PROJECT REPORT

IOT BASED SMART CROP PROTECTION SYSTEM FOR AGRICULTURE

TEAM ID: PNT2022TMID49700

TEAM LEAD: GRACELIN THANGAM S

TEAM MEMBER 1: LEELARANI M

TEAM MEMBER 2: POUN ESAKKI E

TEAM MEMBER 3: RATHNA PRIYA J

Project Report Format

1. **INTRODUCTION**
   - 1.1 Project Overview
   - 1.2 Purpose
2. **LITERATURE SURVEY**
   - 2.1 Existing problem
   - 2.2 References
   - 2.3 Problem Statement Definition
3. **IDEATION & PROPOSED SOLUTION**
   - 3.1 Empathy Map Canvas
   - 3.2 Ideation & Brainstorming
   - 3.3 Proposed Solution
   - 3.4 Problem Solution fit
4. **REQUIREMENT ANALYSIS**
   - 4.1 Functional requirement
   - 4.2 Non-Functional requirements
5. **PROJECT DESIGN**
   - 5.1 Data Flow Diagrams
   - 5.2 Solution & Technical Architecture
   - 5.3 User Stories
6. **PROJECT PLANNING & SCHEDULING**
   - 6.1 Sprint Planning & Estimation
   - 6.2 Sprint Delivery Schedule
   - 6.3 Reports from JIRA
7. **CODING & SOLUTIONING (Explain the features added in the project along with code)**
   - 7.1 Feature 1
   - 7.2 Feature 2
   - 7.3 Database Schema (if Applicable)
8. **TESTING**
   - 8.1 Test Cases
   - 8.2 User Acceptance Testing
9. **RESULTS**
   - 9.1 Performance Metrics
10. **ADVANTAGES & DISADVANTAGES**
11. **CONCLUSION**
12. **FUTURE SCOPE**
13. **APPENDIX**
    - Source Code
    - GitHub & Project Demo Link

1 INTRODUCTION:

1.1 OVERVIEW

This is a Smart Agriculture System project based on Internet Of Things (IoT), that can measure soil moisture and temperature conditions for agriculture 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.

In this project we have not used any hardware. Instead of real soil and temperature conditions, sensors IBM IoT Simulator is used which can transmit soil moisture temperature as required.

- **Project requirements**: Node-RED, IBM Cloud, IBM Watson IoT, Node.js, IBM Device, IBM IoT Simulator, Python 3.7, Open Weather API platform.

- **Project Deliverables**: Application for IoT based Smart Agriculture System

1.2 PURPOSE

IoT based farming improves the entire agriculture system by monitoring the field in real-time. With the help of IoT in agriculture not only saves the time but also reduces the extravagant use of resources such as water and electricity.

Sometimes due to over or less supply of water in the agricultural field crops may not grow proper. Using IoT supply of water and growth of plants can be satisfied to a greater extent. The flow of water can be controlled from the application.

1.2.1 SCOPE OF WORK

- Create a device in IBM Cloud Account.
- Install Node-RED and configure the nodes that we want to use in the project.
- Create the open weather map account and get the API key and the weather conditions using API key in the Node-RED.
- Create a web application for user interaction for observation and control actions.

2 LITERATURE SURVEY:

2.1 EXISTING PROBLEM

- In agriculture water is needed for the crops for their growth. If the Soil gets dry it is necessary to supply water. But sometime if the farmer doesn't visit the field, it is not possible to know the condition of soil.
- Sometimes over supply of water or less supply of water affects the growth of crops.
- Sometimes if the weather/temperature changes suddenly it is necessary to take certain actions.
- Specific crops grow better in specific conditions, they may get damaged due to bad weather.

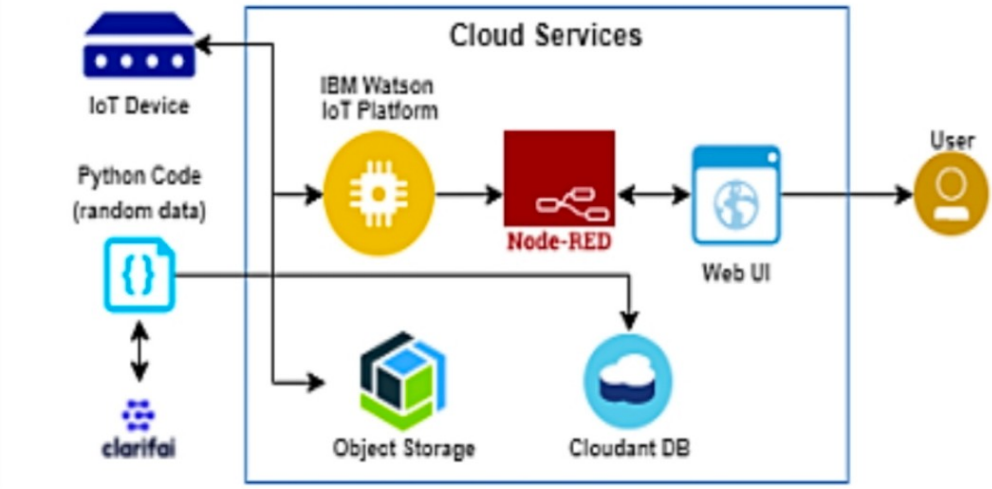
2.1 PROPOSED SOLUTION

- 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 services to 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 and displayed on the application.

- Real time weather conditions can also be known by using different weather APIs from different websites and displayed on our application.

3 THEORITICAL ANALYSIS:

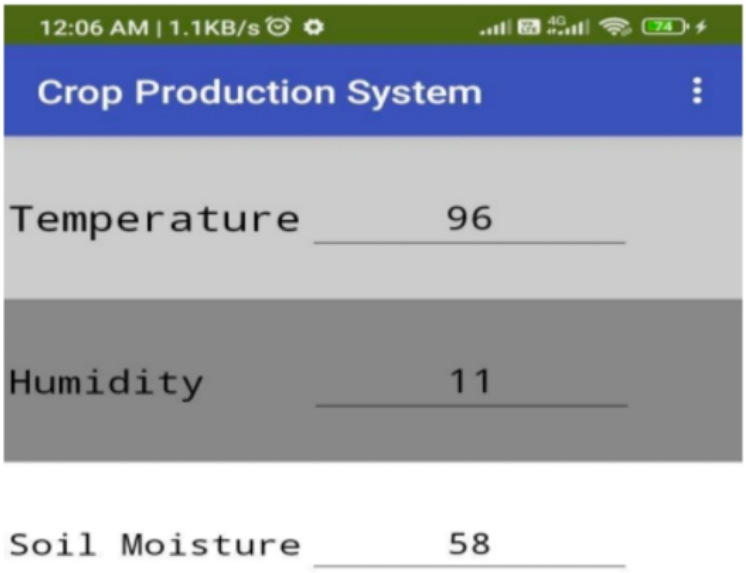
3.1 BLOCK DIAGRAM



3.2 Hardware / Software Designing

1. Create a device in IBM Cloud.
2. Connect the device to IBM Simulator to get the weather conditions.
3. Build Node-RED flow to build a web application to display the weather conditions and control the devices.
4. Get the real time weather condition data from open weather map and integrate it in the Node-RED.
5. Control the working of the web application to the devices by python coding.

4 EXPERIMENTAL INVESTIGATION:

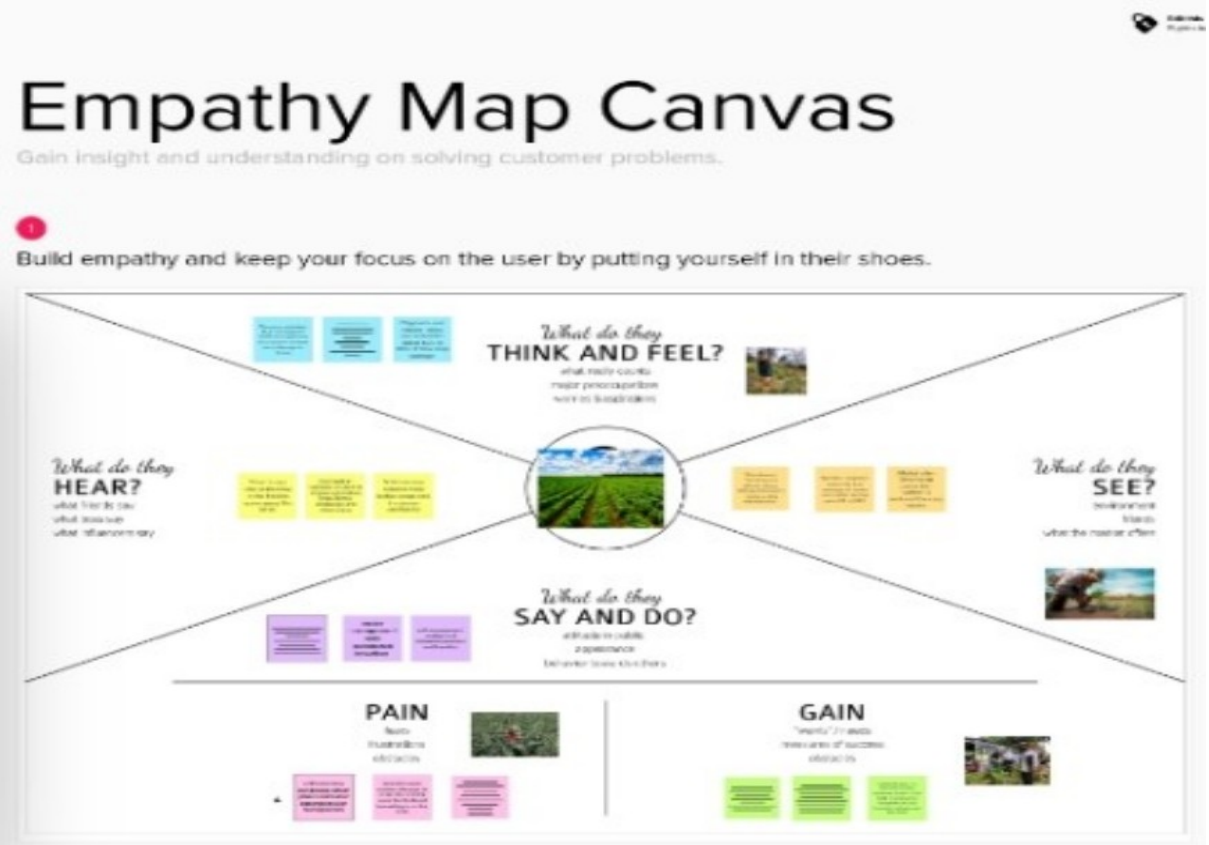


FIG(A) WATSON IOT SIMULATOR

Literature Survey:

AUTHOR NAME	YEAR	TOPIC
Artur Frankiewicz; Rafal Cupek	2013 IECON 2013-39th annual conference of the IEEE industrial electronics society pages:7543-7547	Smart passive infrared sensor-Hardware platform
Discant, A. Rogozan, C. Rusu and A. Bensrhair	2007-30th International Spring Seminar on Electronics Technology (ISSE), Cluj-Napoca, 2007, pp. 100-105. doi: 10.1109/ISSE.2007.4432828 Volume:01 Pages:859-862, DOI:10.1109/ICCSNT.2015.7490876, IEEE Conference Publications.	Sensors for Obstacle Detection
Mustapha, Baharuddin, AladinZayegh, and Rezaul K. Begg.	Artificial Intelligence, Modelling and Simulation (AIMS), 2013 1st International Conference on. IEEE, 2013	Ultrasonic And Infrared Sensors Performance in A Wireless Obstacle Detection System
Padmashree S. Dhake, Sumedha S. Borde	International Journal of Advanced Technology in Engineering and Science, www.ijates.com Volume No.02, Issue No. 03, March 2014.	Embedded Surveillance System Using PIR Sensor
DR. R. Bulli Babu, CH. JonathanSoumith, T. Cherishma Sri Lakshmi & R. Keshav Rao Kluniversity	India Global Journal of Computer Science and Technology: A Hardware & Computation Volume 15 Issue 2 Version 1.0 Year 2015 Type	GSM based Agriculture Monitoring and Controlling System
Dugyala Karthik, R.Ramesh Babu	International Journal of Advanced Information Science and Technology (IJAIST), ISSN: 2319:2682 Vol.6, No.11, November 2017	Smart Crop Protection System with Image Capture over IOT

Empathy Map Convas

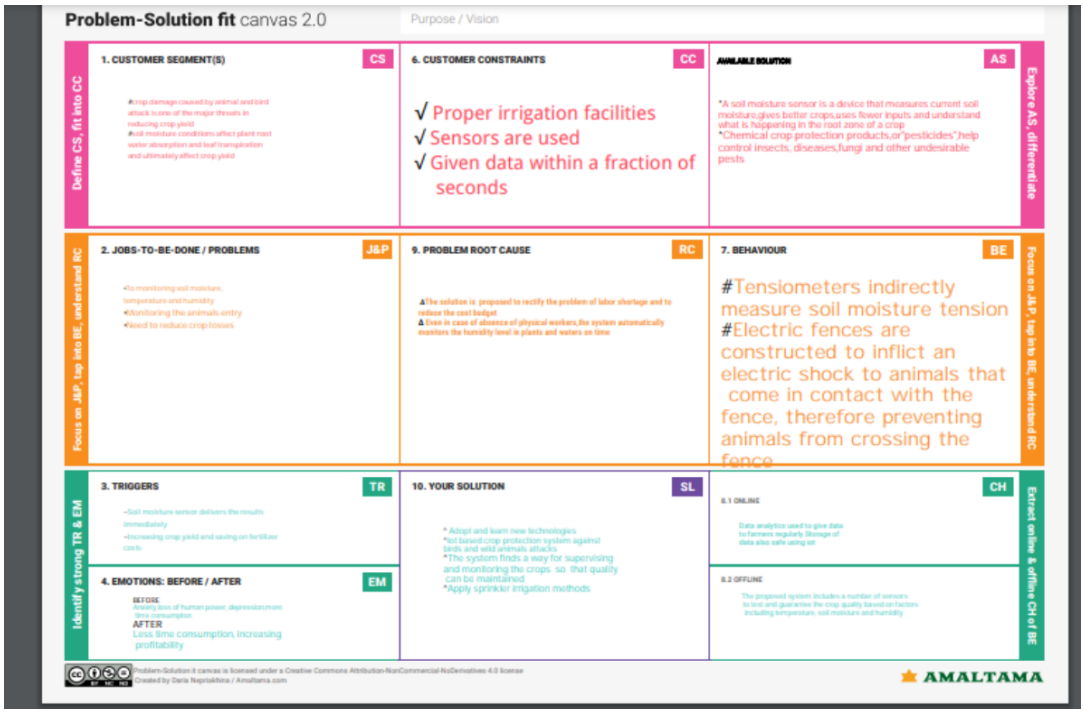


Idention& Brain Storming

Proposed Solution

S. No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	• Crop damage caused by animal and birds attacks is one of the major threats in reducing crop yield. • Soil moisture conditions affect plant root water absorption and leaf transpiration.
2.	Idea / Solution description	• Put an electric fence around the planting place. • A layer of organic matter like straw covers the bare ground between plants and helps to maintain soil moisture.
3.	Novelty /Uniqueness	• IOT Based crop protection system against birds and wild animal attacks • Sprinkler irrigation is a method of applying irrigation water which is similar to natural rainfall. • Soil moisture (SMM) devices provide information about the water status of soil. knowing the soil water status can help you plan when to irrigate and how much water to apply.
4.	Social Impact / Customer Satisfaction	• Sprinkler irrigation for irrigating crops where the plant population per unit area is very high, this system is suitable. • Soil moisture includes increasing crop yields, saving water.
5.	Business Model (Revenue Model)	• Soil moisture sensors aid good irrigation management. Good irrigation management gives better crops, uses fewer inputs, and increases profitability.
6.	Scalability of the Solution	• They are simple to use and easy to install. • In addition to agricultural use, they can also be used for pollution and global warming. • They are equipped with wireless chip so that they can be remotely controlled.

Problem Solution Fit



Requirement Analysis

Project Design phase-II

Solution Requirements (Functional and Non functional)

Date	15 October 2022
Team ID	PNT2022TMID49700
Project Name	Project: Iot based smart crop protection system
Maximum Marks	4 marks

Functional Requirements:

Following are the functional requirements of the proposed solution

FR NO.	Functional Requirement(Epic)	Sub Requirements(story/ sub-Task)
FR-1	User Requirements	Control the animals and birds Monitoring soil moisture, temperature and humidity Automatic sprinkler irrigation system
FR-2	User Registration	Download the app Registration through Gmail Create an account Follow the instructions
FR-3	User Confirmation	Confirmation via Email Confirmation via OTP Confirmation via phone
FR-4	User Delivery	Product will be delivered to registered addresses

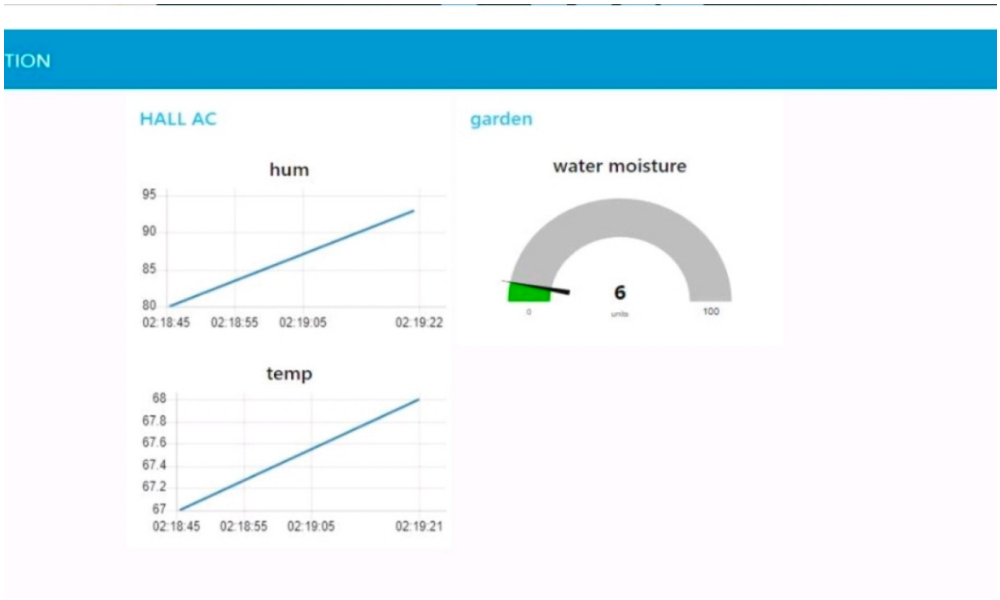
		▯ Free installation and 2 years warranty
FR-5	User Payments	▯ Pay via UPI/Net Banking ▯ Pay via Debit/ Credit/ ATM Card ▯ Pay via Cash on delivery
FR-6	Product feed back	▯ Through phone calls ▯ Through Google forms ▯ Through Email

Non- Functional Requirements:

Following are the non- Functional requirements of the proposed solution

NFR NO.	Non-Functional Requirement	Description
NFR-1	Usability	Have an easily to understand guidebook. simpler to use the product is easy to use even by farmers who are illiterate
NFR-2	Security	Applications security requires two-step authorization. Password and passkeys will be given out based on the needs of the users.
NFR-3	Reliability	This project will help farmers in protecting their fields and save them from significant financial losses. Hardware needs to be checked and maintained regularly.
NFR-4	Performance	IOT devices and sensors are used to indicate the farmer by a massage when animals try to enter into the field and also we use an SD card module that helps to store a specified sounds to scare the animals.
NFR-5	Availability	Are available soil moisture, temperature, humidity and irrigation value.
NFR-6	Scalability	Since this system uses computer vision techniques integrated with IBM cloudant services helps efficiently to retrieve images in large scale thus improving scalability.

project Design



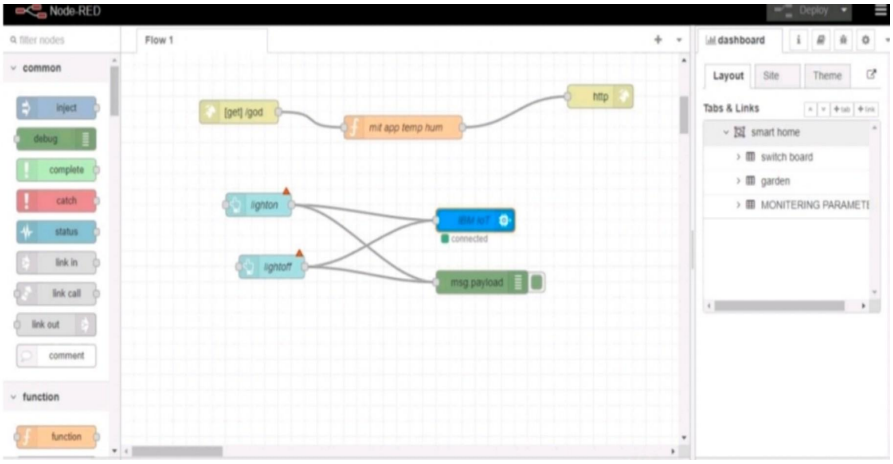
FIG(C) DEVICE CONTROL ACTION

In this project we send the weather data through IoT Simulator shown in fig(a) 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 show in fig (b1 & b2). Web Application is build using Node-RED. We have created 2 tabs:

1. IoT Smart Agriculture.
2. Graphical Representation.

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 in fig(c).

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 from open weather map:

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

6 RESULT:

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

7 ADVANTAGES & DISADVANTAGES:

7.1 ADVANTAGES

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

7.1 DISADVANTAGES:

- Smart Agriculture requires internet connectivity continuously, but rural parts cannot fulfill this requirement.
- Any faults in the sensors can cause great loss in the agriculture, due to wrong records and the actions of automated processes. • IoT devices need much money to implement.

8 APPLICATIONS:

- Precision Farming that is farming processes can be made more controlled and accurate.
- Live monitoring can be done of all the processes and the conditions on the agricultural field.
- All the controls can be made just on the click. • Quality can be maintained.

9 CONCLUSION:

A IoT Web Application is built for smart agricultural system using 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 and reducing the usage of extravagant resources like electricity and water IoT can be implemented in most of the places.

A. SOURCE CODE

```
import random
import ibmiotf.application
import ibmiotf.device
from time import sleep
import sys
#IBM Watson Device Credentials.
organization = "3xaptt"
deviceType = "NodeMCU"
deviceId = "12345"
authMethod = "use-token-auth"
authToken = "12345678"
def myCommandCallback(cmd):
print("Command received: %s" % cmd.data['command'])
status=cmd.data['command']
if status=="sprinkler_on":
print ("sprinkler is ON")
else :
print ("sprinkler is OFF")
#print(cmd)
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()
#Connecting to IBM watson.
deviceCli.connect()
while True:
#Getting values from sensors.
temp_sensor = round( random.uniform(0,80),2)
PH_sensor = round(random.uniform(1,14),3)
camera = ["Detected","Not Detected","Not Detected","
Not Detected","Not Detected","Not Detected",]
camera_reading = random.choice(camera)
flame = ["Detected","Not Detected","Not Detected","
Not Detected","Not Detected","Not Detected",]
flame_reading = random.choice(flame)
moist_level = round(random.uniform(0,100),2)
water_level = round(random.uniform(0,30),2)
#storing the sensor data to send in json format to
cloud.temp_data = { 'Temperature' :
temp_sensor }
PH_data = { 'PH Level' : PH_sensor }camera_data =
{ 'Animal attack' : camera_reading}
flame_data = { 'Flame' : flame_reading }
```

```
moist_data = { 'Moisture Level' : moist_level}
water_data = { 'Water Level' : water_level}
# publishing Sensor data to IBM Watson for every 5-10
seconds.
success = deviceCli.publishEvent("Temperature sensor",
"json", temp_data, qos=0)
sleep(1)
if success:
print (" .....publish ok..... ")
print ("Published Temperature = %s C" % temp_sensor, "to
IBM Watson")
success = deviceCli.publishEvent("PH sensor", "json",
PH_data, qos=0)
sleep(1)
if success:
print ("Published PH Level = %s" % PH_sensor, "to IBM
Watson")
success = deviceCli.publishEvent("camera", "json", camera_data
, qos=0)
sleep(1)
if success:
```

```
print ("Published Animal attack %s " % camera_reading, "to
 IBM Watson")
success = deviceCli.publishEvent("Flame sensor", "json",
flame_data, qos=0)
sleep(1)
if success:
print ("Published Flame %s " % flame_reading, "to IBM
Watson")
success = deviceCli.publishEvent("Moisture sensor", "json",
moist_data, qos=0)
sleep(1)
if success:
print ("Published Moisture Level = %s " % moist_level, "to
IBM Watson")
success = deviceCli.publishEvent("Water sensor", "json",
water_data, qos=0)
sleep(1)
if success:
print ("Published Water Level = %s cm" % water_level, "to
IBM Watson")
print ("")
#Automation to control sprinklers by present temperature
an to send alert message to IBM Watson.
```

```
if (temp_sensor > 35):
print("sprinkler-1 is ON")
success = deviceCli.publishEvent("Alert1", "json",{ 'alert1' :
"Temperature(%s) is high, sprinkerlers are turned ON" %temp_
sensor }
 , qos=0)
sleep(1)
if success:
print( 'Published alert1 : ', "Temperature(%s) is high,
 sprinkerlers are turned ON" %temp_sensor,"to IBM Watson")
print("")
else:
print("sprinkler-1 is OFF")
print("")
#To send alert message if farmer uses the unsafe fertilizer to
 crops.
if (PH_sensor > 7.5 or PH_sensor < 5.5):
success = deviceCli.publishEvent("Alert2", "json",{ 'alert2' : "
Fertilizer PH level(%s) is not safe,use other fertilizer" %PH_
sensor } ,
qos=0)
```

```
sleep(1)
if success:
print('Published alert2 : ' , "Fertilizer PH level(%s) is
not
safe,use other fertilizer" %PH_sensor,"to IBM Watson
")
print ("")
# To send alert message to farmer that animal
attack on crops.
if (camera_reading == "Detected"):
success = deviceCli.publishEvent("Alert3", "json", { '
alert3' : "Animal attack on crops detected" }, qos=0)
sleep(1)
if success:
print('Published alert3 : ' , "Animal attack on crops
 detected","to IBM Watson","to IBM Watson")
print("")
#To send alert message if flame detected on crop
land and turn ON the splinkers to take immediate
action.
if (flame_reading == "Detected"):
print("sprinkler-2 is ON")
success = deviceCli.publishEvent("Alert4", "json", { '
alert4' :
"Flame is detected crops are in danger,sprinklers
turned ON" }, qos=0)
```

```
print( 'Published alert4 : ' , "Flame is detected crops are in

danger,sprinklers turned ON","to IBM Watson")
print("")
else:
print("sprinkler-2 is OFF")
print("")
#To send alert message if Moisture level is LOW and to
Turn ON Motor-1 for irrigation.
if (moist_level < 20):
print("Motor-1 is ON")
success = deviceCli.publishEvent("Alert5", "json", { 'alert5'
: "Moisture level(%s) is low, Irrigation started" %moist_
level }, qos=0)
sleep(1)
if success:
print('Published alert5 : ' , "Moisture level(%s) is low,
Irrigation started" %moist_level,"to IBM Watson" )
print("")
else:
print("Motor-1 is OFF")
print("")
```

```
#To send alert message if Water level is HIGH and to Turn
 ON Motor-2 to take water out.
if (water_level > 20):
print("Motor-2 is ON")
success = deviceCli.publishEvent("Alert6", "json", { 'alert6' :
 "Water level(%s) is high, so motor is ON to take water
out " %water_level }, qos=0)
sleep(1)
if success:
print('Published alert6 : ' , "water level(%s) is high, so
motor is ON to take water out " %water_level,"to IBM
Watson" )
print("")
else:
print("Motor-2 of OFF")
print("")
#command recived by farmerdeviceCli.commandCallback =
myCommandCallback
# Disconnect the device and application from the cloud
deviceCli.disconnect()
```

project Demo Link

https://youtu.be/4Qhbwwd9Mv8