

**IoT BASED SMART CROP PROTECTION SYSTEM  
FOR AGRICULTURE**

**PROJECT REPORT**

*Submitted By*

**TEAM ID PNT2022TMID42171**

**PANNEERSELVAM.M (623519106023)  
(TEAMLEAD)**

**PON KUMAR. M (623519106024)**

**MAHAVISHNU.R (623519106016)**

*in partial fulfilment for the award of the*

*degree of*

**BACHELOR OF TECHNOLOGY**

*IN*

**AVS COLLEGE OF TECHNOLOGY,  
SALEM-636106**

**ANNA UNIVERSITY: CHENNAI 600 025**

**NOVEMBER 2022**

# **TABLE OF CONTENTS**

<b>CHAPTER NO</b>	<b>TITLE</b>
<b>1</b>	<b>INTRODUCTION</b>  1.1 PROJECT OVERVIEW 1.2 PURPOSE
<b>2</b>	<b>LITERATURE SURVEY</b>  2.1 EXISTING PROBLEM  2.2 REFERENCES  2.3 PROBLEM STATEMENT DEFINITION
<b>3</b>	<b>IDEATION AND PROPOSED SOLUTION</b>  3.1 EMPATHY MAP CANVAS  3.2 IDEATION AND BRAINSTORMING  3.3 PROPOSED SOLUTION  3.4 PROBLEM-SOLUTION FIT
<b>4</b>	<b>REQUIREMENT ANALYSIS</b>  4.1 FUNCTIONAL REQUIREMENT  4.2 NON- FUNCTIONAL REQUIREMENT
<b>5</b>	<b>PROJECT DESIGN</b>  5.1 DATA FLOW DIAGRAM  5.2 SOLUTION AND TECHNOLOGY ARCHITECTURE

### 5.3 USER-STORIES

<b>6</b>	<b>PROJECT PLANNING AND SCHEDULING</b>
	6.1 SPRINT PLANNING AND ESTIMATION
	6.2 SPRINT DELIVERY SCHEDULE
	6.3 REPORT FROM JIRA
<b>7</b>	<b>CODING AND SOLUTIONS</b>
	7.1 FEATURE 1
	7.2 FEATURE 2
	7.3 DATABASE SCHEMA
<b>8</b>	<b>TESTING</b>
	8.1 TEST CASES
	8.2 USER ACCEPTANCE TESTING
<b>9</b>	<b>RESULT</b>
	9.1 PERFORMANCE METRICS
<b>10</b>	<b>ADVANTAGES AND DISADVANTAGES</b>
<b>11</b>	<b>CONCLUSION</b>
<b>12</b>	<b>FUTURE SCOPE</b>
<b>13</b>	<b>APPENDIX</b>
	13.1 SOURCE CODE
	13.2 GITHUB & PROJECT DEMO LINK
<b>14</b>	<b>REFERENCES</b>

# CHAPTER 1

## INTRODUCTION

### 1.1 Project Overview

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

### 1.2 Purpose

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

## **CHAPTER 2**

### **LITERATURE SURVEY**

#### **2.1 Existing Problem**

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

#### **2.2 REFERENCES**

Shishir Bagal , Krunal Mahajan , Riya Parate , Ekta Zade , Shubham Khante (2021) have investigated the title of “Smart Crop Protection System Using IOT” . The Smart protection system defines that this project help to farmer for the protection of a farm. We have designed this project for the only secure from animals but we this project have the provision to secure from the human beings also. This can achieve by the help of IOT device that we are discuss in this paper. The SCPS work on the battery so that this project can be easily portable and also we are add

solar panels and converter modules this can help the battery to charge from solar energy. The IOT device is used to indicate the farmer by a message while someone enter into the farm and we are used SD card module that helps to store a specified sound to fear the animals.

## **2.3 Problem Statement Definition**

Most of the farmers are facing many problems nowadays due to many reasons. Our problem to solve is the invasion of various species such as birds and animals that harm the crops that are being cultivated. Various types of species such as birds and animals come to the cultivation field according to the crop that is being cultivated and also according to the season of cultivation. Some wild animals enter the field during night times when the field is near a forest region or when the farm cultivates some fruits and other crops that attract animals.

## IDEATION AND PROPOSED SOLUTION

### 3.1 Empathy Map Canvas

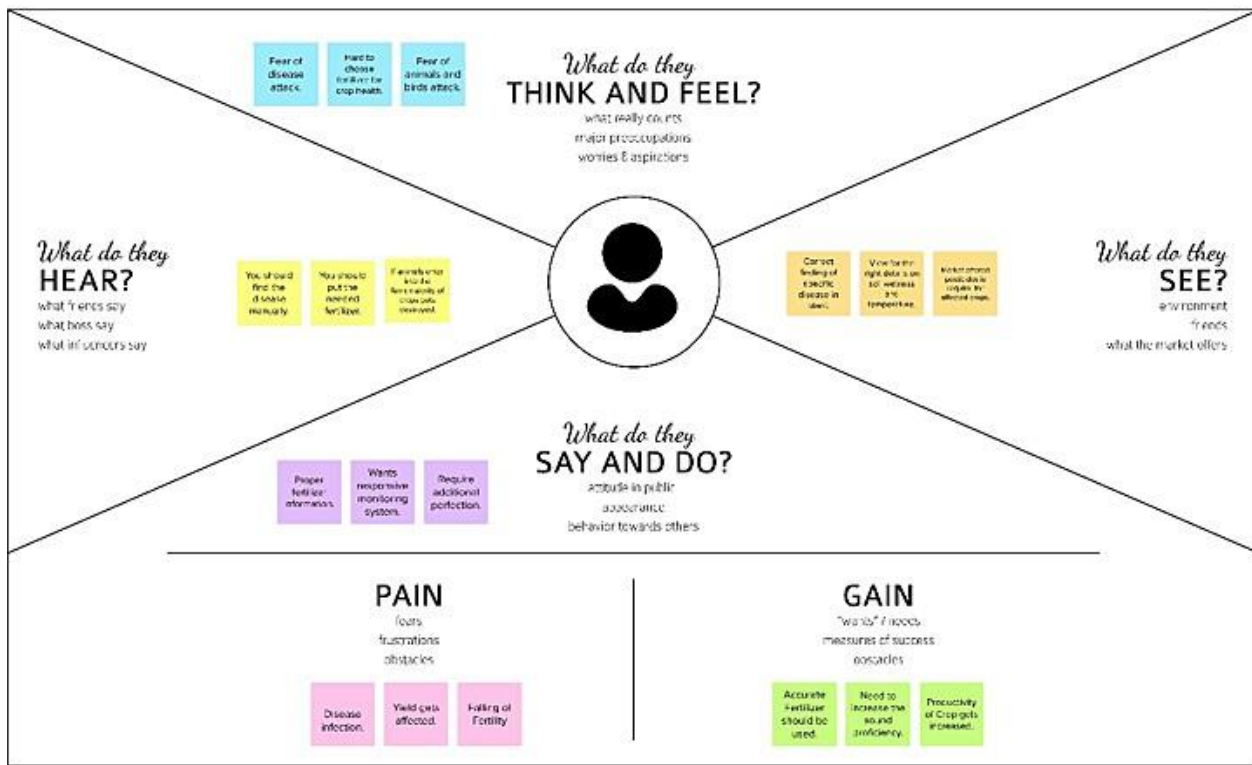


Figure3.1

### 3.2 Ideation & Brainstroming

Crops in farms are many times ravaged by local animals like buffaloes, cows, goats, birds, and fire 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 and fire. This is aarduino Uno based system using microcontroller. This system uses a motion sensor to detect wild animals approaching near the field and smoke sensor to detect the fire. In such a case the sensor signals the microcontroller to take action. If there is a smoke, it immediately turns ON the motor. This ensures complete safety of crops from animals and from fire thus protecting the farmer's loss. This is aarduino. Uno

based system using microcontroller. This system uses a motion sensor to detect wild animals approaching near the field and smoke sensor to detect the fire. In such a case the sensor signals the microcontroller to take action.

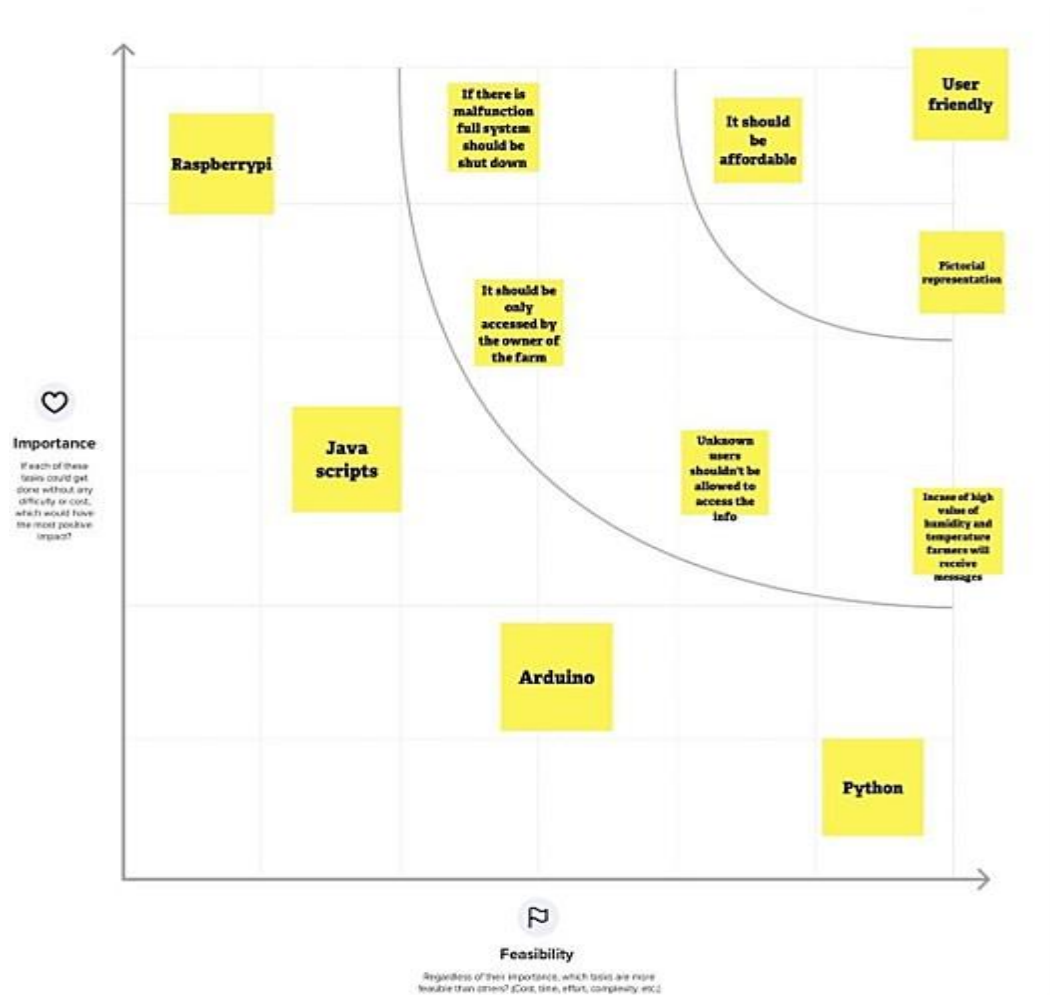


Figure3.2

### 3.3 Proposed Solution

Moisture sensor is interfaced with Arduino Microcontroller to measure the moisture level in soil and relay is used to turn ON and 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. The optimum temperature required for crop cultivation is maintained using sprinklers. IOT based fertilizing methods are followed, to minimize the



negative effects on growth of crops while using fertilizers.

The PIR sensor and UV sensors detect the motion of animals and birds for a particular arrange. The thermal radiation temperature of humans at different ages is fed to the system so there won't be any false alarm. If any invasion of animals is found, the camera focuses on the region and the processed image is sent to the farmer. After seeing the image of the animal that entered, they can decide to take any actions. A fence is built around the field to prevent large animals from entering where the sensors are placed at all the corners of the field fully covering the entire region.

### 3.4 PROBLEM-SOLUTION FIT

Define CL, PR into CL Focus on PR, TR into BE, understand RC Identify strong TR & EM	<b>1. CUSTOMER SEGMENT(S)</b> <span>CS</span> <ul style="list-style-type: none"><li>Farmers who trying to protect Crops from various problems</li></ul>	<b>6. CUSTOMER LIMITATIONS</b> <span>CL</span> <small>RC, BUDGET, DEVICES</small> <ul style="list-style-type: none"><li>Limited supervision.</li><li>Limited financial Constrains.</li><li>Lack of man power.</li></ul>	<b>5. AVAILABLE SOLUTIONS</b> <span>AS</span> <small>PLUSES &amp; MINUSES</small> <ul style="list-style-type: none"><li>Automation in irrigation.</li><li>CCTV Camera to monitor and supervise the crops.</li><li>Alarm system to give alert while animals attacks the crops.</li></ul>	Explore AS, offer services Focus on PR, TR into BE, understand RC Extract online & offline CH at BE
	<b>2. PROBLEMS / PAINS</b> <span>PR</span> <small>ITS FREQUENCY</small> <ul style="list-style-type: none"><li>Crops are not irrigated properly.</li><li>Improper maintenance of crops.</li><li>Lack of knowledge among farmers in usage of fertilizers and hence crops are affected.</li><li>Requires protecting Crops from Wild animals attacks, birds and pests.</li></ul>	<b>9. PROBLEM ROOT / CAUSE</b> <span>RC</span> <ul style="list-style-type: none"><li>Due to insufficient labour forces.</li><li>Due to various environmental factors such as temperature climate, topography and soil quality which results in crop destruction.</li><li>Due to high ammonia, urea, potassium and high PH level fertilizers.</li><li>Crops are damaged and it affects growth.</li></ul>	<b>7. BEHAVIOR</b> <span>BE</span> <small>ITS INTENSITY</small> <ul style="list-style-type: none"><li>Asks suggestions from surrounding peoples and implement the recent technologies.</li><li>Consumes more time in crop land.</li><li>Searching for an alternative solution for an existing solution.</li></ul>	
	<b>3. TRIGGERS TO ACT</b> <span>TR</span> <ul style="list-style-type: none"><li>By seeing surrounding Crop land with installing machineries.</li><li>Hearing about innovative technologies and effective solutions.</li></ul>	<b>10. YOUR SOLUTION</b> <span>SL</span> <ul style="list-style-type: none"><li>Moisture sensor is interfaced with Arduino Microcontroller to measure the moisture level in soil and relay is used to turn ON and OFF the motor pump for managing the excess water level. It will be updated to authorities through IOT.</li><li>Temperature sensor connected to microcontroller is used to monitor the temperature in the field. The optimum temperature required for crop cultivation is maintained using sprinklers.</li><li>IOT based fertilizing methods are followed, to minimize the negative effects on growth of crops while using fertilizers.</li><li>Image processing techniques with IOT is followed for crop protection against animal attacks.</li></ul>	<b>8. CHANNELS of BEHAVIOR</b> <span>CH</span> <div>ONLINE Using different platforms /social media to describe the working and uses of smart Crop protection devices.</div> <div>OFFLINE<ul style="list-style-type: none"><li>Giving awareness among farmers about the application of the device.</li></ul></div>	
	<b>4. EMOTIONS</b> <span>EM</span> <small>BEFORE / AFTER</small> <ul style="list-style-type: none"><li>Mental frustrations due to insufficent production of crops.</li><li>Felt smart enough to follow the available technologies with minimum cost.</li></ul>			

Figure3.4

# **CHAPTER 4**

## **REQUIREMENT ANALYSIS**

### **4.1 Functional Requirement**

Following are the functional requirements of the proposed solution.

1. User Registration ,Registration through Form Registration through Gmail Registration through LinkedIN
2. User Confirmation ,Confirmation via Email Confirmation via OTP
3. Tracking Expense Helpful insights about money management
4. Alert Message Give alert mail if the amount exceeds the budget limit Category This application shall allow users to add categories of their expenses

### **4.2 Non Functional requirement**

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

1. Usability You will able to allocate money to different priorities and also help you to cut down on unnecessary spending
2. Security More security of the customer data and bank account details.
3. Reliability Used to manage his/her expense so that the user is the path of financial stability. It is categorized by week, month, and year and also helps to see more expenses made. Helps to define their own categories.
4. NFR-4 Performance The types of expense are categories along with an option .Throughput of the system is increased due to light weight database support.
5. NFR-5 Availability Able to track business expense and monitor important for maintaining healthy cash flow. NFR-6 Scalability The ability to appropriately handle increasing demands.

# CHAPTER 5

## PROJECT DESIGN

### 5.1 DATA FLOW DIAGRAM

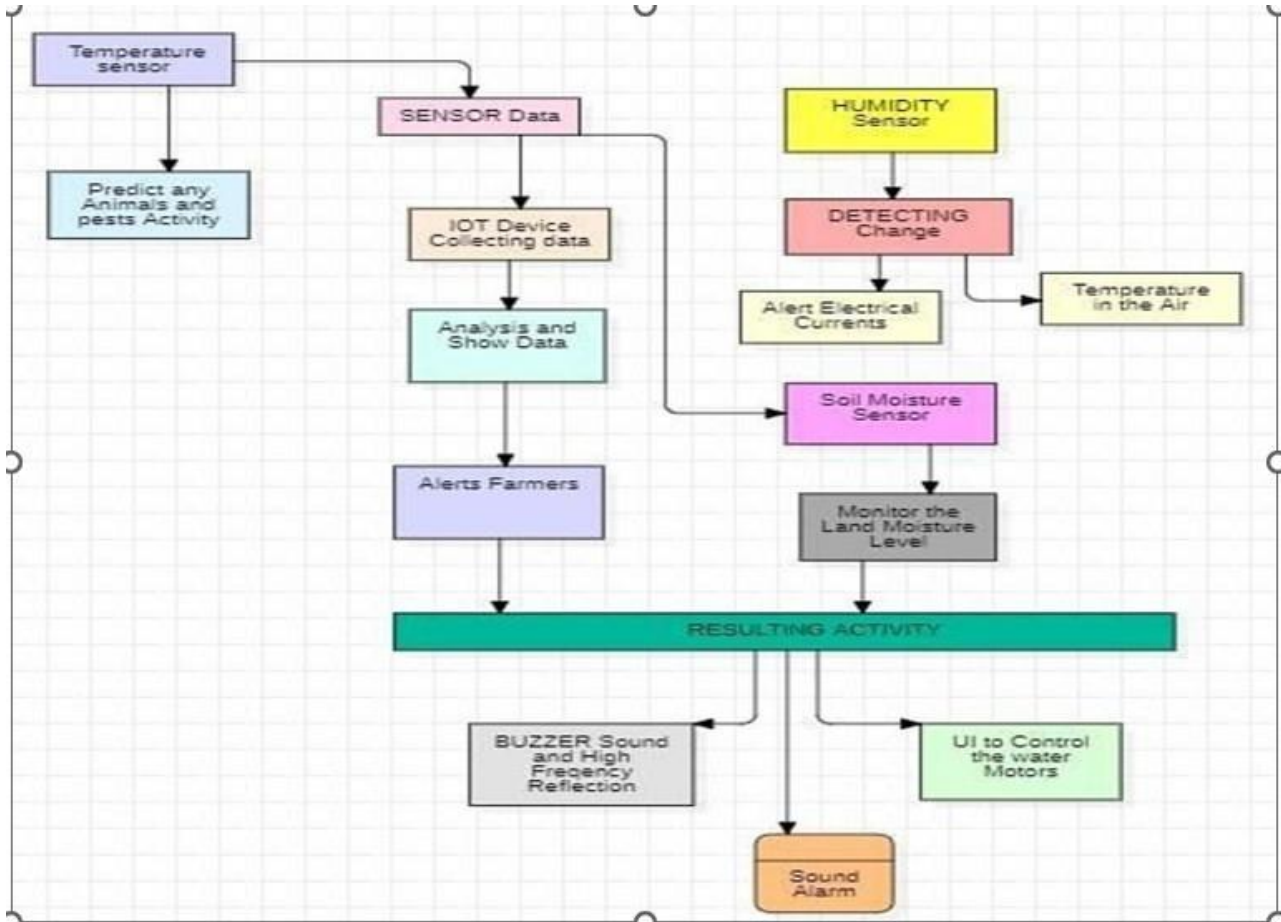


Figure5.1

### 5.2 Technical Architecture

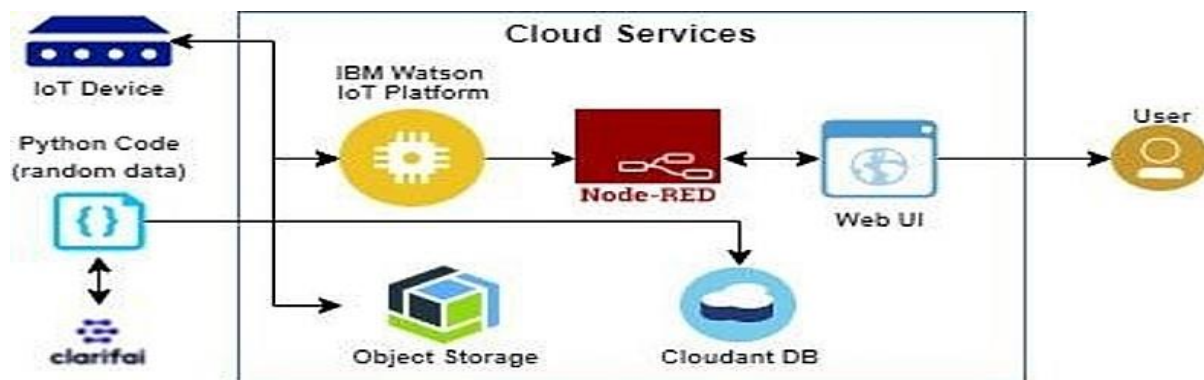
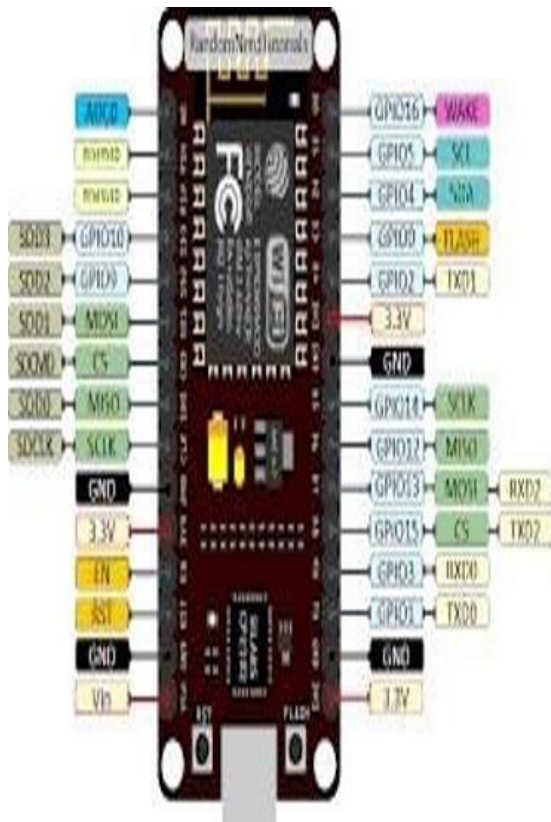


Figure5.2



**Figure5.2.1**



**Figure5.2.2**



## 5.3 USER-STORIES

SPRINT	FUNCTIONAL REQUIREMENT	USER STORY NUMBER	USER STORY/TASK	STORY POINTS	PRIORITY
Sprint-1		US-1	Create the IBM Cloud services which are being used in this project.	7	high
Sprint-1		US-2	Create the IBM Cloud services which are being used in this project.	7	high
Sprint-2		US-3	IBM Watson IoT platform acts as the mediator to connect the web application to IoT devices, so create the IBM Watson IoT platform.	5	medium
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	6	high
Sprint-3		US-1	Configure the connection security and create API keys that are used in the Node-RED service for accessing the IBM IoT Platform.	10	high
Sprint-3		US-3	Create a Node-RED service	8	high
Sprint-3		US-2	Develop a python script to publish random	6	medium

			sensor data such as temperature, moisture, soil and humidity to the IBM IoT platform		
Sprint-3		US-1	After developing python code, commands are received just print the statements which represent the control of the devices.	8	high
Sprint-4		US-3	Publish Data to The IBM Cloud	5	high
Sprint-4		US-2	Create Web UI in Node- Red	8	high
Sprint-4		US-1	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	6	high

## CHAPTER 6

### PROJECT PLANNING & SCHEDULING

#### 6.1 Sprint Planning & Estimation

<b>Sprint</b>	<b>Functional Requirement (Epic)</b>	<b>User Story Number</b>	<b>User Story / Task</b>	<b>Story Points</b>	<b>Priority</b>	<b>Team Members</b>
Sprint1	SensorData (python script)	USN-1	The Data of sensor which are feed to the Raspberrypi. Here we are using python script to generate a random sensor data.	3	High	Panneer selvam.M (Team leader)
Sprint1	Automation (python script)	USN-2	Some activities are made to automation to overcome insufficient labour force in the field. Hence that also included in python script to Implement automation .	5	High	Maha Vishnu (Team Member)
Sprint2	IBM IOT platform	USN-3	To send the raspberrypi data to IOT platform, we create an IBM IOT platform and connect the raspberrypi to the device created in IBM IOT.	5	High	Pon kumar (Team Member)

Sprint3	Node RED service	USN-4	To access the IBM IOT platform from external application or from external UI Node red service is established.	5	High	Maha vishnu (Team Member)
Sprint3	API Key	USN-5	To protect the IBM IOT platform creating an API Key.		High	Pon kumar (Team Member)
Sprint4	User Application	USN-6	To monitor and control the field sensors the User is provided with an User application created by MIT app Inventor	8	High	Panneer selvam (Team Leader) Maha Vishnu (Team Member)

## 6.2 SPRINT DELIVERY SCHEDULE

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	8	6 Days	24 Oct 2022	29 Oct 2022	8	29 Oct 2022
Sprint-2	5	6 Days	31 Oct 2022	05 Nov 2022	5	05 Nov 2022
Sprint-3	8	6 Days	07 Nov 2022	12 Nov 2022	8	12 Nov 2022

Sprint-4	8	6 Days	14 Nov 2022	19 Nov 2022	8	19 Nov 2022
----------	---	--------	-------------------	----------------	---	----------------

## 6.3 REPORT FROM JIRA    REQUIRED SOFTWARE

1. CLARIFAI
2. IBMWATSONIOTPLATFORM
3. PYTHONIDLE
4. NODERED
5. MITAPPINVENTOR **CLARIFAI:**

Clarifai provides an end-to-end platform with the easiest to use UI and API in the market. Clarifai Inc. is an artificial intelligence (AI) company that specializes in computer vision and uses machine learning and deep neural networks to identify and analyse images and videos.

The company offers its solution via API, mobile SDK, and on-premise solutions.

### STEP 1:



clarifai - Google Search

Clarifai Community

The World's AI | Clarifai

The World's AI | Clarifai


+


clarifai.com/signup

1,000 free operations per month.  
Pre-trained models to get you started.  
Easy-to-use, drag and drop UI for faster training, evaluation and deployment.  
SOTA transfer learning to build custom models faster.

AND GAIN ACCESS NOW

TRUSTED BY ENTERPRISES  
WORLDWIDE



 **clarifai**

## Log in to Clarifai

or [signup for free.](#)

Email \*

panneer6374971009@gmail.com

Password \*


.....

LOG IN

[Forgot your password?](#)

About Help Center Terms of Service Privacy Policy

©2022 Clarifai, Inc.

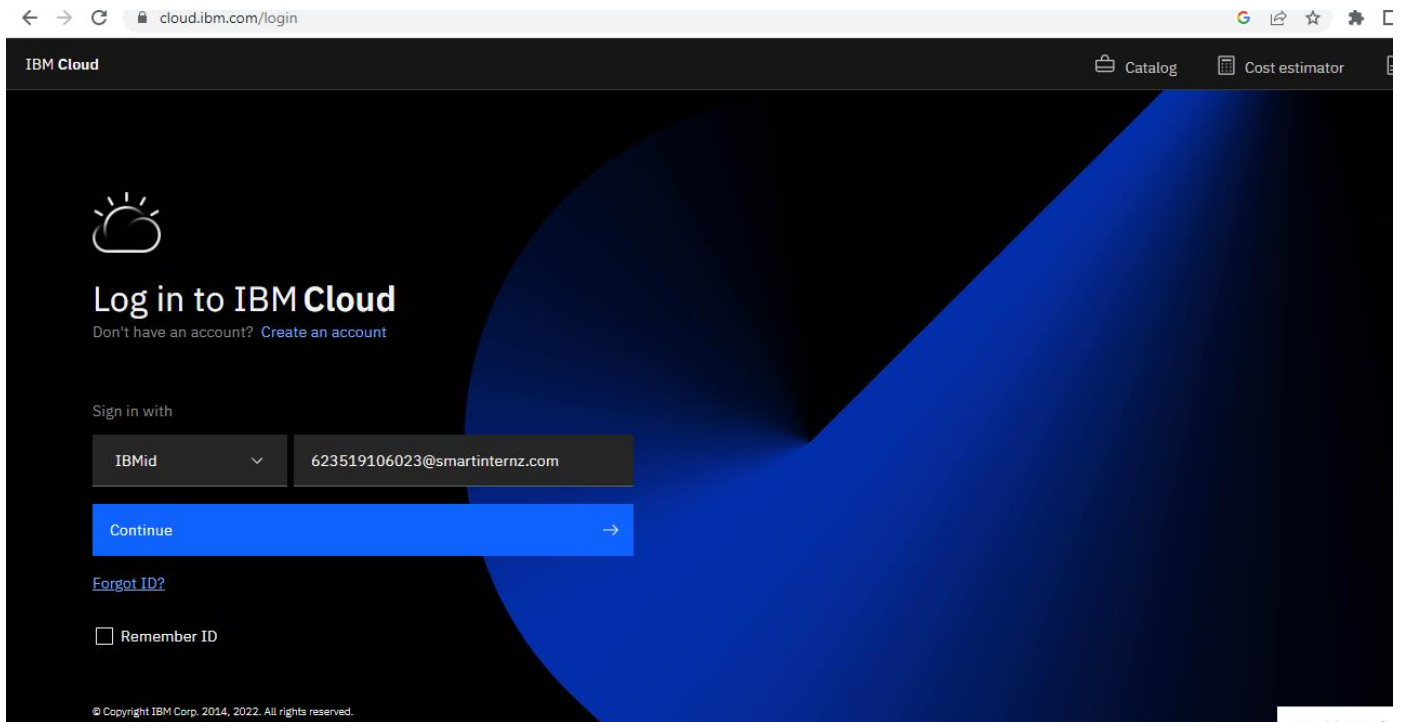


# IBM WATSON IoT PLATFORM:

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

1. IBM Watson IoT Platform
2. Node-RED Service
3. Cloudbant DB

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



LOGIN:

Figure6.3.3

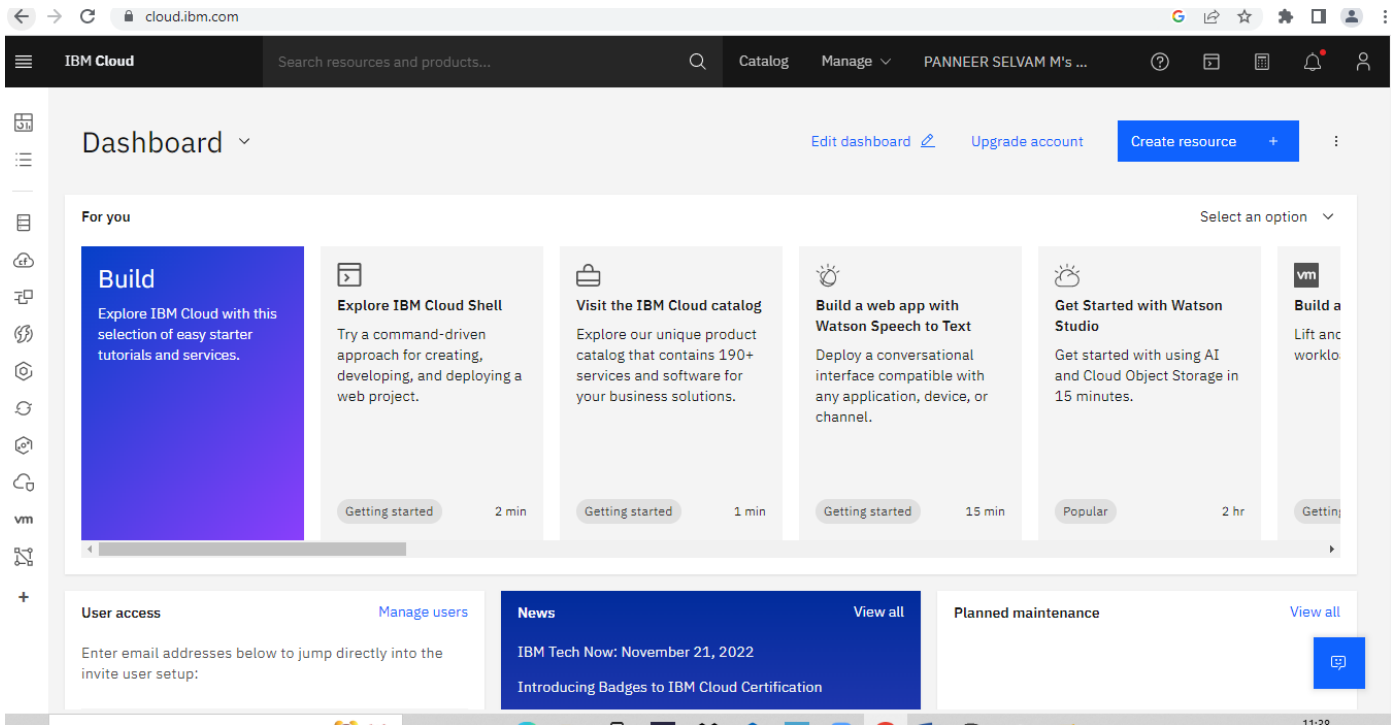
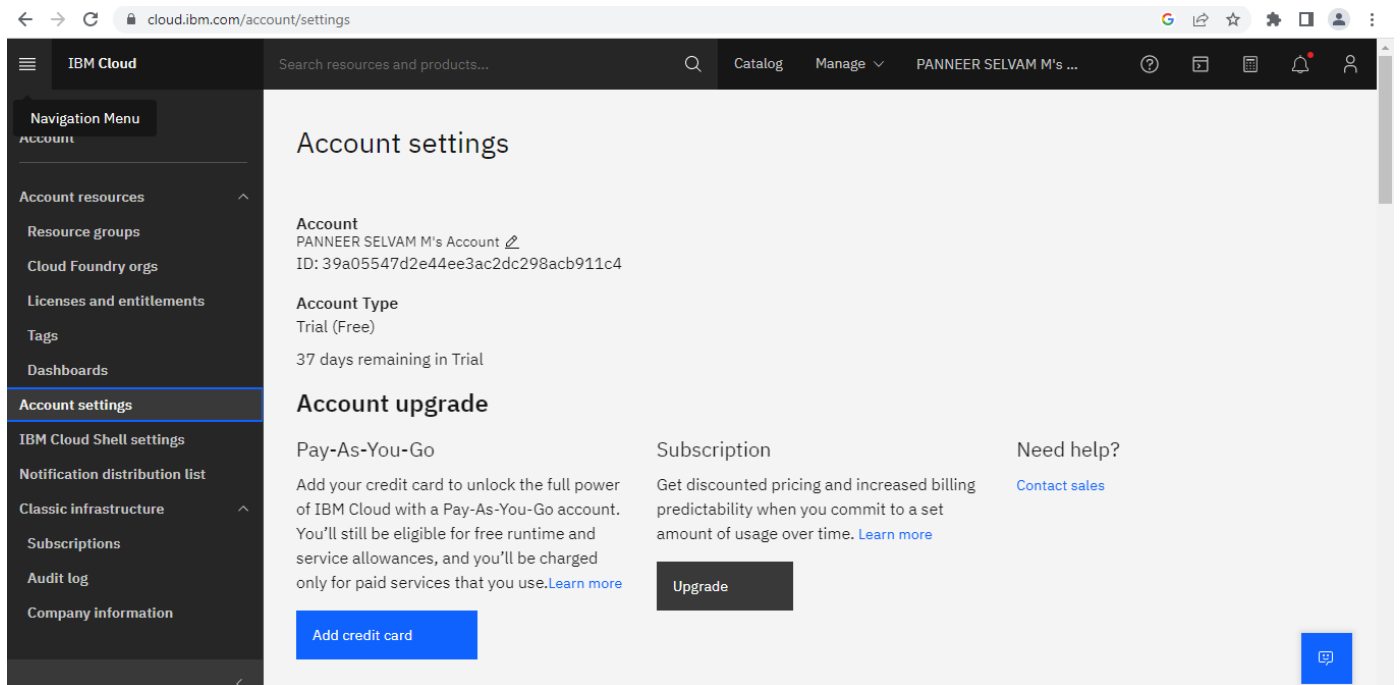


Figure6.3.4



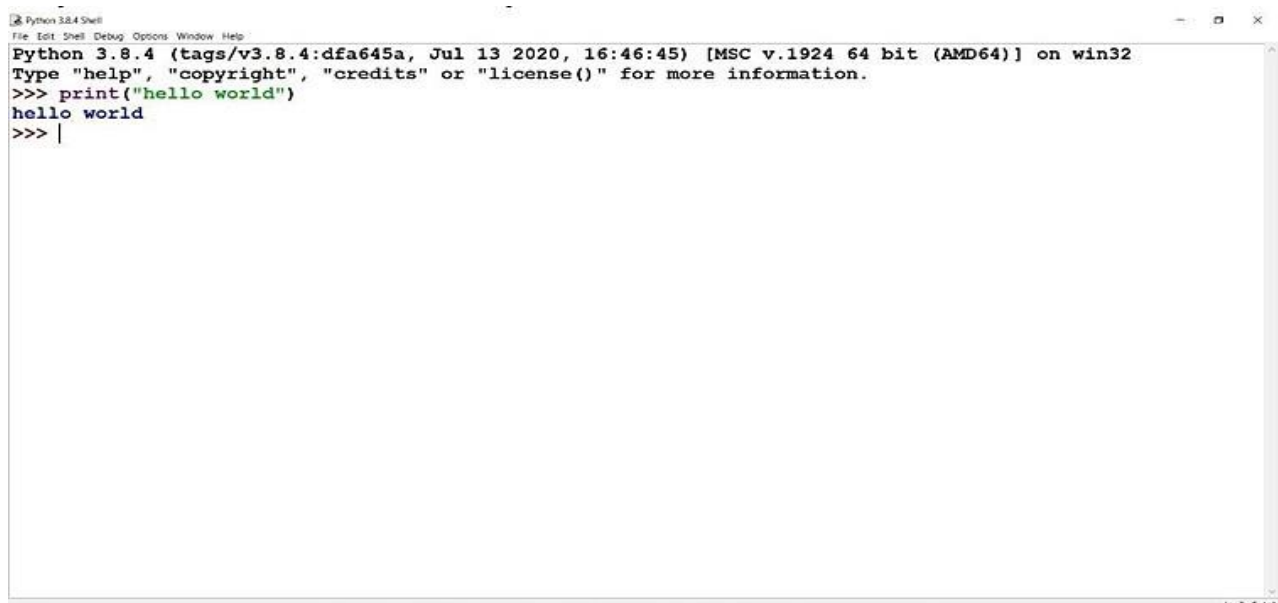
**Figure6.3.5**

## PYTHON IDLE INSTALISATION

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

### STEP 1:

Python is installed successfully



**Figure6.3.6**

### STEP 2:

1. The required python libraries are installed.
2. Watson IoT Python SDK to connect to IBM Watson IoT Platform using python code is installed

### 3. pip install wiotp-stk

```

C:\Users\swast> pip install wiotp-stk

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

C:\Users\swast> pip install wiotp-stk
Collecting wiotp-stk
  Downloading wiotp-stk-0.11.0.tar.gz (96 kB)
    |#####| 96 kB 130 kB/s
Collecting iso8601>=1.1.0
  Downloading iso8601-1.1.0-py3-none-any.whl (9.9 kB)
Requirement already satisfied: pytz>=2018.9 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from wiotp-stk) (2020.1)
Collecting pyyaml>=3.13
  Downloading PyYAML-5.4.0-cp38-cp38-win_amd64.whl (155 kB)
    |#####| 155 kB 126 kB/s
Collecting paho-mqtt>=1.5.0
  Downloading paho-mqtt-1.5.1.tar.gz (99 kB)
    |#####| 99 kB 172 kB/s
Collecting requests>=2.23.0
  Downloading requests-2.28.1-py3-none-any.whl (62 kB)
    |#####| 62 kB 155 kB/s
Collecting requests-toolbelt>=0.8.0
  Downloading requests-toolbelt-0.10.1-py3.py3-none-any.whl (54 kB)
    |#####| 54 kB 281 kB/s
Collecting charset-normalizer<3, >=2
  Downloading charset-normalizer-2.1.1-py3-none-any.whl (39 kB)
Collecting idna<4, >=2.5
  Downloading idna-3.4-py3-none-any.whl (61 kB)
    |#####| 61 kB 40 kB/s
Collecting certifi>=2017.4.17
  Downloading certifi-2022.9.24-py3-none-any.whl (181 kB)
    |#####| 181 kB 261 kB/s
Collecting urllib3<1.22, >=1.21.1
  Downloading urllib3-1.26.12-py3-none-any.whl (148 kB)
    |#####| 148 kB 177 kB/s
Building wheels for collected packages: wiotp-stk, paho-mqtt
  Building wheel for wiotp-stk (setup.py) ... done
  Created wheel for wiotp-stk: filename=wiotp-stk-0.11.0-py3-none-any.whl size=97110 sha256=2f750f4e9164044a4c53080c81c3938d0a7415cd751cc52423a130211571
  Stored in directory: c:\users\swast\appdata\local\pip\cache\wheels\46\41\69\352062eb12981d66cd12ec6d3835d1b53d0ef4412cdcfa1d9
  Building wheel for paho-mqtt (setup.py) ... done
  Created wheel for paho-mqtt: filename=paho-mqtt-1.5.1-py3-none-any.whl size=65420 sha256=6d15d6cb401fc2e16c9120be42f7b6322795180013f91a32c25e220b10064b
  Stored in directory: c:\users\swast\appdata\local\pip\cache\wheels\6a\40\01\c85c827e9d9367ec54704b371e56e795d0acc013aadc9f
Successfully built wiotp-stk paho-mqtt
Installing collected packages: iso8601, pyyaml, paho-mqtt, charset-normalizer, idna, certifi, urllib3, requests, requests-toolbelt, wiotp-stk
Successfully installed certifi-2022.9.24 charset-normalizer-2.1.1 idna-3.4 iso8601-1.1.0 paho-mqtt-1.5.1 pyyaml-5.4 requests-2.28.1 requests-toolbelt-0.10.1 urllib3-1.26.12 wiotp-stk-0.11.0
WARNING: You are using pip version 20.1.1, however, version 22.3 is available.
You should consider upgrading via the 'c:\users\swast\appdata\local\programs\python\python38\python.exe -m pip install --upgrade pip' command.

C:\Users\swast>

```

Figure6.3.7

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

```

C:\Users\swast> pip install --upgrade "ibm-watson>=5.0.0"

Collecting ibm-watson>=5.0.0
  Downloading ibm-watson-5.0.0.tar.gz (171 kB)
    |#####| 171 kB 142 kB/s
Installing build dependencies ... done
Getting requirements to build wheel ... done
Preparing wheel metadata ... done
Collecting ibm-cloud-sdk-core>=1.*, <=1.1.0
  Downloading ibm-cloud-sdk-core-1.1.0-py3-none-any.whl (83 kB)
    |#####| 83 kB 152 kB/s
Requirement already satisfied, skipping upgrade: requests<3.0, >=2.0 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from ibm-watson>=5.0.0) (2.28.1)
Collecting websockets-client>=1.1.0
  Downloading websockets-client-1.1.0-py3-none-any.whl (60 kB)
    |#####| 60 kB 191 kB/s
Requirement already satisfied, skipping upgrade: python-dateutil>=2.5.1 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from ibm-watson>=5.0.0) (2.8.1)
Collecting PyJWT<3.0, >=2.4.0
  Downloading PyJWT-2.6.0-py3-none-any.whl (20 kB)
Requirement already satisfied, skipping upgrade: urllib3<2.0.0, >=1.26.0 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from ibm-cloud-sdk-core>=1.*, <=1.1.0 ibm-watson>=5.0.0) (1.26.12)
Requirement already satisfied, skipping upgrade: certifi>=2017.4.17 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from requests<3.0, >=2.0 ibm-watson>=5.0.0) (2022.9.24)
Requirement already satisfied, skipping upgrade: charset-normalizer<3, >=2 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from requests<3.0, >=2.0 ibm-watson>=5.0.0) (2.1.1)
Requirement already satisfied, skipping upgrade: idna<4, >=2.5 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from requests<3.0, >=2.0 ibm-watson>=5.0.0) (3.4)
Requirement already satisfied, skipping upgrade: charset-normalizer<3, >=2 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from requests<3.0, >=2.0 ibm-watson>=5.0.0) (2.1.1)
Requirement already satisfied, skipping upgrade: idna<4, >=2.5 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from requests<3.0, >=2.0 ibm-watson>=5.0.0) (3.4)
Building wheels for collected packages: ibm-watson
  Building wheel for ibm-watson (PEP 517) ... done
  Created wheel for ibm-watson: filename=ibm-watson-5.0.0-py3-none-any.whl size=370740 sha256=30640b0c1c5d00b24e5c521168530db77dc19754ee5f075d0ff406956
  Stored in directory: c:\users\swast\appdata\local\pip\cache\wheels\14\1d\43\c03a353c386b7a570115f03a0d0ff62b20ac04e99802b7a2
Successfully built ibm-watson
Installing collected packages: PyJWT, ibm-cloud-sdk-core, websockets-client, ibm-watson
Successfully installed PyJWT-2.6.0 ibm-cloud-sdk-core-1.1.0 ibm-watson-5.0.0 websockets-client-1.1.0
WARNING: You are using pip version 20.1.1, however, version 22.3 is available.
You should consider upgrading via the 'c:\users\swast\appdata\local\programs\python\python38\python.exe -m pip install --upgrade pip' command.

C:\Users\swast>

```

Figure6.3.8

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

```
C:\Users\swast> pip install ibm-cos-sdk
Collecting ibm-cos-sdk
  Downloading ibm-cos-sdk-2.12.0.tar.gz (55 kB)
    |#####| 55 kB 611 kB/s
Collecting ibm-cos-sdk-core==2.12.0
  Downloading ibm-cos-sdk-core-2.12.0.tar.gz (956 kB)
    |#####| 956 kB 251 kB/s
Collecting ibm-cos-sdk-s3transfer==2.12.0
  Downloading ibm-cos-sdk-s3transfer-2.12.0.tar.gz (135 kB)
    |#####| 135 kB 242 kB/s
Collecting jmespath<1.0.0,>=0.10.0
  Downloading jmespath-0.10.0-py2.py3-none-any.whl (28 kB)
Collecting python-dateutil<3.0.0,>=2.8.1
  Downloading python-dateutil-2.8.1-py2.py3-none-any.whl (247 kB)
    |#####| 247 kB 261 kB/s
Requirement already satisfied: requests<3.0,>=2.27.1 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from ibm-cos-sdk-core==2.12.0->ibm-cos-sdk) (2.28.1)
Requirement already satisfied: urllib3<1.27,>=1.26.0 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from ibm-cos-sdk-core==2.12.0->ibm-cos-sdk) (1.26.12)
Requirement already satisfied: six<1.5 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from python-dateutil<3.0.0,>=2.8.1->ibm-cos-sdk-core==2.12.0->ibm-cos-sdk) (1.15.0)
Requirement already satisfied: certifi<2017.4.17 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from requests<3.0,>=2.27.1->ibm-cos-sdk-core==2.12.0->ibm-cos-sdk) (2021.9.24)
Requirement already satisfied: idna<3,>=2.5 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from requests<3.0,>=2.27.1->ibm-cos-sdk-core==2.12.0->ibm-cos-sdk) (3.4)
Requirement already satisfied: charset-normalizer<3,>=2 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from requests<3.0,>=2.27.1->ibm-cos-sdk-core==2.12.0->ibm-cos-sdk) (2.1.1)
Building wheels for collected packages: ibm-cos-sdk, ibm-cos-sdk-core, ibm-cos-sdk-s3transfer
  Building wheel for ibm-cos-sdk (setup.py) ... done
  Created wheel for ibm-cos-sdk: filename=ibm_cos_sdk-2.12.0-py3-none-any.whl size=77020 sha256=d6f03caad730d01209c70f7f93c70f4721a75731100f94c734e01cd10e
  Stored in directory: c:\users\swast\appdata\local\pip\Cache\wheels\42\15\fd\6a6f6b6d5a67246e030014100f9d39a7c6d180a226d6
  Building wheel for ibm-cos-sdk-core (setup.py) ... done
  Created wheel for ibm-cos-sdk-core: filename=ibm_cos_sdk_core-2.12.0-py3-none-any.whl size=562952 sha256=c7f8e90e7731d00073c3402317715d7131a7397b3b0d0a0010a1f99d7
  Stored in directory: c:\users\swast\appdata\local\pip\Cache\wheels\4\1\5\fd\6a6f6b6d5a67246e030014100f9d39a7c6d180a226d6
  Building wheel for ibm-cos-sdk-s3transfer (setup.py) ... done
  Created wheel for ibm-cos-sdk-s3transfer: filename=ibm_cos_sdk_s3transfer-2.12.0-py3-none-any.whl size=49709 sha256=47c3930a6d01a3d0f7c1d25d7211a6f83fac5f0275d9f4dc1a0f77
  Stored in directory: c:\users\swast\appdata\local\pip\Cache\wheels\0\7\5\fd\6a6f6b6d5a67246e030014100f9d39a7c6d180a226d6
Successfully built ibm-cos-sdk ibm-cos-sdk-core ibm-cos-sdk-s3transfer
Installing collected packages: jmespath, python-dateutil, ibm-cos-sdk-core, ibm-cos-sdk-s3transfer, ibm-cos-sdk
  Attempting uninstall: python-dateutil
    Found existing installation: python-dateutil 2.8.1
    Uninstalling python-dateutil-2.8.1:
      Successfully uninstalled python-dateutil-2.8.1
Successfully installed ibm-cos-sdk-2.12.0 ibm-cos-sdk-core-2.12.0 ibm-cos-sdk-s3transfer-2.12.0 jmespath-0.10.0 python-dateutil-2.8.1
WARNING: You are using pip version 19.1.1; however, version 22.1 is available.
You should consider upgrading via the 'c:\users\swast\appdata\local\programs\python\python38\python.exe -m pip install --upgrade pip' command.

C:\Users\swast>
```

Figure6.3.9

1. pip install -U ibm-cos-sdk

```
Command Prompt
ADMIN: You are using pip version 20.1.1; however, version 22.1 is available.
You should consider upgrading via the 'c:\users\swast\appdata\local\programs\python\python38\python.exe -m pip install --upgrade pip' command.

C:\Users\swast> pip install -U ibm-cos-sdk
Requirement already up-to-date: ibm-cos-sdk in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (2.12.0)
Requirement already satisfied, skipping upgrade: ibm-cos-sdk-s3transfer==2.12.0 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from ibm-cos-sdk) (2.12.0)
Requirement already satisfied, skipping upgrade: ibm-cos-sdk-core==2.12.0 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from ibm-cos-sdk) (2.12.0)
Requirement already satisfied, skipping upgrade: jmespath<1.0.0,>=0.10.0 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from ibm-cos-sdk) (0.10.0)
Requirement already satisfied, skipping upgrade: requests<3.0,>=2.27.1 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from ibm-cos-sdk-core==2.12.0->ibm-cos-sdk) (2.28.1)
Requirement already satisfied, skipping upgrade: urllib3<1.27,>=1.26.0 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from ibm-cos-sdk-core==2.12.0->ibm-cos-sdk) (1.26.12)
Requirement already satisfied, skipping upgrade: python-dateutil<3.0.0,>=2.8.1 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from ibm-cos-sdk-core==2.12.0->ibm-cos-sdk) (2.8.1)
Requirement already satisfied, skipping upgrade: charset-normalizer<3,>=2 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from requests<3.0,>=2.27.1->ibm-cos-sdk-core==2.12.0->ibm-cos-sdk) (2.1.1)
Requirement already satisfied, skipping upgrade: idna<3,>=2.5 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from requests<3.0,>=2.27.1->ibm-cos-sdk-core==2.12.0->ibm-cos-sdk) (3.4)
Requirement already satisfied, skipping upgrade: certifi<2017.4.17 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from requests<3.0,>=2.27.1->ibm-cos-sdk-core==2.12.0->ibm-cos-sdk) (2021.9.24)
Requirement already satisfied, skipping upgrade: six<1.5 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from python-dateutil<3.0.0,>=2.8.1->ibm-cos-sdk-core==2.12.0->ibm-cos-sdk) (1.15.0)
WARNING: You are using pip version 20.1.1; however, version 22.1 is available.
You should consider upgrading via the 'c:\users\swast\appdata\local\programs\python\python38\python.exe -m pip install --upgrade pip' command.

C:\Users\swast>
```

Figure6.3.10

1. pip install boto3

```
WARNING: You are using pip version 20.1.1; however, version 22.1 is available.  
You should consider upgrading via the 'c:\users\swast\appdata\local\program\python\python\python.exe -m pip install --upgrade pip' command.  
  
C:\Users\swast>pip install boto3  
Collecting boto3  
  Downloading boto3-1.26.0-py3-none-any.whl (132 kB)  
    |#####| 132 kB 148 kB/s  
Collecting s3transfer<0.7.0,>=0.6.0  
  Downloading s3transfer-0.6.0-py3-none-any.whl (79 kB)  
    |#####| 79 kB 113 kB/s  
Collecting botocore<3.30.0,>=1.29.0  
  Downloading botocore-1.29.0-py3-none-any.whl (9.8 MB)  
    |#####| 9.8 MB 2.2 MB/s  
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in c:\users\swast\appdata\local\program\python\python\lib\site-packages (from boto3) (0.10.0)  
Requirement already satisfied: urllib3<1.27,>=1.25.4 in c:\users\swast\appdata\local\program\python\python\lib\site-packages (from botocore<3.30.0,>=1.29.0>boto3) (1.26.12)  
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in c:\users\swast\appdata\local\program\python\python\lib\site-packages (from botocore<3.30.0,>=1.29.0>boto3) (2.8.2)  
Requirement already satisfied: six<=1.5 in c:\users\swast\appdata\local\program\python\python\lib\site-packages (from python-dateutil<3.0.0,>=2.1>botocore<3.30.0,>=1.29.0>boto3) (1.15.0)  
Installing collected packages: botocore, s3transfer, boto3  
Successfully installed boto3-1.26.0 botocore-1.29.0 s3transfer-0.6.0  
WARNING: You are using pip version 20.1.1; however, version 22.1 is available.  
You should consider upgrading via the 'c:\users\swast\appdata\local\program\python\python\python.exe -m pip install --upgrade pip' command.  
  
C:\Users\swast>
```

Figure6.3.11

1. pip install resources

```
C:\Users\swast>pip install resources  
Collecting resources  
  Downloading resources-0.0.1.tar.gz (1.7 kB)  
Building wheels for collected packages: resources  
  Building wheel for resources (setup.py) ... done  
  Created wheel for resources: filename=resources-0.0.1-py3-none-any.whl size=4370 sha256=38115eb3a96bfb54f0f22303a8aeebaac976211e26ae94f962441ec118e  
  Stored in directory: c:\users\swast\appdata\local\pip\cache\wheels\3b\1d\00\45ae57c7b932d145a0963f711c6d22f9af5306e74d80f2f288f  
Successfully built resources  
Installing collected packages: resources  
Successfully installed resources-0.0.1  
WARNING: You are using pip version 20.1.1; however, version 22.1 is available.  
You should consider upgrading via the 'c:\users\swast\appdata\local\program\python\python\python.exe -m pip install --upgrade pip' command.  
  
C:\Users\swast>
```

Figure6.3.12

1. pip install cloudant

```
Contract Point
You should consider upgrading via the 'C:\Users\hassan\AppData\Local\Programs\Python\Python37\python.exe -m pip install --upgrade pip' command.

C:\Users\hassan> pip install cloudant
Collecting cloudant
  Downloading cloudant-2.15.0-py3-none-any.whl (80 kB)
Requirement already satisfied: requests<3.0.0, >=2.7.0 in c:\users\hassan\AppData\Local\Programs\Python\Python37\lib\site-packages (from cloudant) (2.26.1)
Requirement already satisfied: charset-normalizer<3.0, >=2.0.0 in c:\users\hassan\AppData\Local\Programs\Python\Python37\lib\site-packages (from requests<3.0.0, >=2.7.0->cloudant) (2.1.1)
Requirement already satisfied: urllib3<2.0, >=1.25.1 in c:\users\hassan\AppData\Local\Programs\Python\Python37\lib\site-packages (from requests<3.0.0, >=2.7.0->cloudant) (1.26.12)
Requirement already satisfied: certifi<2021.4.12 in c:\users\hassan\AppData\Local\Programs\Python\Python37\lib\site-packages (from requests<3.0.0, >=2.7.0->cloudant) (2021.9.24)
Requirement already satisfied: ibmcloud==2.5 in c:\users\hassan\AppData\Local\Programs\Python\Python37\lib\site-packages (from requests<3.0.0, >=2.7.0->cloudant) (2.5)
Installing collected packages: cloudant
Successfully installed cloudant-2.15.0
WARNING: You are using pip version 20.1.1; however, version 21.1 is available.
You should consider upgrading via the 'C:\Users\hassan\AppData\Local\Programs\Python\Python37\python.exe -m pip install --upgrade pip' command.

C:\Users\hassan>
```

**Figure6.3.13**  
**PROJECT DEVELOPMENT**

**STEP 1:** Write a python code for randomize Soil Moisture ,Temperature, Humidity and Animal detection.

```
ooo.py - C:\Users\DELL\AppData\Local\Programs\Python\Python37\ooo.py (3.7.0)
File Edit Format Run Options Window Help

import time
import sys
import ibmiotf.application
import ibmiotf.device
import random

#Provide your IBM Watson Device Credentials
organization = "iirt5j7"
deviceType = "ebod"
deviceId = "12345"
authMethod = "token"
authToken = "12345678"

# Initialize GPIO
def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data['command'])
    status=cmd.data['command']
    if status=="lighton":
        print ("led is on")
    elif status == "lightoff":
        print ("led is off")
    else :
        print ("please send proper command")

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

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

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

while True:
    #Get Sensor Data from DHT11
```

**Figure6.3.14**



STEP 2: Run the python code it send data to IBM IoT Watson Platform.

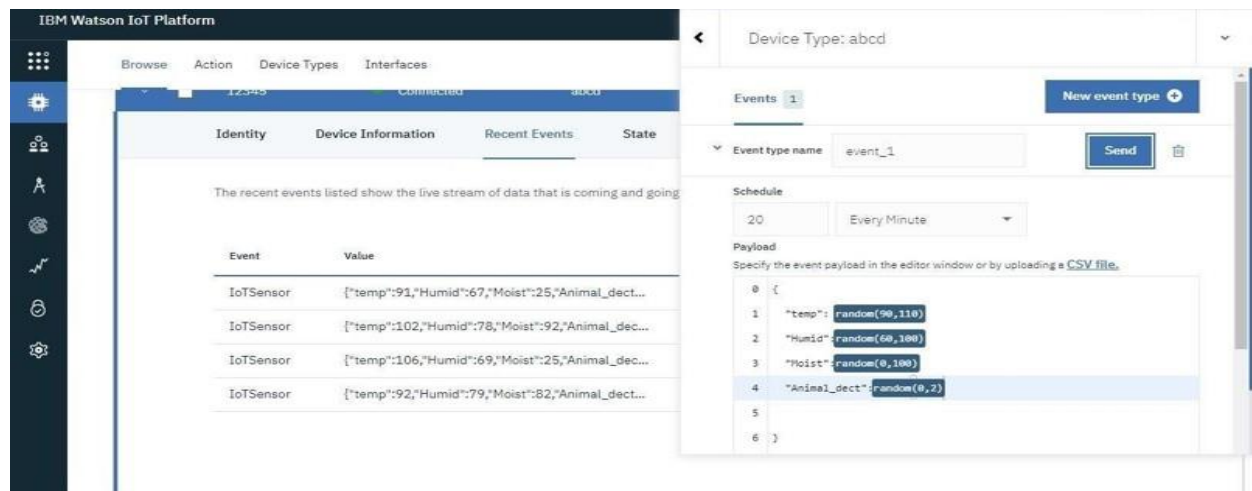


Figure6.3.15

STEP 3: Open Node-RED flow dashboard.

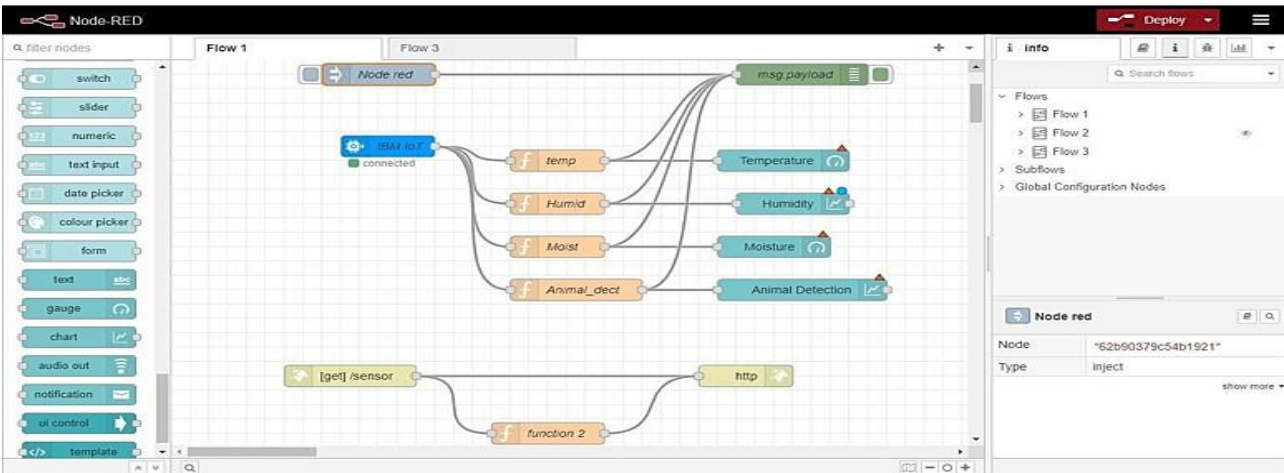


Figure6.3.16

STEP 4: Open Node-RED user interface to show the Soil Moisture, Humidity and Temperature value in gauge.

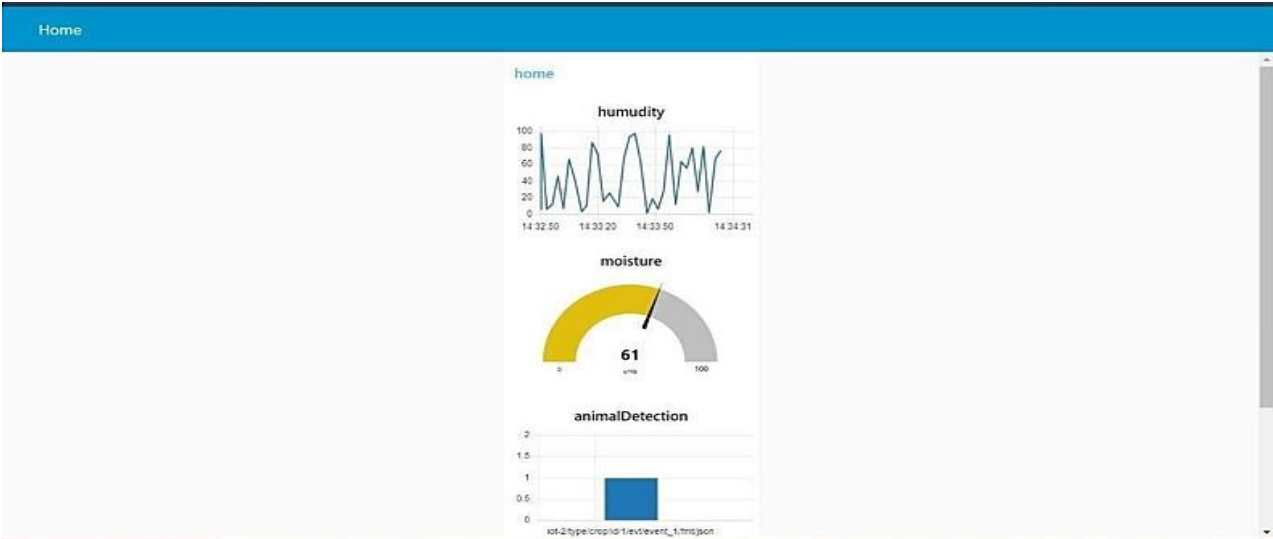


Figure6.3.17

# CHAPTER 7

## CODING AND SOLUTIONS

### 7.1 FEATURE

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

#### **Source Code:**

```
import time import sys
import ibmiotf.application
import ibmiotf.device import
random #Provide your IBM
Watson Device
Credentialsorganization =
"wu5b55" deviceType =
"crop1" deviceId = "1234"
authMethod =
"token" authToken = "1234567890"

# Initialize GPIOtry:

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

        #.....

except Exception as e:

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

# Connect and send a datapoint "hello" with value "world" into the
```

cloud as an event of type "greeting" 10 times

```
deviceCli.connect()while True:
```

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

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

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

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

```
def myOnPublishCallback():
```

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

```
success = deviceCli.publishEvent("IoTSensor",  
"json", data, qos=0,on_publish=myOnPublishCallback) if not  
success:
```

```
print("Not connected to IoT")time.sleep(10) deviceCli.commandCallback  
= myCommandCallback
```

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

## 7.2 FEATURE 2

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

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

## Output:

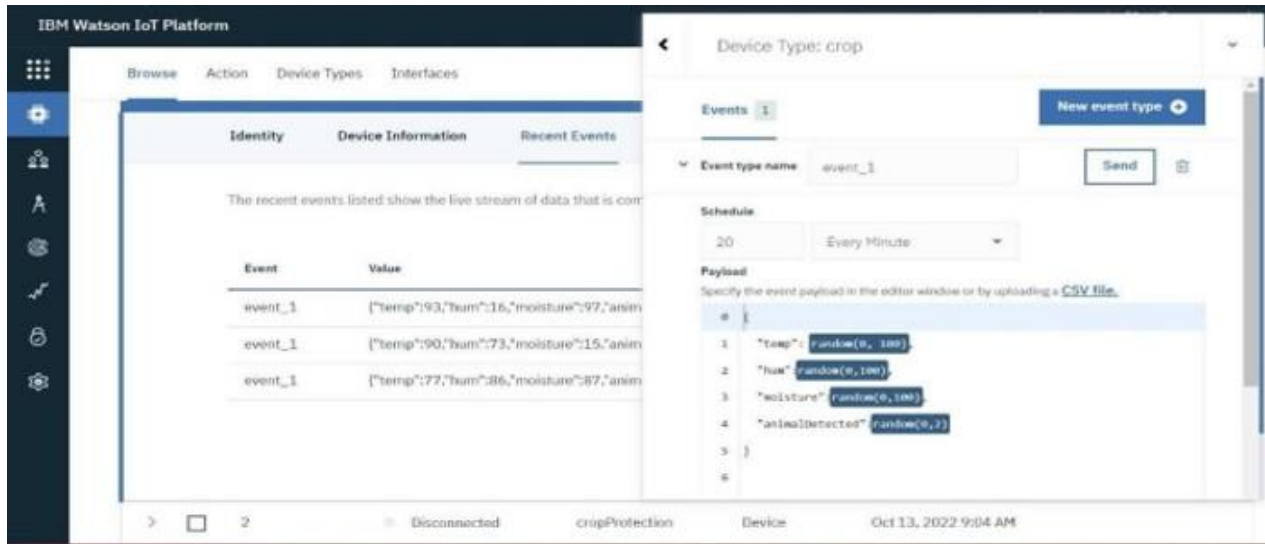


Figure7.2

## 7.3 DATABASE SCHEMA

### PYTHON CODE TO IBM

```
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random

#Provide your IBM Watson Device Credentials
organization = "wu5b55"
deviceType = "crop1"
deviceId = "1234"
authMethod = "token"
authToken = "1234567890"

# Initialize GPIO
try:
    deviceOptions = {"org": organization, "type": deviceType, "id":
deviceId, "auth-method": authMethod, "auth-token": authToken}
    deviceCli = ibmiotf.device.Client(deviceOptions)
except Exception as e:
    print("Caught exception connecting device: %s" % str(e))
    sys.exit()

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

True:

```
#Get Sensor Data from DHT11
temp=random.randint(0,100)
Hum=random.randint(0,100)
moisture=random.randint(0,100)
data = { 'temperature' : temp, 'Humidity': Hum,
'Moisture':moisture }

#print data
def myOnPublishCallback():
    print ("Temperature = " + str(temp)+" C
          Humidity = " + str(hum)+ " moisture =
          " + str(moisture) + "to IBM Watson")
    success = deviceCli.publishEvent("IoTSensor", "json", data,qos=0,
on_publish=myOnPublishCallback)
    if not success:
        print("Not connected to IoTF")
        time.sleep(10)

deviceCli.commandCallback = myCommandCallback

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

## CHAPTER 8

### TESTING

#### 8.1 TEST CASES

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	4	2	3	20
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	11	2	4	20	37
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	24	14	13	26	77

#### 8.2 USER ACCEPTANCE TESTING

This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	7	0	0	7
Client Application	51	0	0	51

Security	2	0	0	2
Outsource Shipping	3	0	0	3
Exception Reporting	9	0	0	9
Final Report Output	4	0	0	4
Version Control	2	0	0	2

# CHAPTER 9

## RESULT

### 9.1 PERFORMANCE METRICS

#### MIT APP INVENTOR:

STEP 1: MIT APP inventor to design the APP.

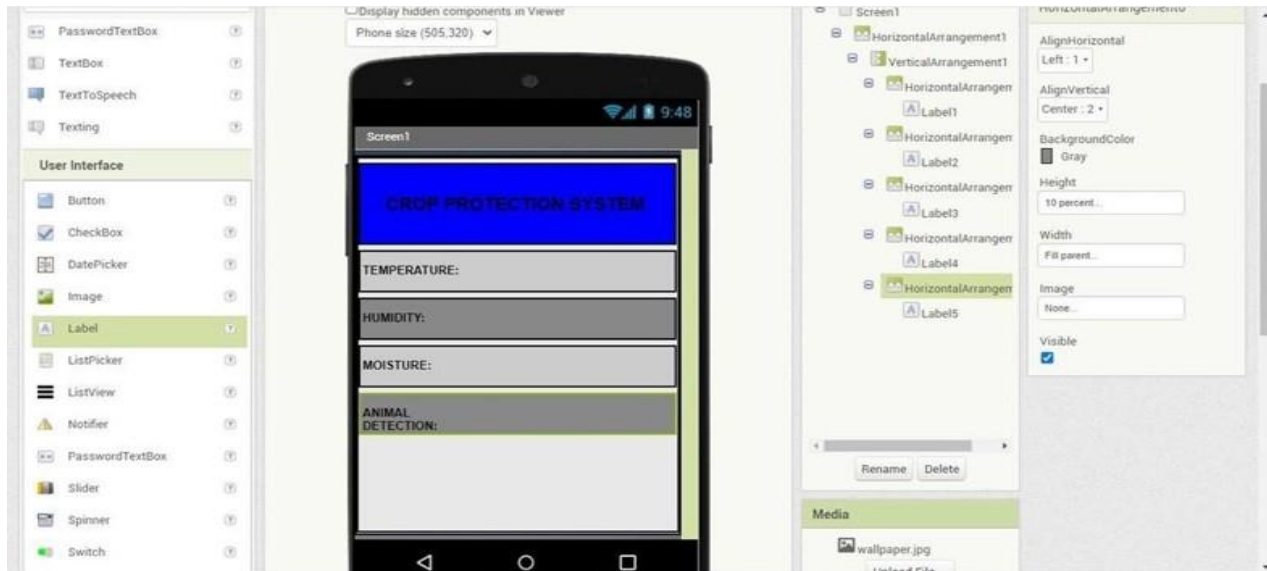


Figure 9.1



Figure 9.1.1

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





**Figure 9.1.2**

## CHAPTER 10

### ADVANTAGES AND DISADVANTAGES

#### Advantages:

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

#### Disadvantages:

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

#### APPLICATIONS:

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

## **CHAPTER 11**

### **CONCLUSION**

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

## **CHAPTER 12**

### **FUTURE SCOPE**

Study and analysis of the developed Crop protection systems for its cost effectiveness with the development of Arduino based variable frequency Ultrasonic bird deterrent circuit. outline of the crop damage caused by a particular Wild animal if the behavioral features of the With the reduced cost in the smart phones.

## CHAPTER 13

### APPENDIX

#### 13.1 SOURCE CODE

The sourcecode has been uploadedin github. To refer the final soursecode click [“SOURCE CODE”](#)

#### 13.2 GITHUB & PROJECT DEMO LINK

##### GITHUB LINK

The githublink :"[GITHUB LINK](#)"

##### PROJECT DEMO LINK

The ProjectDemo link: "[DEMO LINK](#)"

## CHAPTER 14

### REFERENCE

1. Priyanka Deotale, Prasad Lokulwar (2021) have presented the paper titled “Smart Crop Protection System from Wild animals Using IoT” . Crops in the agricultural land are destroyed by the domestic animals and wild animals ,it is one of the reason for low productivity . Farmers can't be there for entire 24 hours so we have make use of IOT to control the animals destroying the field . Once the animal is detected the system will alarm and start lighting in the corner of the farm . It will not harm any animals and we can also protect the crops.
2. N.S. Gogul Dev , K.S. Sreenesh, P.K. Binu (2019) has presented a paper titled “IoT Based Automated Crop Protection System”. Low productivity of crops is one of the main problems faced by the farmers in our country. This can be because of two main reasons. Crops destroyed by wild animals and because of bad weather condition. This paper provides a solution to the destruction of crops by animals. This system will provide a complete technical solution using the Internet of things (IOT) to the farmers to prevent their crops from wild animals and provide information to the farmers to maximize their production. Animals are detected using PIR sensors and cameras where animals are identified using TensorFlow image processing Techniques. Raspberry PI is used as the processing unit of the system and sound buzzers are used to emit the ultrasound frequencies