IOT BASED SMART CROP PROTECTION SYSTEM FOR AGRICULTURE

Team ID: PNT2022TMID53952

TEAM MEMBERS

JANAKI M,

JANANI R P,

ARUNA A,

JEBASHALOMIE IMMANUEL.

INTRODUCTION

Crops in farms are many times ravaged by local animals like buffaloes, cows, goats, birds etc. this leads to huge losses for the farmers. It is not possible for farmers to barricade entire fields or stay on field 24 hours and guard it. So here we propose automatic crop protection system from animals. This is a microcontroller based system using PIC family microcontroller. The microcontroller will now sound an alarm to woo the animal away from the field as well as sends SMS to the farmer so that he may about the issue and come to the spot in case the animal don't turn away by the alarm. This ensures complete safety of crop from animals thus protecting farmers loss. Our main purpose of the project is to develop intruder alert to the farm, to avoid losses due to animal and fire. These intruder alert and protect the crop that damaging that indirectly increase yield of the crop. The develop system will not harmful and injurious to animal as well as human beings. Theme of project is to design a intelligent security system for farm protecting by using embedded system. The existing system, mainly provide the surveillance functionality. Also these system don't provide protection from wild animals, especially in such an application area. They also need to take actions based on the type of animal that tries to enter the area, as different methods are adopted to prevent different animals from entering restricted areas. The other commonly used method by farmer in order to prevent the crop vandalization by animals include building physical barriers, use of electric fences and manual surveillance and various such exhaustive and dangerous method.

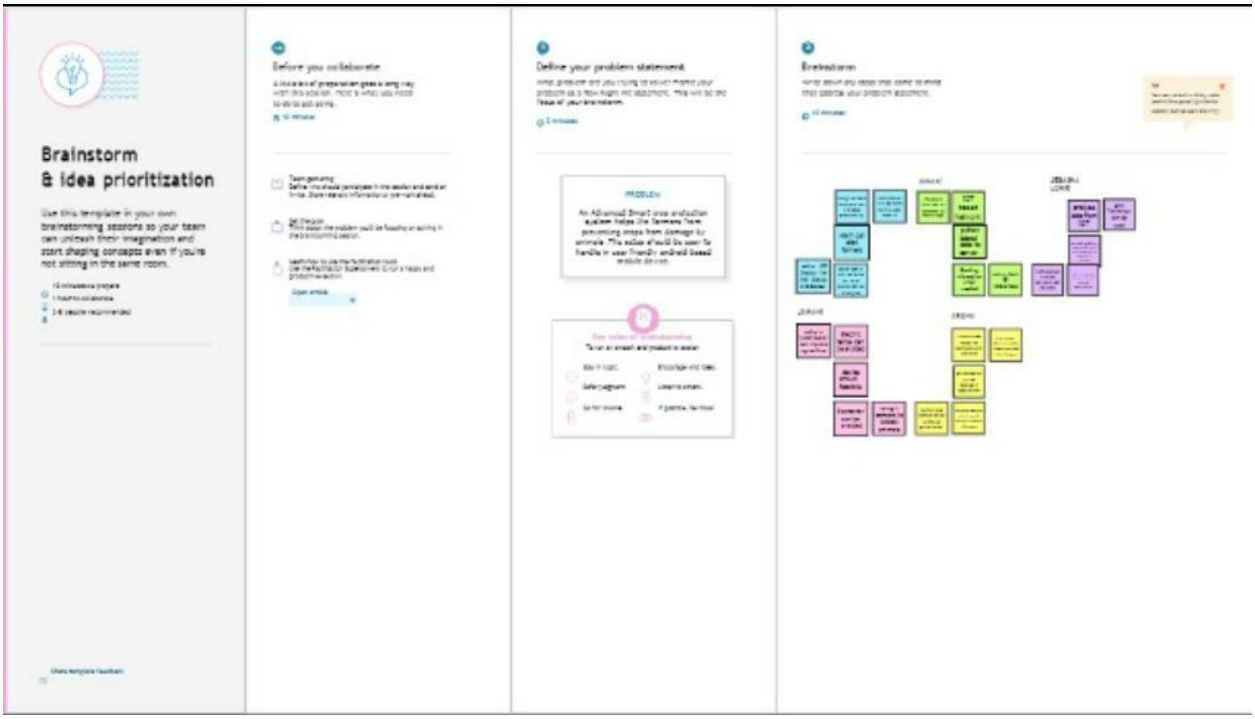
PROBLEM STATEMENT :

In the world, economy of many countries are dependent upon the agriculture. In spite of economic development agriculture is the backbone of the economy. Crops in forms are many times ravaged by local animals like buffaloes, cows, goats, birds and fire etc. this leads to huge loss for the farmers. It is not possible for farmers to blockade to entire fields or stay 24 hours and guard it. Agriculture meets food requirements of the people and produces several raw materials for industries. But because of animal interference and fire in agricultural lands, there will be huge loss of crops. Crops will be totally getting destroyed.

IDEATION PHASE AND PROPOSED SOLUTION

EMPATHY MAP CANVAS:

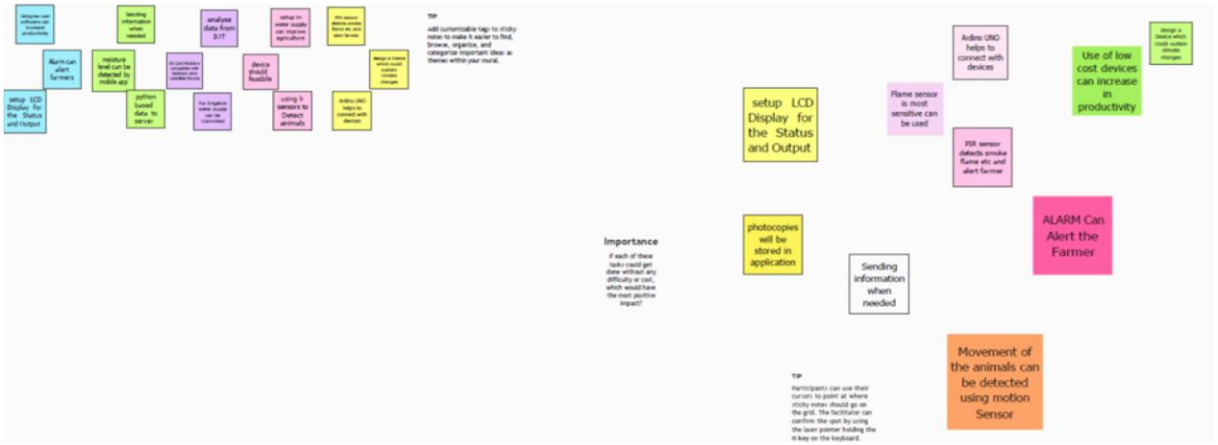
IDEATION PHASE AND BRAINSTORMING:



Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes



Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes

PROPOSED SOLUTION:

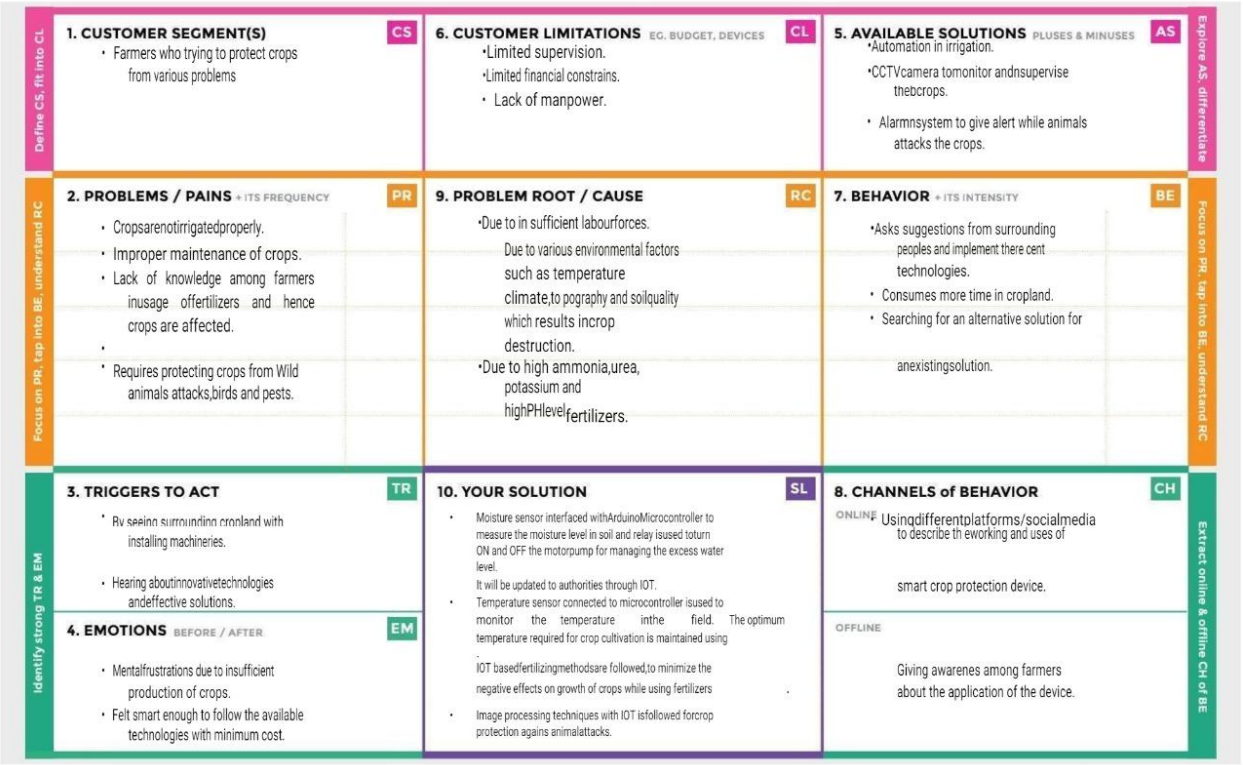
S.NO.	Parameter	Description
1.	Problem Statement. (Problem to be solved)	✓ Crops are not irrigated properly due to insufficient labour forces. ✓ Improper maintenance of crops against various environmental factors such as temperature climate, topography and soil quantity which results in crop destruction. ✓ Requires protecting crops from wild animals attacks birds and pests.
2.	Idea /Solution Description.	✓ Moisture sensor is interfaced with Arduino Microcontroller to measure the moisture level in soil and relay is used to turn ON & OFF the motor pump for managing the excess water level. It will be updated to authorities through IOT. ✓ Temperature sensor connected to microcontroller is used to monitor the temperature in the field. ✓ Image processing techniques with IOT is followed for crop protection against animal attack.
3.	Novelty / Uniqueness.	✓ Automatic crop maintenance and protection using embedded and IOT Technology.
4.	Social Impact / Customer satisfaction.	✓ This proposed system provides many facilities which helps the farmers to maintain the crop field without much loss.
5.	Business Model (Revenue Model).	✓This prototype can be developed as product with minimum cost with high performance.
6.	Scalability of the solution	✓This can be developed to a scalable product by using solution sensors and transmitting the data through Wireless Sensor Network and Analysing the data in cloud and operation is performed using robots.

PROBLEM SOLUTION FIT:

ProjectTitle:IOT Based Smart Crop Protection System for Agriculture

Project DesignPhase-I.SolutionFit

Team ID: PNT2022TMID53952



REQUIREMENT ANALYSIS

SOLUTION REQUIREMENT:

FUNCTIONAL REQUIREMENTS :

- **Following are the functional requirements of the proposed solution.**

S.NO.	Functional Requirement.	Sub Requirement.
1.	User Visibility	Sense animals nearing the crop field & sounds alarm to woo them away as well as sends SMS to farmer using cloud service.
2.	User Reception	The Data like values of Temperature, Humidity, Soil moisture Sensors are received via SMS.
3.	User Understanding	Based on the sensor data value to get the information about the present of farming land.
4.	User Action	The User needs take action like destruction of crop residues, deep plowing, crop rotation, fertilizers, strip cropping, scheduled planting operations.

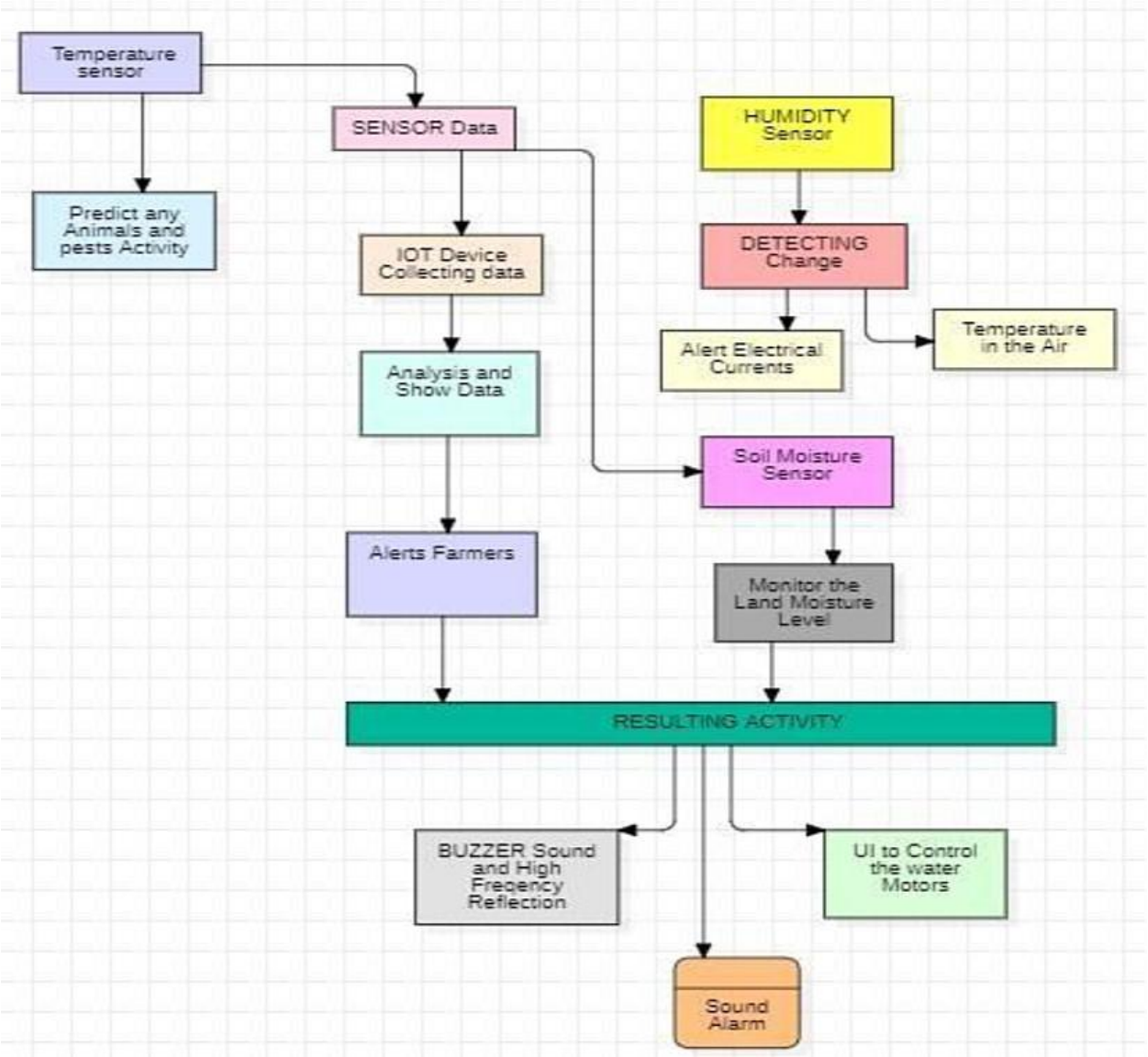
Following are the non functional requirements of the proposed solution.

NON FUNCTINAL REQUIREMENT:

S.NO.	Non-Functional Requirement.	Description.
1.	Usability	Mobile Support Users must be able to interact in the same roles & tasks on computers & mobile devices where practical, given mobile capabilities.
2.	Security	Data requires secure access to must register and communicate securely on devices and authorized users of the system who exchange information must be able to do.
3.	Reliability	It has a capacity to recognize the disturbance near the field and doesn't give a false caution signal.
4.	Performance	Must provide acceptable response times to users regardless of the volume of data that is stored and the analytics that occurs in background. Bidirectional, near real-time communications must be supported. This requirement is related to the requirement to support industrial and device protocols at the edge.
5.	Availability	IOT Solutions and domains demand highly available systems for 24 x 7 operations. Isn't a critical production application, which means that operations or productiondon't go down if the IOT solution is down.
6.	Scalability	System must handle expanding load & data retention needs that are based on the upscaling of the solution scope, such as extra manufacturing facilities and extra buildings.

PROJECT DESIGN

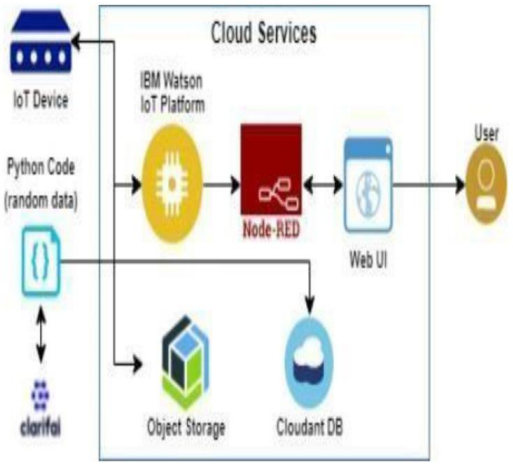
DATA FLOW DIAGRAM:



SOLUTION AND TECHNICAL ARCHITECTURE:

SPRINT PLANNING AND ESTIMATION

TECHNICAL ARCHITECTURE:



BLOCK DIAGRAM:

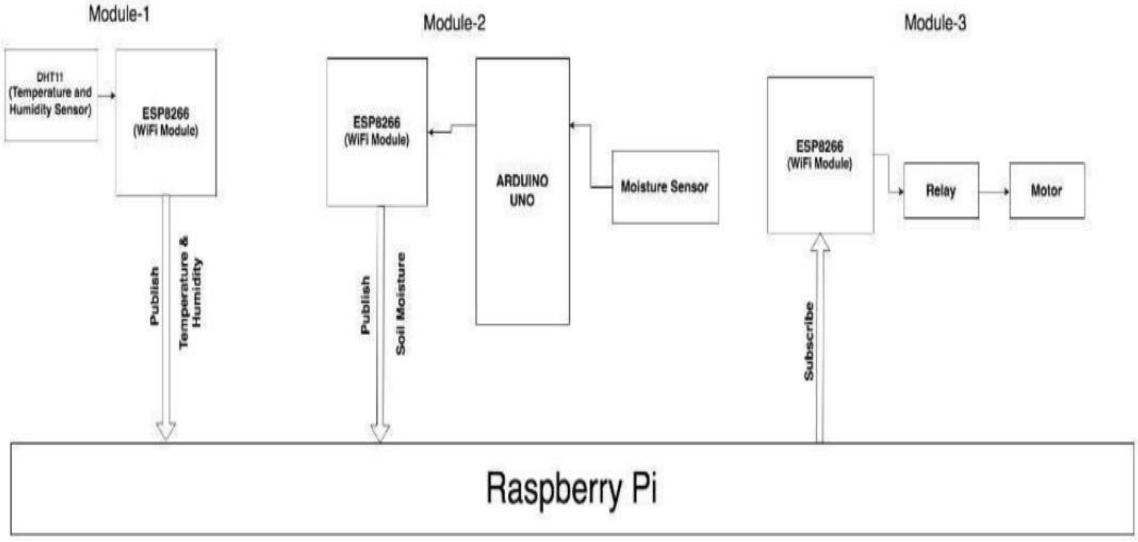


Table-1: Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g., Mobile Application	HTML, CSS, JavaScript / Angular JS / Node Red.
2.	Application Logic-1	Logic for a process in the application	Java / Python
3.	Application Logic-2	Logic for a process in the application	IBM Watson STT service
4.	Application Logic-3	Logic for a process in the application	IBM Watson Assistant
5.	Database	Data Type, Configurations etc.	MySQL, NoSQL, etc.
6.	Cloud Database	Database Service on Cloud	IBM DB2.
7.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service or Local Filesystem
8.	External API-1	Purpose of External API used in the application	IBM Weather API, etc.
9.	IoT Model	Purpose of IoT Model is for integrating the sensors with a user interface.	IBM IoT Platform
10.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration :	Local, Cloud Foundry, Kubernetes, etc.

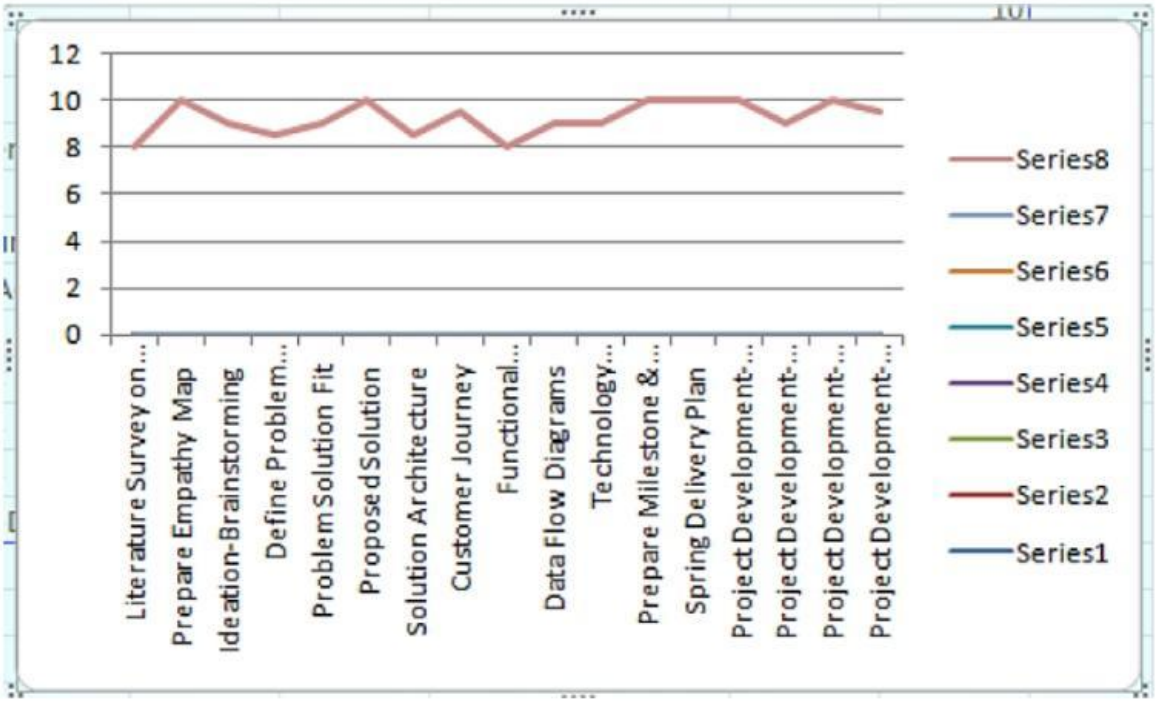
PROJECT PLANNINGAND SCHEDULING

SPRINT PLANNINGAND ESTIMATION:

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

Velocity:
Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity iteration unit (story points per day)

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$



CODING AND SOLUTIONING

FEATURE-1

```
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 83ddcfb774c54cfd81d7a67ba69a0678'),)
COS_ENDPOINT = "https://s3.jp-tok.cloud-object-storage.appdomain.cloud"
COS_API_KEY_ID = "kn05el2QeCyawCFMRytUXLFirKVxw8v5HAIRvDKsIHmu"
COS_AUTH_ENDPOINT = "https://iam.cloud.ibm.com/identity/token"
```

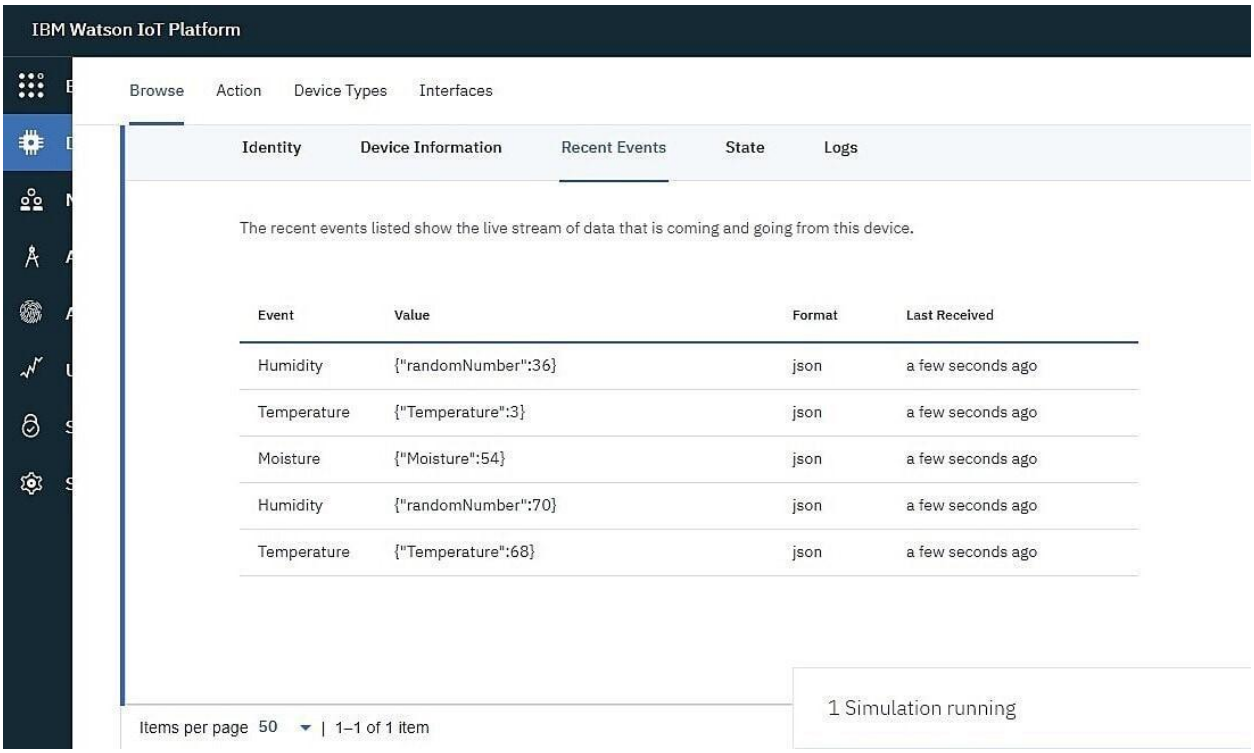
```
COS_RESOURCE_CRN = "crn:v1:bluemix:public:cloudantnosqldb:eu-gb:a/98d92dfd0ccf4f32a116d3d0fe24e15c:02d1fcad-1310-4403-93a6-a0eabc4c768b::"

clientdb = Cloudant("apikey-v2-d8mn8ful7bxv3pw2cq0o1p1d8z3icznh8qu8y2xsv5", "400eef0a90d31fd7fa41c9dd0a2baa4b", url="https://cbf0b64e-c2d3-4404-be21-36565dc150b9-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(commamd=="lighton"):
        print('lighton')
    elif(command=="lightoff"):
        print('lightoff')
    elif(command=="motoron"):
        print('motoron')
    elif(command=="motoroff"):
        print('motoroff')

myConfig = {
    "identity": {
        "orgId": "tw9ckq",
```

```
        "typeId": "jade",
        "deviceId": "7010"
    },
    "auth": {
        "token": "9944893843"
    }
}

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



database_name = "sample1"
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 = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    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='a6100c6f4fb74e79ad8b57b1db2f0235',
        inputs=[resources_pb2.Input(data=resources_pb2.Data(image=resources_pb2.Image(base64=file_bytes))
        )])
    response = stub.PostModelOutputs(request, metadata=metadata)
    print(response)
    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.inwrite(picname+'.jpg',frame)
                multi_part_upload('Jade', picname+'.jpg', picname+'.jpg')
                json_document={"link":COS_ENDPOINT+'/'+'Jade'+'/'+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", data=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()
```



Features

Output: Digital pulse high (3V) when triggered (mo on detected) digital low when idle (no mo on detected). Pulse lengths are determined by resistors and capacitors on the PCB and differ from sensor to sensor. Power supply: 5V-12V input voltage for most modules (they have a 3.3V regulator), but 5V is ideal in case the regulator has different specs.

BUZZER

Specifications:

- RatedVoltage : 6V DC
- Opera ng Voltage : 4 to 8V DC
- Rated Current*: ≤30mA

- SoundOutput at 10cm* : ≥85dB
- Resonant Frequency : 2300 ±300Hz

- Tone: Con nuous A buzzer is a loud noise maker.

Most modern ones are civil defense or air- raid sirens, tornado sirens, or the sirens on emergency service vehiclessuch as ambulances, police cars and fire trucks. There are two general types, pneuma c and electronic.

FEATURE-2:

i. Goodsensi vity to Combus ble gas in wide range .
ii. Highsensi vity to LPG, Propane and Hydrogen .
iii. Longlife and low cost.
iv. Simpledrive circuit.

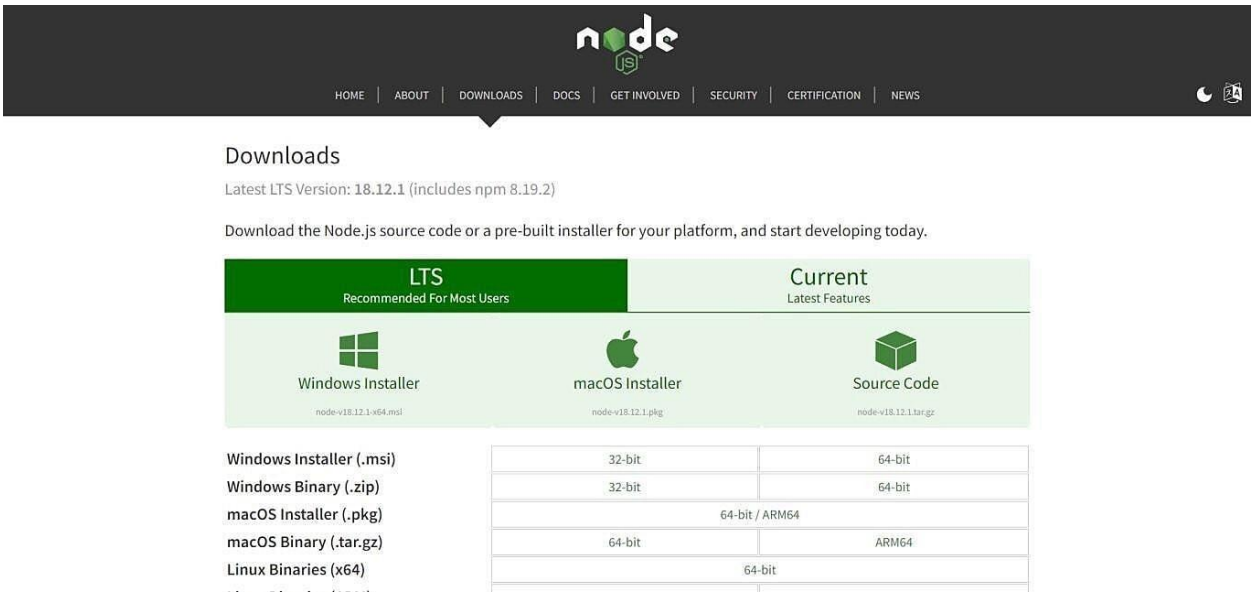
TESTING

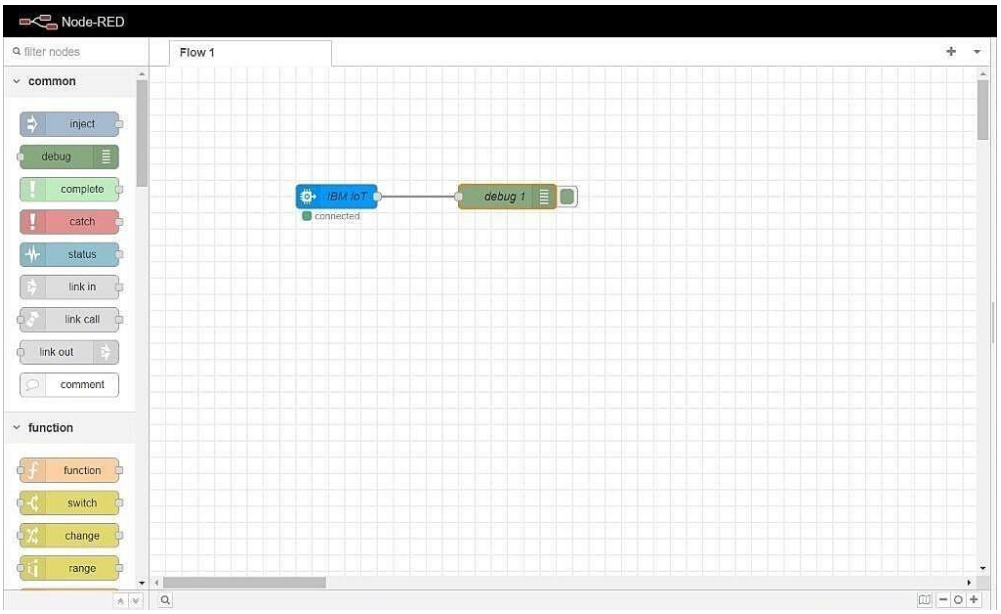
TEST CASES:

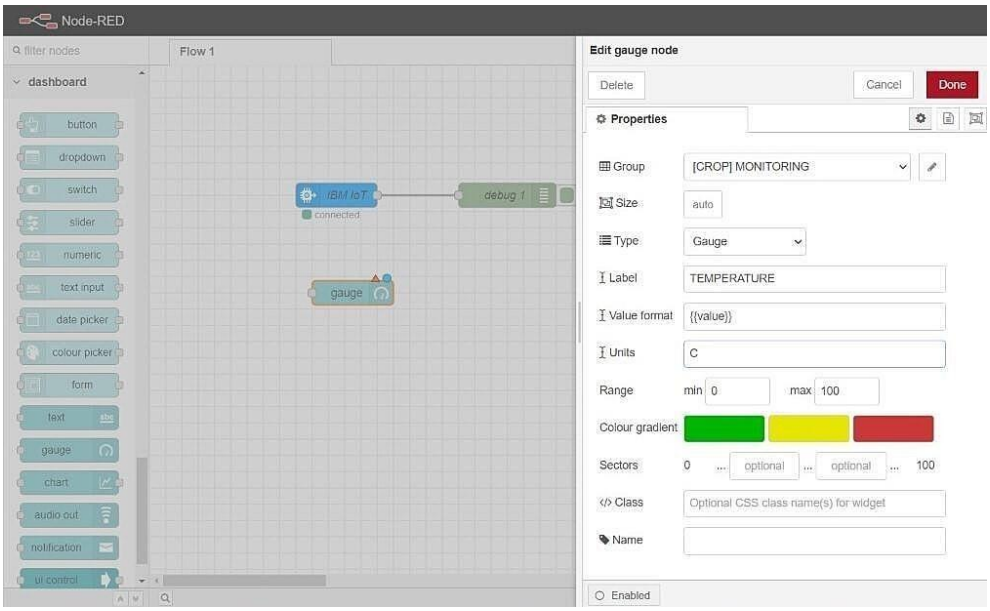
sno	parameter	Values	Screenshot
1	Model summary	-	
2	accuracy	Training accuracy- 95% Validation accuracy- 72%	

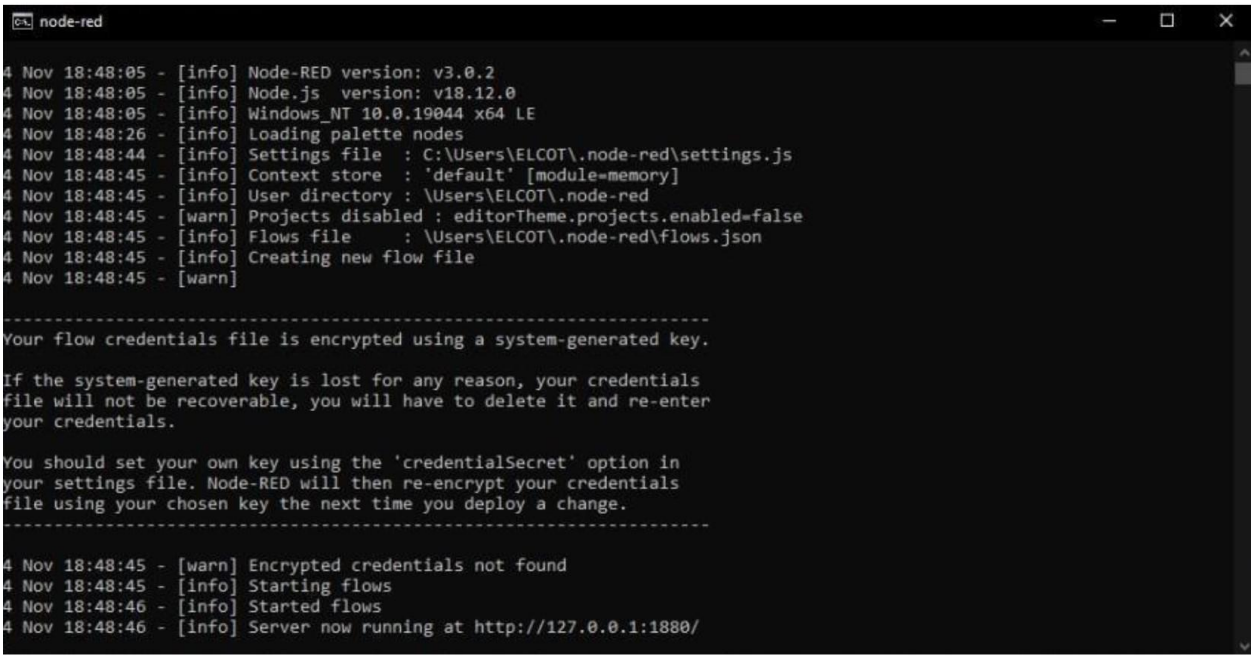
3	Confidence score	Class detected-80% Confidence score-80%	

User Acceptance Testing:




Node-RED
filter nodes
Flow 1
common
inject
debug
complete
catch
status
link in
link call
link out
comment
function
function
switch
change
range
IBM IoT
connected
debug 1


Node-RED
filter nodes
Flow 1
dashboard
button
dropdown
switch
slider
numeric
text input
date picker
colour picker
form
text
gauge
chart
audio out
notification
ui control
IBM IoT
connected
debug 1
gauge
Edit gauge node
Delete
Cancel
Done
Properties
Group
[CROP] MONITORING
Size
auto
Type
Gauge
Label
TEMPERATURE
Value format
{{value}}
Units
C
Range
min 0
max 100
Colour gradient
Sectors
0
optional
optional
100
Class
Optional CSS class name(s) for widget
Name
Enabled



Result:

The problem of crop vandalization by wild animals and fire has become a major social problem in current time.

It requires urgent attention as no effective solution exists till date for this problem. Thus this project carries a great social relevance as it aims to address this problem. This project will help farmers in protecting their orchards and fields and save them from significant financial losses and will save them from the unproductive efforts that they endure for the protection their fields. This will also help them in achieving better crop yields thus leading to their economic wellbeing.

CONCLUSION:

The project developed will help people in protecting the crops without much hard practices.

FUTURE SCOPE:

In the future, there will be very large scope, this project can be made based on Image processing in which wild animal and fire can be detected by cameras and if it comes towards farm then system will be directly activated through wireless networks. Wild animals can also be detected by using wireless networks such as laser wireless sensors and by sensing this laser or sensor's security system will be activated.

REFERENCES:

i. Mr.Pranav shitap, Mr.Jayesh redij, Mr.Shikhar Singh, Mr.Durvesh Zagade, Dr. Sharada Chougule. Department of ELECTRONICS AND TELECOMMUNICATION ENGINEERING, Finolex Academy of Management and technology, ratangiri, India. N.Penchalaiah, D.Pavithra, B.Bhargavi, D.P.Madhurai,

ii K.EliyasShaik,S.Md.sohaib.Assitant Professor, Department of CSE,AITS, Rajampet,India UG Student, Department of CSE,AITS,Rajampet, India.

iii. Mr.P.Venkateswara Rao, Mr.Ch Shiva Krishna ,MR M Samba Siva ReddyLBRCE,LBRCE,LBRCE.

iv. Mohit Korche,Sarthak Tokse, ShubhamShirbhate, Vaibhav Thakre,S. P. Jolhe(HOD). Students , Final Year,Dept.of Electrical engineering,Government College of engineering,Nagpur head of dept.,Electrical engineering,Government College of engineering,Nagpur.

APPENDIX

SOURCE CODE

```
import me importsys import ibmio .applica on # to installpip
install ibmio impor bmio .device


# Provide your IBM Watson Device Creden als organiza on = "8gyz7t" #
replace the ORG ID deviceType = "weather_monitor" # replace the Device
type deviceId = "b827ebd607b5" # replace Device ID authMethod = "token"
authToken = "LWVpQPaVQ166HWN48f" # Replace the authtoken
def myCommandCallback(cmd): # func on for Callbackif


    cm.data['command'] == 'motoron':


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


else:

if 'interval' not in cmd.data: print("Error - command is missing

  requiredinforma on: 'interval'")


  interval = cmd.data['interval']


elif cmd.command == "print": if 'message' not in cmd.data: print("Error -
commandis missing requiredinforma on: 'message'")
else:output = cmd.data['message'] print(output)
```

```
try:

      deviceOp ons = {"org": organiza on, "type": deviceType, "id": deviceId, "authmethod":
   authMethod,
                          "auth-token": authToken}              deviceCli
= ibmio .device.Client(deviceOp ons) #
.....

exceptExcep on as e: print("Caught excep on connec ng device: %s"
      % str(e)) sys.exit()

   # Connect and send a datapoint "hello" with value "world" into the cloud as an event of type "gree ng"
   10 mes
deviceCli.connect()

while True: deviceCli.commandCallback =
      myCommandCallback

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

SENSOR.PY

```
   import me import
   sysimport    ibmio
   .applica on impor
   bmio .device
import random

   # Provide your IBM Watson Device Creden als organiza on = "8gyz7t" #
   replace the ORG ID deviceType = "weather_monitor" # replace the Device
   type deviceId = "b827ebd607b5" # replace Device ID authMethod = "token"
   authToken = "LWVpQPaVQ166HWN48f" # Replace the authtoken
```

```
def myCommandCallback(cmd):

      print("Command received: %s" % cmd.data['command'])
  print(cmd)

try:
            deviceOp ons = {"org": organiza on, "type": deviceType, "id": deviceId,
  "auth-method": authMethod, "auth-token": authToken}
  deviceCli = ibmio .device.Client(deviceOp ons)
            #.....

exceptExcep on as e:
          print("Caught excep on connec ng device: %s" % str(e)) sys.exit()

  # Connect and send a datapoint "hello" with value "world" into the cloud as an event of type "gree ng"
  10 mes
deviceCli.connect()

while True:
        temp=random.randint(0,1
  00)
  pulse=random.randint(0,100)
        soil=random.randint(0,100)

        data = { 'temp' : temp, 'pulse': pulse ,'soil':soil}
        #print data          def
  myOnPublishCallback():
            print ("Published Temperature = %s C" % temp, "Humidity = %s %%" % pulse,"Soil
  Moisture = %s %%" % soil,"to IBM Watson")

        success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0,
  on_publish=myOnPublishCallback)                    if not success:
  print("Not connected to
        IoTF") me.sleep(1)

        deviceCli.commandCallback = myCommandCallback
```

Disconnect the device and applica on from the cloud deviceCli.disconnect()

Node-RED FLOW :

[
{
"id":"625574ead9839b34
",
"type":"ibmiotout", "z":"630c8601c5ac3295",
"authen ca on":"apiKey",
"apiKey":"ef745d48e395ccc0",
"outputType":"cmd",
"deviceId":"b827ebd607b5",
"deviceType":"weather_monitor",
"eventCommandType":"data",
"format":"json",
"data":"data",
"qos":0,
"name":"IBM
IoT",
"service":"regist
ere d", "x":680,
"y":220,
"wires":[]
},
{
"id":"4cff18c3274cccc4", "type":"ui_bu on",
"z":"630c8601c5ac3295",
"name":"",
"group":"716e956.00eed6c",
"order":2,
"width":"0",
"height":"0",

"passthru":false,
"label":"MotorON",
"tool p":"",
"color":"",
"bgcolor":"",
"className":"",
"icon":"",
"payload":"{\"command\":\"motoron\"}",
"payloadType":"str",
"topic":"motoron",
"topicType":"s
tr", "x":360,
"y":160, "wires":[["625574ead9839b34"]]},
{
"id":"659589baceb4e0b0",
"type":"ui_bu on", "z":"630c8601c5ac3295",
"name":"",
"group":"716e956.00eed6c",
"order":3,
"width":"0",
"height":"0",
"passthru":true,
"label":"MotorOF
F",
"tool p":"",
"color":"",
"bgcolor":"",
"className":"",
"icon":"",
"payload":"{\"command\":\"motoroff\"}",
"payloadType":"str",
"topic":"motoroff",
"topicType":"s
tr", "x":350,

"y":220, "wires":[["625574ead9839b34"]]}, {"id":"ef745d48e395ccc0", "type":"ibmiot",
"name":"weather_monitor", "keepalive":"60",
"serverName":"",
"cleansession":true,
"appId":"",
"shared":false},

{"id":"716e956.00eed6c",
"type":"ui_group",
"name":"Form",
"tab":"7e62365e.b7e6b8
", "order":1,
"disp":true,
"width":"6",
"collapse":fal
se},
{"id":"7e62365e.b7e6b8",
"type":"ui_tab",
"name":"contorl",
"icon":"dashboard
", "order":1,
"disabled":false,
"hidden":false}
]

[

{

"id":"b42b5519fee73ee2", "type":"ibmio n",
"z":"03acb6ae05a0c712",
"authen ca on":"apiKey",
"apiKey":"ef745d48e395ccc0",

"inputType":"evt",
"logicalInterface":"",

```
"ruleId":"",
"deviceId":"b827ebd607b5",
"applica onId":"",
"deviceType":"weather_monitor", "eventType":"+",
"commandType":"",
"format":"json",
"name":"IBMIoT",
"service":"registered",
"allDevices":"",
"allApplica ons":"",
"allDeviceTypes":"",
"allLogicalInterfaces":"",
"allEvents":true,
"allCommands":"",
"allFormats
":"",
"qos":0,
"x":270,
"y":180,
   "wires":[["50b13e02170d73fc","d7da6c2f5302ffaf","a949797028158f3f","a71f164bc3 78bcf1"]]
},
{
"id":"50b13e02170d73fc
",
"type":"func on",
"z":"03acb6ae05a0c712
", "name":"Soil
Moisture",
   "func":"msg.payload = msg.payload.soil;\nglobal.set('s',msg.payload);\nreturn
msg;", "outputs":1, "noerr":
0,
"ini alize
":"",
"finalize":"",
"libs":
[],
```

```
"x":490,
"y":120,
"wires":[["a949797028158f3f","ba98e701f55f04fe"]]
},
{
"id":"d7da6c2f5302ffaf", "type":"func on",
"z":"03acb6ae05a0c712",
"name":"Humidity",
  "func":"msg.payload = msg.payload.pulse;\nglobal.set('p',msg.payload)\nreturn
msg;", "outputs":1, "noerr":
0,
"ini alize
":"",
"finalize":"",
"l
i
b
s
":[
],
"x
":
48
0,
"y":260, "wires":[["a949797028158f3f","70a5b076eeb80b70"]]
},
{
"id":"a949797028158f3f
",
"type":"debug",
"z":"03acb6ae05a0c712
", "name":"IBMo/p",
"ac ve":true,
"tosidebar":true,
"console":false,
"tostatus":false,
"complete":"payload",
"targetType":"msg",
```

"statusVal":"",
"statusType":"auto",
"x":780,
"y":180,
"wires":[]
},

{
"id":"70a5b076eeb80b70",
"type":"ui_gauge",
"z":"03acb6ae05a0c712",
"name":"",
"group":"f4cb8513b95c98a4",
"order":6,
"width":"0",
"height":"0",
"gtype":"gage",
" tle":"Humidity",
"label":"Percentage(%)",
"format":"{{value}}
", "min":0,
"max":"100",
"colors":["#00b500","#e6e600","#ca3838"], "seg1":"",
"seg2":"",
"className
":"","x":86
0,
"y":260,
"wires":[]
},
{
"id":"a71f164bc378bcf1", "type":"func on",
"z":"03acb6ae05a0c712",
"name":"Temperature",
"func":"msg.payload=msg.payload.temp;\nglobal.set('t',msg.payload);\nreturn msg;", "outputs":1,
"noerr":

0,
"ini alize
":"",
"finalize":"",
"l
i
b
s
":[
],
"x
":
49
0,
"y":360,

"wires":[["8e8b63b110c5ec2d","a949797028158f3f"]]
},
{
"id":"8e8b63b110c5ec2d",
"type":"ui_gauge",
"z":"03acb6ae05a0c712",
"name":"",
"group":"f4cb8513b95c98a4",
"order":11,
"width":"0",
"height":"0",
"gtype":"gage",
" tle":"Temperature",
"label":"DegreeCelcius",
"format":"{{value}}",
"min":0,
"max":"100",
"colors":["#00b500","#e6e600","#ca3838"], "seg1":"",
"seg2":"",
"className
":"",

"x":790,
"y":360,
"wires":[]
},
{
"id":"ba98e701f55f04fe",
"type":"ui_gauge",
"z":"03acb6ae05a0c712",
"name":"",
"group":"f4cb8513b95c98a4",
"order":1,
"width":"0",
"height":"0",
"gtype":"gage",
" tle":"Soil Moisture",
"label":"Percentage(%)",
"format":"{{value}}
", "min":0,
"max":"100",
"colors":["#00b500","#e6e600","#ca3838"], "seg1":"",
"seg2":"",
"className
":"",
"x":790,
"y":120,
"wires":[]
},
{
"id":"a259673baf5f0f98
", "type":"h pin",
"z":"03acb6ae05a0c712
", "name":"",
"url":"/sensor",
"method":"ge

```
t",
"upload":fal
s e,
"swaggerDoc"
:"", "x":370,
"y":500,
"wires":[["18a8cdbf7943d27a"]]
},
{
"id":"18a8cdbf7943d27a", "type":"func on",
"z":"03acb6ae05a0c712",
"name":"h pfunc on",
  "func":"msg.payload{\"pulse\":global.get('p'),\"temp\":global.get('t'),\"soil\":global.get( 's')};\nreturn
  msg;",
"outputs":1,
"noerr":0,

"ini alize":"",
"finalize":"",
"l
i
b
s
":[
],
"x
":
63
0,
"y":500, "wires":[["5c7996d53a445412"]]
},
{
"id":"5c7996d53a445412
",
"type":"h presponse",
"z":"03acb6ae05a0c712
", "name":"",
```

```
"statusCode":"",
"header
s":{},
"x":870,
"y":500,
"wires":[]
},
{
"id":"ef745d48e395ccc0",
"type":"ibmiot",
"name":"weather_monitor",
"keepalive":"60",
"serverName":"",
"cleansession":true,
"appId":"",
"shared":false},
{
"id":"f4cb8513b95c98a4", "type":"ui_group",
"name":"monitor",
"tab":"1f4cb829.2fd
ee8 ", "order":2,
"disp": true, "width
":"6",
"collapse":f
alse,
"className
":""
},
{
"id":"1f4cb829.2fdee8",
"type":"ui_tab",
"name":"Home",
"icon":"dashboard
", "order":3,
"disabled":false,
```

"hidden":false }