

# FINAL DELIVERABLES

## PROJECT REPORT

TEAM ID	PNT2022TMID23722
PROJECT TITLE	IoT BASED SMART CROP PROTECTION SYSTEM FOR AGRICULTURE

### PROJECT OBJECTIVES:

- Gain information of Watson IoT Platform.
- linking IoT devices to the Watson IoT platform and
- exchanging the sensor data.
- Gain information on Cloudant DB
- Gain information on using the Clarifai service
- Gain information of storing images in IBM Object Storage and retrieving images
- Creating a Web Application through which the user interacts with the device.

### LITERATURE SURVEY:

#### Abstract:

Most of the farmers are facing many problems nowadays due to many reasons. Our problem to solve is the invasion of various species such as birds and animals that harm the crops that are being cultivated. Various types of species such as birds and animals come to the cultivation field according to the crop that is being cultivated and also according to the season of cultivation. Some wild animals enter the field during night times when the field is near a forest region or when the farm cultivates some fruits and other crops that attract animals. Some animals cross the field in search of food and water and also the birds enter the field for food and they damage all the crops. When the animals enter the field they not only eat food but they also damage the entire field by walking upon the crops and also by spoiling the food crops. The birds, by entering the field they come to eat seeds of the crops and also they tend to drag the crops and ruin the entire field. Some birds enter the field to eat the insects and pests in the field.

Here to solve this situation we are proposing a solution using IOT(Internet of Things) where we use various types of sensors to monitor the entire field and using the help of the internet we tend to send the message to the farmer or the person who is responsible for solving the crisis that is currently occurring. The types of sensors we use will also give the information of the humidity level in the field, the temperature of the field, and detection of animals using their thermal radiation and also we process the information and give them in the form of graphs and images to the farmers for easy understanding.

## **PROJECT FLOW:**

The device will sense the animals and birds using the Clarifai service. If any animal or bird is detected the image will be captured and stored in the IBM Cloud object storage. It also generates an alarm and avoids animals from destroying the crop. The image URL will be stored in the IBM Cloudant DB service. The device will also monitor the soil moisture levels, temperature, and humidity values and send them to the IBM IoT Platform. The image will be retrieved from Object storage and displayed in the web application. A web application is developed to visualize the soil moisture, temperature, and humidity values. Users can also control the motors through web applications.

## **Implementation:**

Firstly, we should create codes for connecting the sensors to the Arduino and connecting the Arduino to the Wifi module and connecting them to the Internet. Then we should create codes to monitor and intimate messages about humidity and temperature on a regular basis, and codes should be written for PIR sensor and UV sensor to make sure that the motion detection of animals is being intimidated and preventive measures are taken. The preventive measure for every problem should be given according to the problem that arose and the codes for every problem and their solution should be fed on the cloud to access and as a result if the person doesn't know what to do in this type of situation then they can refer to the solutions. Codes should be written to not to intimate humans and also there should be power backup for the system to function efficiently. The backup system is solar and all the products used should consume less power and function more efficiently. The system should be made in a way that it can function more effectively even when there is very low data rate. The program should be coded in such a way.

## **To Accomplish this, we have to complete all the activities and tasks listed below:**

1. Create and configure IBM Cloud Services
  - o Create IBM Watson IoT Platform
  - o Create a device & configure the IBM IoT Platform
    - o Create Node-RED service
    - o Create a database in Cloudant DB to store location data
    - o Create a cloud object storage service and create a bucket to store the images
2. 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.
3. Develop a web Application using Node-RED Service.
4. 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

## **Conclusion:**

As a result of this system, we can detect the changes in the field easily and intimate the farmers about it and also we can take precautions and do remedies accordingly. Here we use very low power consuming highly efficient components that give us accurate results and also they perform at low data rate conditions without any lag and help in finding the remedies. This crop protection system helps in detection of all kinds of external dangers and it saves time and money to the farmers before any loss that may occur. With the help of this system the farmers can be in a peaceful environment at ease without any pressure.

## PROBLEM STATEMENT:

Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	Farmer	Monitor my crops	There are some disturbances	Of birds, animals & insects	Very frustrated and depressed about my field
PS-2	Farmer	Prevent animals from attacking my field	There is no easy and helpful technology	Of many kinds of birds & animals attack according to the type of cultivation	Unable to do anything many times

## PROPOSED SOLUTION FIT:

<b>1. CUSTOMER SEGMENT(S)</b> Define CS, fit into CL Farmer's ! Who's not near his field	<b>6. CUSTOMER LIMITATIONS</b> CO, BUDGET, DEVICES CL 1) High adoption costs; security concerns. 2) Not aware of the implementation of IoT in agriculture.	<b>5. AVAILABLE SOLUTIONS</b> PLUSES & MINUSES AS Monitor different parameters and mobile or web application make easily to farm the crop field.
<b>2. PROBLEMS / PAINS</b> ITS FREQUENCY PR It's difficult to monitor and control Ain't known if the application doesn't work properly.	<b>9. PROBLEM ROOT / CAUSE</b> RC 1) If temperature, PH level, humidity & light intensity makes the serious cause for the environment. 2) Farmer affected by less productivity which will affect in their	<b>7. BEHAVIOR</b> ITS INTENSITY BE Direct related: Tries to find a solution to prevent this problem Indirect related: Located in rural where internet connectivity might not be strong enough to facilitate fast transmission speeds.
<b>3. TRIGGERS TO ACT</b> TR Create opportunities to lift people out of poverty in developing nations. (Over 60%)	<b>10. YOUR SOLUTION</b> SL IoT based Smart crop protection system for agriculture !! It help farmers grow more food on less land by protection crops from pests, diseases and weeds as well as raising productivity	<b>8. CHANNELS of BEHAVIOR</b> CH ONLINE: The Data send through application for the farmers to know about the farms. OFFLINE: The control action is taken by the farmers to monitor the farms.
<b>4. EMOTIONS</b> BEFORE / AFTER EM BEFORE: Finances, Heavy work overload and conflict in relationship. AFTER: It will easier to make more yield		

## REQUIRED SOFTWARE:

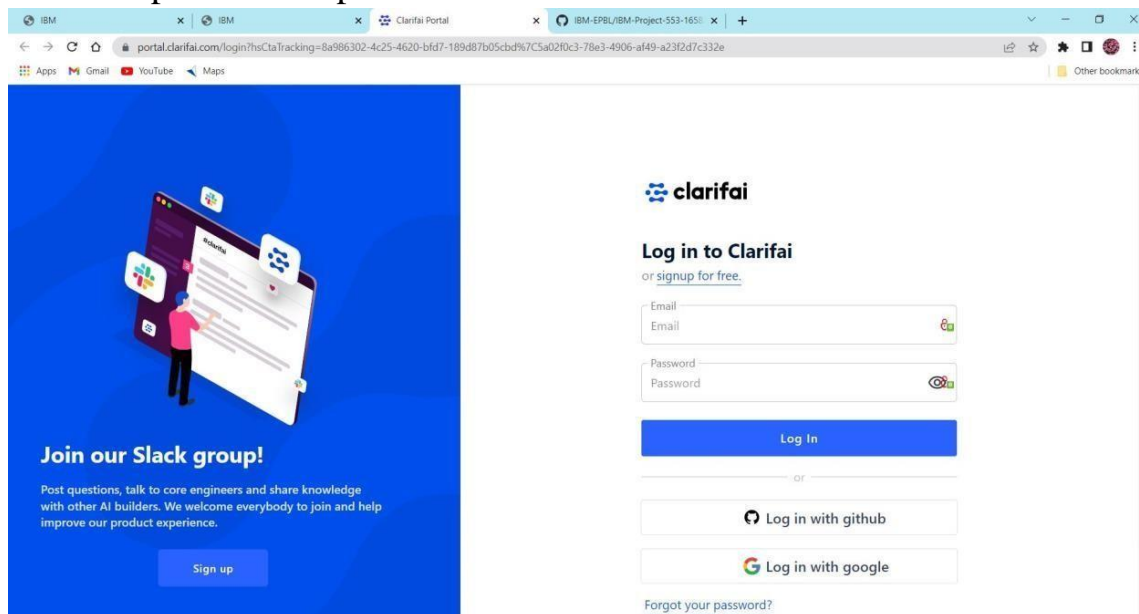
- CLARIFAI
- IBM WATSON IOT PLATFORM
- PYTHON IDLE
- NODE RED
- MIT APP INVENTOR

## CLARIFAI:

Clarifai provides an end-to-end platform with the easiest to use UI and API in the market. Clarifai Inc. is an artificial intelligence (AI) company that specializes in computer vision and uses machine learning and deep neural networks to identify and analyse images and videos. The company offers its solution via API, mobile SDK, and on-premise solutions.

### STEP 1:

- Open Clarifai portal in web browser.




### STEP 2:

- Signup using the required user mail and password

Already have an ICT Academy account? [Sign in.](#)

————— Create an account using —————

 ICT Academy

————— or create a new one here —————

Full Name

Public Username

Email (required) !

*\*"pskavya" is not a valid email address.  
This is what you will use to login.*

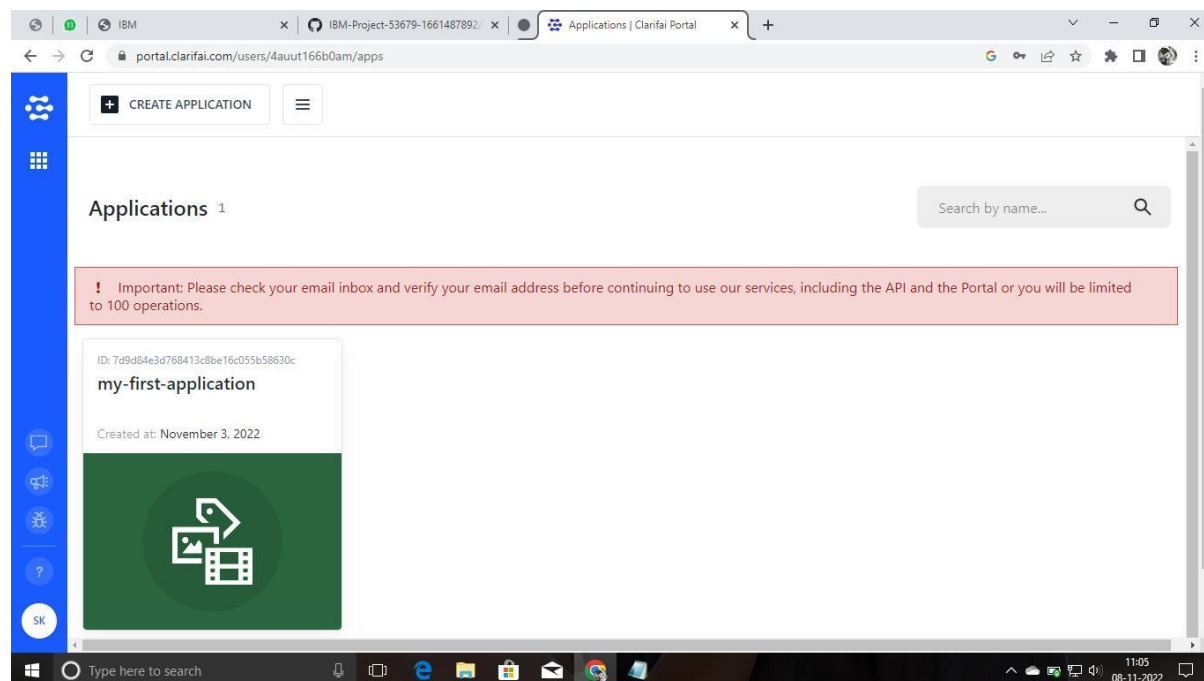
Password

☐ I agree to the ICT Academy [Terms of Service](#)

**Create Account**

### STEP 3:

Finally, Created an account



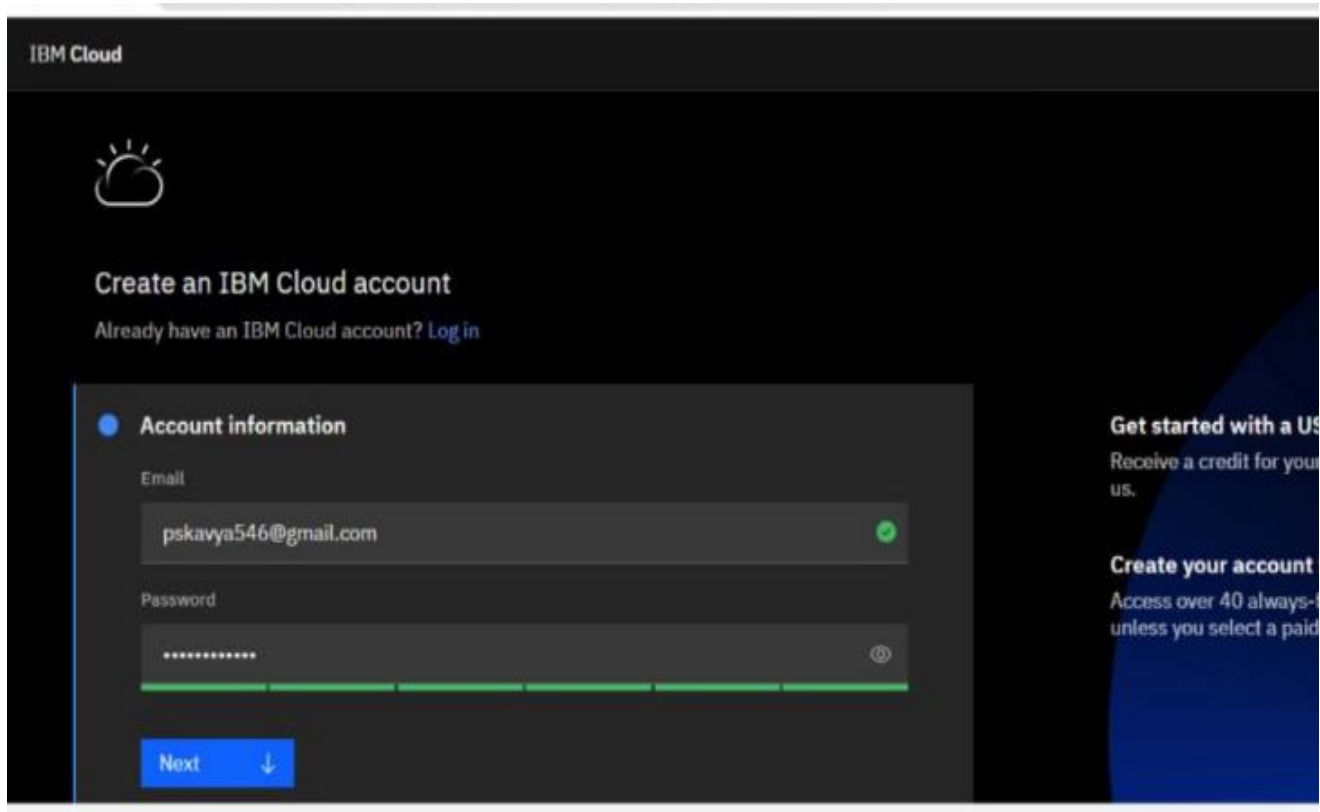
### IBM WATSON IoT PLATFORM:

We need to have basic knowledge of the following cloud services:

- IBM Watson IoT Platform
- Node-RED Service
- Cloudant DB

We need to create an IBM Cloud Account to complete this project.

## LOGIN:



 [courses.ictacademy.skillsnetwork.site/courses/course-v1:IBMDDeveloperSkillsNetwork+CC0103EN+v3/courseware/703354f1007245aabaeb47447e732f7c/d90d4a7252b14c5186fe06bd513386e...](https://courses.ictacademy.skillsnetwork.site/courses/course-v1:IBMDDeveloperSkillsNetwork+CC0103EN+v3/courseware/703354f1007245aabaeb47447e732f7c/d90d4a7252b14c5186fe06bd513386e...)

Please check the box and click on the Open tool button below to obtain a unique Feature Code and copy it. You will be able to apply it using the instructions that follow in a subsequent lab in the course.

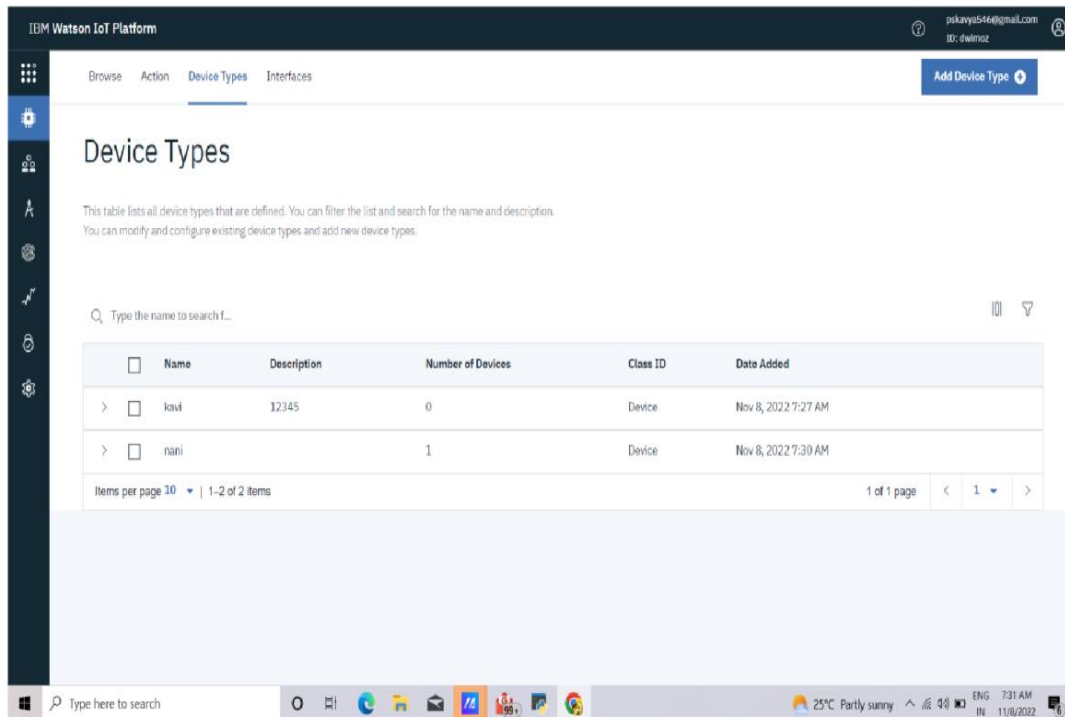
**NOTE:** If you have already applied your IBM Cloud feature code in another course/lab to create an IBM Cloud account or extend its trial, please skip this item, as the code can only be used once.

### LTI Consumer (External resource)

Open Tool 

[< Previous](#) [Next >](#)





## PYTHON IDLE INSTALLATION:

Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific problems.

### STEP 1:

- Python is installed successfully

```
Python 3.8.4 Shell
File Edit Shell Debug Options Window Help
Python 3.8.4 (tags/v3.8.4:dfa645a, Jul 13 2020, 16:46:45) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("hello world")
hello world
>>> |
```

## STEP 2:

- The required python libraries are installed.
- Watson IoT Python SDK to connect to IBM Watson IoT Platform using python code is installed
- `pip install wiotp-sdk`

```
Command Prompt
[msi quit()] or Ctrl-Z plus Return to exit
>>> quit()

C:\Users\swast>pip --version
pip 20.1.1 from c:\users\swast\appdata\local\programs\python\python38\lib\site-packages\pip (python 3.8)

C:\Users\swast>pip install wiotp-sdk
Collecting wiotp-sdk
  Downloading wiotp-sdk-0.11.0.tar.gz (96 kB)
    |#####| 96 kB 130 kB/s
Collecting iso8601>=0.1.12
  Downloading iso8601-1.1.0-py3-none-any.whl (9.9 kB)
Requirement already satisfied: pytz>=2018.9 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from wiotp-sdk) (2020.1)
Collecting pyyaml>=3.13
  Downloading PyYAML-6.0-cp38-cp38-win_amd64.whl (155 kB)
    |#####| 155 kB 126 kB/s
Collecting paho-mqtt>=1.5.0
  Downloading paho-mqtt-1.6.1.tar.gz (99 kB)
    |#####| 99 kB 172 kB/s
Collecting requests>=2.21.0
  Downloading requests-2.28.1-py3-none-any.whl (62 kB)
    |#####| 62 kB 155 kB/s
Collecting requests_toolbelt>=0.8.0
  Downloading requests_toolbelt-0.10.1-py2.py3-none-any.whl (54 kB)
    |#####| 54 kB 281 kB/s
Collecting charset-normalizer<3,>=2
  Downloading charset-normalizer-2.1.1-py3-none-any.whl (39 kB)
Collecting idna<4,>=2.5
  Downloading idna-3.4-py3-none-any.whl (61 kB)
    |#####| 61 kB 40 kB/s
Collecting certifi>=2017.4.17
  Downloading certifi-2022.9.24-py3-none-any.whl (161 kB)
    |#####| 161 kB 261 kB/s
Collecting urllib3<4.27,>=1.21.1
  Downloading urllib3-1.26.12-py2.py3-none-any.whl (140 kB)
    |#####| 140 kB 177 kB/s
Building wheels for collected packages: wiotp-sdk, paho-mqtt
  Building wheel for wiotp-sdk (setup.py) ... done
  Created wheel for wiotp-sdk: filename=wiotp_sdk-0.11.0-py3-none-any.whl size=97110 sha256=2f750f6e916d4044ba4c5368d6c81c3938d6a7416cbd533cc52423a13021b571
  Stored in directory: c:\users\swast\appdata\local\pip\cache\wheels\46\41\69\352062eb129b1d46c4d32ec9d3835d3b5248ef4432cdcfa3d9
  Building wheel for paho-mqtt (setup.py) ... done
  Created wheel for paho-mqtt: filename=paho_mqtt-1.6.1-py3-none-any.whl size=65428 sha256=6d15dbdc481fc2e1dc91265a42fba6322395188013f91a32c25e225b108646
  Stored in directory: c:\users\swast\appdata\local\pip\cache\wheels\6a\48\01\c895c027e969367ec5470bf371ee56e795a9acc6a19aadcf9
Successfully built wiotp-sdk paho-mqtt
Installing collected packages: iso8601, pyyaml, paho-mqtt, charset-normalizer, idna, certifi, urllib3, requests, requests-toolbelt, wiotp-sdk
Successfully installed certifi-2022.9.24 charset-normalizer-2.1.1 idna-3.4 iso8601-1.1.0 paho-mqtt-1.6.1 pyyaml-6.0 requests-2.28.1 requests-toolbelt-0.10.1 urllib3-1.26.12 wiotp-sdk-0.11.0
WARNING: You are using pip version 20.1.1; however, version 22.3 is available.
You should consider upgrading via the 'c:\users\swast\appdata\local\programs\python\python38\python.exe -m pip install --upgrade pip' command.

C:\Users\swast>
```

- Python client library for IBM Text to Speech is installed
- `pip install --upgrade "ibm-watson">=5.0.0`



```
Command Prompt
C:\Users\swast>pip install --upgrade "ibm-watson>=5.0.0"
Collecting ibm-watson>=5.0.0
  Downloading ibm-watson-6.1.0.tar.gz (373 kB)
    | 373 kB 142 kB/s
    Installing build dependencies ... done
    Getting requirements to build wheel ... done
    Preparing wheel metadata ... done
Collecting ibm-cloud-sdk-core==3.*>=3.3.6
  Downloading ibm-cloud-sdk-core-3.16.0-py3-none-any.whl (83 kB)
    | 83 kB 152 kB/s
Requirement already satisfied, skipping upgrade: requests<3.0,>=2.0 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from ibm-watson>=5.0.0) (2.28.1)
Collecting websocket-client==1.1.0
  Downloading websocket-client-1.1.0-py2.py3-none-any.whl (68 kB)
    | 68 kB 195 kB/s
Requirement already satisfied, skipping upgrade: python-dateutil>=2.5.3 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from ibm-watson>=5.0.0) (2.8.1)
Collecting PyJWT<3.0.0,>=2.4.0
  Downloading PyJWT-2.6.0-py3-none-any.whl (20 kB)
Requirement already satisfied, skipping upgrade: urllib3<2.0.0,>=1.26.0 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from ibm-cloud-sdk-core==3.*>=3.3.6->ibm-watson>=5.0.0) (1.26.12)
Requirement already satisfied, skipping upgrade: certifi>=2017.4.17 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from requests<3.0,>=2.0->ibm-watson>=5.0.0) (2022.9.24)
Requirement already satisfied, skipping upgrade: charset-normalizer<3,>=2 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from requests<3.0,>=2.0->ibm-watson>=5.0.0) (2.1.1)
Requirement already satisfied, skipping upgrade: six>=1.5 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from python-dateutil>=2.5.3->ibm-watson>=5.0.0) (1.15.0)
Building wheels for collected packages: ibm-watson
  Building wheel for ibm-watson (PEP 517) ... done
  Created wheel for ibm-watson: filename=ibm_watson-6.1.0-py3-none-any.whl size=370748 sha256=50648bccc54ee0ba245cc521468536cd77a9cf975fccc5f975bddf9ba6956
  Stored in directory: c:\users\swast\appdata\local\pip\cache\wheels\34\b4\cd\829a351c802b7a578115fe7ddaedff62b29ae84e90882c7e2
Successfully built ibm-watson
Installing collected packages: PyJWT, ibm-cloud-sdk-core, websocket-client, ibm-watson
Successfully installed PyJWT-2.6.0 ibm-cloud-sdk-core-3.16.0 ibm-watson-6.1.0 websocket-client-1.1.0
WARNING: You are using pip version 20.1.1; however, version 22.3 is available.
You should consider upgrading via the 'c:\users\swast\appdata\local\programs\python\python38\python.exe -m pip install --upgrade pip' command.

C:\Users\swast>
```

- Required Libraries for cloud object storage is installed.
- pip install ibm-cos-sdk

```
Command Prompt
C:\Users\swast>pip install ibm-cos-sdk
Collecting ibm-cos-sdk
  Downloading ibm-cos-sdk-2.12.0.tar.gz (55 kB)
    | 55 kB 411 kB/s
Collecting ibm-cos-sdk-core==2.12.0
  Downloading ibm-cos-sdk-core-2.12.0.tar.gz (956 kB)
    | 956 kB 251 kB/s
Collecting ibm-cos-sdk-s3transfer==2.12.0
  Downloading ibm-cos-sdk-s3transfer-2.12.0.tar.gz (135 kB)
    | 135 kB 242 kB/s
Collecting jmespath<1.0.0,>=0.10.0
  Downloading jmespath-0.10.0-py2.py3-none-any.whl (24 kB)
Collecting python-dateutil<3.0.0,>=2.8.2
  Downloading python-dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
    | 247 kB 142 kB/s
Requirement already satisfied: requests<3.0,>=2.27.1 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from ibm-cos-sdk-core==2.12.0->ibm-cos-sdk) (2.28.1)
Requirement already satisfied: urllib3<1.27,>=1.26.9 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from ibm-cos-sdk-core==2.12.0->ibm-cos-sdk) (1.26.12)
Requirement already satisfied: six>=1.5 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from python-dateutil<3.0.0,>=2.8.2->ibm-cos-sdk-core==2.12.0->ibm-cos-sdk) (1.15.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from requests<3.0,>=2.27.1->ibm-cos-sdk-core==2.12.0->ibm-cos-sdk) (2022.9.24)
Requirement already satisfied: idna<4,>=2.5 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from requests<3.0,>=2.27.1->ibm-cos-sdk-core==2.12.0->ibm-cos-sdk) (3.4)
Requirement already satisfied: charset-normalizer<3,>=2 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from requests<3.0,>=2.27.1->ibm-cos-sdk-core==2.12.0->ibm-cos-sdk) (2.1.1)
Building wheels for collected packages: ibm-cos-sdk, ibm-cos-sdk-core, ibm-cos-sdk-s3transfer
  Building wheel for ibm-cos-sdk (setup.py) ... done
  Created wheel for ibm-cos-sdk: filename=ibm_cos_sdk-2.12.0-py3-none-any.whl size=73926 sha256=a6f65caad036b69209e205e7f0b1855bf4a721a71f535188f94c734e01c3be
  Stored in directory: c:\users\swast\appdata\local\pip\cache\wheels\21\5f\fd\6a04bb45aad71bc8b30083436f9d39ef7c4fd1b69d226d
  Building wheel for ibm-cos-sdk-core (setup.py) ... done
  Created wheel for ibm-cos-sdk-core: filename=ibm_cos_sdk_core-2.12.0-py3-none-any.whl size=562952 sha256=c7f8e89def7511d484073c5092533731d8715bad59fb3deda5ad0d38a3f99d7
  Stored in directory: c:\users\swast\appdata\local\pip\cache\wheels\ca\3d\78\48c57e974477098f3acc8c1df482b1de8a8cf8c69d668ee7
  Building wheel for ibm-cos-sdk-s3transfer (setup.py) ... done
  Created wheel for ibm-cos-sdk-s3transfer: filename=ibm_cos_sdk_s3transfer-2.12.0-py3-none-any.whl size=89769 sha256=67c5983a4abd0be33db07cb1d35d7216ebef83fec9e5f8275d9f8e51ceb777
  Stored in directory: c:\users\swast\appdata\local\pip\cache\wheels\c0\7a\17\13b53ca7d27a1062a47c58ba1c2ff3832795b698c8dbd46
Successfully built ibm-cos-sdk ibm-cos-sdk-core ibm-cos-sdk-s3transfer
Installing collected packages: jmespath, python-dateutil, ibm-cos-sdk-core, ibm-cos-sdk-s3transfer, ibm-cos-sdk
  Attempting uninstall: python-dateutil
    Found existing installation: python-dateutil 2.8.1
    Uninstalling python-dateutil-2.8.1:
      Successfully uninstalled python-dateutil-2.8.1
Successfully installed ibm-cos-sdk-2.12.0 ibm-cos-sdk-core-2.12.0 ibm-cos-sdk-s3transfer-2.12.0 jmespath-0.10.0 python-dateutil-2.8.2
WARNING: You are using pip version 20.1.1; however, version 22.3 is available.
You should consider upgrading via the 'c:\users\swast\appdata\local\programs\python\python38\python.exe -m pip install --upgrade pip' command.

C:\Users\swast>
```

- pip install -U ibm-cos-sdk

```
Command Prompt
WARNING: You are using pip version 20.1.1; however, version 22.3 is available.
You should consider upgrading via the 'c:\users\swast\appdata\local\programs\python\python38\python.exe -m pip install --upgrade pip' command.

C:\Users\swast>pip install -U ibm-cos-sdk
Requirement already up-to-date: ibm-cos-sdk in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (2.12.0)
Requirement already satisfied, skipping upgrade: ibm-cos-sdk-s3transfer==2.12.0 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from ibm-cos-sdk) (2.12.0)
Requirement already satisfied, skipping upgrade: ibm-cos-sdk-core==2.12.0 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from ibm-cos-sdk) (2.12.0)
Requirement already satisfied, skipping upgrade: jmespath<1.0.0,>=0.10.0 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from ibm-cos-sdk) (0.10.0)
Requirement already satisfied, skipping upgrade: requests<3.0,>=2.27.1 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from ibm-cos-sdk-core==2.12.0->ibm-cos-sdk) (2.28.1)
Requirement already satisfied, skipping upgrade: urllib3<1.27,>=1.26.9 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from ibm-cos-sdk-core==2.12.0->ibm-cos-sdk) (1.26.12)
Requirement already satisfied, skipping upgrade: python-dateutil<3.0.0,>=2.8.2 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from ibm-cos-sdk-core==2.12.0->ibm-cos-sdk) (2.8.2)
Requirement already satisfied, skipping upgrade: charset-normalizer<3,>=2 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from requests<3.0,>=2.27.1->ibm-cos-sdk-core==2.12.0->ibm-cos-sdk) (2.1.1)
Requirement already satisfied, skipping upgrade: idna<4,>=2.5 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from requests<3.0,>=2.27.1->ibm-cos-sdk-core==2.12.0->ibm-cos-sdk) (3.4)
Requirement already satisfied, skipping upgrade: certifi>2017.4.17 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from requests<3.0,>=2.27.1->ibm-cos-sdk-core==2.12.0->ibm-cos-sdk) (2022.9.24)
Requirement already satisfied, skipping upgrade: six>=1.5 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from python-dateutil<3.0.0,>=2.8.2->ibm-cos-sdk-core==2.12.0->ibm-cos-sdk) (1.15.0)
WARNING: You are using pip version 20.1.1; however, version 22.3 is available.
You should consider upgrading via the 'c:\users\swast\appdata\local\programs\python\python38\python.exe -m pip install --upgrade pip' command.

C:\Users\swast>
```

- pip install boto3

```
Command Prompt
WARNING: You are using pip version 20.1.1; however, version 22.3 is available.
You should consider upgrading via the 'c:\users\swast\appdata\local\programs\python\python38\python.exe -m pip install --upgrade pip' command.

C:\Users\swast>pip install boto3
Collecting boto3
  Downloading boto3-1.26.0-py3-none-any.whl (132 kB)
    |#####| 132 kB 148 kB/s
Collecting s3transfer<0.7.0,>=0.6.0
  Downloading s3transfer-0.6.0-py3-none-any.whl (79 kB)
    |#####| 79 kB 113 kB/s
Collecting botocore<1.30.0,>=1.29.0
  Downloading botocore-1.29.0-py3-none-any.whl (9.8 MB)
    |#####| 9.8 MB 2.2 MB/s
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from boto3) (0.10.0)
Requirement already satisfied: urllib3<1.27,>=1.25.4 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from botocore<1.30.0,>=1.29.0->boto3) (1.26.12)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from botocore<1.30.0,>=1.29.0->boto3) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from python-dateutil<3.0.0,>=2.1->botocore<1.30.0,>=1.29.0->boto3) (1.15.0)
Installing collected packages: botocore, s3transfer, boto3
Successfully installed boto3-1.26.0 botocore-1.29.0 s3transfer-0.6.0
WARNING: You are using pip version 20.1.1; however, version 22.3 is available.
You should consider upgrading via the 'c:\users\swast\appdata\local\programs\python\python38\python.exe -m pip install --upgrade pip' command.

C:\Users\swast>
```

- pip install resources

```
Command Prompt
C:\Users\swast>pip install resources
Collecting resources
  Downloading resources-0.0.1.tar.gz (3.7 kB)
  Building wheels for collected packages: resources
  Building wheel for resources (setup.py) ... done
  Created wheel for resources: file=resources-0.0.1-py3-none-any.whl size=4370 sha256=38113eb3ac96cbfb54f0f22303a68ae5a6ac976211e26ae94f9b2441ec318e
  Stored in directory: c:\users\swast\appdata\local\pip\cache\wheels\b3\1d\00\45ae97c7b92d145a0963f711c6d22f9af5306e74c88f2f28fd
Successfully built resources
Installing collected packages: resources
Successfully installed resources-0.0.1
WARNING: You are using pip version 20.1.1; however, version 22.3 is available.
You should consider upgrading via the 'c:\users\swast\appdata\local\programs\python\python38\python.exe -m pip install --upgrade pip' command.
C:\Users\swast>
```

- pip install cloudant

```
Command Prompt
You should consider upgrading via the 'c:\users\swast\appdata\local\programs\python\python38\python.exe -m pip install --upgrade pip' command.
C:\Users\swast>pip install cloudant
Collecting cloudant
  Downloading cloudant-2.15.0-py3-none-any.whl (80 kB)
    | 80 kB 395 kB/s
Requirement already satisfied: requests<3.0.0,>=2.7.0 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from cloudant) (2.28.1)
Requirement already satisfied: charset-normalizer<3,>=2 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from requests<3.0.0,>=2.7.0->cloudant) (2.1.1)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from requests<3.0.0,>=2.7.0->cloudant) (1.26.12)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from requests<3.0.0,>=2.7.0->cloudant) (2022.9.24)
Requirement already satisfied: idna<4,>=2.5 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from requests<3.0.0,>=2.7.0->cloudant) (3.4)
Installing collected packages: cloudant
Successfully installed cloudant-2.15.0
WARNING: You are using pip version 20.1.1; however, version 22.3 is available.
You should consider upgrading via the 'c:\users\swast\appdata\local\programs\python\python38\python.exe -m pip install --upgrade pip' command.
C:\Users\swast>
```

## DATA FROM PYTHON TO IBM:

Python code to generate random data and pass it to IBM Watson IoT platform

### Source Code:

```
import
time
import
sys
```

```

import cv2
import numpy as np
import wiot.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, resource_pb2
from clarifai_grpc.grpc.api.status import status_code_pb2

#This is how you authenticate
metadata = (('authorization', 'key 0620e202302b4508b90eab7efe7475e4'),)
COS_ENDPOINT = "https://s3.jp-tok.cloud-object-storage.appdomain.cloud"
COS_API_KEY_ID = "g5d4q08EIgv4TWUCJj4hfEzgalqEjrDbE82AJDWLAOHO"
COS_AUTH_ENDPOINT = "https://iam.cloud.ibm.com/identity/token"
COS_RESOURCE_CRN = "crn:v1:bluemix:public:cloud-object-
storage:global:a/c2fa2836eaf3434bbc8b5b58fefff3f0:62e450fd-4c82-4153-
ba41-ccb53adb8111::"
clientdb = cloudant("apikey-
W2njldnwtjO16V53LAVUCqPwc2aHTLmlj1xXvtdGKJBn",
"88cc5f47c1a28afbfb8ad16161583f5a", url="https://d6c89f97-cf91-48b7-b14b-
c99b2fe27c2f-bluemix.cloudantnosqldb.appdomain.cloud")
clientdb.connect()

#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="oauth"),
                        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
        #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_fileobj method will automatically execute a multi-
part upload
        #in 5 MB chunks size
        with open(file_path, "rb") as file_data:
            cos.Object(bucket_name, item_name).upload_fileobj(
                Fileobj=file_data,
                Config=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 myCommandCallback(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=="motoron"):
        print('motoron')
    elif(command=="motoroff"):
        print('motoroff')
myConfig = {
    "identity": {
        "orgId": "chytun",
        "typeId": "NodeMCU",
        "deviceId": "12345"
    },
    "auth": {
        "token": "12345678"
    }
}

client = wiot.sdk.device.DeviceClient(config=myConfig, logHandlers=None)
client.connect()

database_name = "sample"
my_database = clientdb.create_database(database_name)
if my_database.exists():
    print(f'({database_name})' successfully created.")
cap=cv2.VideoCapture("garden.mp4")
if(cap.isOpened()==True):
    print('File opened')
else:
    print('File not found')

while(cap.isOpened()):
    ret, frame = cap.read()
    gray = cv3.cvtColor(frame, cv2.COLOR_BGR@GRAY)
    imS= cv2.resize(frame, (960,540))

```



```

cv2.imwrite('ex.jpg',imS)
with open("ex.jpg", "rb") as f:
    file_bytes = f.read()
#This is the model ID of a publicly available General model. You may
use any other public or custom model ID.
request = service_pb2.PostModelOutputsRequest(
    model_id='e9359dbe6ee44dbc8842ebe97247b201',

inputs=[resources_pb2.Input(data=resources_pb2.Data(image=resources_pb2.I
mage(base64=file_bytes))

    ))
response = stub.PostModelOutputs(request, metadata=metadata)
if response.status.code != status_code_pb2.SUCCESS:
    raise Exception("Request failed, status code: " +
str(response.status.code))
detect=False
for concept in response.outputs[0].data.concepts:
    #print('%12s: %.f' % (concept.name, concept.value))
    if(concept.value>0.98):
        #print(concept.name)
        if(concept.name=="animal"):
            print("Alert! Alert! animal detected")
            playsound.playsound('alert.mp3')
            picname=datetime.datetime.now().strftime("%y-%m-%d-%H-
%M")
            cv2.imwrite(picname+'.jpg',frame)
            multi_part_upload('Dhakshesh', picname+'.jpg',
picname+'.jpg')

json_document=({"link":COS_ENDPOINT+'/'+'Dhakshesh'+'/'+'picname+'.jpg'})
new_document = my_database.create_document(json_document)
if new_document.exists():
    print(f"Document successfully created.")
    time.sleep(5)
    detect=True
moist=random.randint(0,100)
humidity=random.randint(0,100)
myData=({'Animal':detect,'moisture':moist,'humidity':humidity})
print(myData)
if(humidity!=None):
    client.publishEvent(eventId="status",msgFormat="json",
daya=myData, qos=0, onPublish=None)
    print("Publish Ok..")
    client.commandCallback = myCommandCallback
    cv2.imshow('frame',imS)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
client.disconnect()
cap.release()
cv2.destroyAllWindows()

```

## DATA GENERATION IOT PLATFORM:

Source code is deployed on IBM Watson IoT platform to generate sensor data.

### Source Code:

```
{  
  
    "temperature": random(0, 100),  
  
    "humidity": random(0, 100),  
  
    "moisture": random(0, 100),  
  
    "animalDetected": random(0,2)  
  
}
```

### Output:

The screenshot displays the IBM Watson IoT Platform interface. The main window shows the 'Recent Events' tab for a device named 'cropProtection'. The events table lists three events, each with a payload containing temperature, humidity, moisture, and animalDetected values. A modal window is open for configuring a new event type named 'event\_1'. The 'Schedule' is set to 'Every Minute' with a delay of '20'. The 'Payload' is a JSON object with the same structure as the events shown in the table.

Event	Value
event_1	{"temp":93,"hum":16,"moisture":97,"anim
event_1	{"temp":90,"hum":73,"moisture":15,"anim
event_1	{"temp":77,"hum":86,"moisture":87,"anim

Device Type: crop

Events: 1

Event type name: event\_1

Schedule: 20 Every Minute

Payload:

```
{  
  "temp": random(0, 100),  
  "hum": random(0, 100),  
  "moisture": random(0, 100),  
  "animalDetected": random(0, 2)  
}
```

Device: cropProtection

Status: Disconnected

Time: Oct 13, 2022 9:04 AM

## PYTHON CODE TO IBM:

```
import time
import sys

import ibmiotf.application
import ibmiotf.device import
random

#Provide your IBM Watson Device Credentials
organization = "wu5b55"

deviceType = "crop1"
deviceId = "1234"
authMethod = "token"
authToken = "1234567890"

# Initialize GPIO

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 thecloud as
an event of type "greeting" 10 times
deviceCli.connect()
```



```
while True:
```

```
    #Get Sensor Data from DHT11
```

```
    temp=random.randint(0,100)
```

```
    Hum=random.randint(0,100)
```

```
    moisture=random.randint(0,100)
```

```
    data = { 'temperature' : temp, 'Humidity': Hum,  
    'Moisture':moisture }
```

```
#print data
```

```
    def myOnPublishCallback():
```

```
        print ("Temperature = " + str(temp)+" C Humidity = " +  
        str(hum)+ " moisture = " + str(moisture) + "to IBM Watson")
```

```
    success = deviceCli.publishEvent("IoTSensor", "json",data,qos=0,  
    on_publish=myOnPublishCallback)
```

```
    if not success:
```

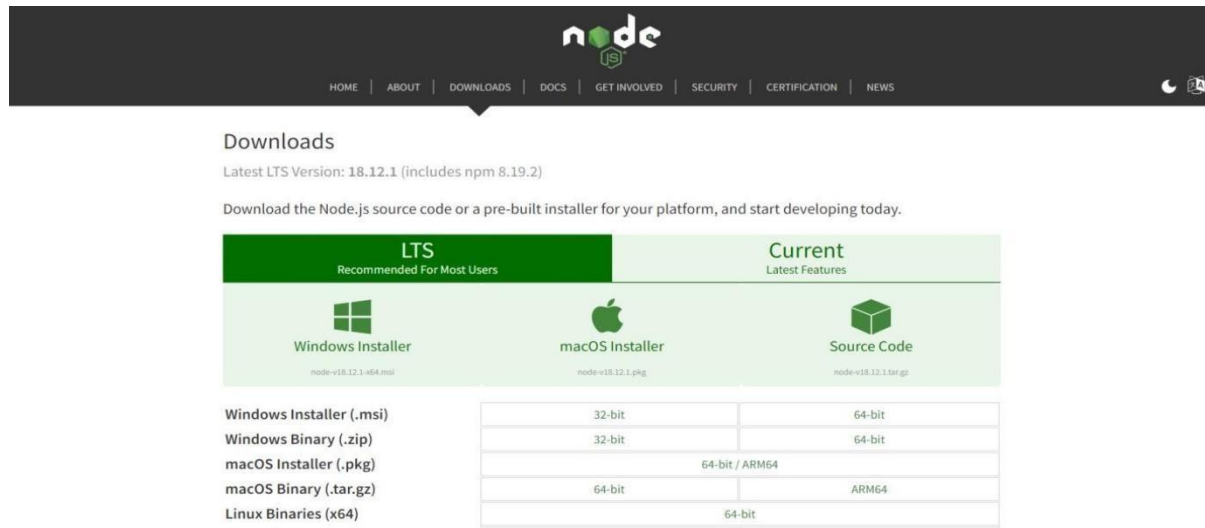
```
        print("Not connected to IoT")  
        time.sleep(10)
```

```
    deviceCli.commandCallback = myCommandCallback
```

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

## NODE-JS CONNECTION:

### STEP1: Download and Install NODE JS.



The image shows the Node.js Downloads page. At the top, there's a navigation bar with links: HOME, ABOUT, DOWNLOADS, DOCS, GET INVOLVED, SECURITY, CERTIFICATION, and NEWS. Below the navigation bar, the 'Downloads' section is highlighted. It states 'Latest LTS Version: 18.12.1 (includes npm 8.19.2)' and 'Download the Node.js source code or a pre-built installer for your platform, and start developing today.' There are two main tabs: 'LTS Recommended For Most Users' and 'Current Latest Features'. Under the 'LTS' tab, there are three options: 'Windows Installer' (node-v18.12.1-x64.msi), 'macOS Installer' (node-v18.12.1.pkg), and 'Source Code' (node-v18.12.1.tar.gz). Below these, there's a table showing the available binaries for different platforms and architectures.

Platform	Architecture	File Name
Windows	32-bit	node-v18.12.1-x86.msi
	64-bit	node-v18.12.1-x64.msi
macOS	64-bit	node-v18.12.1.pkg
	ARM64	node-v18.12.1.pkg
Linux	64-bit	node-v18.12.1-linux-x64.tar.gz
	ARM64	node-v18.12.1-linux-arm64.tar.gz

### STEP2: Setup node.js and configure command prompt for error check .open node-red from the generated link.

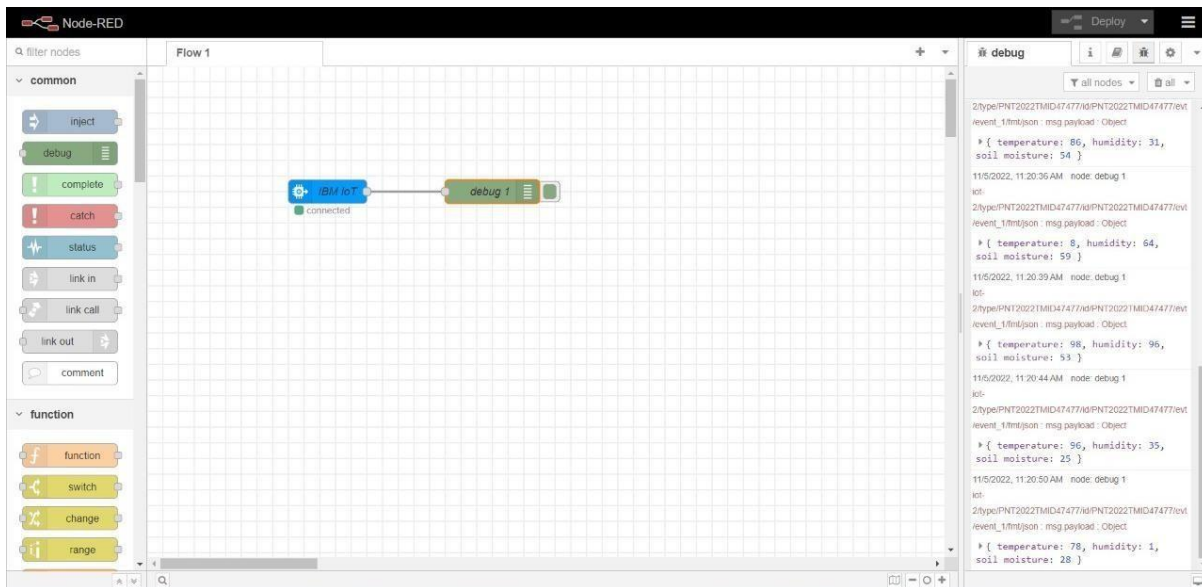
```
node-red
4 Nov 18:48:05 - [info] Node-RED version: v3.0.2
4 Nov 18:48:05 - [info] Node.js version: v18.12.0
4 Nov 18:48:05 - [info] Windows_NT 10.0.19044 x64 LE
4 Nov 18:48:26 - [info] Loading palette nodes
4 Nov 18:48:44 - [info] Settings file : C:\Users\ELCOT\.node-red\settings.js
4 Nov 18:48:45 - [info] Context store : 'default' [module=memory]
4 Nov 18:48:45 - [info] User directory : \Users\ELCOT\.node-red
4 Nov 18:48:45 - [warn] Projects disabled : editorTheme.projects.enabled=false
4 Nov 18:48:45 - [info] Flows file : \Users\ELCOT\.node-red\flows.json
4 Nov 18:48:45 - [info] Creating new flow file
4 Nov 18:48:45 - [warn]

-----
Your flow credentials file is encrypted using a system-generated key.

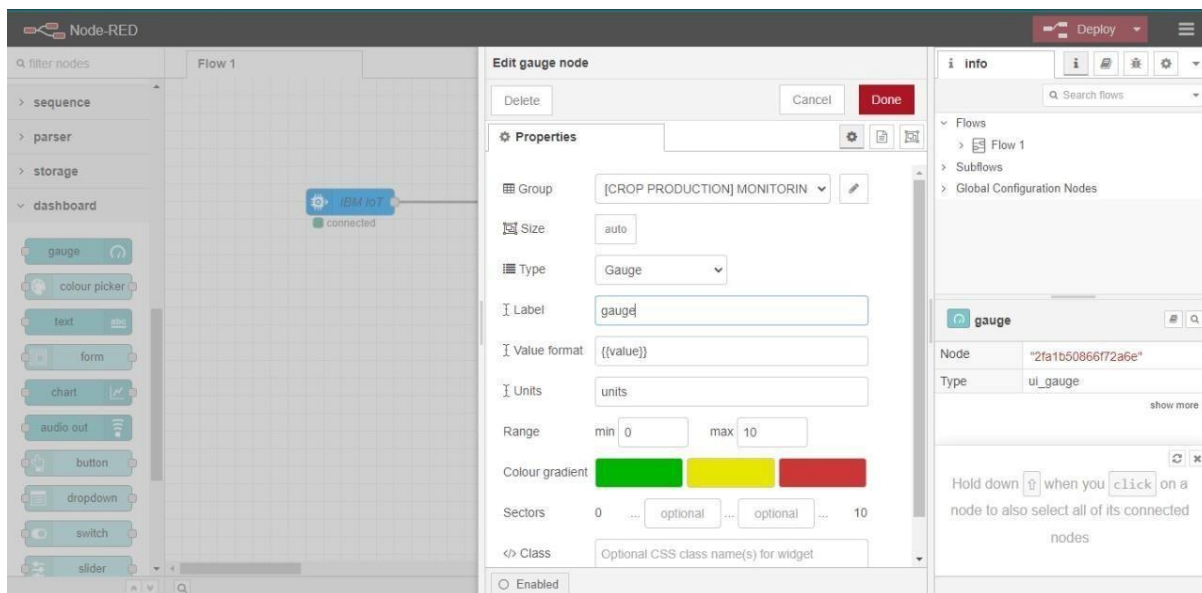
If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.

You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.
-----
4 Nov 18:48:45 - [warn] Encrypted credentials not found
4 Nov 18:48:45 - [info] Starting flows
4 Nov 18:48:46 - [info] Started flows
4 Nov 18:48:46 - [info] Server now running at http://127.0.0.1:1880/
```

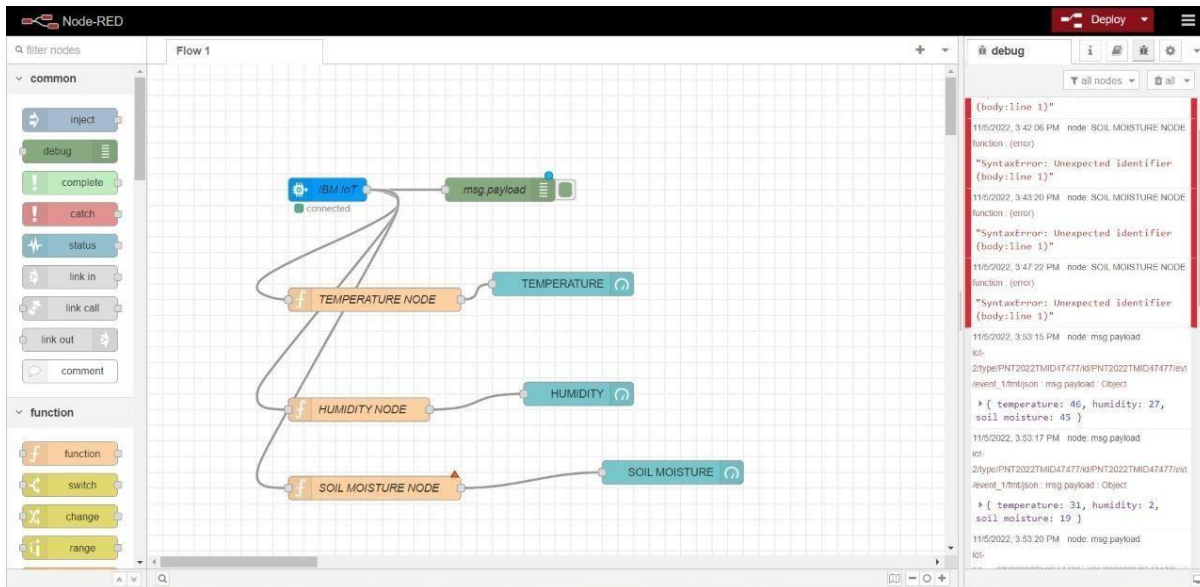
### STEP3: Connect IBM IOT in and Debug 1 and Deploy.



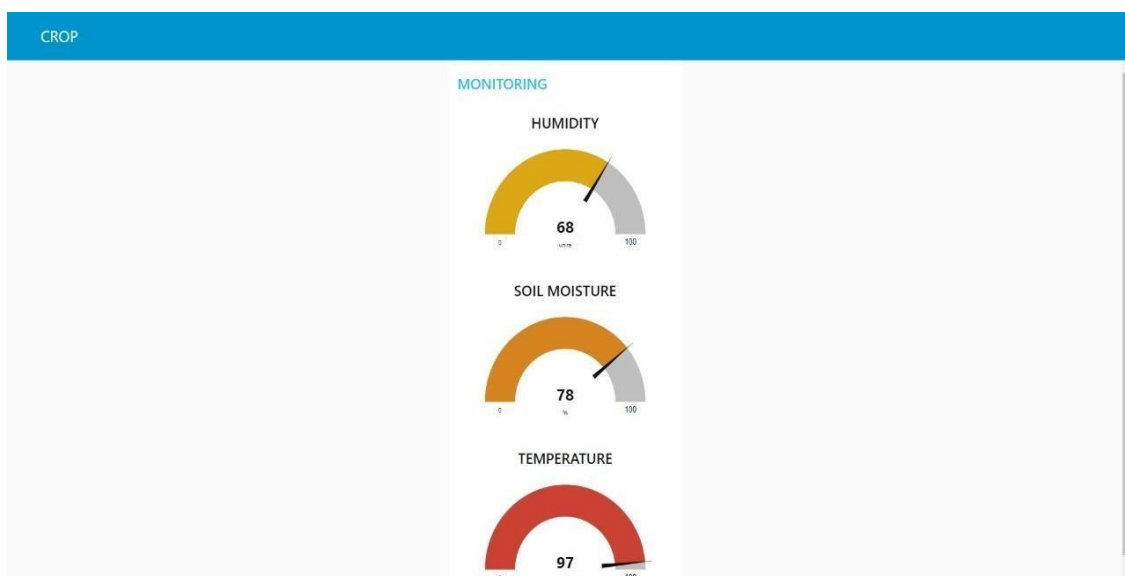
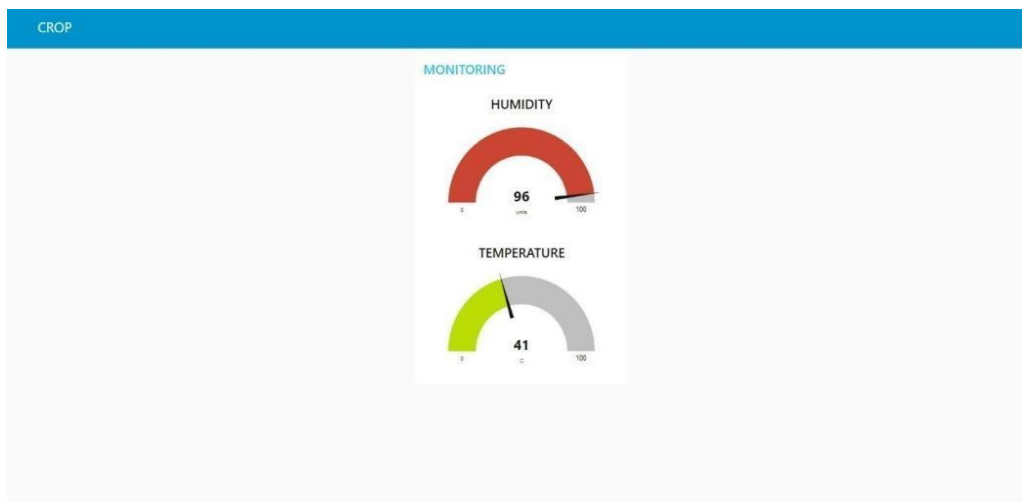
### STEP4: Edit gauge node (Here the gauge nodes are named as Temperature, Humidity and Soilmoisture).





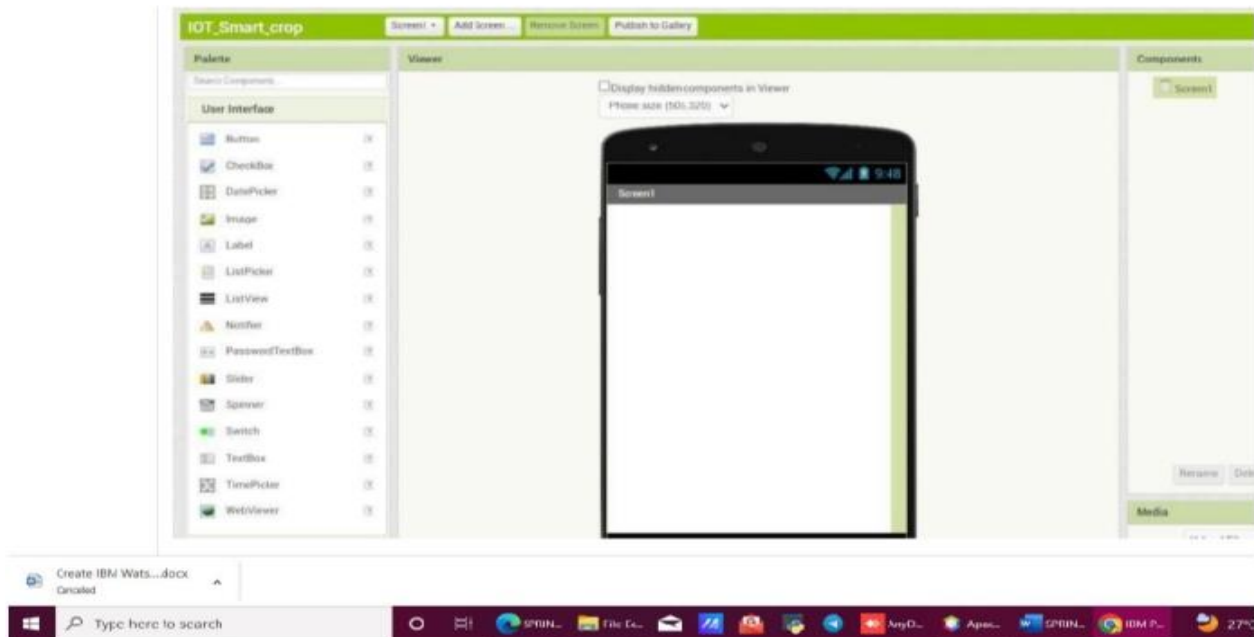


STEP3: Generate the some output from recent events.

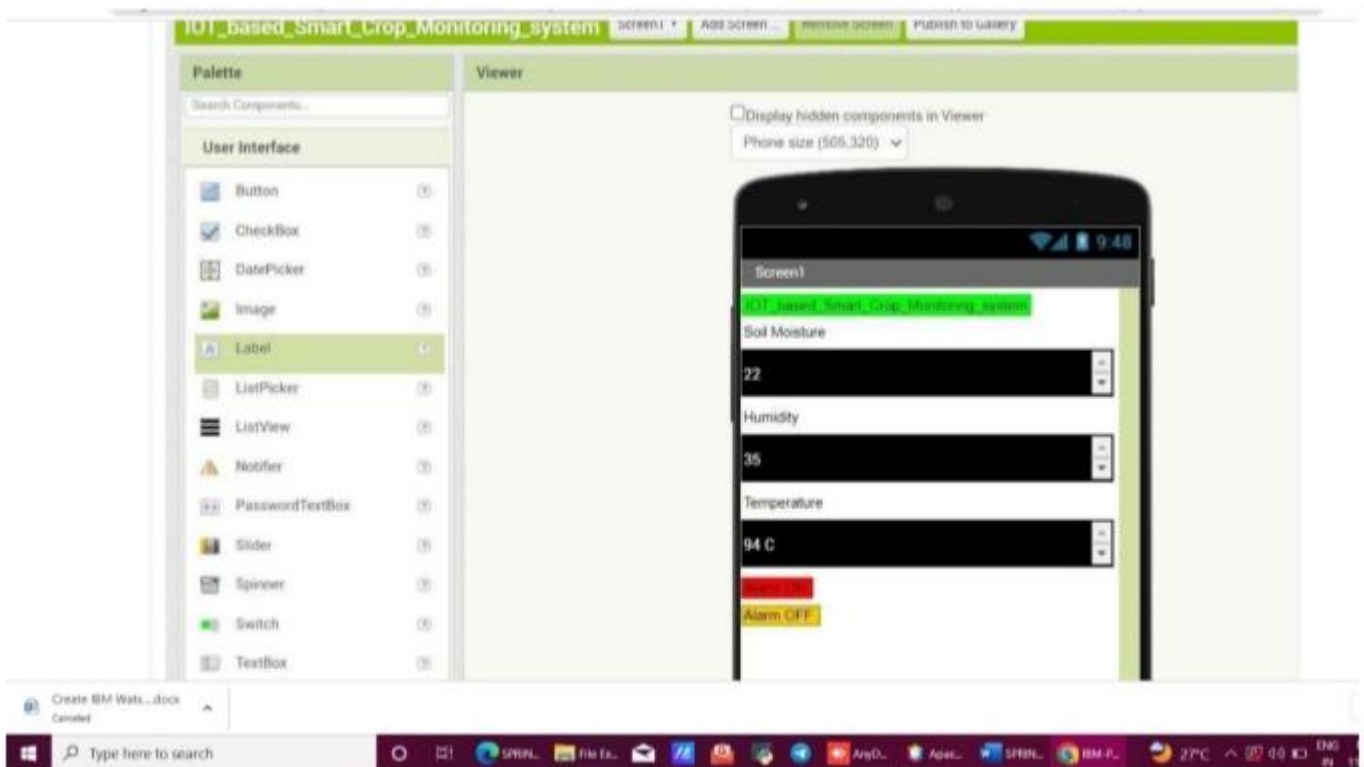


## MIT APP INVENTOR:

STEP 1: MIT APP inventor to design the APP.



STEP 2: Customize the App interface to Display the Values.



## **ADVANTAGES:**

- Farmers can monitor the health of farm animals closely, even if they are physically distant.
- Smart farming systems reduce waste, improve productivity and enable management of a greater number of resources through remote sensing.
- High reliance.
- Enhanced Security.

## **DISADVANTAGES:**

- Farms are located in remote areas and are far from access to the internet.
- A farmer needs to have access to crop data reliably at any time from any location, so connection issues would cause an advanced monitoring system to be useless.
- High Cost
- Equipment needed to implement IoT in agriculture is expensive.

## **APPLICATIONS:**

- Monitoring the crop field with the help of sensors (light, humidity, temperature, soil moisture, etc.)
- Automating the irrigation system
- Soil Moisture Monitoring (including conductivity and pH)