PROJECT REPORT

IOT BASED SMART CROP PROTECTION SYSTEM FOR AGRICULTURE

TEAM ID: PNT2022TMID06411

ROLL NUMBER	NAME
730419106004	ASHWINI . A
730419106005	BOOPATHI.S
730419106027	NAGANIRANJANA.J
730419106037	RAJALAKSHMI.S

CONTENT

1. **INTRODUCTION**
2. **LITERATURE SURVEY**
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 requirement
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
7. **CODING & SOLUTIONING**
   - 7.1 Feature 1
   - 7.2 Feature 2
   - 7.3 Feature 3
8. **RESULTS**
   - 8.1 Performance Metrics
9. **ADVANTAGES & DISADVANTAGES**
10. **CONCLUSION**
11. **FUTURE SCOPE**
12. **APPENDIX**
    - Source Code
    - GitHub & Project Demo Link

ABSTRACT

IOT Based Smart Crop-Protection for Agriculture monitoring is a system describes how to monitor crop field. It is developed by using sensors and according to the decision from a server based on sensed data, the irrigation and monitoring system is enhanced. Through wireless transmission the sensed data is forwarded to web server database. If the irrigation is automated, then the moisture and temperature fields are decreased below the potential range. The user can monitor and control the system remotely with the help of application which provides a web interface to user. By smart Agriculture monitoring system and one of the oldest ways in agriculture is the manual method of checking the parameters. In this method farmers by themselves verify all the parameter and calculate the reading. It aims at making agriculture smart using automation and IoT. The cloud computing devices are used at the end of the system that can create a whole computing system from sensors to tools that observe data from agriculture field. It proposes a novel methodology for smart farming by including a smart sensing system and smart irrigator system through wireless communication technology. This system is cheap at cost for installation. Here one can access and control the agriculture system in laptop, cell phone or a computer.

Chapter – 1

INTRODUCTION

A system using sensors that monitor different conditions of environment like humidity, temperature etc., the processor and GUI module is used. The field condition is sent to the farmer via mobile text messages. With this system Soil moisture, humidity and energy efficiency are managed. A system is proposed for intelligent agriculture monitoring system based on IOT technology. The main aim of this project is to help farmers to automate their farms by providing them with a Web App through which they can monitor the parameters of the field like Temperature, humidity etc. and control the equipment like water motor and other devices remotely via internet without their actual presence in the field.

Crops in the farms are many times devastated by the wild as well as domestic animals and low productivity of crops is one of the reasons for this. It is not possible to stay 24 hours in the farm to sentinel the crops. So, to surmount this issue an automated perspicacious crop aegis system is proposed utilizing Internet of Things (IOT). The system consists of esp8266 (node MCU), soil moisture sensor, dihydrogen monoxide sensor, GPRS and GSM module, servo motor, dihydrogen monoxide pump, etc. to obtain the required output. As soon as any kineticism is detected the system will engender an alarm to be taken and the lights will glow up implemented at every corner of the farm. This will not harm any animal and the crops will stay forfended.

Chapter-2

LITERATURE SURVEY

Si no.	Journal Details	Inference
1.	S. Giordano, I. Seitanidis, M. Ojo, D. Adami and F. Vignoli, "IoT solutions for crop protection against wild animal attacks," 2018 IEEE International Conference on Environmental Engineering (EE), 2018, pp. 1-5, doi: 10.1109/EE1.2018.8385275.	The creation of an Internet of Things application for crop security to stop animal invasions in the agricultural field is discussed in the study. To guard against damage from weather events and wild animal attacks, agriculture is furnished with a repelling and monitoring system. In this paper, heterogeneous sensors and actuators are coordinated with the cloud to interact and create a platform for new services in the area.They utilised wireless technologies, in particular in the peripheral area, such as 6LoWPAN, WiFi, Zigbee, etc., and used an advanced IoT gateway to communicate with the data centre. The life expectancy of the devices prior to deployment is another crucial aspect that had to take into account. To achieve this goal, they chosen low energy-consuming motes that have batteries and solar panels for energy harvesting.

2.	S. R. Prathibha, A. Hongal and M. P. Jyothi, "IOT Based Monitoring System in Smart Agriculture," 2017 International Conference on Recent Advances in Electronics and Communication Technology (ICRAECT), 2017, pp. 81-84, doi: 10.1109/ICRAECT.2017.52	The purpose of the paper is to deploy smart agriculture using automation and emerging technologies, such as IoT. The key to increasing the output of productive crops is to keep an eye on environmental elements. The paper's feature involves utilising sensors with a single CC3200 chip to monitor the temperature andhumidity in an agricultural field. The CC3200 camera interface allows for the collection of photographs and the transmission of those images through MMS to farmers mobile devices over Wi-Fi
3	S. K. Roy, A. Roy, S. Misra, N. S. Raghuwanshi and M. S. Obaidat, "AID: A prototype for Agricultural Intrusion Detection using Wireless Sensor Network," 2015 IEEE International Conference on Communications (ICC), 2015, pp. 7059-7064, doi: 10.1109/ICC.2015.7249452.	In this work, they suggest a hardware prototype that uses a Wireless Sensor Network (WSN) to detect intruders in a field of crops called Agricultural Intrusion Detection (AID.When a trespasser enters the field, AID assists in setting off alarms in the farmer's home and sends a text message to the farmer's cellphone at the same time. Theyconstructed and install wireless sensor boards with Advanced Virtual RISC (AVR) microcontrollers across an outside environment in order to execute the suggested scheme.
4	I. Nanda, C. Sahithi, M. Swath, S. Maloji and V. K. Shukla, "IIOT Based Smart Crop Protection and Irrigation System," 2020 Seventh International Conference on Information TechnologyTrends (ITT), 2020, pp. 118-125, doi: 10.1109/ITT51279.2020.9320783.	A monitoring strategy for farm safety against animal assaults and climate change circumstances is produced by the study. Smart farming usually makes use of IIoT advancements to highlight the standard of agriculture. It includes many controls and sensors. The ARM Cortex-A board, which uses 3W, is the key component in the process for WSN. The ARM Cortex-A board is equipped with a variety of sensors, includinga camera, PIR sensor, LDR sensor, HC-SR04 ultrasonic sensor, and DHT 11 humidity and temperature sensor. When there is movement within the scope, the PIR activates, the camera begins to record, and the data is saved both on-board and in the IoT cloud. Immediately after the data is recorded, information is automatically generated usinga SIM900A unit to alert about any interference with the weather conditions information that DHt11 has obtained. The notification of the threshold rate will be

		issued to the cell phone or the website in the event of a deviation.
5	P.Navaneetha, R.Ramiya Devi, S.Venn ila, P.Manikandan , Dr.S.Saravanan "IOT Based Crop Protection System ag ainst Birds andWild Animal Attacks" INTERNATIONAL JOURNAL OF IN NOVATIVE RESEARCH IN TECHN OLOGY,IJIRT 149166April 2020 | IJI RT | Volume 6 Issue 11 | ISSN: 2349-6002	They suggested an automated method for protecting crops from fire and animals. This system is based on an Arduino Uno microcontroller. This device employs a motion sensor to find approaching wild animals close to the field and a smoke sensor to find a fire. The sensor instructs the microcontroller to operate in this situation. In order to get the animals out of the field, the microcontroller now sounds an alarm. It also calls the farmer and sends an SMS to let him know about the problem in case the animals don't leave after hearing the warning. The motor is turned on right away if there is smoke. This guarantees the entire safety of crops against animals and fire, safeguarding the environment.
6	S. Yadahalli, A. Parmar and A. Deshpande, "Smart Intrusion Detection System for Crop Protection by using Arduino," 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA), 2020, pp. 405-408, doi: 10.1109/ICIRCA48905.2020.9182868.	A motion sensor and an IR sensor have been employed in the suggested work, which effectively detects any movement of intruders. Additionally, because a camera is being employed here, the timing of entry and exit is also being recorded. It is simple to carry out this plan to safeguard crops, and it may be done without endangering people or animals. Additionally, the system's components are not overly expensive, making it highly viable. As a result, this product can be utilised to safeguard farm crops. For agricultural purposes, it may be quite helpful.

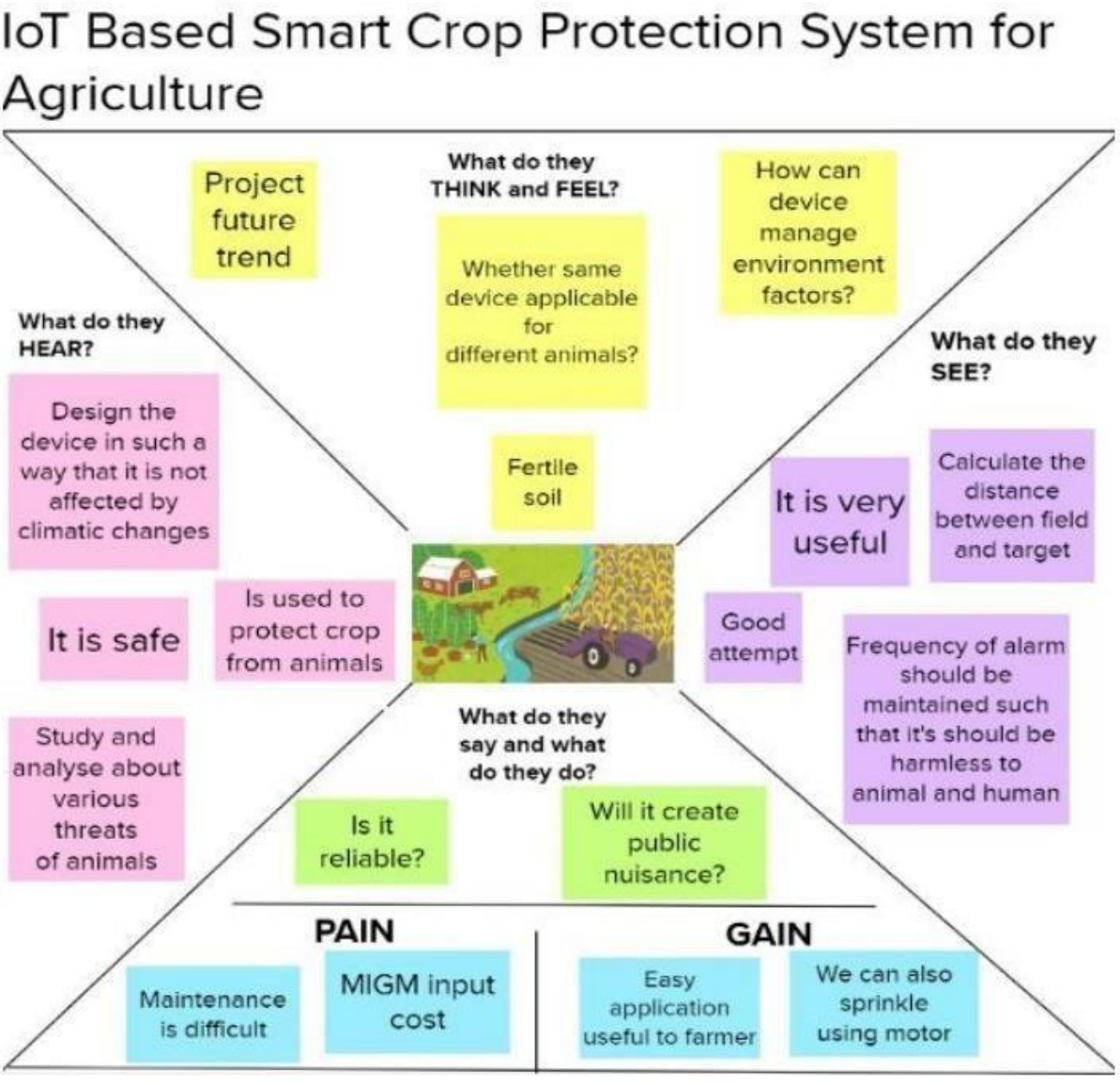
7	Shishir Bagal, Krunal Mahajan, Riya Parate, Ekta Zade, Shubham Khante, "Smart Crop Protection System Using IOT“, April 2021| IJIRT | Volume 7 Issue 11 | ISSN: 2349-6002	The approach employed in the intelligent crop protection system is defined in the study. The major goal of this initiative is to warn the farmer and create dread in him or her about the possibility of farm theft and animal cruelty. This is accomplished using SCPS. In order to make this device portable, they added solar panels and converter modules, which will enable the battery to be charged using solar energy. The SCPS operates on the battery. The IOT gadget is used to alert the farmer when someone enters the farm, and an SD card module is used to store a specific sound that makes the animals frightened.

Chapter-3

IDEATION & PROPOSED SOLUTION

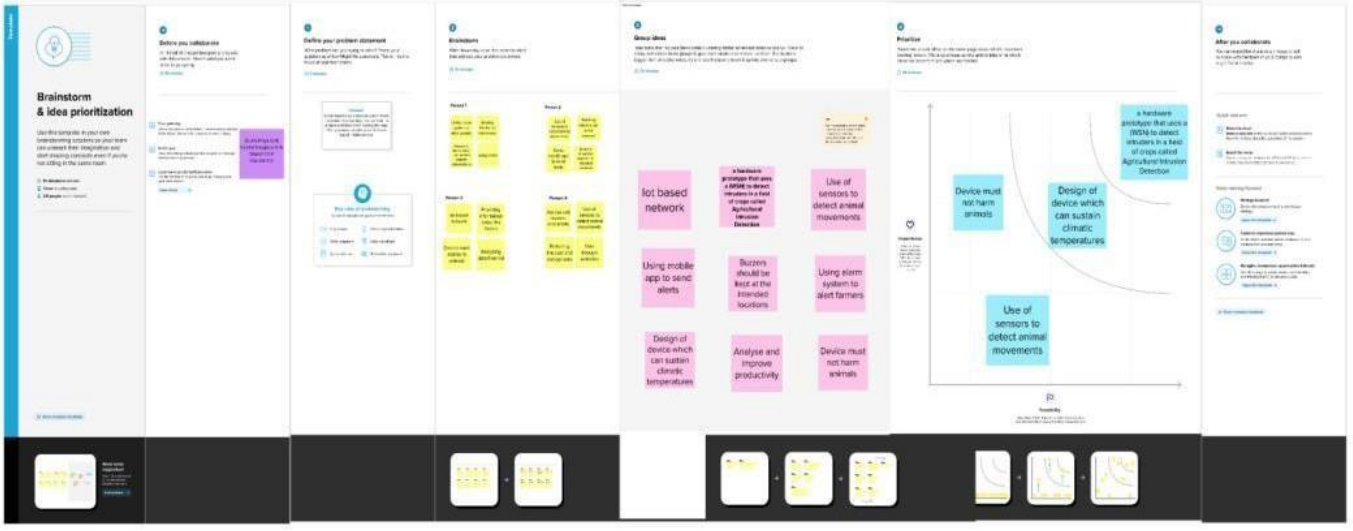
3.1 Empathy Map Canvas:

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes. It is a useful tool to helps teams better understand their users. Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.



3.2 Ideation & Brainstorming:

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions. Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

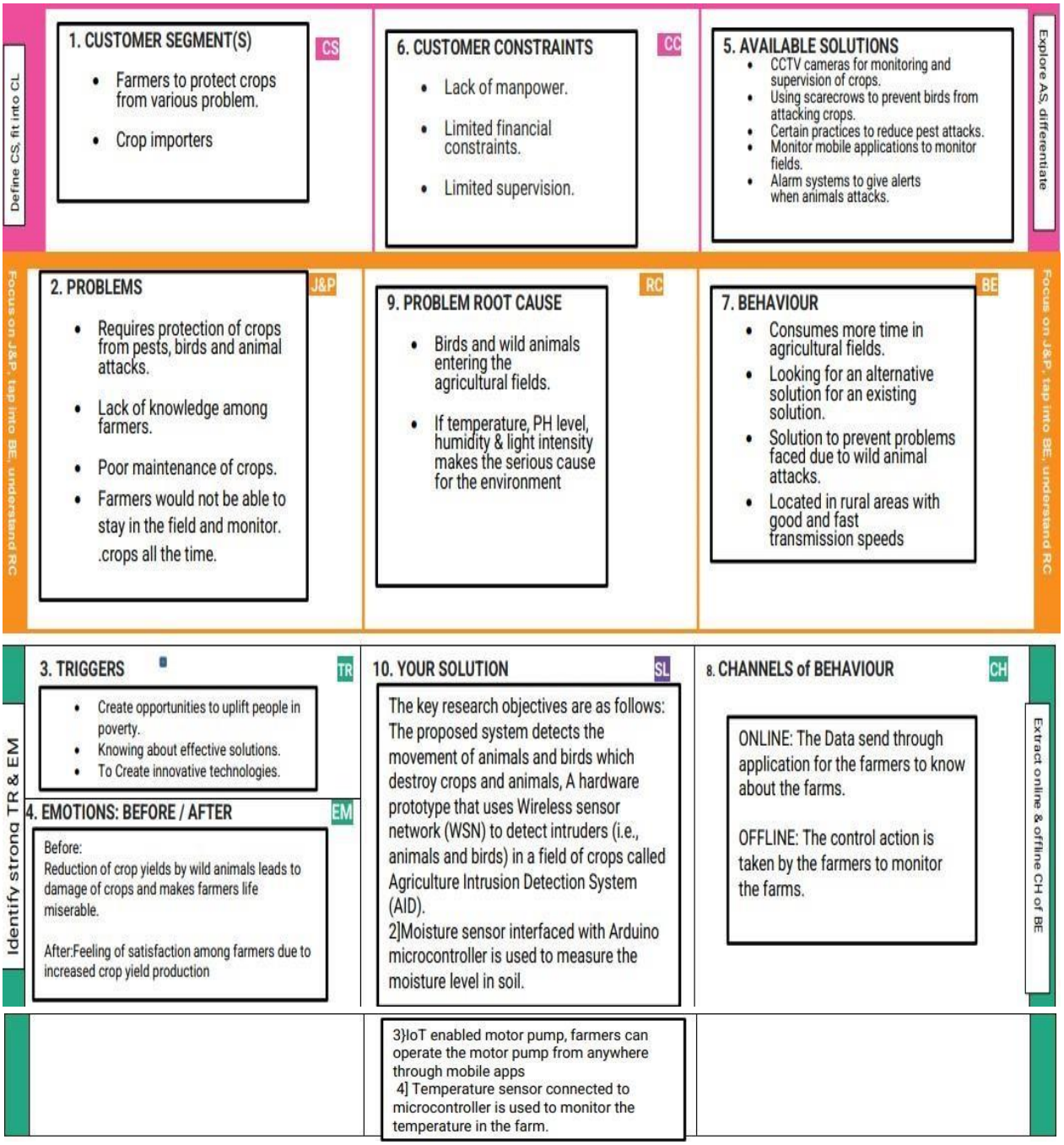


3.3 Proposed Solution:

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	An intelligent crop protection system helps the farmers in protecting the crop from the animals and birds which destroy the crop. This system also helps farmers to monitor the soil moisture levels in the field and also the temperature and humidity values near the field. The motors and sprinklers in the field can be controlled using the mobile application.
2.	Idea / Solution description	• The proposed system detects the movement of animals and birds which destroy crops and animals, A hardware prototype that uses Wireless sensor network (WSN) to detect intruders (i.e., animals and birds) in a field of crops called Agriculture Intrusion Detection System (AID). • Moisture sensor interfaced with Arduino microcontroller is used to measure the moisture level in soil. • IoT enabled motor pump, farmers can operate the motor pump from anywhere through mobile apps. • Temperature sensor connected to microcontroller is used to monitor the temperature in the farm.
3.	Novelty / Uniqueness	The uniqueness of this system is that it detects the movement of animals and alerts the farmers and senses the moisture level and automatically turns on sprinklers without any human intervention.
4.	Social Impact / Customer Satisfaction	• The device should not harm the animals and birds. • The device should can sustain the climatic temperature • It is not possible for famers to protect the field from animals and birds at all times.

		• Improves the productivity, saves lives of farmers. • Simple solution to suite the farmer community.
5.	Business Model (Revenue Model)	• As the product usage can be understood by everyone, it is easy for them to use it properly for their safest organization. • The product is advertised all over the platforms. Since it is economical, even helps small scale farming land from disasters. • This growth is attributed to the rising product demand in light of the surging of crop production. Therefore, surging government standards and regulations for the improvement of the crop production.
6.	Scalability of the Solution	• Crop protection systems can be scaled up in many different ways. • This notion offered an effective IoT-based, real-time crop protection model based on this methodology for enhancing crop productivity. • With the addition of various technologies, this system can be scaled. Even in unfavourable circumstances, additional sensors can be utilised to monitor animal position and movement as well as soil moisture levels. • The use of this is also possible in farms and gardens. Monitoring the locations can also be done using websites and mobile apps.

3.4 Problem Solution fit:



Chapter-4

REQUIREMENT ANALYSIS

4.1 Functional Requirement:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	• User registration • Registration through phone number • Register through gmail • Manual registration • Registration through webpage
FR-2	User Confirmation	• Confirmation via Email • Confirmation via OTP • Confirmation via phone call
FR-3	User Requirement	• Crop protection from birds and animals • Automatic sprinkler system • Monitor soil humidity temperature of the farm
FR-4	User login	• A dashboard is created for the user and the logincredentials are given to him • Using this login credentials, the user will be ableto login to his/her account • If the user enters incorrect username or password then the system will notify them using an errormessage • By using this user can able to autosave their login credentials in their Web page itself so need to enter the details again and again
FR-5	User Notification	• User gets alert message regarding animal activitynear the farmlands • If soil moisture levels are low user will getnotifications to turn on the sprinkler
FR-6	System functionality	• Detects movement of animal around the field usingsensor circuit and sends notification to the farmer and also detects the soil moisture level. • Soil moisture level detected is stored and sent tofarmer via app Farmer can control motors and sprinklers via app

4.2 Non-functional Requirements:

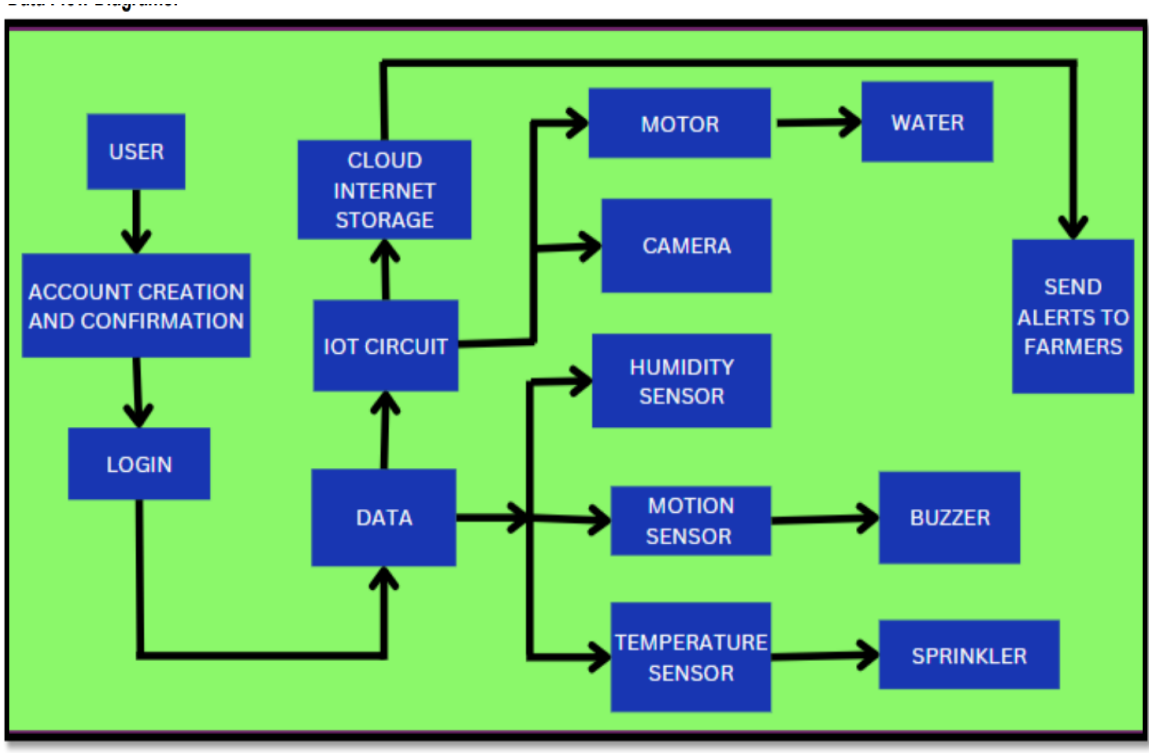
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	• Have a clear and self-explanatory manual. • Easier to use • Even an illiterate farmer have to use the product without any difficulties
NFR-2	**Security**	• Application has to be secured with 2 step authorisation Passwords and passkeys will be assigned as per the users need.
NFR-3	**Reliability**	• Hardware requires a regular checking and service Software may be updated periodically Immediate alert is provided in case of any system failure.
NFR-4	**Performance**	• The application must have a good user interface It should have a minimal energy requirement it has to save water and energy
NFR-5	**Availability**	• All the features will be available when the user requires. • It depends on the need of the farmer and the customization the user has done.
NFR-6	**Scalability**	• The product has to cover all the space of land irrespective of the size or area of a farm field.

Chapter-5

PROJECT DESIGN

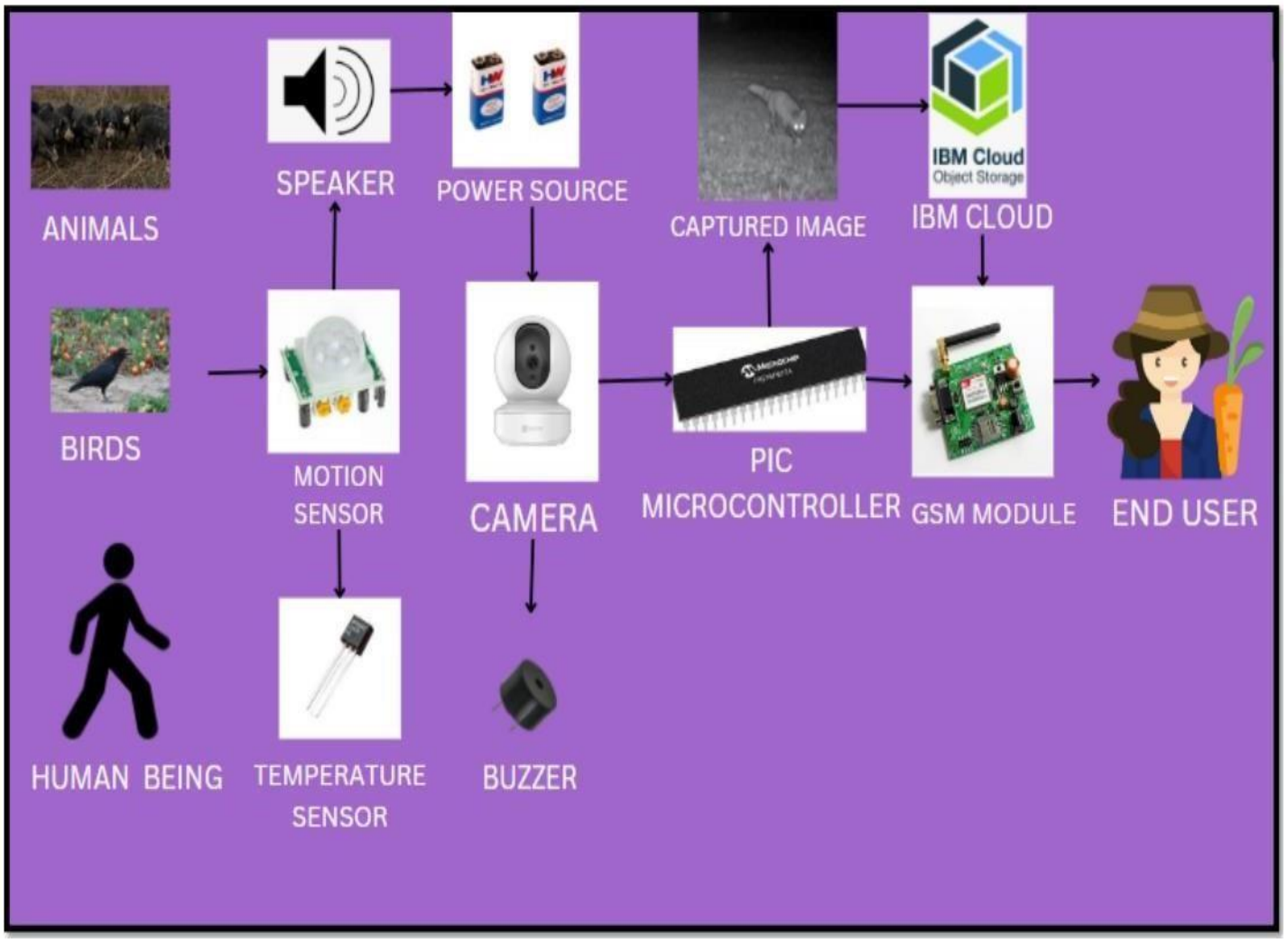
5.1 Data Flow Diagrams:



5.2 Solution & Technical Architecture:

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

- Find the best tech solution to solve existing business problems.
- Describe the structure, characteristics, behavior, and other aspects of the software to project stakeholders.
- Define features, development phases, and solution requirements.
- Provide specifications according to which the solution is defined, managed, and delivered.


ANIMALS
SPEAKER
POWER SOURCE
CAPTURED IMAGE
IBM Cloud
Object Storage
IBM CLOUD
BIRDS
MOTION
SENSOR
CAMERA
PIC
MICROCONTROLLER
GSM MODULE
END USER
HUMAN BEING
TEMPERATURE
SENSOR
BUZZER

Explanation for the Architecture Diagram:

- The device will detect 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 avoid animals from destroying the crop.
- It also generates an alarm and avoid 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

5.3 User Stories:

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard.	High	Sprint-1
Customer	confirmation	USN-2	As a user, I will receive confirmation emailonce I have registered for the application.	I can receive confirmation email & click confirm.	High	Sprint-1
Administrator	Login	USN-3	As a user, I can register for the applicationthrough Facebook.	I can register & access the dashboard with FacebookLogin.	Low	Sprint-1
Temperature Sensing	Dashboard	USN-4	Detects the temperature which is stored inthe database.	Temperature data is stored.	Medium	Sprint-1
Motor pump	Controlling motorpump	USN-5	It is used to control the pump and sprinklers.	Switching on and off using themobile app.	High	Sprint-2
Animal detection	Dashboard	USN-6	Here if an animal is detected buzzer will turn on and notification is sent.	I will receive a notification.	High	Sprint-2
Warehouse Management	Collecting the database of crops	USN-7	Here farmer needs to update the data.	data is stored.	Low	Sprint-3
Customer (Web user)	Usage	USN-8	User can view the webpage and get information.	I can view the webpage in the available devices.	High	Sprint-3
Customer Care Executive	Action	USN-9	User can solve the problem when some faces anyusage issues.	I can solve the issue with the help of customer care.	High	Sprint-4
Administrator	Administration	USN-10	Used to store any information.	I can store the gained information.	High	Sprint-4

Chapter – 6

PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1		US-1	Create the TINKER CAD account which are being used in this project.	6	High	Ashwini.A Boopathi.S Naganiranjana.J Rajalakshmi.S
Sprint-1		US-2	Using Arduino UNO, buzzer and various sensors creating a birds and animal detection circuit and also creating a circuit to check the moisture level in the soil	4	Medium	Ashwini.A Boopathi.S Naganiranjana.J Rajalakshmi.S
Sprint-2		US-3	IBM Watson IoT platform acts as the mediatorto connect the web application to IoT devices, so create the IBM Watson IoT platform.	5	Medium	Ashwini.A Boopathi.S Naganiranjana.J Rajalakshmi.S
Sprint-2		US-4	In order to connect the IoT device to the IBM cloud, create a device in the IBM Watson IoT platform and get the device credentials.	5	High	Ashwini.A Boopathi.S Naganiranjana.J Rajalakshmi.S
Sprint-3		US-1	Configure the connection security and createAPI keys that are used in the Node-RED service for accessing the IBM IoT Platform.	10	High	Ashwini.A Boopathi.S Naganiranjana.J Rajalakshmi.S
Sprint-3		US-2	Create a Node-RED service.	10	High	Ashwini.A Boopathi.S Naganiranjana.J Rajalakshmi.S

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-3		US-1	Develop a python script to publish random sensor data such as temperature, moisture,soil and humidity to the IBM IoT platform	7	High	Ashwini.A Boopathi.S NaganiranjaJ Rajalakshmi.S
Sprint-3		US-2	After developing python code, commands are received just print the statements which represent the control of the devices.	5	Medium	Ashwini.A Boopathi.S NaganiranjaJ Rajalakshmi.S
Sprint-4		US-3	Publish Data to The IBM Cloud	8	High	Ashwini.A Boopathi.S NaganiranjaJ Rajalakshmi.S
Sprint-4		US-1	Create Web UI in Node- Red	10	High	Ashwini.A Boopathi.S Naganiranja.J Rajalakshmi.S
Sprint-4		US-2	Configure the Node-RED flow to receive data from the IBM IoT platform and also use Cloudant DB nodes to store the received sensor data in the cloudant DB	10	High	Ashwini.A Boopathi.S Naganiranjan a.J Rajalakshmi.S

Sprint	Total StoryPoints	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

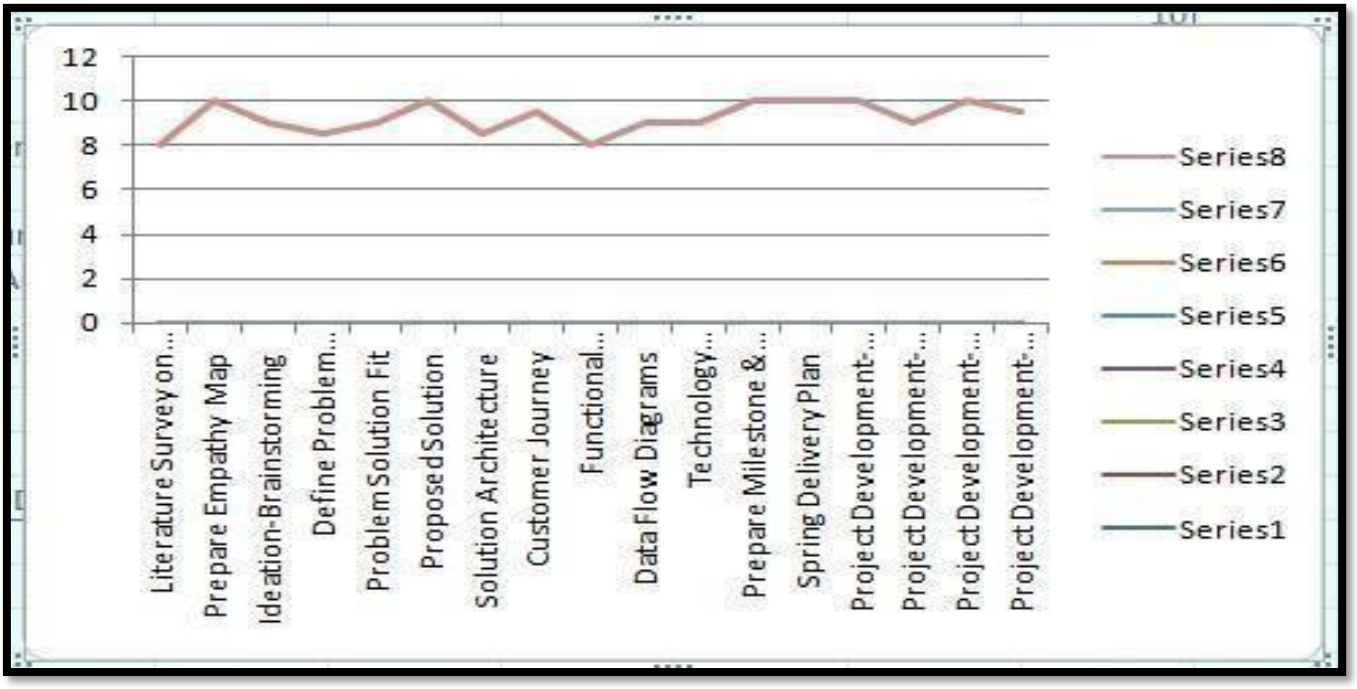
Velocity:

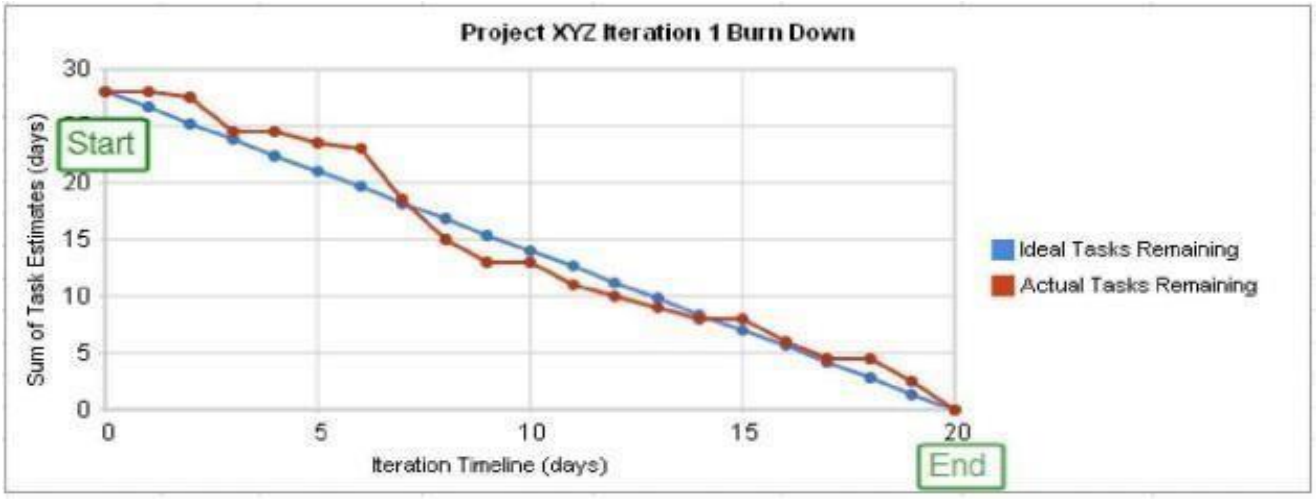
Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) periteration unit (story points per day)

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

Burndown Chart:

A burndown chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burndowncharts can be applied to any project containing measurable progress overtime.





6.2 Sprint Delivery Schedule:

TITLE	DESCRIPTION	DATE
Literature Survey on The Selected Project and Information Gathering	A Literature Survey is a compilation summary of research done previously in the given topic. Literature survey can be taken from books, research paper online or from any source.	19 September 2022
Prepare Empathy Map	Empathy Map is a visualization tool which can be used to get a better insight of the customer	19 September 2022
Ideation-Brainstorming	Brainstorming is a group problem solving session where ideas are shared, discussed and organized among the team members.	19 September 2022
Define Problem Statement	A Problem Statement is a concise description of the problem or issues a project seeks to address. The problem statement identifies the current state, the desired future state and any gaps between the two.	19 September 2022
Problem Solution Fit	This helps us to understand the thoughts of the customer their likes, behaviour, emotions etc.	12 October 2022
Proposed Solution	Proposed solution shows the current solution and it helps is going towards the desired result until it is achieved.	12 October 2022
Solution Architecture	Solution Architecture is a very complex process I.e it has a lot of sub-processes and branches. It helps in understanding the components and features to complete our project.	12 October 2022
Customer Journey	It helps us to analyse from the perspective of a customer, who uses our project.	15 October 2022
Functional Requirement	Here functional and nonfunctional requirements are briefed. It has specific features like usability, security, reliability, performance, availability and scalability.	15 October 2022
Data Flow Diagrams	Data Flow Diagram is a graphical or visual representation using a standardized set of symbols and notations to describe a business's operations through data movement.	15 October 2022
Technology Architecture	Technology Architecture is a more well defined version of solution architecture. It helps us analyze and understand various technologies that needs to be implemented in the project.	15 October 2022
Prepare Milestone & Activity List	It helps us to understand and evaluate our own progress and accuracy so far.	29 October 2022

Spring Delivery Plan	Sprint planning is an event in scrum that kicks off the sprint. The purpose of sprint planning is to define what can be delivered in the sprint and how that work will be achieved.	Completed

Chapter-7

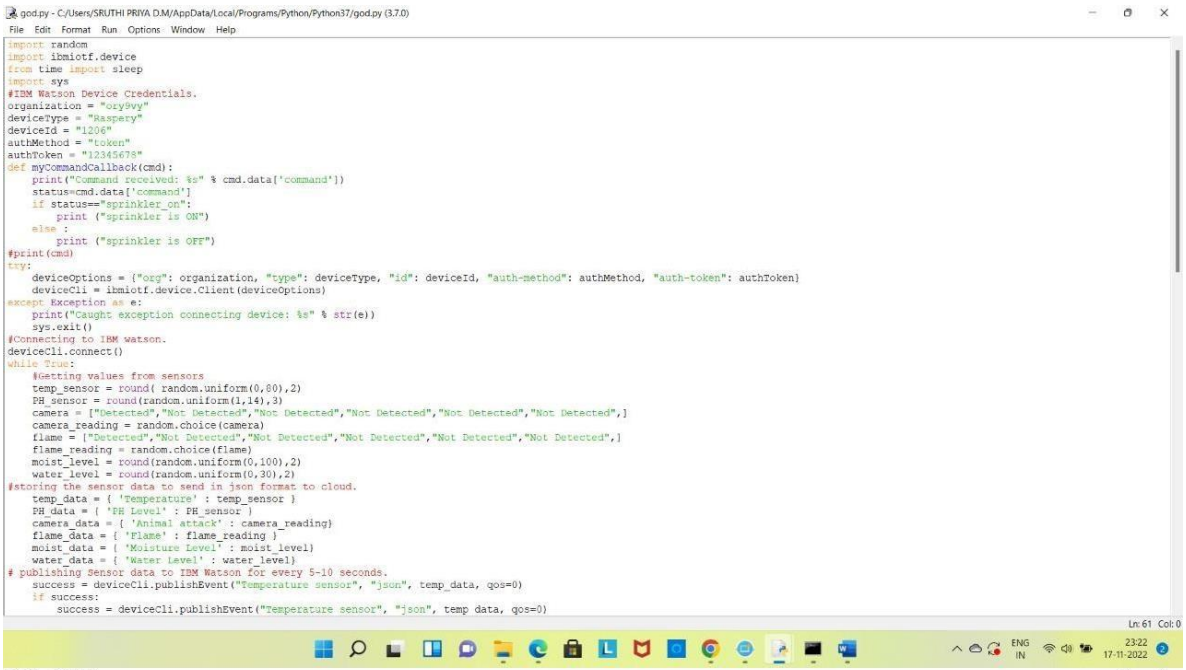
CODING & SOLUTIONING

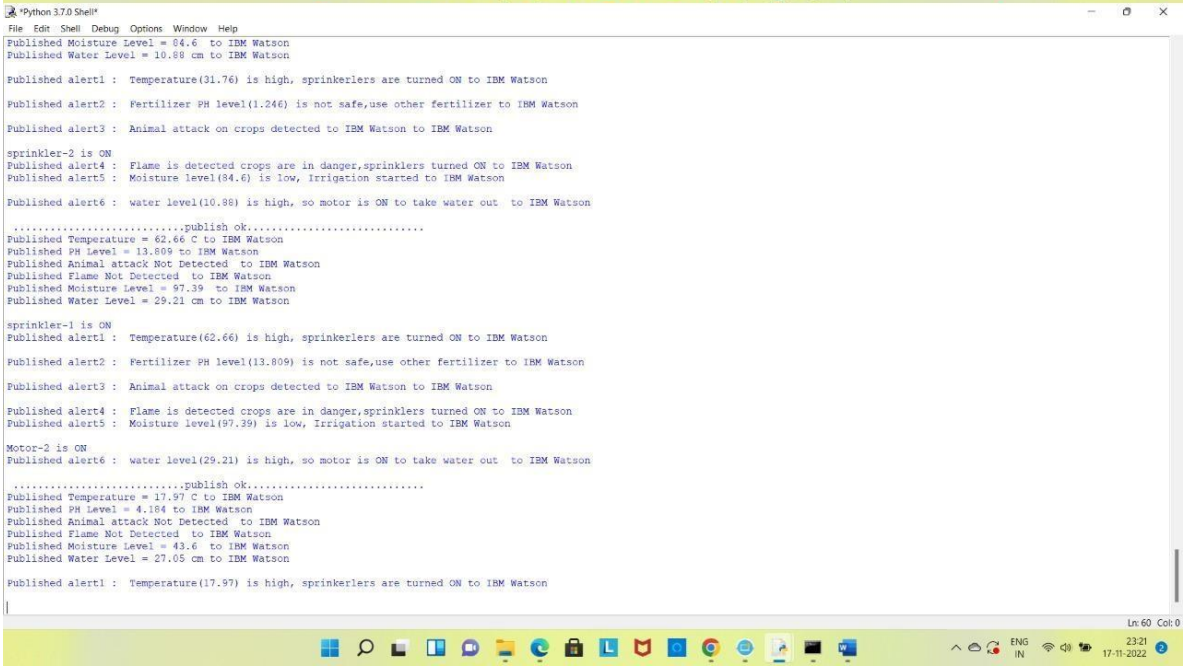
7.1 Feature 1: Coding for detecting temperature, soil moisture, humidity, sprinkler on off , ph level and movement of animals and birds.

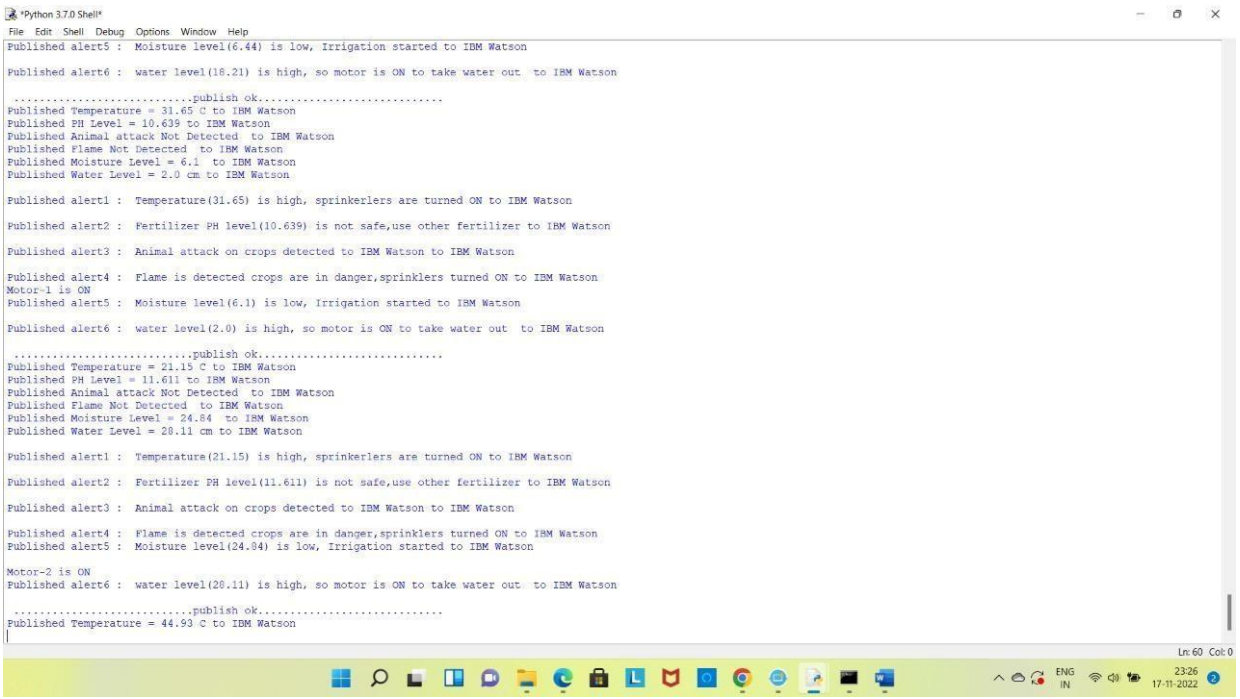
```
import random
import ibmiotf.device
from time import sleep
import sys
#IBM Watson Device Credentials.
organization = "ory9vy"
deviceType = "Raspery"
deviceId = "1206"
authMethod = "token"
authToken = "12345678"
def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data['command'])
    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}
```

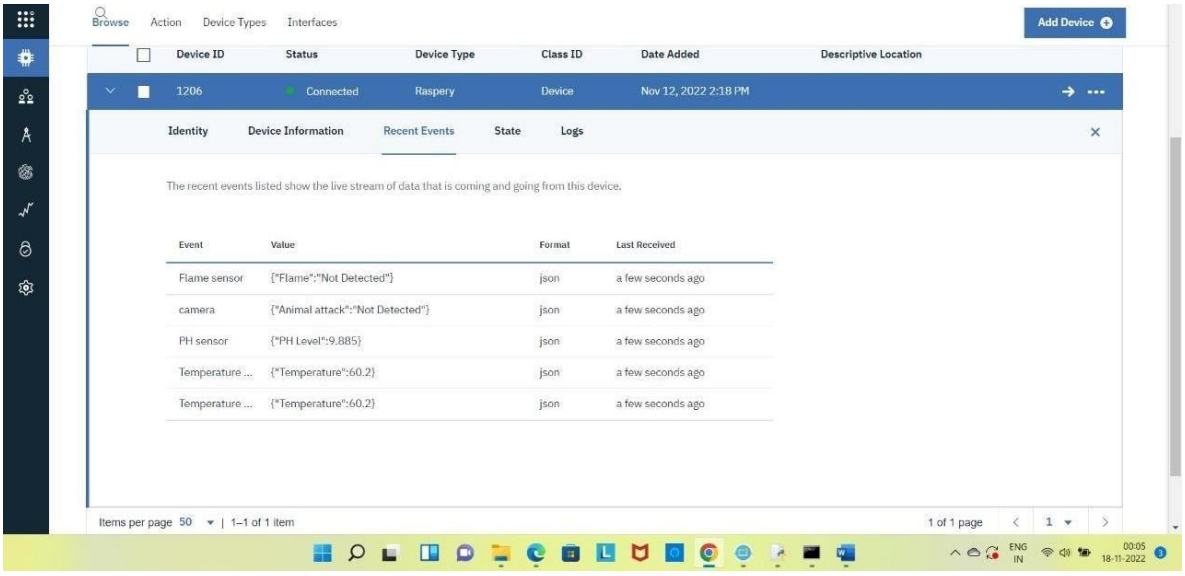
```
    flame_data = { 'Flame' : flame_reading }
    moist_data = { 'Moisture Level' : moist_level}
    water_data = { 'Water Level' : water_level}
# publishing Sensor data to IBM Watson for every 5-10 seconds.
    success = deviceCli.publishEvent("Temperature sensor", "json", temp_data, qos=0)
    if success:
        success = deviceCli.publishEvent("Temperature sensor", "json", temp_data,
qos=0)
        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("")
#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
```

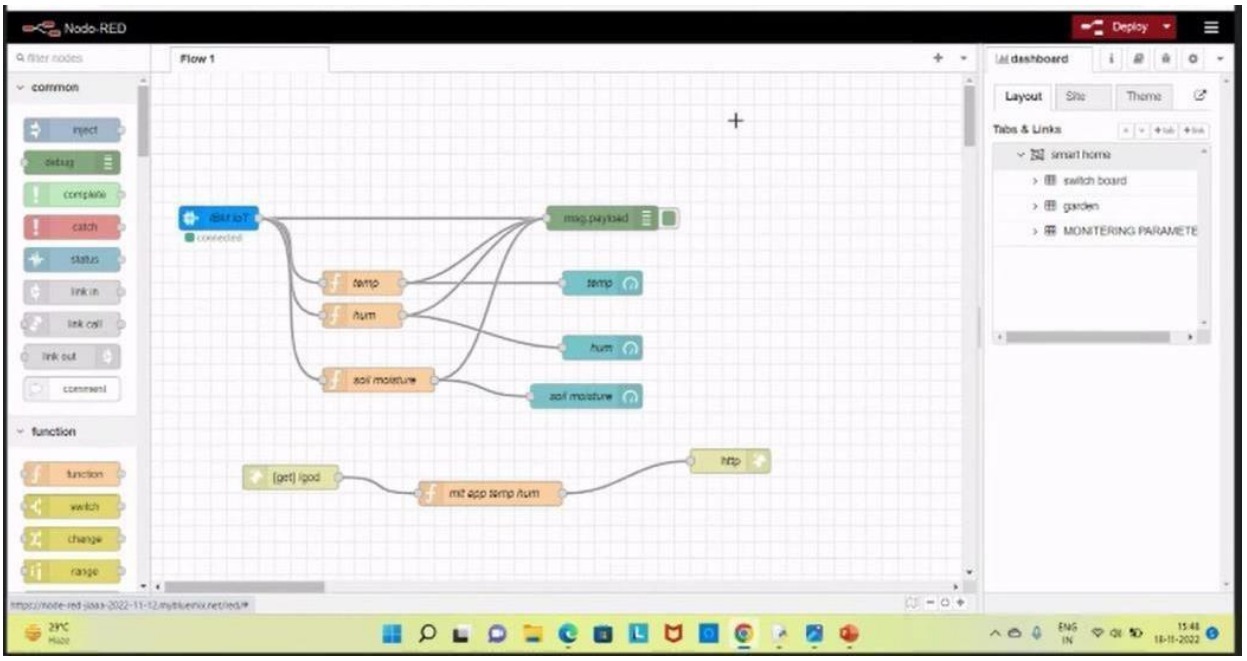
```
crops detected" }, qos=0)
    sleep(1)
    if success:
       print('Published alert3 : ' , "Animal attack on crops detected","to IBM Watson","to
IBM Watson")
    print("")
#To send alert message if flame detected on crop land and turn ON the splinkers to
take immediate action.
    if (flame_reading == "Detected"):
        print("sprinkler-2 is ON")
    success = deviceCli.publishEvent("Alert4", "json", { 'alert4' : "Flame is detected
crops are in danger,sprinklers turned ON" }, qos=0)
    sleep(1)
    if success:
        print( 'Published alert4 : ' , "Flame is detected crops are in danger,sprinklers
turned ON","to IBM Watson")
#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("")
#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("")
#command recived by farmer
deviceCli.commandCallback =  myCommandCallback
# Disconnect the device and application from the cloud
deviceCli.disconnect()
```

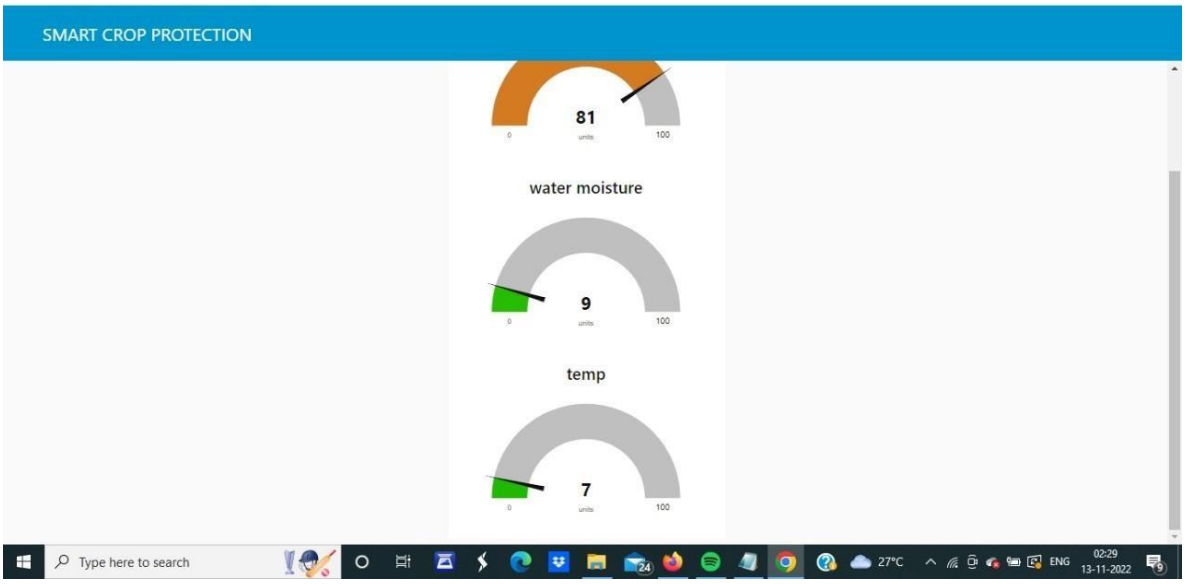

god.py - C:/Users/SRUTHI PRIYA D.M/AppData/Local/Programs/Python/Python37/god.py (3.7.0)
File Edit Format Run Options Window Help
import random
import ibmiotf.device
from time import sleep
import sys
#IBM Watson Device Credentials.
organization = "ory9vy"
deviceType = "Raspery"
deviceId = "1206"
authMethod = "token"
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)
if success:
success = deviceCli.publishEvent("Temperature sensor", "json", temp data, qos=0)
Ln: 61 Col: 0
ENG IN
23:22
17-11-2022

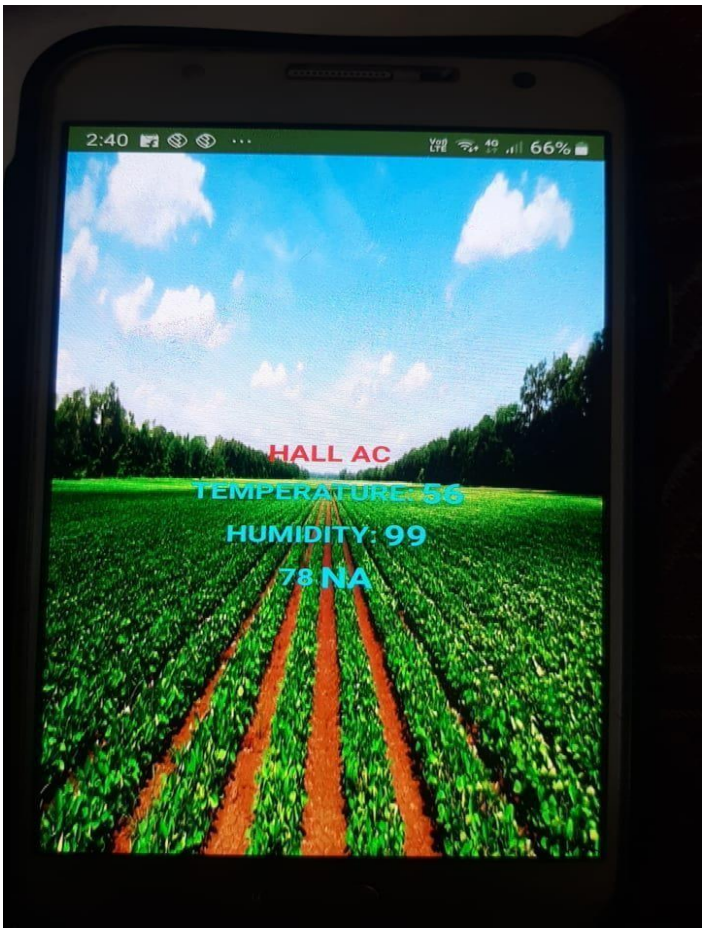

Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Published Moisture Level = 64.6 to IBM Watson
Published Water Level = 10.88 cm to IBM Watson
Published alert1 : Temperature(31.76) is high, sprinkerlers are turned ON to IBM Watson
Published alert2 : Fertilizer PH level(1.246) is not safe,use other fertilizer to IBM Watson
Published alert3 : Animal attack on crops detected to IBM Watson to IBM Watson
sprinkler-2 is ON
Published alert4 : Flame is detected crops are in danger,sprinklers turned ON to IBM Watson
Published alert5 : Moisture level(84.6) is low, Irrigation started to IBM Watson
Published alert6 : water level(10.88) is high, so motor is ON to take water out to IBM Watson
.....publish ok.....
Published Temperature = 62.66 C to IBM Watson
Published PH Level = 13.809 to IBM Watson
Published Animal attack Not Detected to IBM Watson
Published Flame Not Detected to IBM Watson
Published Moisture Level = 97.39 to IBM Watson
Published Water Level = 29.21 cm to IBM Watson
sprinkler-1 is ON
Published alert1 : Temperature(62.66) is high, sprinkerlers are turned ON to IBM Watson
Published alert2 : Fertilizer PH level(13.809) is not safe,use other fertilizer to IBM Watson
Published alert3 : Animal attack on crops detected to IBM Watson to IBM Watson
Published alert4 : Flame is detected crops are in danger,sprinklers turned ON to IBM Watson
Published alert5 : Moisture level(97.39) is low, Irrigation started to IBM Watson
Motor-2 is ON
Published alert6 : water level(29.21) is high, so motor is ON to take water out to IBM Watson
.....publish ok.....
Published Temperature = 17.97 C to IBM Watson
Published PH Level = 4.184 to IBM Watson
Published Animal attack Not Detected to IBM Watson
Published Flame Not Detected to IBM Watson
Published Moisture Level = 43.6 to IBM Watson
Published Water Level = 27.05 cm to IBM Watson
Published alert1 : Temperature(17.97) is high, sprinkerlers are turned ON to IBM Watson
Ln: 60 Col: 0
ENG IN
23:21
17-11-2022


Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Published alert5 : Moisture level(6.44) is low, Irrigation started to IBM Watson
Published alert6 : water level(18.21) is high, so motor is ON to take water out to IBM Watson
.....publish ok.....
Published Temperature = 31.65 C to IBM Watson
Published PH Level = 10.639 to IBM Watson
Published Animal attack Not Detected to IBM Watson
Published Flame Not Detected to IBM Watson
Published Moisture Level = 6.1 to IBM Watson
Published Water Level = 2.0 cm to IBM Watson
Published alert1 : Temperature(31.65) is high, sprinkerlers are turned ON to IBM Watson
Published alert2 : Fertilizer PH level(10.639) is not safe,use other fertilizer to IBM Watson
Published alert3 : Animal attack on crops detected to IBM Watson to IBM Watson
Published alert4 : Flame is detected crops are in danger,sprinklers turned ON to IBM Watson
Motor-1 is ON
Published alert5 : Moisture level(6.1) is low, Irrigation started to IBM Watson
Published alert6 : water level(2.0) is high, so motor is ON to take water out to IBM Watson
.....publish ok.....
Published Temperature = 21.15 C to IBM Watson
Published PH Level = 11.611 to IBM Watson
Published Animal attack Not Detected to IBM Watson
Published Flame Not Detected to IBM Watson
Published Moisture Level = 24.84 to IBM Watson
Published Water Level = 20.11 cm to IBM Watson
Published alert1 : Temperature(21.15) is high, sprinkerlers are turned ON to IBM Watson
Published alert2 : Fertilizer PH level(11.611) is not safe,use other fertilizer to IBM Watson
Published alert3 : Animal attack on crops detected to IBM Watson to IBM Watson
Published alert4 : Flame is detected crops are in danger,sprinklers turned ON to IBM Watson
Published alert5 : Moisture level(24.84) is low, Irrigation started to IBM Watson
Motor-2 is ON
Published alert6 : water level(20.11) is high, so motor is ON to take water out to IBM Watson
.....publish ok.....
Published Temperature = 44.93 C to IBM Watson
Ln: 60 Col: 0
ENG IN
23:26
17-11-2022


Browse
Action
Device Types
Interfaces
Add Device
Device ID
Status
Device Type
Class ID
Date Added
Descriptive Location
1206
Connected
Raspery
Device
Nov 12, 2022 2:18 PM
Identity
Device Information
Recent Events
State
Logs
The recent events listed show the live stream of data that is coming and going from this device.
Event
Value
Format
Last Received
Flame sensor
{"Flame":"Not Detected"}
json
a few seconds ago
camera
{"Animal attack":"Not Detected"}
json
a few seconds ago
PH sensor
{"PH Level":9.885}
json
a few seconds ago
Temperature ...
{"Temperature":60.2}
json
a few seconds ago
Temperature ...
{"Temperature":60.2}
json
a few seconds ago
Items per page 50 | 1–1 of 1 item
1 of 1 page
1
ENG IN
00:05
18-11-2022


Node-RED
Deploy
Flow 1
common
inject
debug
complete
catch
status
link in
link call
link out
comment
function
function
switch
change
range
dashboard
Layout
Site
Theme
Tabs & Links
smart home
switch board
garden
MONITERING PARAMETE
msg.payload
temp
hum
soil moisture
temp
hum
soil moisture
[get] /god
mit app temp /hum
http
29°C

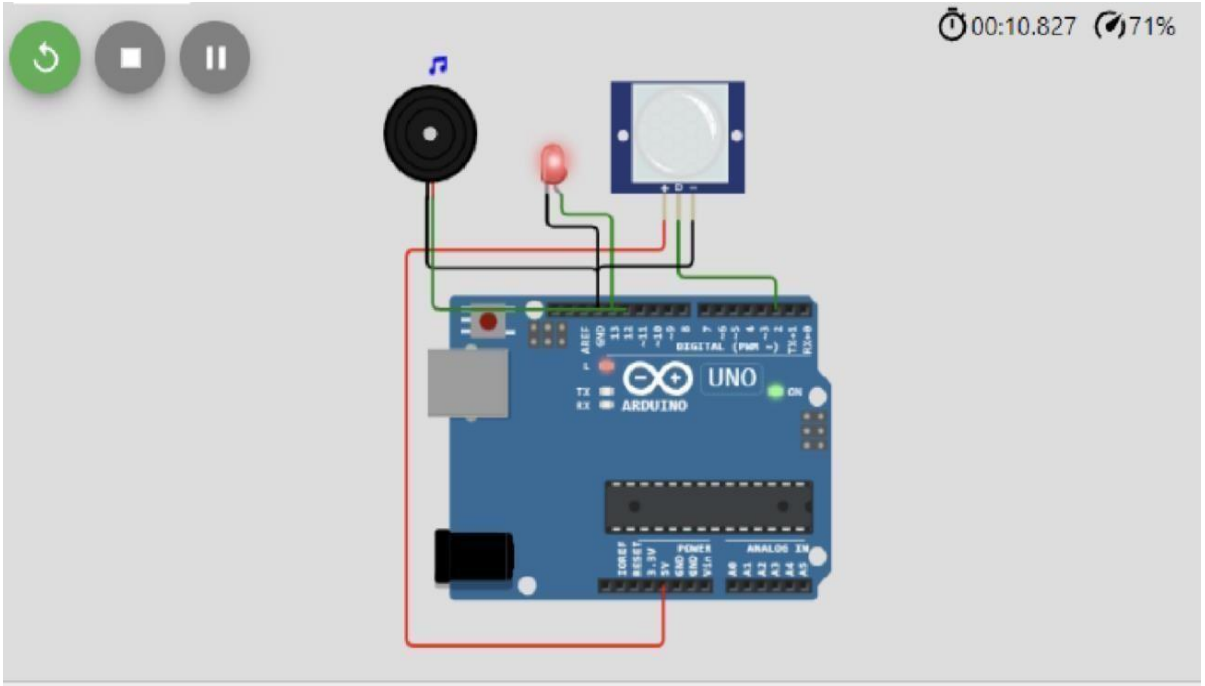

SMART CROP PROTECTION
81
water moisture
9
temp
7
Type here to search
27°C
ENG
02:29
13-11-2022



7.2 Feature 2: Movement of birds and animals in arduino

```
 from pygame import

mixer  class  SoundPlayer:
      def___init (self, sound_file):
          mixer.init(44100, -16, 2, 2048)
          self.sound
          =mixer.Sound(sound_file)
      def play(self):
          self.sound.pla
          y()
      import
      time
      class
      FPS:
        def___init (self):
           self.frame_count
           = 0
           self.elapsed_time
        = 0 def start(self):
           self.start_time =
       time.time() def stop(self):
            self.stop_time =
```

```
 time.time() self.frame_count
      += 1
      self.elapsed_time += (self.stop_time-self.start_time)
 def count(self):
      return
self.frame_count def
elapsed(self):
    return
self.elapsed_time def
fps(self):
  if
     self.elapsed_time=
     =0:return 0
   else:
       return self.frame_count/self.elapsed_time
```

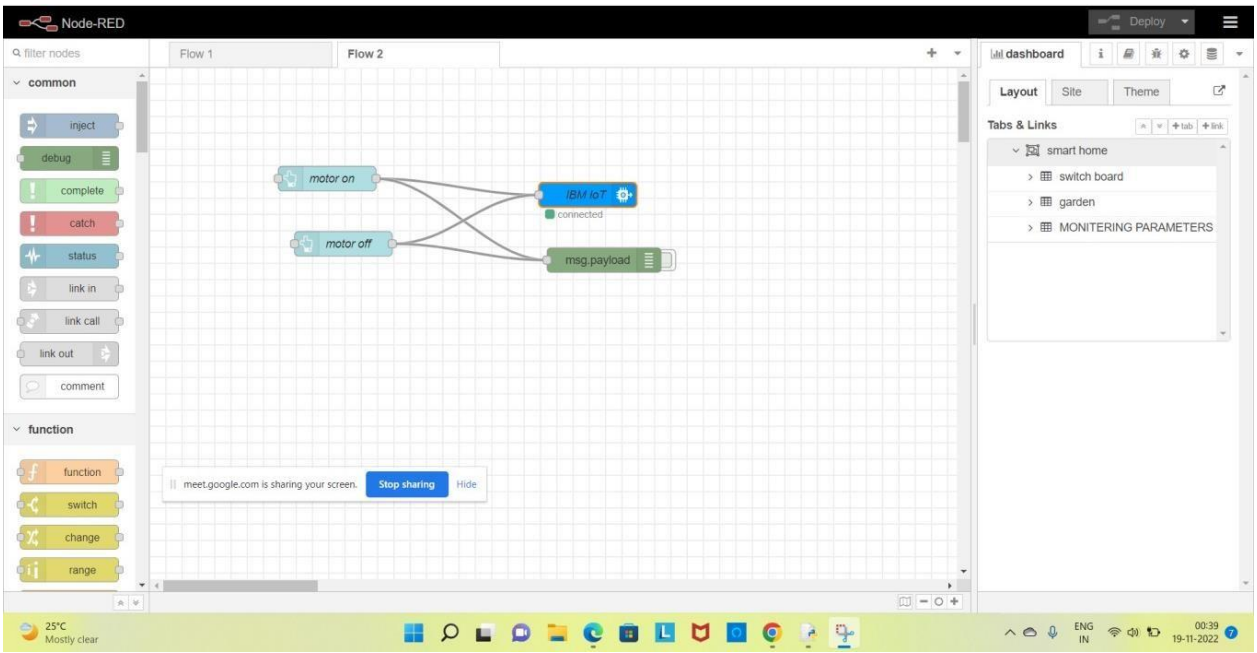



```
Motion detected!
Camera activated!
Pictures taken!
Buzzer activated!
```

7.3 **Feature 3:** Code for motor on off

```
# Provide your IBM Watson Device Credentials organization = "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): # function for Callback if 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 requiredinformation: 'interval'")

interval = cmd.data['interval']

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


try:

deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "authmethod": authMethod,
"auth-token": authToken}                deviceCli
= ibmiotf.device.Client(deviceOptions) # .....

exceptException as e:
print("Caught exception connecting device: %s" % str(e)) sys.exit()

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

while True:
deviceCli.commandCallback = myCommandCallback

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

 SENSOR.PY

import time import sysimport ibmiotf.application importibmiotf.device
import random

# Provide your IBM Watson Device Credentials organization = "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:
deviceOptions = {"org": organization, "type": deviceType, "id": deviceId,
"auth-method": authMethod, "auth-token": authToken} deviceCli =
ibmiotf.device.Client(deviceOptions)
#.....

exceptException 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:
temp=random.randint(0,100) 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") time.sleep(1)

deviceCli.commandCallback = myCommandCallback

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



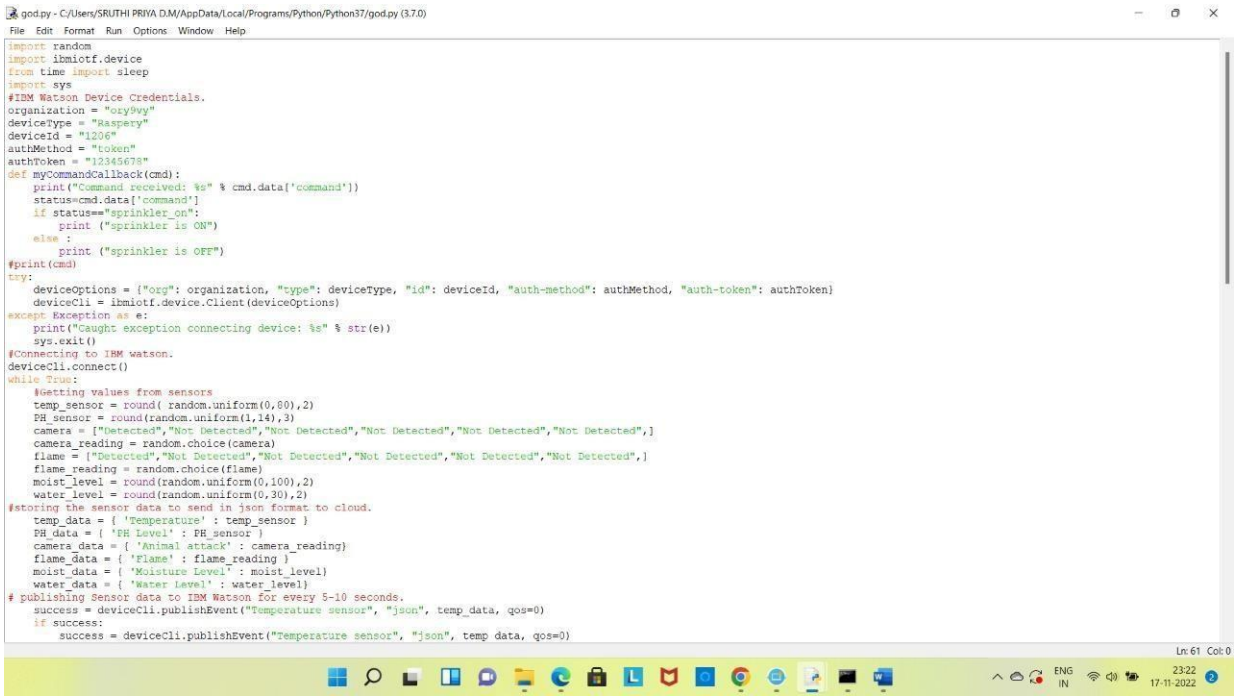
```
import time
import sys
import ibmiotf.application  # to install pip install ibmiotf
import ibmiotf.device

# Provide your IBM Watson Device Credentials
organization = "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):  # function for Callback

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

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

    if cmd.command == "setInterval":

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


try:
    deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "aut
                     "auth-token": authToken}
    deviceCli = ibmiotf.device.Client(deviceOptions)
# ..............................................
```

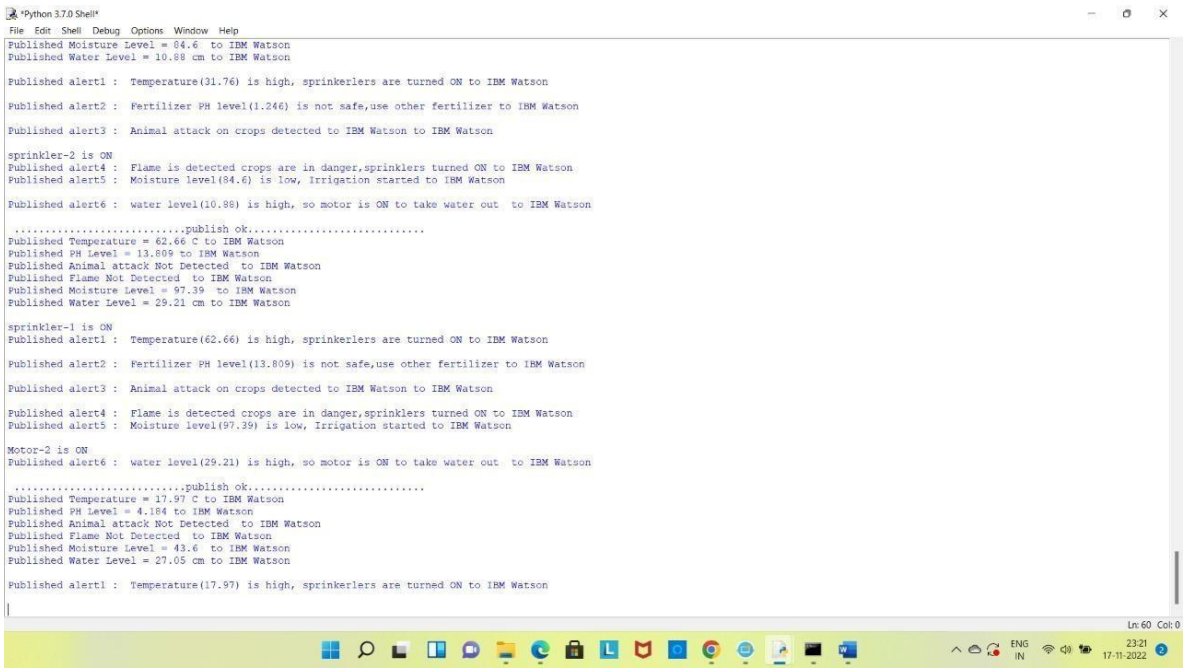
```
Published Temperature = 92 C Humidity = 63 % Soil Moisture = 42 % to IBM Watson
Published Temperature = 15 C Humidity = 84 % Soil Moisture = 29 % to IBM Watson
Published Temperature = 81 C Humidity = 58 % Soil Moisture = 48 % to IBM Watson
Published Temperature = 99 C Humidity = 81 % Soil Moisture = 5 % to IBM Watson
Published Temperature = 35 C Humidity = 36 % Soil Moisture = 71 % to IBM Watson
Published Temperature = 73 C Humidity = 77 % Soil Moisture = 8 % to IBM Watson
Published Temperature = 37 C Humidity = 82 % Soil Moisture = 37 % to IBM Watson
Published Temperature = 19 C Humidity = 66 % Soil Moisture = 14 % to IBM Watson
Published Temperature = 59 C Humidity = 21 % Soil Moisture = 19 % to IBM Watson
Published Temperature = 63 C Humidity = 23 % Soil Moisture = 83 % to IBM Watson
Published Temperature = 72 C Humidity = 93 % Soil Moisture = 69 % to IBM Watson
Published Temperature = 66 C Humidity = 27 % Soil Moisture = 28 % to IBM Watson
Published Temperature = 51 C Humidity = 92 % Soil Moisture = 27 % to IBM Watson
Published Temperature = 21 C Humidity = 49 % Soil Moisture = 97 % to IBM Watson
Published Temperature = 42 C Humidity = 70 % Soil Moisture = 68 % to IBM Watson
Published Temperature = 75 C Humidity = 17 % Soil Moisture = 87 % to IBM Watson
Published Temperature = 100 C Humidity = 56 % Soil Moisture = 49 % to IBM Watson
Published Temperature = 20 C Humidity = 68 % Soil Moisture = 53 % to IBM Watson
Published Temperature = 80 C Humidity = 76 % Soil Moisture = 22 % to IBM Watson
Published Temperature = 3 C Humidity = 77 % Soil Moisture = 36 % to IBM Watson
Published Temperature = 53 C Humidity = 35 % Soil Moisture = 9 % to IBM Watson
Published Temperature = 47 C Humidity = 61 % Soil Moisture = 10 % to IBM Watson
Published Temperature = 47 C Humidity = 63 % Soil Moisture = 19 % to IBM Watson
Published Temperature = 2 C Humidity = 94 % Soil Moisture = 41 % to IBM Watson
Published Temperature = 61 C Humidity = 10 % Soil Moisture = 83 % to IBM Watson
Published Temperature = 48 C Humidity = 11 % Soil Moisture = 96 % to IBM Watson
Published Temperature = 8 C Humidity = 34 % Soil Moisture = 43 % to IBM Watson
Published Temperature = 94 C Humidity = 25 % Soil Moisture = 88 % to IBM Watson
Published Temperature = 5 C Humidity = 70 % Soil Moisture = 41 % to IBM Watson
Published Temperature = 77 C Humidity = 20 % Soil Moisture = 50 % to IBM Watson
Published Temperature = 9 C Humidity = 77 % Soil Moisture = 7 % to IBM Watson
Published Temperature = 85 C Humidity = 49 % Soil Moisture = 79 % to IBM Watson
Published Temperature = 1 C Humidity = 53 % Soil Moisture = 94 % to IBM Watson

========== RESTART: C:\Users\murug\Desktop\motor.py ==========
2022-11-10 00:49:51,101   ibmiotf.device.Client      INFO    Connected successful
ly: d:8gyz7t:weather_monitor:b827ebd607b5
MOTOR ON IS RECEIVED
MOTOR OFF IS RECEIVED
```

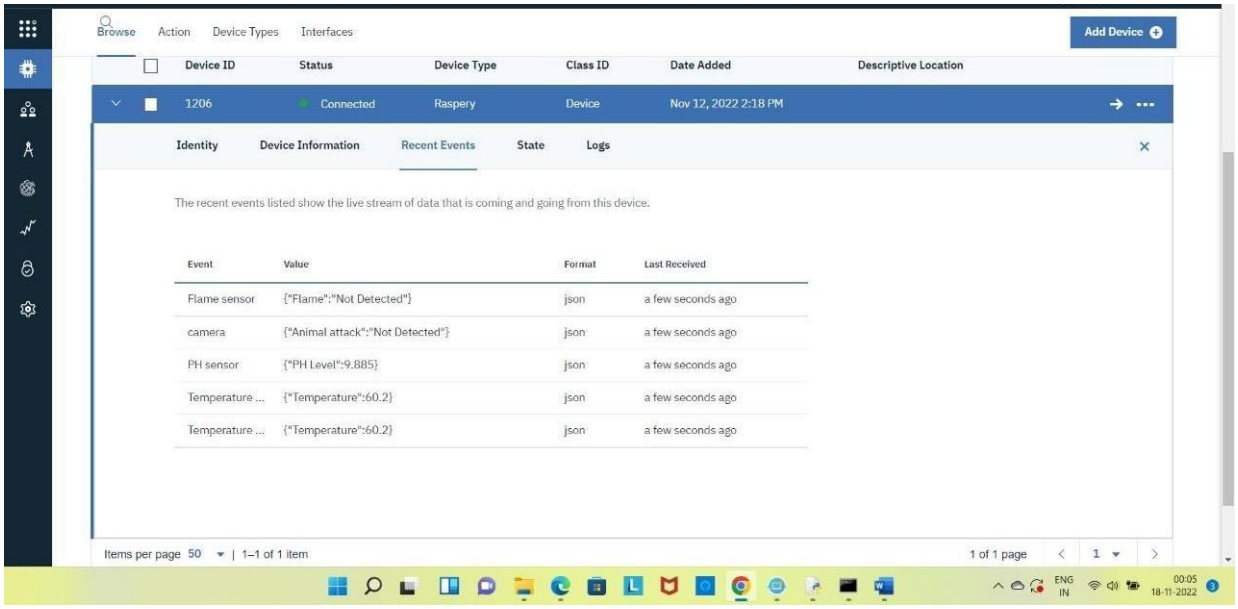
Chapter-8

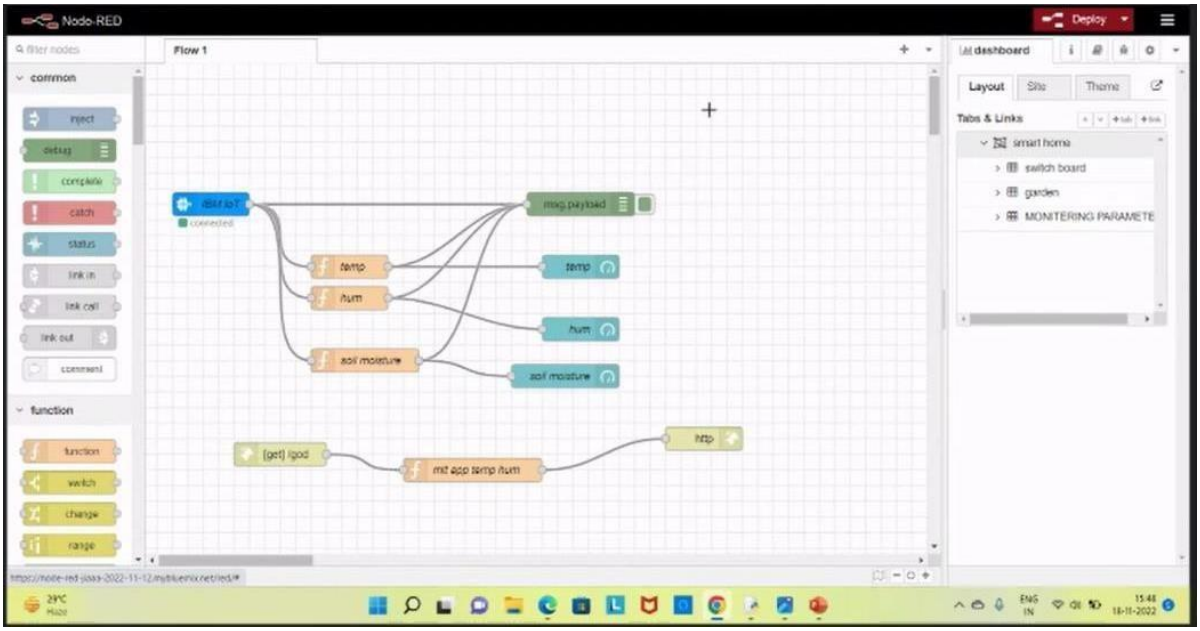
RESULTS

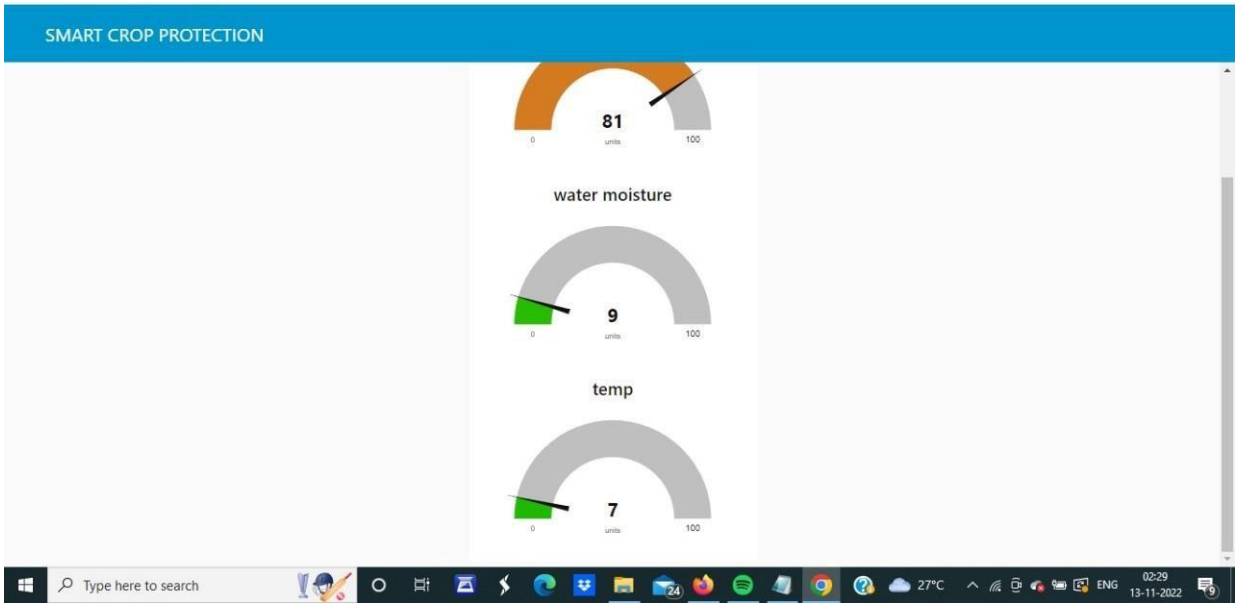
Detection of temperature, soil moisture, humidity, ph level and movement of animals and birds.

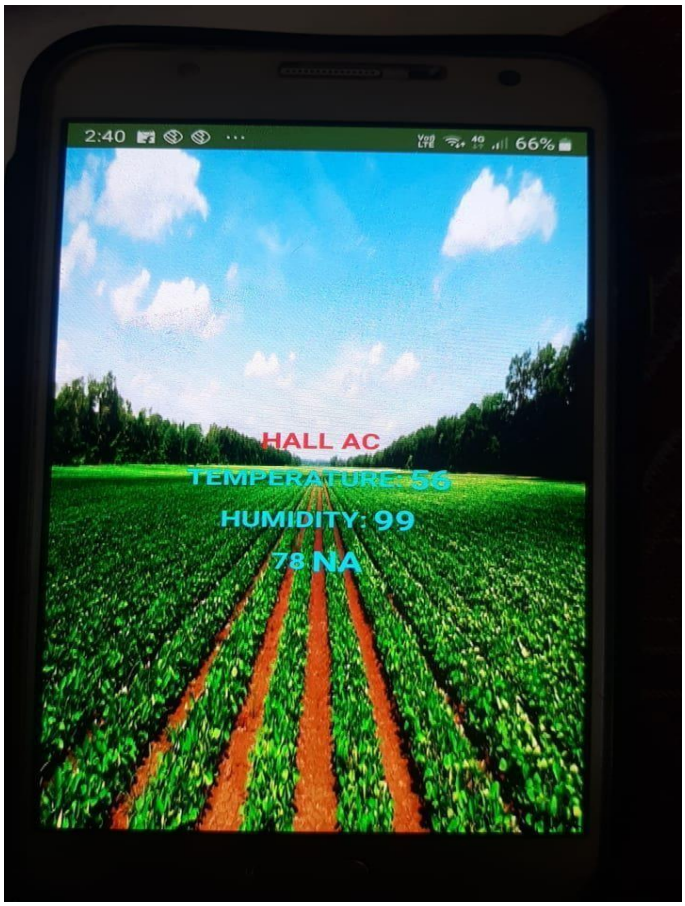




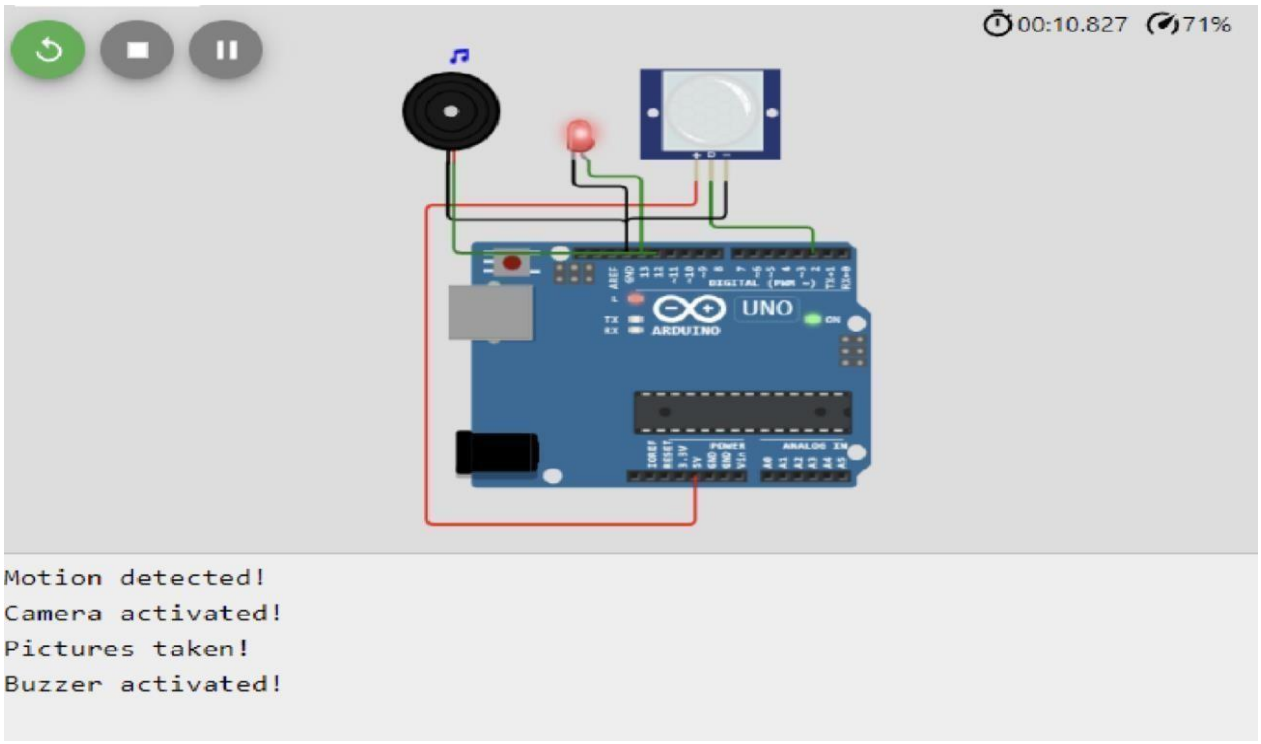

Browse
Action
Device Types
Interfaces
Add Device
Device ID
Status
Device Type
Class ID
Date Added
Descriptive Location
1206
Connected
Raspery
Device
Nov 12, 2022 2:18 PM
Identity
Device Information
Recent Events
State
Logs
The recent events listed show the live stream of data that is coming and going from this device.
Event
Value
Format
Last Received
Flame sensor
{"Flame":"Not Detected"}
json
a few seconds ago
camera
{"Animal attack":"Not Detected"}
json
a few seconds ago
PH sensor
{"PH Level":9.885}
json
a few seconds ago
Temperature ...
{"Temperature":60.2}
json
a few seconds ago
Temperature ...
{"Temperature":60.2}
json
a few seconds ago
Items per page 50
1–1 of 1 item
1 of 1 page
1


Node-RED
Deploy
Flow 1
common
inject
debug
complete
catch
status
link in
link call
link out
comment
function
function
switch
change
range
dashboard
Layout
Site
Theme
Tabs & Links
smart home
switch board
garden
MONITERING PARAMETE
temp
hum
soil moisture
msg.payload
http
[get] igod
mit app temp hum

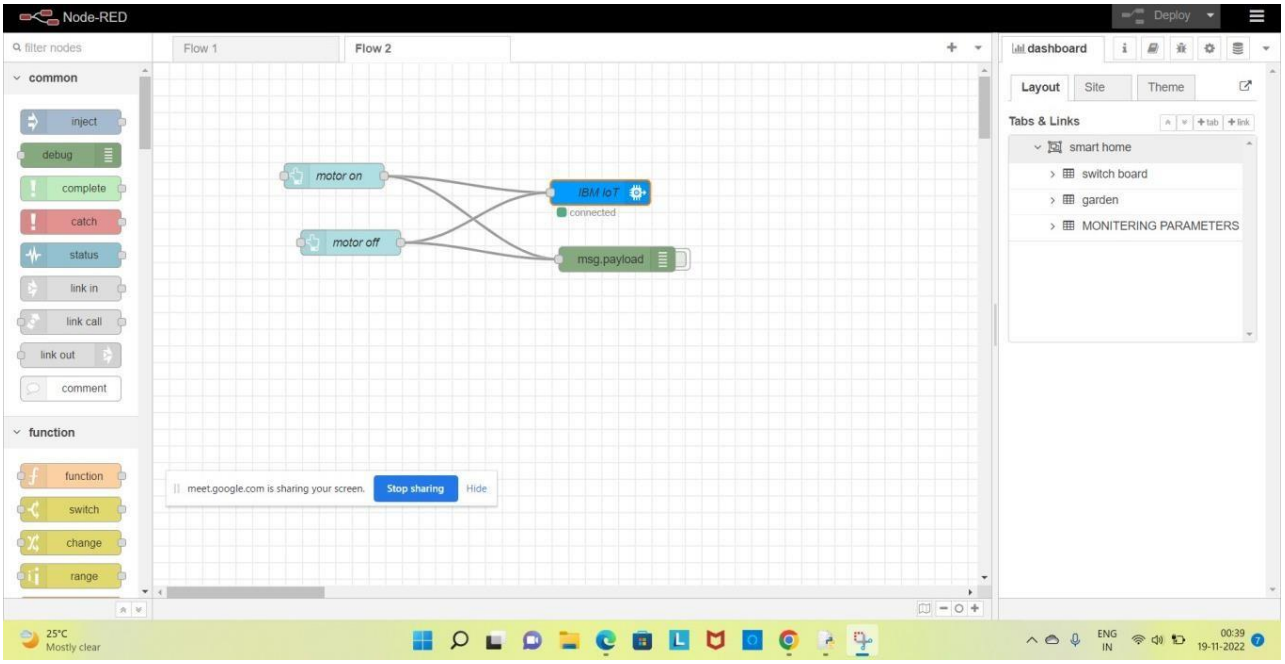

SMART CROP PROTECTION
81
water moisture
9
temp
7
Type here to search
27°C
ENG
02:29
13-11-2022

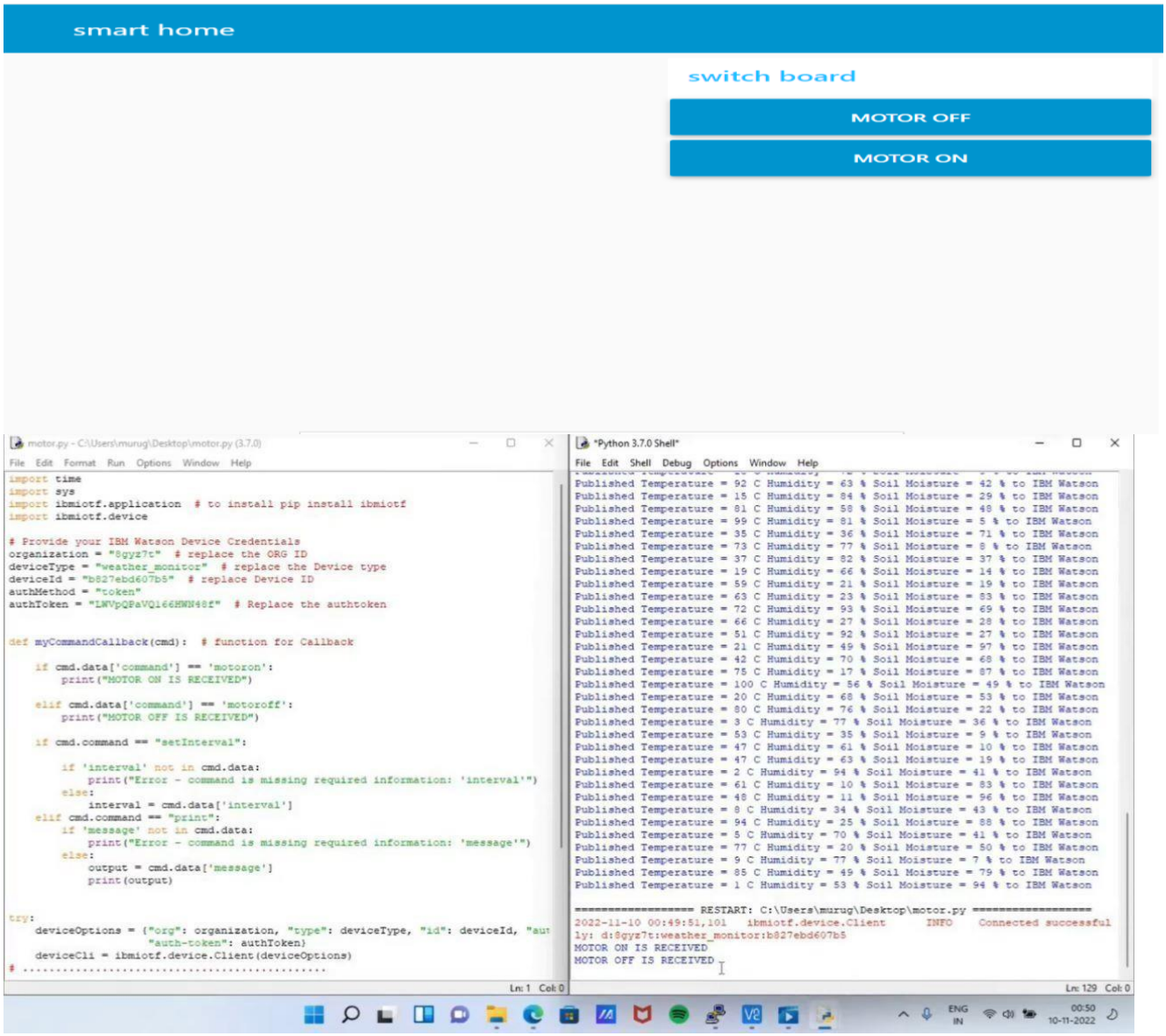

2:40
66%
HALL AC
TEMPERATURE:56
HUMIDITY: 99
78 NA

Detection of movement of birds and animals



Motor on and off




smart home
switch board
MOTOR OFF
MOTOR ON
motor.py - C:\Users\murug\Desktop\motor.py (3.7.0)
File Edit Format Run Options Window Help
import time
import sys
import ibmiotf.application # to install pip install ibmiotf
import ibmiotf.device
# Provide your IBM Watson Device Credentials
organization = "8gyz7t" # replace the ORG ID
deviceType = "weather_monitor" # replace the Device type
deviceId = "b827ebd607b5" # replace Device ID
authMethod = "token"
authToken = "LWVpQPaVQ166HWW48f" # Replace the authtoken
def myCommandCallback(cmd): # function for Callback
if cmd.data['command'] == 'motoron':
print("MOTOR ON IS RECEIVED")
elif cmd.data['command'] == 'motoroff':
print("MOTOR OFF IS RECEIVED")
if cmd.command == "setInterval":
if 'interval' not in cmd.data:
print("Error - command is missing required information: 'interval'")
else:
interval = cmd.data['interval']
elif cmd.command == "print":
if 'message' not in cmd.data:
print("Error - command is missing required information: 'message'")
else:
output = cmd.data['message']
print(output)
try:
deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "aut
"auth-token": authToken}
deviceCli = ibmiotf.device.Client(deviceOptions)
Ln: 1 Col: 0
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Published Temperature = 92 C Humidity = 63 % Soil Moisture = 42 % to IBM Watson
Published Temperature = 15 C Humidity = 84 % Soil Moisture = 29 % to IBM Watson
Published Temperature = 81 C Humidity = 58 % Soil Moisture = 48 % to IBM Watson
Published Temperature = 99 C Humidity = 81 % Soil Moisture = 5 % to IBM Watson
Published Temperature = 35 C Humidity = 36 % Soil Moisture = 71 % to IBM Watson
Published Temperature = 73 C Humidity = 77 % Soil Moisture = 8 % to IBM Watson
Published Temperature = 37 C Humidity = 82 % Soil Moisture = 37 % to IBM Watson
Published Temperature = 19 C Humidity = 66 % Soil Moisture = 14 % to IBM Watson
Published Temperature = 59 C Humidity = 21 % Soil Moisture = 19 % to IBM Watson
Published Temperature = 63 C Humidity = 23 % Soil Moisture = 83 % to IBM Watson
Published Temperature = 72 C Humidity = 93 % Soil Moisture = 69 % to IBM Watson
Published Temperature = 66 C Humidity = 27 % Soil Moisture = 28 % to IBM Watson
Published Temperature = 51 C Humidity = 92 % Soil Moisture = 27 % to IBM Watson
Published Temperature = 21 C Humidity = 49 % Soil Moisture = 97 % to IBM Watson
Published Temperature = 42 C Humidity = 70 % Soil Moisture = 68 % to IBM Watson
Published Temperature = 75 C Humidity = 17 % Soil Moisture = 87 % to IBM Watson
Published Temperature = 100 C Humidity = 56 % Soil Moisture = 49 % to IBM Watson
Published Temperature = 20 C Humidity = 68 % Soil Moisture = 53 % to IBM Watson
Published Temperature = 80 C Humidity = 76 % Soil Moisture = 22 % to IBM Watson
Published Temperature = 3 C Humidity = 77 % Soil Moisture = 36 % to IBM Watson
Published Temperature = 53 C Humidity = 35 % Soil Moisture = 9 % to IBM Watson
Published Temperature = 47 C Humidity = 61 % Soil Moisture = 10 % to IBM Watson
Published Temperature = 47 C Humidity = 63 % Soil Moisture = 19 % to IBM Watson
Published Temperature = 2 C Humidity = 94 % Soil Moisture = 41 % to IBM Watson
Published Temperature = 61 C Humidity = 10 % Soil Moisture = 83 % to IBM Watson
Published Temperature = 48 C Humidity = 11 % Soil Moisture = 96 % to IBM Watson
Published Temperature = 8 C Humidity = 34 % Soil Moisture = 43 % to IBM Watson
Published Temperature = 94 C Humidity = 25 % Soil Moisture = 88 % to IBM Watson
Published Temperature = 5 C Humidity = 70 % Soil Moisture = 41 % to IBM Watson
Published Temperature = 77 C Humidity = 20 % Soil Moisture = 50 % to IBM Watson
Published Temperature = 9 C Humidity = 77 % Soil Moisture = 7 % to IBM Watson
Published Temperature = 85 C Humidity = 49 % Soil Moisture = 79 % to IBM Watson
Published Temperature = 1 C Humidity = 53 % Soil Moisture = 94 % to IBM Watson
========== RESTART: C:\Users\murug\Desktop\motor.py ==========
2022-11-10 00:49:51,101 ibmiotf.device.Client INFO Connected successful
ly: d:8gyz7t:weather_monitor:b827ebd607b5
MOTOR ON IS RECEIVED
MOTOR OFF IS RECEIVED
Ln: 129 Col: 0
ENG
IN
00:50
10-11-2022

Chapter – 9

ADVANTAGES AND DISADVANTAGES

Advantages:

- Farms can be monitored and controlled remotely.
- Increase in convenience to farmers.
- Less Manpower.
- Better standards of living.

Disadvantages:

- Lack of internet/connectivity issues.
- Added cost of internet and internet gateway infrastructure.
- Farmers wanted to adapt the use of WebApp.

Chapter – 10

CONCLUSION

This system focuses on developing devices and tool to manage, display and alert the users using the advantages of a wireless sensor network system. It aims at making agriculture smart using automation and IoT. The cloud computing devices are used at the end of the system that can create a whole computing system from sensors to tools that observe data from agriculture field. It proposes a novel methodology for smart farming by including a smart sensing system and smart irrigator system through wireless communication technology. Thus, the objective of the project to implement an IoT system in order to help farmers to control and monitor their farms has been implemented successfully.

Chapter – 11

FUTURE SCOPE

Agriculture domains encounters with many challenges starting from soil parameters, seed sowing, crop growth and its quality, weed handling, disease management till harvesting and storing crop. Artificial intelligence driven techniques along with other available tools and automation can address these challenges and proven the revolution in agriculture. Most popular AI application in agriculture is use of Robot and Drones, they perform almost all task like humans even at a faster rate with accuracy. From literature review it is clear that precision farming is probable by integrating sensors, cameras, data analytics, GPS and remote sensing. Image recognitions software's, IoT sensors can be used for disease recognition at primary stages and hence crop health can be supervised which increases superior quality production with minimum loss. Table 1 demonstrate the various applications in view of Smart Agriculture for improved evolution as well as superiority. Still there are several challenges associated with AI and IoT application in smart agriculture which is the promising future to be explored area for researchers. Some of major challenges are: • Awareness issues • Hardware implementation challenges • Cost of software and hardware • Network management • Energy management • Privacy issues • Security challenges • Interoperability of systems with the induction of Computer vision, Deep learning, Big data also agriculture sector has influenced a lot. Researchers can integrate IoT sensors along with smart systems and computational optimization algorithms to overcome the limitations/shortcomings. Smart Agriculture has a budding potential towards productivity, precision, optimization, adaptive resource management and intelligent food traceability. It will contribute to environment also in terms of efficient use of water, prevent disease contamination and precise use of pesticides.

Chapter – 12

APPENDIX

Source Code

```
import random
import ibmiotf.device
from time import sleep
import sys
#IBM Watson Device Credentials.
organization = "ory9vy"
deviceType = "Raspery"
deviceId = "1206"
authMethod = "token"
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)
    if success:
        success = deviceCli.publishEvent("Temperature sensor", "json", temp_data, qos=0)
        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("")
#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)
    sleep(1)
    if success:
        print( 'Published alert4 : ' , "Flame is detected crops are in danger,sprinklers turned
ON","to IBM Watson")
#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("")
#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("")
#command recived by farmer
deviceCli.commandCallback =  myCommandCallback
# Disconnect the device and application from the cloud
deviceCli.disconnect()
```

GitHub

https://github.com/IBM-EPBL/IBM-Project-31913-1660206388