifai_channel import clarifaichannel
from clarifai_grpc.grpc.api import service_pb2_grpc
stub = service_pb2_grpc.V2Stub (Clarifaichannel.get_grpc_channel())
from clarifai_grpc.grpc.api import service_pb2, resources_pb2
from clarifai_grpc.grpc.api.status import status_code_pb2
#This is how you authenticate.
metadata=(('authorization', 'key bc885e5165d74ef48f42f6f6a2c9eb87'),)
COS_ENDPOINT = "https://s3.jp-tak.cloud-object-storage.appdomain.cloud" # Current list avaiable at https://control.cloud-object-storage.cloud.ibm.com/v2/endpoints
COS_API_KEY_ID = "zep-ctim079907XFAE7170cmFLLUCOA25" # eg "MOOYSELW4JTT MB-B-2y8TLYBIQQRanc--P3byk"
COS_AUTH_ENDPOINT="https://iam.cloud.ibm.com/identity/token"
COS_RESOURCE_CRN-"crn:v1:bluemix:public:cloud-object-storage:global:a/6b644a3fda97448c23eeef263ed6:199ab1e5-0d9d-420f-8e4a-98d86804368::"# eg crn:v1:bluemix:public:clo
clientdb = Cloudant ("apikey-v2-16u3crmdpkghhxefdikvpssoh5fwezrmuup5fv5g3ubz", "b0ab115f5d3e6255eabb978e7e2f0e1",url="https://apikey-v2-16u3crmdpkghhxefdikvpssoh5fwezr

#Create resource
COS = ibm_boto3.resource("s3",
    ibm_api_key_id=COS_API_KEY_ID,
    ibm _service_instance_id=COS_RESOURCE_CRN,
    ibm_auth_endpoint= COS_AUTH_ENDPOINT,

   config-Config(signature_version="auth"),
   endpoint_url-cOS_ENDPOINT
)
def multi_part_upload (bucket_name, item_name, file_path):
    try:
        print("Starting file transfer for (0) to bucket: (1)\n".format(item_name, bucket_name))

        #set 5 MB chunks
        part size 1024 1024 5
```

Ln: 6 Col: 11

Type here to search ENG 13:50 18-11-2022

python ibm.1.py - C:\Users\ELCOT\AppData\Local\Programs\Python\Python37\python ibm.1.py (3.7.0)

File Edit Format Run Options Window Help

```
        #set 5 MB chunks
        part size 1024 1024 5
       #set threadhold to 15 MB
       file threshold 1024 1024 *15
       #set the transfer threshold and chunk size
      transfer_config= ibm boto3.s3.transfer.Transferconfig(
        multipart threshold-file_threshold,
        multipart chunksize-part_size
     )
     #the upload fileob) method will automatically execute a multi-part upload
     #in 5 chunks for all files over 15 MB
     with open(file_path, "b") as file data:
        cos.Object (bucket_name, item_name).upload_fileobj(1
        fileobj=file_data
        confug=transfer_config
       )
    print("Transfer for {0} Complete!\n".format(item_name))
     except clientError as be:
       print ("CLIENT ERROR: {0}\n".format(be))
       except Exception as e:
     print("Unable to complete multi-part upload: {0}".format(e))
def nyCommandCallback(cmd):
      print ("Command received:%s"% cmd.data)
      command-cmd.data['command']
       print (command)
      if(command lighton'):
        print('lighton')
       elif (command lightoff'):
      print ('lightoff')
      elif (command=='motoroff'):
      print ('motoroff')
myConfig={
        "identity":{
       "orgId": "rvp7mx",
       "typeid": "NodeMCU",
       "deviceId": "12345"
}
    "auth": {
    "token":"12345678"
    }
```

Ln: 74 Col: 22

Type here to search

ENG 14:04 18-11-2022

ibm publish.py - C:\Users\VS\OneDrive\Desktop\ibm publish.py (3.7.0)

File Edit Format Run Options Window Help

```
authMethod = "token"
authToken = "12345678"

# 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
        moist=random.randint (0,100)
        temp=random.randint(-20,125)
        hum=random.randint(0,100)

        data = { 'moist':moist, 'temp' : temp, 'hum': hum}
        #print data
        def myOnPublishCallback():
            print ("Published temp = %s C" % temp, "hum = %s %%" % hum,"moist = %s %%" % moist, "to IBM Watson")

        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()
```

Snipping Tool

Screenshot copied to clipbaord and saved

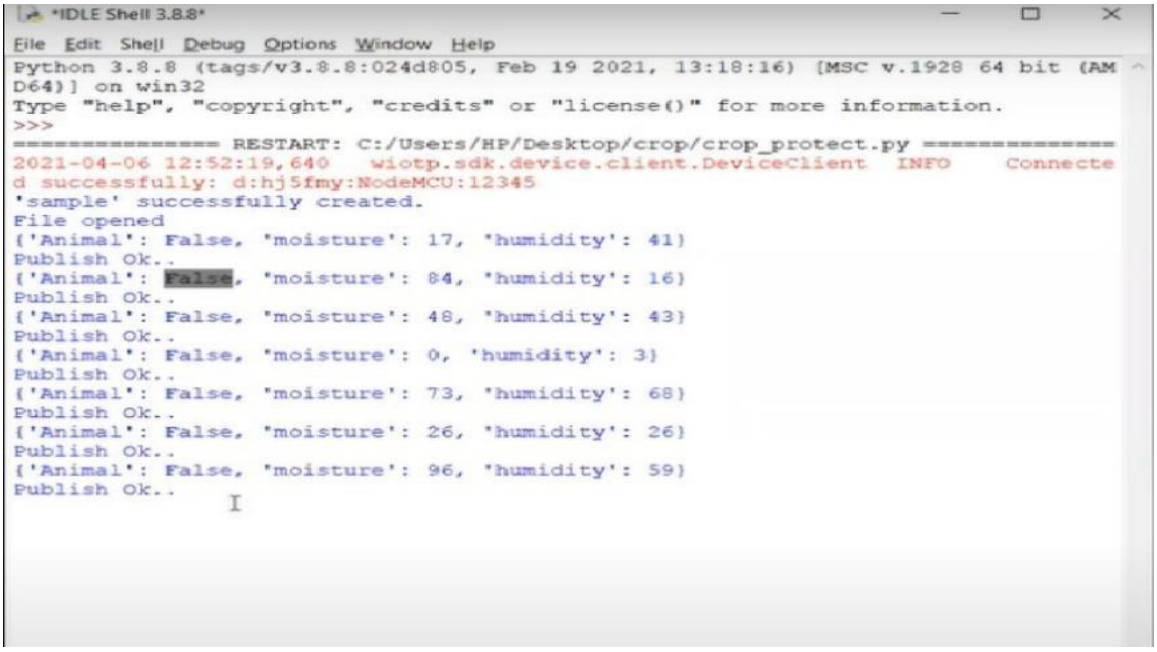
Select here to mark up and share the image

Ln: 39 Col: 0

30°C Cloudy

ENG IN 13:54 16-11-2022

Output:



WOKWI

Create your project in the online platform of the WOKWI and execute it using the IBM Credential.

CODE

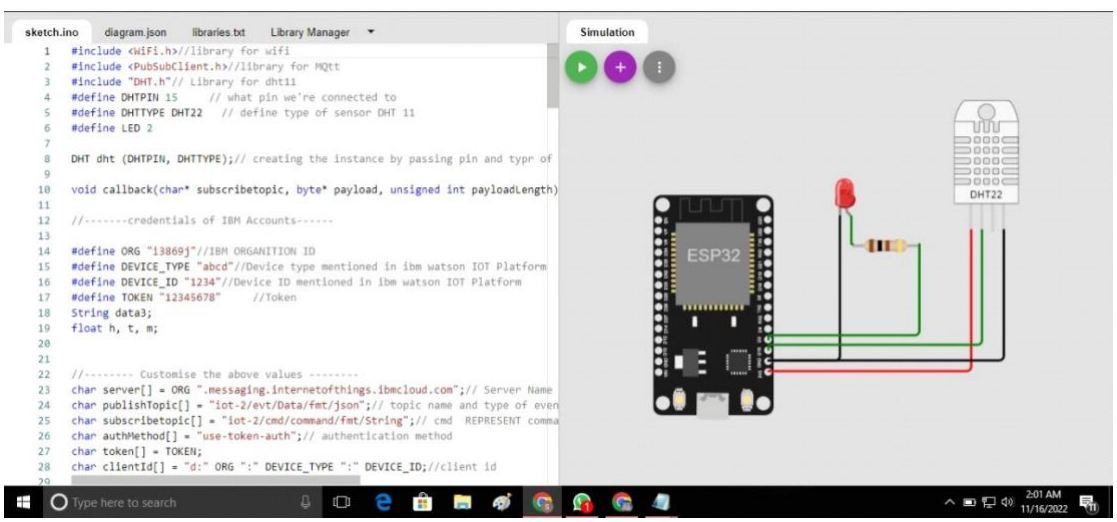
```
#include <WiFi.h>//library for wifi

#include <PubSubClient.h>//library for MQtt

#include "DHT.h"// Library for dht11

#define DHTPIN 15     // what pin we're connected to

#define DHTTYPE DHT22   // define type of sensor DHT 11

#define LED 2


DHT dht (DHTPIN, DHTTYPE);// creating the instance by passing pin and typr of dht connected


void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);


//-------credentials of IBM Accounts------
```

```
#define ORG "rvp7mx"//IBM ORGANITION ID
#define DEVICE_TYPE "abcd"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "1234"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "12345678"     //Token
String data3; float h, t, m;



//-------- Customise the above values --------
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";// Server Name char
publishTopic[] = "iot-2/evt/Data/fmt/json";// topic name and type of event perform and
format in which data to be send
char subscribetopic[] = "iot-2/cmd/command/fmt/String";// cmd  REPRESENT command type
AND COMMAND IS TEST OF FORMAT STRING
char authMethod[] = "use-token-auth";// authentication method char
token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;//client id



//-----------------------------------------
WiFiClient wifiClient; // creating the instance for wificlient
PubSubClient client(server, 1883, callback ,wifiClient); //calling the predefined client id by passing
parameter like server id,portand wificredential



void setup()// configureing the ESP32
{
  Serial.begin(115200);
dht.begin();
  pinMode(LED,OUTPUT);
  delay(10);
  Serial.println();
```

```
  randomSeed(analogRead(0));
wificonnect();   mqttconnect();
}

void loop()// Recursive Function
{

  h = dht.readHumidity();   t
= dht.readTemperature();
m = random(100);
  Serial.print("temp:");
  Serial.println(t);
  Serial.print("Humid:");
  Serial.println(h);
  Serial.print("moist:");
  Serial.println(m);

  PublishData(t, h,m);
delay(1000);   if
(!client.loop()) {
mqttconnect();
  }
{
    if(m<=100){
      Serial.print("motor is ON Automatically When LOW Moist in (moist<=100) ");
Serial.print("\n");
    }
else{
      Serial.println("Moist level is normal");
      }
```

```
  }
}
```

Wokwi output:

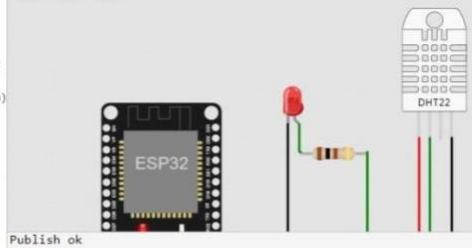


sketch.ino | diagram.json | libraries.txt | Library Manager

```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQtt
#include "DHT.h"// Library for dht11
#define DHTPIN 15     // what pin we're connected to
#define DHTTYPE DHT22   // define type of sensor DHT 11
#define LED 2

DHT dht (DHTPIN, DHTTYPE);// creating the instance by passing pin and typr of

void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)

//-------credentials of IBM Accounts------

#define ORG "13869j"//IBM ORGANITION ID
#define DEVICE_TYPE "abcd"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "1234"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "12345678"     //Token
String data3;
float h, t, m;


//-------- Customise the above values --------
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";// Server Name
char publishTopic[] = "iot-2/evt/Data/fmt/json";// topic name and type of even
char subscribetopic[] = "iot-2/cmd/command/fmt/String";// cmd  REPRESENT comma
char authMethod[] = "use-token-auth";// authentication method
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;//client id
```

Simulation

00:10.885 88%




```
Publish ok
motor is ON Automatically When LOW Moist in (moist<=100)
temp:24.00
Humid:40.00
moist:27.00
Sending payload: {"temp":24.00,"Humid":40.00,"Moist":27.00}
Publish ok
```

Type here to search

2:01 AM
11/16/2022