IoT Based Smart Crop Protection System for Agriculture

Team Members: (Team Id: PNT2022TMID05121)

Yugesh M- 921319104233

Suriya Prakash K- 921319104302

Tamilarasan v-921319104208

Vignesh Muthu P-921319104222

Project Report

1. **INTRODUCTION**
   1.1 Project Overview

3. Introduction

Problem Statement - SmartFarmer : IoT Based Smart Crop Protection System for Agriculture

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 guard the crops. An intelligent crop protection system helps the farmers in protection the crop from the animals and birds which destroy the crop. This system shall also include remote monitoring and control of pump to avoid the farmer to visit the farm in night-time.

By this project we can achieve

- Safety of people and animal
- Simple solution to suite the farmer community

Img 1.1

.1 Project Overview

Agriculture, the major sector which defines the growth of the nation and any community that exists on this earth. It requires very intensive care, discipline and patience to reap the yield. In this evolving modern world the number of people who adopt agriculture are only a handful, in the next few years there will be very minimal people who do farming.

This project will help the farmers to grow crops in a whole new way so that they can reap good yield and healthy foods. This system does the following:

1. Autonomous crop monitoring.
2. Irrigation control.
3. Environment sensing.

.2 Purpose

The purpose of this project is to help farmers to grow crops better and have yield, it emanates the need to manually look for the soil condition and decide whether to water the

plant or not, this process becomes very difficult when plants are maintained over a very large area. This project helps the farmers to find a better solution and make them aware of their surroundings.

2.Literature Survey

Introduction:

Nowadays technology has been increasing every day in our life. That IoT is also achieving great things in society. The Internet Of Things (IoT) is useful to formers to monitor the field and secure the field from wild animals. Our project is to help the farmers to keep their fields secure and continuous monitoring from anywhere. One of the major sectors in which IoT has brought major change was the agriculture sector which could be termed as smart Farming or Precision Agriculture. This involves monitoring various parameters which enhances the growth of the plant these factors include:

- Soil Moisture.
- Water Level Indicator.
- Rainfall.
- Temperature.
- Humidity.

By the above-mentioned factors, we could precisely monitor the crop field.

Survey Performed:

Reference 1	Title	Automation and Protection of Agriculture land using IoT
	Authors	Vaishnavi S Electronic and Communication Engineering Department GSSS Institute of Engineering and Technology for Women Mysuru, India
		Poorvika V M Electronic and Communication Engineering Department GSSS Institute of Engineering and Technology for Women Mysuru, India

Inference From Reference 1:

IoT Enabling Technologies described here are:

1. ARDUINO MEGA:

It is an open source IOT development board whose firmware runs on the ESP8266 Wi-Fi module. It performs the operations such as the collection of data from the DHT 11 sensors and if temperature is greater than the threshold then it supplies the water through irrigation. Then if the water level is high the excess of the water is removed through the water pump. If the object is detected through the PIR sensor the needle is rotated and buzzers are switched on.

2. PIR SENSOR:

PIR sensors allow you to sense motion, almost always used to detect whether a human has moved in or out of the sensors range. They are small, inexpensive, low-power, easy to use and don't wear out. Sensitivity range is up to 20 feet (6 meters) 110° x 70° detection range. If the object is detected then is sends information to the Arduino Mega.

3. NEEDLE ROTATOR:

The rotator is used to avoid the animals entering to the field, the nails are been fixed to the to the fence and this is connected to the driver when the animals is passing near the fencing the PIR sensor detects and it sends the information then this needle rotates 90 degree and protects the field.

4. WATER MOTOR PUMP:

The motor is used to pump the excess of water in the field. The pump incorporates an automatic pressure demand switch such that the pump will commence when a tap is opened and turn off when the tap is closed. Alternatively, the pump can be connected to an open hose and a suitable12v switch used to turn the flow On/Off.

Reference 2	Title	IOT in Agricultural Crop Protection and Power Generation
	Authors	Mrs. Sowmya M S Asst. Prof., Department of Electronics and Instrumentation Engineering, JSS Academy of Technical Education Bangalore, India

Inference From Reference 2:

The basic idea beyond using IOT in agriculture is to protect the crops during different seasons. As the many techniques are applied in agricultural sector greenhouse technique is also one of them. As of controlling and monitoring of greenhouse using IOT and some other technologies are

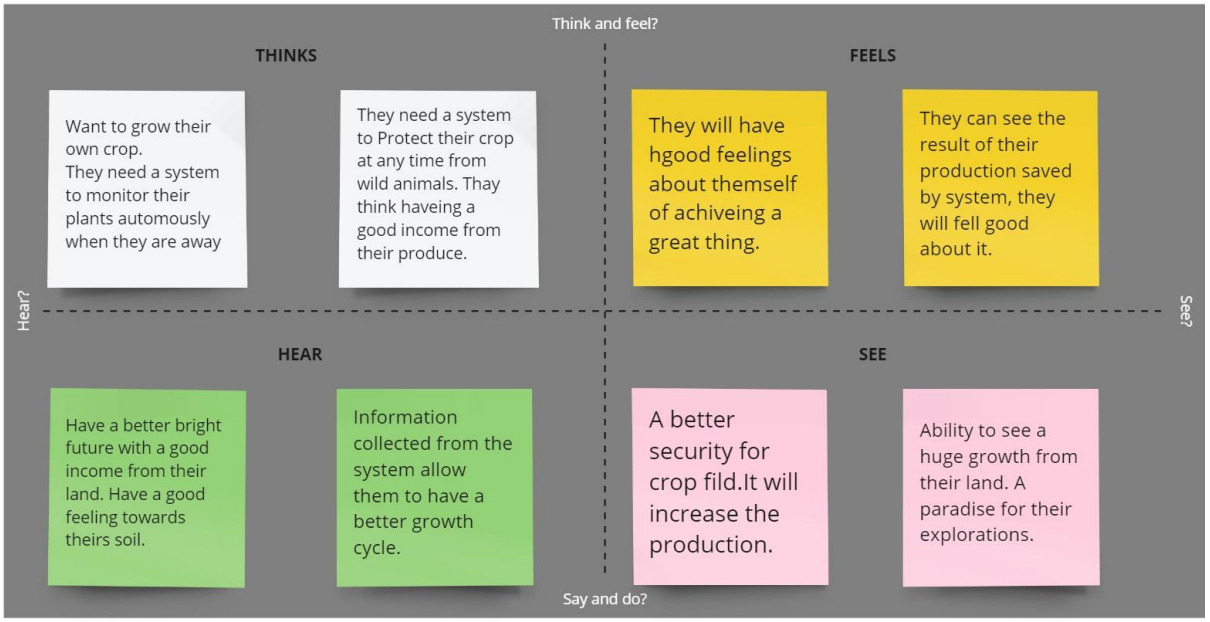
implemented. By observing all the technologies which are carried out on the greenhouse, so we are designing a greenhouse system which can control the parameters in the greenhouse

The propose system is designed by incorporating various hardware and software technologies in to it. The proposed system uses ARM7 LPC2148 microcontroller as a main control unit and all the controlling operations and executions have been carried out by this microcontroller. This system uses multiple sensors to monitor and control various parameters like temperature, moisture content in the soil, humidity etc. The sensors include humidity sensor, IR sensor, moisture sensor and rain sensor/rain switch as shown in the Fig. This sensor provides various readings that will help monitor and control the greenhouse. Entire system works in both automatic mode as well as in manual mode. By using IOT technology proposed system also enables the manual monitoring of green house from anywhere. Power is generated as part of this system using solar panels placed on the roof top of the greenhouse.

3. IDEATION & PROPOSED SOLUTION

3.1.Empathy Map Canvas

An empathy map is a simple, easy-to-digest visual that capture Knowledge about a user's behaviors and attitudes



3.2.Ideation & Brainstorming:

Step-1: Team Gathering, Collaboration and Select the Problem Statement

1

Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

PROBLEM

How might we able to come up with a better solution for sustainable farming using modern technologies?

PROBLEM

How might we able to create a better environment for farmers?

PROBLEM

How might we able to secure their farming feild using morden technologies?

PROBLEM

How might we able to make the life of farmes easier by secureing the crop and grate yield?

Step-2: Brainstorm, Idea listing and Grouping

2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

Person 1

- Automation detection of moveing objects
- Detecting wild animals by using thermal images
- Detecting the bieds by using ultrosonic
- Sending video or image to farmer

Person 2

- Automatic fire detaction
- Water Level detection
- Wather monitoring
- Identifiy the wind direction

Person 3

- Automation water flow control
- Finding soil type and recommanding crop
- Soil Moisture sencing
- Crop tracking

Person 6

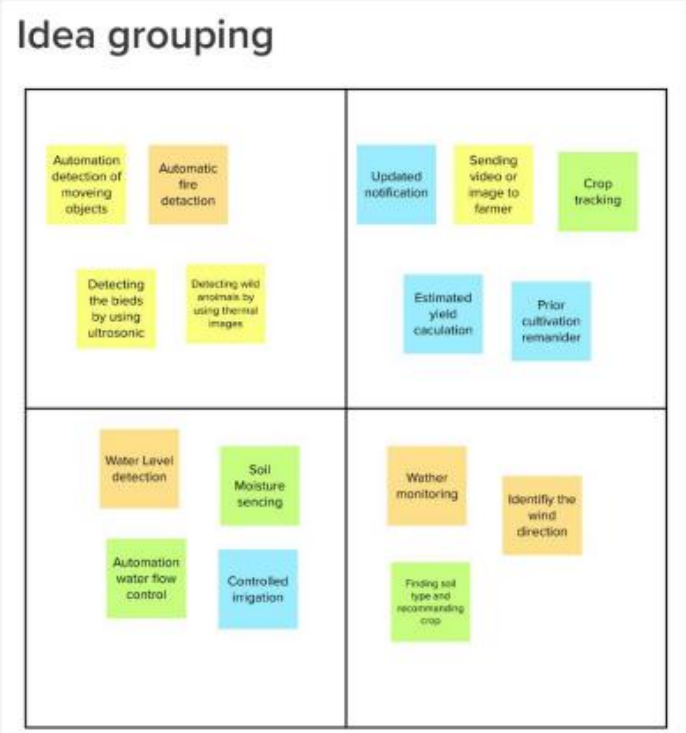
- Prior cultivation remanider
- Estimated yield caculation
- Controlled irrigation
- Updated notification

3

Group ideas

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

20 minutes



Step-3: Idea Prioritization

4

Prioritize

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

20 minutes

3.3.Proposed Solution:

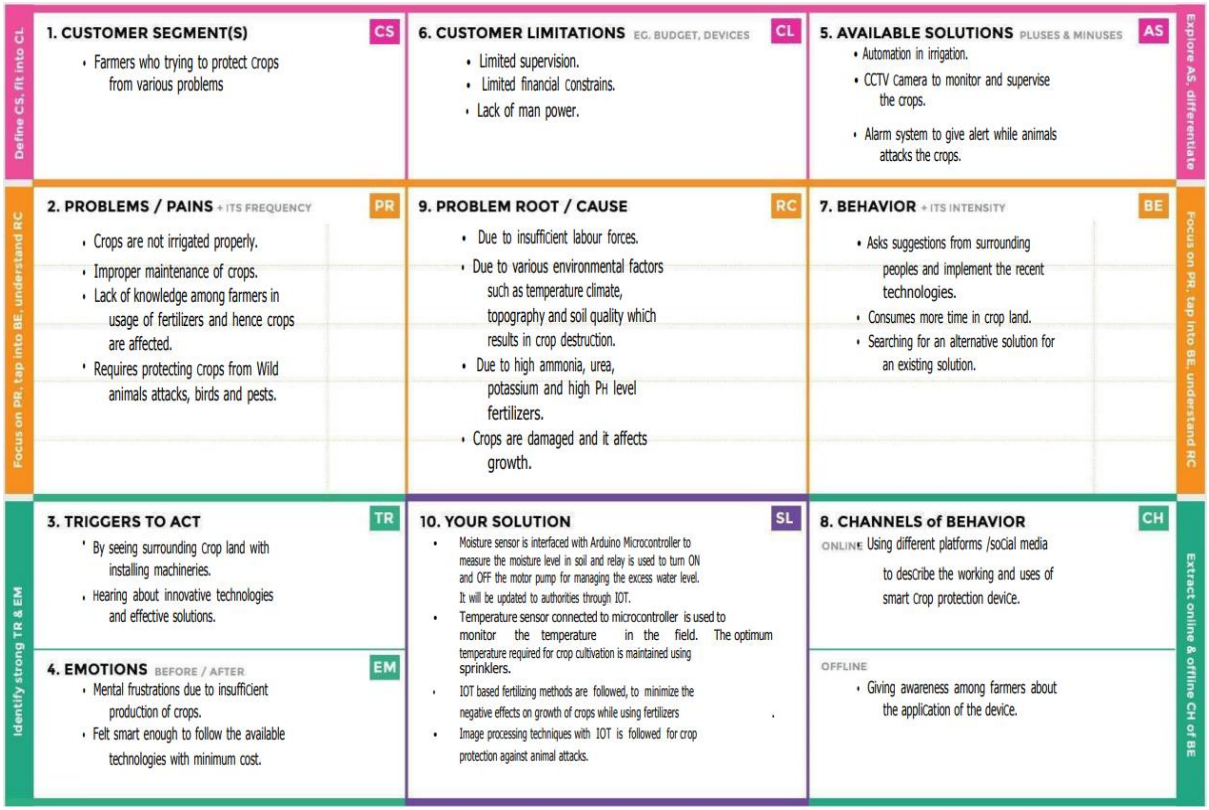
Project team shall fill the following information in proposed solution template.

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	IoT-based agriculture System help the farmer in monitoring different parameters of his field and security for their crops from wild animals using some sensors.
2.	Idea / Solution description	Our system comprises the following elements to come up with a solution: 1. Environment monitoring 2. Water Flow control 3. Detecting wild animals around the field
3.	Novelty / Uniqueness	Our system could function in both solar and battery mode. The inbuilt battery delivers power during the necessary times. It also delivers remote sensing facilities. Detecting animals and producing sound to send away from field.
4.	Social Impact / Customer Satisfaction	Upon implementing customers feel: 1. Seeing nearby adopting better agriculture practice. 2. Better income rates. 3. Better Yield. 4. Stable income. 5. Feeling comfortable with the practices.
5.	Business Model (Revenue Model)	Our system comprises of hardware and software part: 1. Controller (Brain)-8000 2. Solenoid valves-5000 3. Pipe materials- Needed to be provided by the land owner.

		4. Cloud storage of data- 10000/Month. 5. Needed sensors and other- 5000 Roughly sums around – 30000 Additionally, we can generate income by increasing the number of controllers.
6.	Scalability of the Solution	This system of ours is like a Lego which can be stacked and scaled up for a larger growth area.

3.4.Proposed Solution Fit:

AS



4.REQUIREMENT ANALYSIS:

4.1.Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Visibility	Sense animals nearing the crop field & sounds alarm to woo them away as well as sends SMS to farmer using cloud service.
FR-2	User Reception	The Data like values of Temperature, Humidity, Soil moisture Sensors are received via SMS.
FR-3	User Understanding	Based on the sensor data value to get the information about the present of farming

		land.
FR-4	User Action	The User needs take action like destruction of crop residues, deep ploughing, crop rotation, fertilizers, strip cropping, scheduled planting operations.

4.2.Non-functional Requirements:

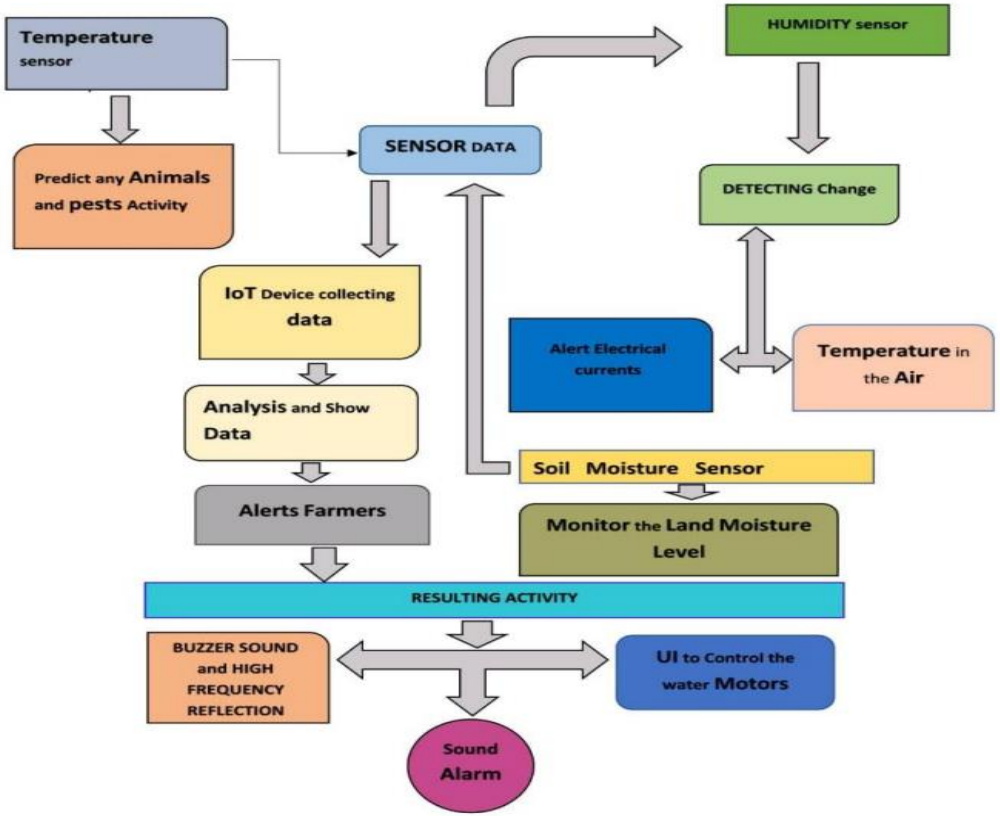
Following are the non-functional requirements of the proposed solution.

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

		manufacturing facilities and extra buildings.

5.Project Design:

5.1.Data Flow Diagrams:



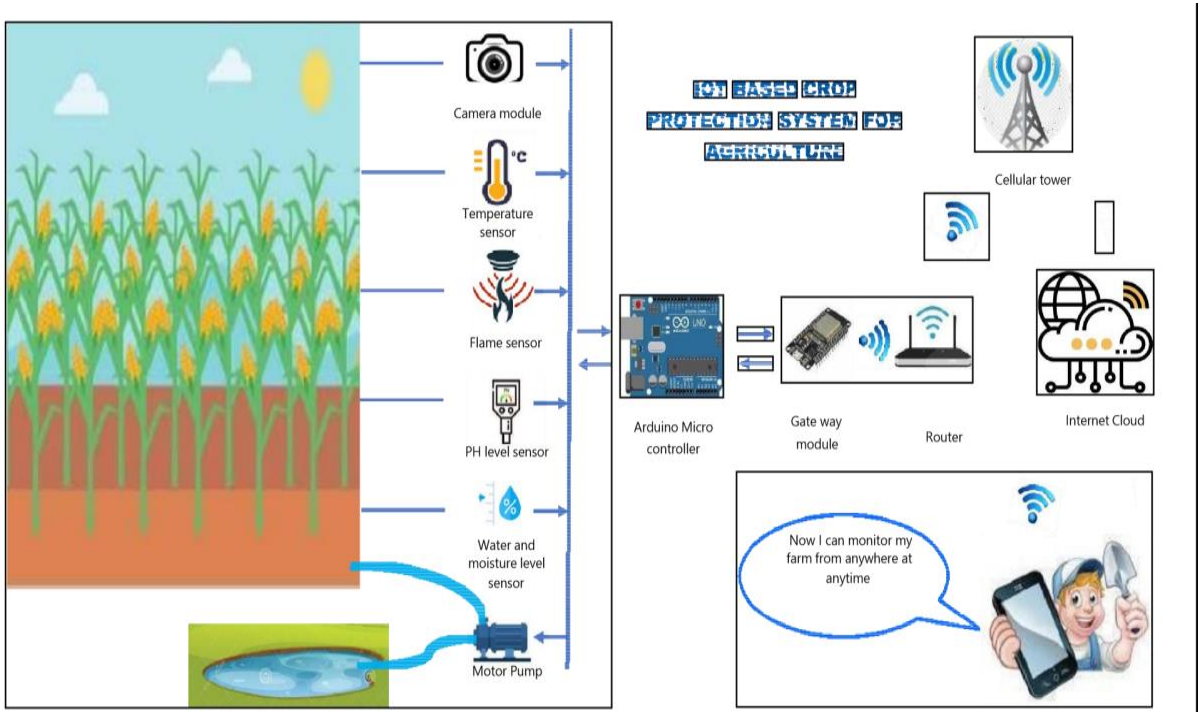
User Stories

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	User can enter into the web application	I can access my account / dashboard	High	Sprint-1
		USN-2	User can register their credentials like email id and password	I can receive confirmation email & click confirm	High	Sprint-1
	Login	USN-3	User can log into the application y entering email and password	I can login to my account	High	Sprint-1

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
	Dashboard	USN-4	User can View the temperature	I can view the data given by the device	High	Sprint-2
		USN-5	User can view the level of sensor monitoring value	I can view the data given by the device	High	Sprint-2
Customer (Web user)	Usage	USN-1	User can view the web page and get the information	I can view the data given by the device	High	Sprint-3
Customer	Working	USN-1	User act according to the alert given by the device	I can get the data work according to it	High	Sprint-3
		USN-2	User turns ON the water motors/Buzzer/Sound Alarm when occur the disturbance on field	I can get the data work according to it	High	Sprint-4
Customer Care Executive	Action	USN-1	User solve the problem when some faces any usage issues	I Can solve the issues when someone fails to understanding g the procedure	High	Sprint-4
Administrator	Administration	USN-1	Use stores every information	I Can store the gained information	High	Sprint-4

5.2.Solution & Technical Architecture:



The Repeller Device:

To improve the energy efficiency of the device, we made use of a Passive Infrared Sensor (PIR) sensor, which activates the driver responsible for the ultrasound generation.

Back-End System:

*We call the" back-end" a system where all the CPU intensive task processes take place.

Weather Monitoring System:

The device communicates over Wi-Fi to the backend system.

6.PROJECT PLANNING & SCHEDULING:

Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	Create the IBM Cloud services which are being used in this project.	6	High	Yugesh M, Suriya Prakash K, Tamilarasan V, Vignesh Muthu P.
Sprint-1		USN-2	Configure the IBM Cloud services which are being used in completing this project.	4	Medium	Yugesh M, Suriya Prakash K, Tamilarasan V, Vignesh Muthu P.
Sprint-2		USN-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	Yugesh M, Suriya Prakash K, Tamilarasan V, Vignesh Muthu P.
Sprint-2		USN-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	Yugesh M, Suriya Prakash K, Tamilarasan V, Vignesh Muthu P.
Sprint-3		USN-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	Yugesh M, Suriya Prakash K, Tamilarasan V, Vignesh Muthu P.
Sprint-3		USN-2	Create a Node-RED service.	10	High	Yugesh M, Suriya Prakash K, Tamilarasan V, Vignesh Muthu P.
Sprint-3		USN-1	Develop a python script to publish random sensor data	7	High	Yugesh M,

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
			such as temperature, moisture, soil and humidity to the IBM IoT platform			Suriya Prakash K, Tamilarasan V, Vignesh Muthu P.
Sprint-3		USN-2	After developing python code, commands are received just print the statements which represent the devices.	5	Medium	Yugesh M, Suriya Prakash K, Tamilarasan V, Vignesh Muthu P.
Sprint-4		USN-3	Publish Data to The IBM Cloud	8	High	Yugesh M, Suriya Prakash K, Tamilarasan V, Vignesh Muthu P.
Sprint-4		USN-1	Create Web UI in Node- Red	10	High	Yugesh M, Suriya Prakash K, Tamilarasan V, Vignesh Muthu P.
Sprint-4		USN-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	Yugesh M, Suriya Prakash K, Tamilarasan V, Vignesh Muthu P.

Project Tracker, Velocity & Burndown Chart: (4 Marks)

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

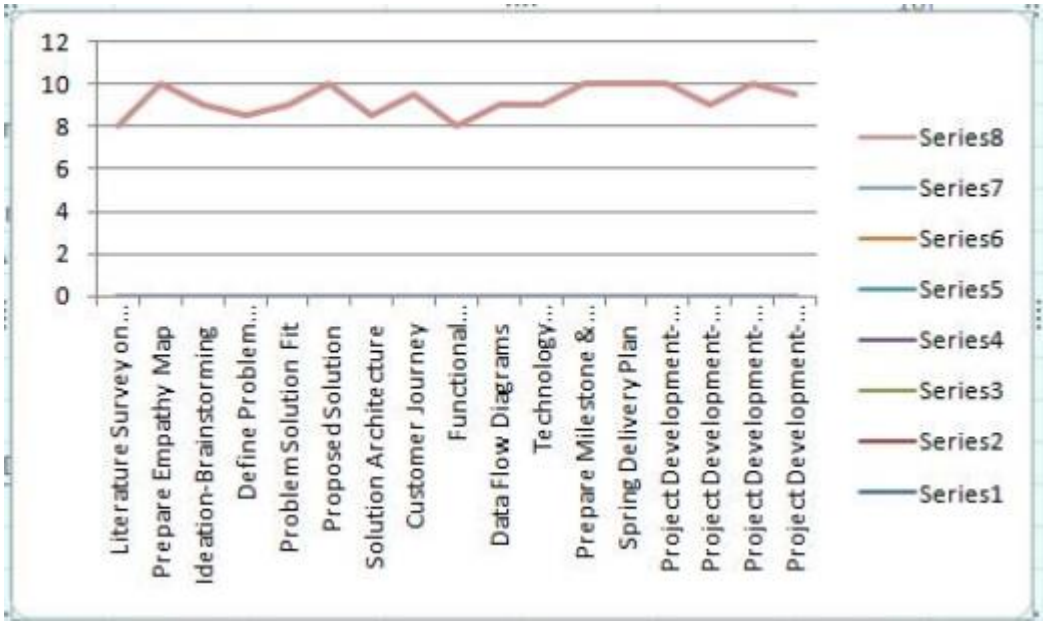
Velocity:

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

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

Burndown Chart:

A burn down 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, burn down charts can be applied to any project containing measurable progress over time.



7.CODING & SOLUTIONING (Explain the features added in the project along with the code):

7.1Feature 1

The unique feature of our project is that we can add up to 5 soil moisture and humidity sensors and we can add up to 4 pumps. The device is also capable of sustaining solar power so that it could operate without any power shortages during day time. It is capable of operating autonomously without any human intervention

7.2Feature 2

The person who connected with the device can only view the data, other than the person connected with the device will not be able to view the sensor readings. It enables a simple device security principle that others can not view and control the sensor readings from the device.

7.3. Feature 3

The user who is connected to the device can view the readings in mobile as well as the desktop. It is both mobile and web responsive so there is no need to install a separate mobile application in the mobil;e devices to view the device status.

10. ADVANTAGES AND DISADVANTAGES :

Advantages:

1. Remote monitoring.
2. Autonomous watering system.
3. Environment monitoring.
4. Enables data security over sharing.
5. Can view the data in both mobile and desktop.

Disadvantages

1. Limited to small area - Requires similar units to cover larger area.
2. Few data discrepancies during bad weather.

11. CONCLUSION:

The aim of this project is to make the life and work of the farmer much easier. This can be achieved using the technique - Precision Farming, this involves autonomous monitoring of crops and other environmental parameters which has an effect on the crop, these environmental conditions are:

1. Environmental Humidity
2. Environmental Temperature.
3. Soil Moisture.
4. Rain Sensing.

Above mentioned are some of the conditions monitored autonomously, threshold parameters for various crops are automatically set upon user input of crop variety to be monitored. By this system one could achieve a good yield and better nutritional crops in their agricultural produce.

12. FUTURE SCOPE:

Future scope of our project relies on the farmers and their feedbacks, in future we are planning to add the following features:

1. One device one farm - Cover the entire farm area with a single device.
2. Pest monitoring system.
3. Estimated yield calculator.
4. Estimated time of cultivation.
5. Individual cloud management dashboard.

13. APPENDIX:

Source Code:

IoT :

```
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <WiFi.h>
#include <WiFiClient.h>
#include <PubSubClient.h>
#include <Adafruit_BMP280.h>
#include <math.h>
#include <Wire.h>
#define BMP_SDA 21
#define BMP_SCL 22
#include <DFRobot_DHT11.h>
DFRobot_DHT11 DHT;
#define DHT11_PIN 4
#define rainAnalog 35
#define rainDigital 34
#define moistureDigital 32

Adafruit_BMP280 bmp280;
const char* ssid     = "";
const char* password = "";
AsyncWebServer server(80);
AsyncEventSource events("/events");
unsigned long lastTime = 0;
unsigned long timerDelay = 1000;
int soil;
int rain;
int rainA;
```

```
float temperature;
float humidity;
float pressure;
float  altitude;
long lastMsg = 0;
int pumpRelayPin = 26;

#define ORG "6jw3v9"
#define DEVICE_TYPE "ESP32"
#define DEVICE_ID "######################"
#define TOKEN "######################"

char servers[] = ORG ".messaging.internetofthings.ibmcloud.com";
char pubTopic1[] = "iot-2/evt/temperature/fmt/json";
char pubTopic2[] = "iot-2/evt/humidity/fmt/json";
char pubTopic3[] = "iot-2/evt/pressure/fmt/json";
char pubTopic4[] = "iot-2/evt/altitude/fmt/json";
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;

WiFiClient wifiClient;
PubSubClient client(servers, 1883, NULL, wifiClient);

// Init BME280
void initBME() {
 if (!bmp280.begin(0x76)) {
  Serial.println("Could not find a valid BMP280 sensor, check wiring!");
  while (1);
 }
}

void getSensorReadings() {
 DHT.read(DHT11_PIN);
 temperature = DHT.temperature;
 humidity = DHT.humidity;
 pressure = bmp280.readPressure() / 100;
 soil = digitalRead(moistureDigital);
 rain = digitalRead(rainDigital);
 rainA = analogRead(rainAnalog);
```

```
  altitude = bmp280.readAltitude(1011.18);
  if(soil == 1){
    digitalWrite(pumpRelayPin, LOW);
  }
  else{
    digitalWrite(pumpRelayPin, HIGH);
  }
}

// Initialize WiFi
void initWiFi() {
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi ..");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print('.');
    delay(1000);
  }
  Serial.println(WiFi.localIP());
}

String processor(const String& var) {
  getSensorReadings();
  //Serial.println(var);
  if (var == "TEMPERATURE") {
    return String(temperature);
  }
  else if (var == "HUMIDITY") {
    return String(humidity);
  }
  else if (var == "PRESSURE") {
    return String(pressure);
  }
  else if (var == "ALTITUDE") {
    return String(altitude);
  }
  else if (var == "RAINING") {
    return String(rain);
  }
  else if (var == "SOIL") {
```

```
   return String(soil);
  }
  return String();
}

const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML><html>
<head>
  <title>Grow Greens Smart</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
   <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.7.2/css/all.css"
integrity="sha384-fnmOCqbTlWIlj8LyTjo7mOUStjsKC4pOpQbqyi7RrhN7udi9RwhKkMHp
vLbHG9Sr" crossorigin="anonymous">
  <link rel="icon" href="data:,">
  <style>
                html {font-family: Arial; display: inline-block; text-align: center;
background-color:#FCF8E8}
    p { font-size: 1.2rem;}
    body {  margin: 0;}
    .topnav { overflow: hidden; color: #6D9886; font-size: 1rem; }
    .content { padding: 20px; }
           .card  {  background-color:  #F2AA4CFF;  box-shadow: 2px 2px 12px 1px
rgba(140,140,140,.5); border-radius: 30px;}
         .cards { max-width: 800px; margin: 0 auto; display: grid; grid-gap: 2rem;grid-
template-columns: repeat(auto-fit, minmax(200px, 1fr)); }
    .reading { font-size: 1.4rem; }
  </style>
</head>
<body>
  <div class="topnav">
    <h1>Grow Greens Smart</h1>
  </div>
  <div class="content">
    <div class="cards">
      <div class="card">
        <p><i class="fas fa-thermometer-half" style="color:#101820FF; font-size:25px"></i>
Temperature</p><p><span class="reading" style = "color:#101820FF"><span id="temp"
style="font-size:1rem;                font-weight:bolder;">%TEMPERATURE%</span>
&deg;C</span></p>
      </div>
```

```
    <div class="card">
                <p><i class="fas fa-tint" style="color:#101820FF; font-size:25px"></i> Humidity</p><p><span class="reading" style="color:#101820FF; font-size:1rem;"><span id="hum" style="font-size:1rem; font-weight:bolder;">%HUMIDITY%</span> &percnt;</span></p>
    </div>
    <div class="card">
                    <p><i class="fas fa-angle-double-down" style="color:#101820FF;font-size:25px"></i> Pressure</p><p><span class="reading" style="color:#101820FF; font-size:1rem;"><span id="pres" style="font-size:1rem; font-weight:bolder;">%PRESSURE%</span> hPa</span></p>
    </div>
    <div class="card">
            <p><i class="fas fa-mountain" style="color:#101820FF; font-size:25px"></i> Altitude</p><p><span class="reading" style="color:#101820FF"><span id="alti" style="font-size:1rem; font-weight:bolder;">%ALTITUDE%</span> m</span></p>
    </div>
    <div class="card">
            <p><i class="fas fa-cloud-rain" style="color:#101820FF; font-size:25px"></i> Raining</p><p><span class="reading" style="color:#101820FF"><span id="rain" style="font-size:1rem; font-weight:bolder;">%RAINING%</span></span></p>
    </div>
    <div class="card">
                <p><i class="fas fa-tree" style="color:#101820FF; font-size:25px"></i> Moisture</p><p><span class="reading" style="color:#101820FF"><span id="soil" style="font-size:1rem; font-weight:bolder;">%SOIL%</span></span></p>
    </div>
   </div>
 </div>
<script>
if (!!window.EventSource) {
 var source = new EventSource('/events');

 source.addEventListener('open', function(e) {
  console.log("Events Connected");
 }, false);
 source.addEventListener('error', function(e) {
  if (e.target.readyState != EventSource.OPEN) {
  console.log("Events Disconnected");
  }
```

```
}, false);

source.addEventListener('message', function(e) {
 console.log("message", e.data);
}, false);

source.addEventListener('temperature', function(e) {
 console.log("temperature", e.data);
 document.getElementById("temp").innerHTML = e.data;
}, false);

source.addEventListener('humidity', function(e) {
 console.log("humidity", e.data);
 document.getElementById("hum").innerHTML = e.data;
}, false);

source.addEventListener('pressure', function(e) {
 console.log("pressure", e.data);
 document.getElementById("pres").innerHTML = e.data;
}, false);

source.addEventListener('altitude', function(e) {
 console.log("latitude", e.data);
 document.getElementById("alti").innerHTML = e.data;
}, false);

source.addEventListener('rain', function(e) {
console.log("Rain", e.data);
 if(e.data == '0')
 document.getElementById("rain").innerHTML = "Raining";
 else
   document.getElementById("rain").innerHTML = "Not Raining";
}, false);

source.addEventListener('soil', function(e) {
console.log("Soil Moisture", e.data);
if(e.data == '1')
   document.getElementById("soil").innerHTML = "Less Water";
 else
   document.getElementById("soil").innerHTML = "Enough Water";
```

```
 }, false);
}
</script>
</body>
</html>)rawliteral";

void setup()  {
  Serial.begin(115200);
  pinMode(rainDigital, INPUT);
  pinMode(moistureDigital, INPUT);
  pinMode(pumpRelayPin, OUTPUT);
  initWiFi();
  initBME();
  // Handle Web Server
  server.on("/", HTTP_GET, [](AsyncWebServerRequest * request) {
    request->send_P(200, "text/html", index_html, processor);
  });

  // Handle Web Server Events
  events.onConnect([](AsyncEventSourceClient * client) {
  if (client->lastId()) {
              Serial.printf("Client reconnected! Last message ID that it got is: %u\n",
client->lastId());
    }
    // send event with message "hello!", id current millis
    // and set reconnect delay to 1 second
    client->send("hello!", NULL, millis(), 10000);
  });
  server.addHandler(&events);
  server.begin();
  if (!client.connected()) {
    Serial.print("Reconnecting client to ");
    Serial.println(servers);
    while (!client.connect(clientId, authMethod, token)) {
      Serial.print(".");
      delay(500);
    }
    Serial.println("Bluemix connected");
  }
}
```

```
void loop() {
  client.loop();
  long now = millis();
  if (now - lastMsg > 3000) {
    lastMsg = now;


    String payload = "{\"temperature\":";
    payload += temperature;
    payload += "}";
    Serial.print("Sending payload: ");
    Serial.println(payload);
    if (client.publish(pubTopic1, (char*) payload.c_str())) {
      Serial.println("Publish ok");
    } else {
      Serial.println("Publish failed");
    }


    String payload1 = "{\"humidity\":";
    payload1 += humidity;
    payload1 += "}";
    Serial.print("Sending payload: ");
    Serial.println(payload1);
    if (client.publish(pubTopic2, (char*) payload1.c_str())) {
      Serial.println("Publish ok");
    } else {
      Serial.println("Publish failed");
    }


    String payload2 = "{\"pressure\":";
    payload2 += pressure;
    payload2 += "}";
    Serial.print("Sending payload: ");
    Serial.println(payload2);
    if (client.publish(pubTopic3, (char*) payload2.c_str())) {
      Serial.println("Publish ok");
    } else {
```

```
    Serial.println("Publish failed");
   }

   String payload3 =
   "{\"altitude\":";payload3 +=
   altitude;
   payload3 += "}";
   Serial.print("Sending
   payload: ");
   Serial.println(payload3);
   if (client.publish(pubTopic4, (char*)
    payload3.c_str())) {Serial.println("Publish ok");
   } else {
    Serial.println("Publish failed");
   }

  }
  if ((millis() - lastTime) >
   timerDelay) {
   getSensorReadings();
   Serial.printf("Temperature = %.2f ºC \n",
   temperature);Serial.printf("Humidity = %.2f \n",
   humidity); Serial.printf("Pressure = %.0f hPa \n",
   pressure); Serial.printf("Altitude = %.0f m \n",
   altitude); Serial.printf("Rain = %d\n", rain);
   Serial.printf("Rain = %d\n",
   rainA);Serial.printf("Soil =
   %d\n", soil); Serial.println();

   // Send Events to the Web Server with the Sensor Readings
   events.send("ping", NULL, millis());
   events.send(String(temperature).c_str(), "temperature",
   millis());events.send(String(humidity).c_str(), "humidity",
   millis()); events.send(String(pressure).c_str(), "pressure",
   millis()); events.send(String(altitude).c_str(), "altitude",
   millis()); events.send(String(rain).c_str(), "rain", millis());
   events.send(String(soil).c_str(), "soil", millis());
   lastTime = millis();
  }}
```

github link : https://github.com/IBM-EPBL/IBM-Project-9289-1658991755