NAALAIYA THIRAN PROJECT - 2022

IOT BASED SMART CROP PROTECTION SYSTEM

A PROJECT REPORT

Submitted by

GEORGE NITHIS KISHORE S	**212919205013**
HARISH J	**212919205014**
SHERLIN A G	**212919205043**
DHINAKARAN K	**212919205011**
JINIL DEMETRIUS V	**212919205018**

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY DEPARTMENT.

ST.JOSEPH COLLEGE OF ENGINEERING,

SRIPERUMBUDUR

ANNA UNIVERSITY::CHENNAI-602117

BONAFIDE CERTIFICATE

Certified that this project report "**IOT BASED SMART CROP PROTECTION SYSTEM FOR AGRICULTURE"** is the bonafide work of "GEORGE NITHIS KISHORE.S(212919205013), HARISH.J(212919205014),SHERLIN A G(212919205043), DHINAKARAN .K(212919205011), JINIL DEMETRIUS.V(212919205018) who carried out the project work under my supervision.

SIGNATURE OF HOD

Mr. S.Muthukumaran

Information Technology Department
St.Joseph College Of Engineering,
Sriperumbudur.

SIGNATURE OF MENTOR

Mrs .K.Karthika

Information Technology Department
St .Joseph College Of Engineering,
Sriperumbudur.

PROJECT CALENDAR

Phase	Phase Description	Week	Dates	Activity Details
1	Preparation Phase (Pre- requisites, Registrations, Environment Set-up, etc.)	2	22 - 27 Aug 2022	Creation GitHub account & collaborate with Project repository in project workspace
2	Ideation Phase (Literature Survey, Empathize, Defining Problem Statement, Ideation)	2	29 Aug – 3rd Sept 2022	Literature survey (Aim, objective, problem statement and need for the project)
		3	5 - 10th Sept 2022	Preparing Empathy Map Canvas to capture the user Pains & Gains
		4	12 - 17 Sept 2022	Listing of the ideas using brainstorming session
3	Project Design Phase -I (Proposed Solution, Problem- Solution Fit, Solution Architecture)	5	19 - 24 Sept 2022	Preparing the proposed solution document
		6	26 Sept - 01 Oct 2022	Preparing problem - solution fit document & Solution Architecture
4	Project Design Phase -II (Requirement Analysis, Customer Journey, Data Flow Diagrams, Technology Architecture)	7	3 - 8 Oct 2022	Preparing the customer journey maps
		8	10 - 15 Oct 2022	Preparing the Functional Requirement Document & Data- Flow Diagrams and Technology Architecture
5	Project Planning Phase (Milestones & Tasks, Sprint Schedules)	9	17 - 22 Oct 2022	Preparing Milestone & Activity List, Sprint Delivery Plan
6	Project Development Phase (Coding & Solutioning, acceptance Testing, Performance Testing)	10	24 - 29 Oct 2022	Preparing Project Development - Delivery of Sprint-1
		11	31 Oct - 5 Nov 2022	Preparing Project Development - Delivery of Sprint-2
		12	7 - 12 Nov 2022	Preparing Project Development - Delivery of Sprint-3
		13	14 - 19 Nov 2022	Preparing Project Development - Delivery of Sprint-4

TABLE OF CONTENTS

CHAPTER NO	CONTENTS	PAGE NO
	ABSTRACT	
	LIST OF FIGURES	
	LIST OF TABLES	
1	INTRODUCTION 1.1 PROJECT OVERVIEW 1.2 PURPOSE	1
2	LITERATURE SURVEY 2.1 EXISTING SOLUTION 2.2 PROBLEM STATEMENT DEFINITION	3
3	IDEATION & PROPOSED SOLUTION 3.1 EMPATHY MAPCANVAS 3.2 IDEATION & BRAINSTORMING 3.3 PROPOSED SOLUTION 3.4 PROBLEM SOLUTION FIT	4
4	REQUIREMENT ANALYSIS 4.1 FUNCTIONAL REQUIREMENT 4.2 NON-FUNCTIONAL REQUIREMENT	11
5	PROJECT DESIGN 5.1 DATA FLOW DIAGRAMS 5.2 SOLUTION & TECHNICAL ARCHITECTURE	14
6	PROJECT PLANNING & SCHEDULING 6.1 SPRINT PLANNING & ESTIMATION	17

	6.2 SPRINT DELIVERY SCHEDULE	
7	**CODING & SOLUTIONING** **7.1 FEATURE**	**20**
8	**TESTING AND RESULTS** **8.1 TEST CASES AND RESULTS**	**37**
9	**PERFORMANCE RESULTS** **9.1 PERFORMANCE METRICES**	**39**
10	**ADVANTAGES & DISADVANTAGES**	**40**
11	**UAT**	**42**
12	**CONCLUSION**	**44**
13	**FUTURE SCOPE**	**45**

SOURCE CODE

GITHUB & PROJECT DEMO LINK

LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO
3.1	EMPATHY MAP	4
3.4	PROBLEM SOLUTION FIT	10
5.1	DATA FLOW DIAGRAM	15
5.2	SOLUTION ARCHITECTURE	16
7.1	LIBRARIES REQUIRED	20
7.2	CREATING RANDOM DATA	21
7.3	RANDOM FLOATING WINDOW FLOAT VALUES	22
7.4	TWILIO	22
7.5	PROGRAM EXECUTION FOR GETTING TEMPERATURE AND HUMIDITY DATA	23
7.6	PROGRAM EXECUTION FOR ANIMAL INTRUSION	24
7.7	SENDING SMS TO THE USER	24
7.8	SMS FUNCTION	25
7.9	TWILIO DASHBOARD	25
7.10	MESSAGE OUTPUT	26
7.11	TWILIO LOGS	26
7.12	OPEN WEATHER API	27
7.13	OPEN WEATHER WEBSITE	27
7.14	GET WEATHER FUNCTION	28
7.15	KEY VALUES AND STATUS OF THE OPEN WEATHER API	28
7.16	GOOGLE SHEETS FOR DEVELOPERS	28
7.17	GOOGLE CLOUD DASHBOARD	30
7.18	UPDATE SHEET FUNCTION	31
7.19	GOOGLE SHEET DATABASE	31
7.20	USAGE OF WHS.UPDATE FUNCTION	32
7.21	GET VALUES FUNCTION	32

7.22	**PROGRAM EXECUTION OUTPUT**	**33**
7.23	**SOIL HUMIDITY CODE**	**33**
7.24	**EXECPTIONAL HANDLING**	**34**
7.25	**WEBPAGE**	**34**
7.26	**WOKWI SIMULATION**	**35**
7.27	**WOKWI SIMULATION**	**36**
7.28	**WOKWI SIMULATION**	**36**
8.1	**DATABASE UPDATE FOR TESTING**	**37**
8.2	**UPDATED VALUE**	**37**
8.3	**MESSAGE OUTPUT TESTCASE**	**38**
8.4	**WEBPAGE**	**38**
9.1	**PERFORMANCE METRIX IN GOOGLE SHEETS API**	**39**
9.2	**LATENCY IN GOOGLE SHEETS API**	**39**
11.1	**UAT EXECUTION**	**42**

LIST OF TABLES

TABLE NO	TITLE	PAGE NO
3.1	IDEATION CHART	5
3.2	PROPOSED SOLUTION	7
4.1	FUNCTIONAL REQUIREMENTS	11
4.2	NON-FUNCTIONAL REQUIREMENT	12
6.1	SPRINT PLANNING AND ESTIMATION	17
6.2	SPRINT DELIVERY PLAN	19

CHAPTER 1
INTRODUCTION

1.1 PROJECT OVERVIEW

Today, technology has penetrated every part of human life. But the contribution of technology to the field of agriculture is considerably low when compared to the other sectors, which saw an incremental growth over the last decade.

The domain of Agriculture contributes the most to the Indian economy and about 1/3rd of India's population is directly dependent on agriculture for their source of income. Considering this, even a small improvement in this sector will make a huge impact on the Indian economy and on the life of farmers. This helps farmers and consumers equally as it is the consumers in the end, who get to enjoy low priced goods without deterioration in quality.

To achieve this, we have to overcome the hurdles faced by farmers, which mostly revolve around crop disease, improper maintenance of crops, lack of details about the quality of soil and intervention of animals and birds. To overcome this, in this project we propose 'An intelligent crop protection system', the main objective of which is to improve the yield and increase the profit for farmers. An intelligent crop protection system uses data from moisture, motion, temperature, humidity sensors and updates the data in real time in IBM cross platform IOT cloud interface. The motors and the sprinkling system are activated based on the data from the sensors. Also when the motion sensor detects motion, the farmer is notified with that through the mobile application. This helps the farmers in protecting the crop from the animals and birds which destroy the crop.

And also ease up the maintenance process. The historical data from sensors are stored in cloud, so this can also be used for soil evaluation and this also helps to plan, which type of crops are to be planted in the upcoming seasons so that the yield is high.

1.2 PURPOSE

A vast majority of the people are invariably affected by the production of crops. Farmers, for example, rely on them for their survival. The consumers, on the other hand, depend on the crops as it provides them with a multitude of utilities. It therefore, becomes essential to protect and maintain these crops. The project aims at improving the farmers'

situation by preventing them from incurring losses due to the damage of crops. Crop failure also deteriorates the quality of the yield thereby decreasing the quality of living.

CHAPTER 2
LITERATURE SURVEY

2.1 EXISTING SOLUTION

In real time, it was learnt that the size of the animal is found out by using several PIR sensors. PIR sensors can be used to determine the height of the animals instead of using a camera for image processing. This reduces the processing time and power. The crop protection is majorly dependent on the moisture content of the soil, the temperature and humidity of the surrounding environment. Additionally, tracking of the damaged crops location is done and the camera is activated only at that instant in order to capture the image.

From the literatures survey performed it is evident that image based animal intrusion identification is not necessary in all situations because it requires high computation power, and the cost of the installation will be high when compared to that of a typical sensor based intrusion identification.

2.3 PROBLEM STATEMENT DEFINITION

Smart crop protection system aims at improving the farmers' situation by preventing them from incurring losses due to the damage of crops. Crop failure also deteriorates the quality of the yield thereby decreasing the quality of living.

SUMMARY

Using several PIR sensors can also prove to be efficient identifying the location of the intrusion and usage of multiple PIR sensors can be used to find the height of the animal and classify the seriousness of intrusion. Moisture control is another aspect where people generally misjudge the effectiveness. So actively, monitoring and automating the process of controlling moisture level will prove helpful. For cloud database in this case the information that needed to be stored in the cloud are in strings and integers for which google sheets database is more than sufficient. Choosing online based weather API proves to be more efficient than that of local sensors for the following since satellite weather data is almost as accurate as local offline sensor outputs.

CHAPTER 3

IDEATION AND PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS

An empathy map is a collaborative visualization used to express clearly what one knows about a particular type of user. It externalizes knowledge about users in order to create a shared understanding of user needs, and aid in decision making.

Empathy maps are split into 4 quadrants (Says, Thinks, Does, and Feels), with the user in the middle. Empathy maps provide a glance into who a user is as a whole.The ***Says*** quadrant contains what the user says or what he needs. The ***Thinks*** quadrant captures what the user is thinking throughout the experience. The ***Does*** quadrant encloses the actions the user takes. The ***Feels*** quadrant is the user's emotional state.

The empathy map for Industry Specific Intelligent Fire Management System is shown in Fig 3.1

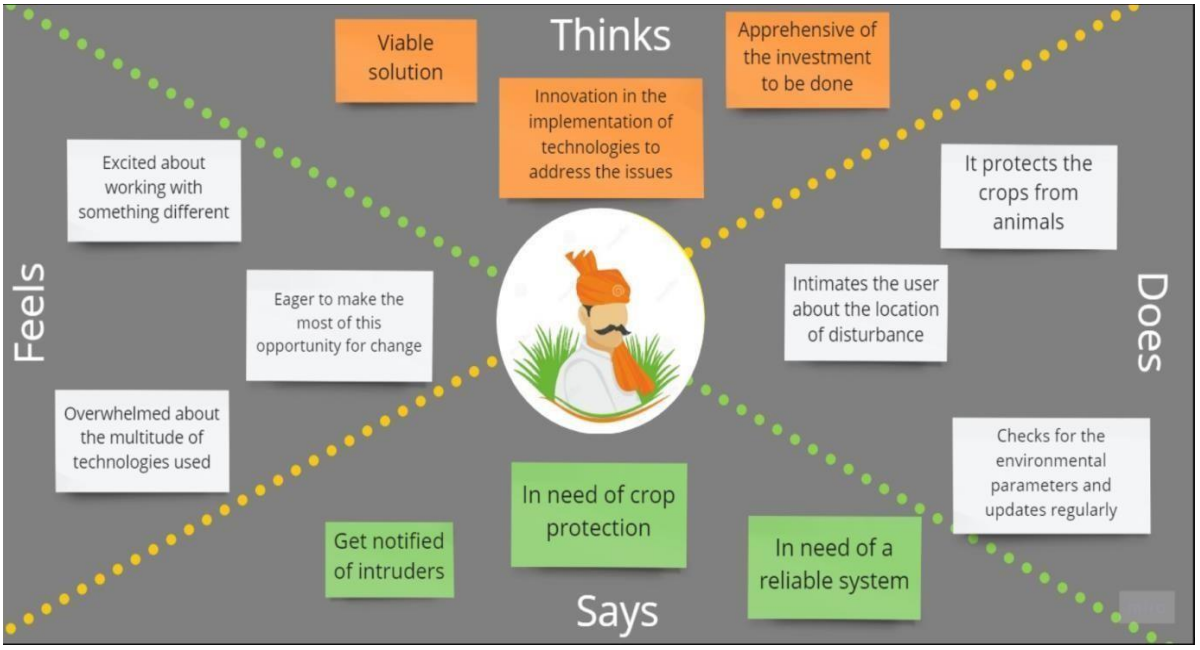


Fig 3.1 Empathy map

3.2 IDEATION AND BRAINSTORMING

Ideation is often closely related to the practice of brainstorming, a specific technique that is utilized to generate new ideas. Brainstorming is usually conducted by getting a group of people together to come up with either general new ideas or ideas for solving a specific problem or dealing with a specific situation. A principal difference between ideation and brainstorming is that ideation is commonly more thought of as being an individual pursuit, while brainstorming is almost always a group activity. Both brainstorming and ideation are processes invented to create new valuable ideas, perspectives, concepts and insights, and both are methods for envisioning new frameworks and systemic problem solving.

The Ideation chart for Industry Specific Intelligent Fire Management System is shown in Table 3.1.

IDEA 1	IDEA 2	IDEA 3
Crop protection from animals using IR motion detectors The farmland is surrounded by fences and each fence is equipped with multiple IR motion detectors in various heights. Location of each motion detector is surveyed and stored in the database.	A user interface system for farmers to analyze the data. The data to the system are sensor data from Humidity sensor, Temperature sensor, PIR Sensor and they are processed using a microcontroller and stored in a database. This database also gives an overview on crop yields, profit and losses for the farmer,	Crop protection from environmental factors such as UV rays, temperature, humidity, moisture content in soil. Using color sensors to detect NPK values of the soil and determining its fertility this data can be used to determine what type of fertilizers to be used. These factors can play a major role in crop protection and crop yield.

Cameras are placed in suitable locations so that we get a complete view over the farmland. When an animal or the intruder enters the field. The IR detectors which are placed in various heights are used to detect the type of the animal which has entered the field and the size of the animal. Alarms can be used to alert when large animals enter the field. And the camera is activated when the IR sensor detects motion. Then the picture is sent to the farmer.	what crop has been sowed and Expenses. This database can be used in the future for analyzing a pattern for best yields, to minimize the expenses and help the farmer take decisions financially.	So having control over these will help to improve the yield. Sensors for UV concentration, Moisture content, temperature are measured and water sprinklers are used to control the parameters accordingly.

3.3 PROPOSED SOLUTION

The proposed solution for IOT Based Smart crop protection system for Agriculture is shown in table 3.2

ATTRIBUTE	DEFINITION	STUDENTS WORK
Novelty	The proposed solution needs to be fundamentally different from what people already know..	A toolbox of novel techniques based on the integration of crop prediction system and Internet of things
Feasibility of idea	Project feasibility is the study of a project's various elements to determine if it has the potential for success.	Earlier crop protection systems required manpower to detect intrusion and protect the crops. But, this project makes use of Iot technology with ultrasonic frequencies being used to prevent the intrusion.
Business model	Create a model for identifying products and services to sell the market to target and also take into account anticipated expenses.	This project can be applied to different sectors of farming. Since it reduces the involvement of humans by bringing into picture new technologies, the cost of manufacturing also drops down. This makes the crop products easily available to the end user.

Social Impact	Social impact is how organizations, businesses or individuals' actions affect the surrounding community.	One of the major factors that has a direct impact on crop protection systems is the area where it is conceived. Different areas mean different types of crops and different types of intrusion. It therefore places a demand on the developer to configure the sensor values in such a way that it suits all kinds of environment and come up with a detection system that is common to all environments yet uniquely identifies the disturbance.
Scalability of solution	Scalability is system that can accommodate expansion without hampering the existing workflow and ensure an increase in the output or efficiency of the process.	Since this system uses computer vision techniques integrated with IBM cloudant services helps efficiently to retrieve images in large scale thus improving scalability.

3.4 PROBLEM SOLUTION FIT

The Problem solution fit simply means that one have found a problem with the customer and that the solution one have realised for it actually solves the customers problem. The problem solution fit is an important step towards the Product-Market Fit. The structure of problem solution fit is given below.

Customer state fit: To make sure one understand the target group, their limitations and their currently available solutions, against which one is going to compete.

Problem-Behavior fit: To help one to identify the most urgent and frequent problems, understand the real reasons behind them and see which behavior supports it.

Communication-Channel fit: To help one to sharpen the communication with strong triggers, emotional messaging and reaching customers via the right channels.

Solution guess: Translate all the validated data one have gathered into a solution that fits the customer state and his/her limitations, solves a real problem and taps into the common behavior of the target group.

The problem solution fit for IOT based smart crop protection system using for agriculture is shown in Fig 3.4

1. Customer Segment • Large scale Farmers • Silos owners	6. Customer Limitations • Animal Intrusions • Effects due to environment • Fertility of soil	5. Available Solutions • Electric fences • Humidity Management Models • Crop Management software
2. Problems / Pains It is difficult for Large scale farmers to manage and protect their resources from animal intrusions and external factors. There is also no specific software to manage and collect all the relevant information.	9. Problem root / Cause • Wild Animals • Environmental Factors (Excess greenhouse gasses, High Temperatures) • Soil fertility	7. Behavior • Gain knowledge on the existing solutions and try to learn more on the products available in this domain.
3. Triggers to act • Real time water sprinklers for controlling humidity • Motion detectors to check on intruders and animals 4. Emotions Before: Stressed, Unprepared, Helpless After: Stress free, Fearless	10. Your Solution • Crop protection from animals using IR motion detectors • A user interface system for farmers to analyze the data • Crop protection from environmental factors such as UV rays, temperature, humidity, moisture content in soil.	8. Channels of Behavior Gather information from websites and journals about the existing models

Fig 3.4 Problem Solution fit

CHAPTER 4
REQUIREMENT ANALYSIS

Requirements analysis is very critical process that enables the success of a system or software project to be assessed. Requirements are generally split into two types: Functional and Non-functional requirements.

4.1 FUNCTIONAL REQUIREMENTS

These are the requirements that the end user specifically demands as basic facilities that the system should offer. All these functionalities need to be necessarily incorporated into the system as a part of the contract. These are represented or stated in the form of input to be given to the system, the operation performed and the output expected. They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements.

The following table 4.1 shows the functional requirements for The IOT based smart crop protection system using for agriculture.

FR NO.	FUNCTIONAL REQUIREMENT (EPIC)	SUB REQUIREMENT (STORY / SUB-TASK)
FR-1	User registration	Download the app Registration through Gmail Create an account Follow the instructions
FR-2	User Confirmation	Confirmation via Email

		Confirmation via OTP
FR-3	Interface sensor	Interface sensor and the application so if animals enter the field it gives alarm.
FR-4	Accessing datasets	Datasets are retrieved from Cloudant DB
FR-5	Mobile application	Motors and sprinklers in the field can be controlled by mobile application.

4.2 NON-FUNCTIONAL REQUIREMENTS

These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other. They are also called non-behavioral requirements.

They basically deal with issues like Portability, Security, Maintainability, Reliability,Scalability,Performance,Reusability,Flexibility.

The following table 4.2 shows the Non-Functional Requirements of IoT Based Smart Crop Protection System.

FR NO.	NON-FUNCTIONAL REQUIREMENT	DESCRIPTION
NFR-1	Usability	The smart protection system defines that this project helps farmers to protect the farm.

NFR-2	Security	We have designed this project to secure the crops from animals.
NFR-3	Reliability	This project will help farmers in protecting their fields and save them from significant financial losses. This will also help them in achieving better crop yields thus leading to their economic well being.
NFR-4	Performance	IOT devices and sensors are used to indicate the farmer by a message when animals try to enter into the field and also we use an SD card module that helps to store a specified sound to scare the animals.
NFR-5	Availability	By developing and deploying resilient hardware and software we can protect the crops from wild animals.
NFR-6	Scalability	Since this system uses computer vision techniques integrated with IBM cloudant services helps efficiently to retrieve images in large scale thus improving scalability

CHAPTER 5

PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyze an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually "say" things that would be hard to explain in words, and they work for both technical and nontechnical audiences, from developer to CEO. That's why DFDs remain so popular after all these years. While they work well for data flow software and systems, they are less applicable nowadays to visualizing interactive, real-time or database-oriented software or systems.

There are four main elements of a DFD — external entity, process, data store, and data flow.

External entity

An external entity, which are also known as terminators, sources, sinks, or actors, are an outside system or process that sends or receives data to and from the diagrammed system. They're either the sources or destinations of information, so they're usually placed on the diagram's edges. External entity symbols are similar across models except for Unified, which uses a stick-figure drawing instead of a rectangle, circle, or square.

Process

Process is a procedure that manipulates the data and its flow by taking incoming data, changing it, and producing an output with it. A process can do this by performing computations and using logic to sort the data, or change its flow of direction. Processes usually start from the top left of the DFD and finish on the bottom right of the diagram.

Data store

Data stores hold information for later use, like a file of documents that's waiting to be processed. Data inputs flow through a process and then through a data store while data outputs flow out of a data store and then through a process.

Data flow

Data flow is the path the system's information takes from external entities through processes and data stores. With arrows and succinct labels, the DFD can show the direction of the data flow.

The data flow diagram for IOT based smart crop protection system using for agriculture is shown in following figure 5.1

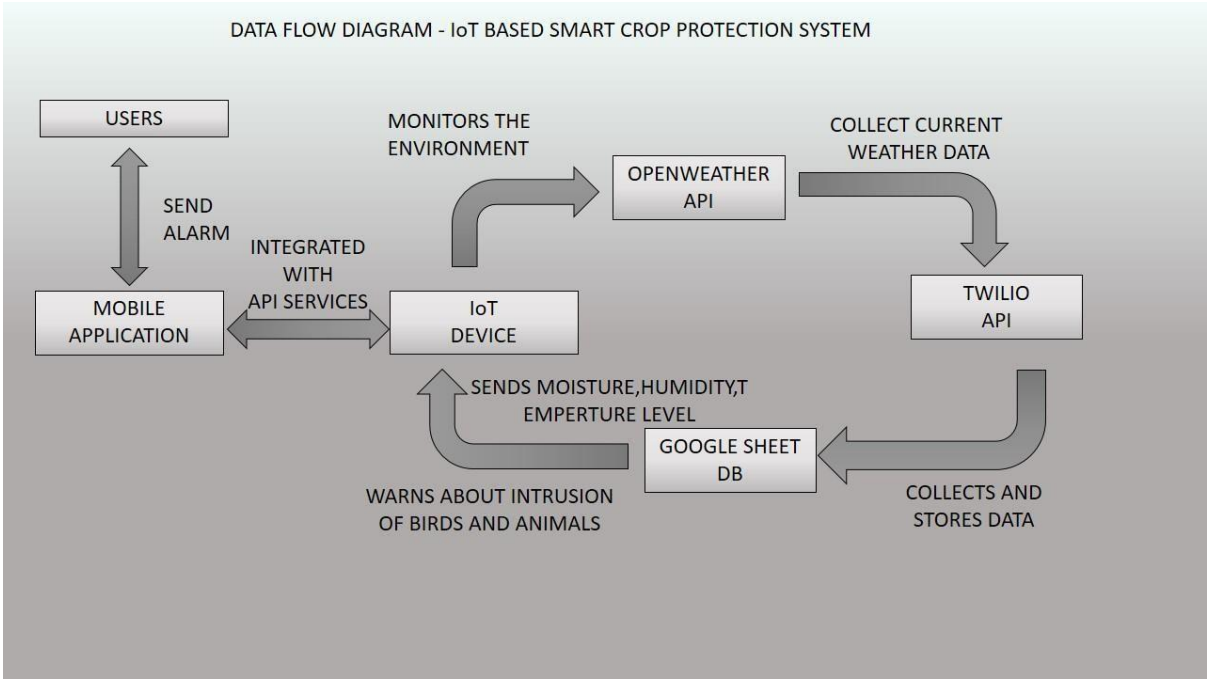


Fig 5.1 Data Flow Diagram for IoT based smart crop protection system

5.2 SOLUTION AND TECHNICAL ARCHITECHTURE

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, behaviour, 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.

The figure 5.2 shows the solution architecture of IoT based smart crop protection system

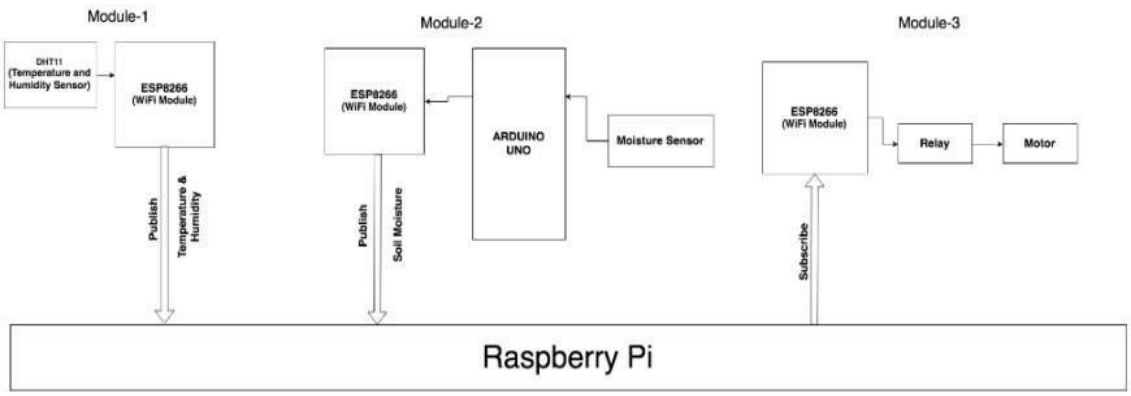


Fig 5.2 Solution Architecture of IoT based smart crop protection system

CHAPTER 6

PROJECT PLANNING & SCHEDULING

6.1 SPRINT PLANNING AND ESTIMATION

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. Sprint planning is done in collaboration with the whole scrum team.

The sprint is a set period of time where all the work is done. However, before leap into action it is necessary to set up the sprint. It need to decide on how long the time box is going to be, the sprint goal, and where it is going to start. The sprint planning session kicks off the sprint by setting the agenda and focus. If done correctly, it also creates an environment where the team is motivated, challenged, and can be successful.

The Table 6.1 shows the sprint planning and estimation of IoT Based Smart Crop Protection System.

SPRINT	FUNCTIONAL REQUIREMENT (EPIC)	USER STORY NUMBER	USER STORY / TASK	STORY POINTS	PRIORITY
Sprint-1	User registration	USN-1	Download the app Register through gmail Create an account Follow the instructions	2	High
Sprint-1	User Confirmation	USN-2	Confirmation via OTP Confirmation via mail	1	High
Sprint-2	Interfacing sensor	USN-3	Interface the sensor with raspberry pi and then the application is developed. So it notifies the user via mail and message whenever there is a movement of animals and birds.	2	Low
Sprint-1	Accessing datasets	USN-4	As a user, I can register for the application through Gmail	2	Medium

Sprint-1	Mobile application	USN-5	As a user, I can log into the application by entering email & password	1	High

6.1 SPRINT DELIVERY SCHEDULE

The sprint delivery plan is scheduled accordingly as shown in the below table 6.2 which consists of the sprints with respective to their duration, sprint start and end date and the releasing data.

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	03 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

CHAPTER 7
CODING & SOLUTIONING

7.1 FEATURE

- Warning and text message to mobile number when animal is found in the farm
- Online database using Google Sheets to store the results of the sensors
- Using weather API to notify if the whether is extreme to the user
- Soil moisture is also monitored and notified to the user through SMS about it, so that the farmer can ensure the required level of moisture in the soil.

Basic explanation:

Libraries required:

```
In [6]: import pandas as pd
        import random
        import requests, json
        import time
        from twilio.rest import Client
        import gspread
        from datetime import datetime
```

Fig 7.1 Libraries required

Pandas :

Pandas is an open source Python package that is most widely used for data science/data analysis and machine learning tasks. It is built on top of another package named Numpy, which provides support for multi-dimensional arrays

Random:

Sometimes we want the computer to pick a random number in a given range, pick a random element from a list, pick a random card from a deck, flip a coin, etc. The random module provides access to functions that support these types of operations.

In this case random is used to generate the random values between specified float values to recreate sensor data.

JSON:

JSON is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays. It is a common data format with diverse uses in electronic data interchange, including that of web applications with servers.

Time:

Python time module allows to work with time in Python. It allows functionality like getting the current time, pausing the Program from executing, etc. In this program the time library is used to get the current time so that the data can be stored using time stamps.

Twilio:

The Twilio Python Helper Library makes it easy to interact with the Twilio API from your Python application. In this program this library is used to send SMS using twilio.

Gspread:

The gspread is a python helper library used to interact with the spread sheets in the google sheets and provide necessary functions to interact with the spread sheets. This library is used to update the values in the sheets database.

Creating Random Data:

```
In [2]: df = pd.DataFrame()
        username = "Pavithran 1904032 "

In [3]: rand_prox_location_1=[]
        constrain_location_1=9.7
        rand_prox_location_2=[]
        constrain_location_2=7
        rand_prox_location_3=[]
        constrain_location_3=9.5
        rand_prox_location_4=[]
        constrain_location_4=9.9
        soil_humidity=[]
        constrain_soil_humidity= 75
        constrain_temp= 295
        constrain_humidity= 80
        constrain_pressure= 1030
        constrain_soil_humidity_low =40
        for i in range(1,100):
            rand_prox_location_1.append(round(random.uniform(0.25,10),2))
            rand_prox_location_2.append(round(random.uniform(0.25,10),2))
            rand_prox_location_3.append(round(random.uniform(0.25,10),2))
            rand_prox_location_4.append(round(random.uniform(0.25,10),2))
            soil_humidity.append(round(random.uniform(1,100),1))
        print("--------",rand_prox_location_4)
        print("--------",rand_prox_location_3)
        print("--------",rand_prox_location_2)
        print("--------",rand_prox_location_1)
        print("--------",soil_humidity)

        -------- [5.21, 7.96, 3.89, 1.73, 5.69, 4.45, 8.86, 1.75, 3.76, 8.3, 7.96, 2.56, 5.3, 1.86, 8.64, 8.3
```

Fig 7.2 Creating Random data

In this particular snippet a data frame ‘df’ is created

And rand_prox_location_1 to ran_rpox_location_4 lists are created so that the random values can be temporarily stored in the runtime. These lists represent the various PIR sensor outputs which are in meters (distance from the pir sensor), constrain_location_1 to constrain_location_4 are the constraints provided for each sensor, which can be changed so that the values for each and every field(farm fields) dimensions.

Similarly soil humidity list and constraints are also created to represent the sensor data from the soil humidity sensor, constraints for the temperature, atmospheric humidity, pressure were also assigned.

In the following random uniform float values are generated and appended in the respective lists.

```
In [69]: df['Location_1']=rand_prox_location_1
         df['Location_2']=rand_prox_location_2
         df['Location_3']=rand_prox_location_3
         df['Location_4']=rand_prox_location_4
         df['soil_humidity']=soil_humidity
         df.to_excel('IBM_sensor_data.xlsx', index = False)
```

Fig 7.3 Random floating uniform float values

The location data and soil humidity data were then stored in an excel file knows as IBMVsensordata for further use.

Sending SMS:

TWILIO:

Fig 7.4 Twilio

Twilio is an American company based in San Francisco, California, which provides programmable communication tools for making and receiving phone calls, sending and receiving text messages, and performing other communication functions using its web service APIs.

Usage in this program:

```
In [73]: for i in range(100):
             time_stamp = datetime.now()
             time.sleep(5)
             temp=[0,0,0,0]
             value=[0,0,0,0]
             animal_intrusion_set=0
             temp_set=0
             soil_humidity_set=0
             pressure_set=0
             atmos_humidity_set=0
             animal_intrusion_text = ' Animal intrusion at'
             temporary_text= ''
             current_temperature,current_pressure,current_humidity,weather_description = get_whether()
             if current_temperature > constrain_temp:
                 temp_set=1
                 temporary_text += ' Temperature is high :'+str(current_temperature) + ' || '
             if current_pressure > constrain_pressure:
                 pressure_set=1
                 temporary_text += ' Pressure is high :'+str(current_pressure) + ' || '
             if current_humidity > constrain_humidity:
                 atmos_humidity_set=1
                 temporary_text += ' Atmosphereic humidity is high :'+str(current_humidity ) + ' || '
```

Fig 7.5 Program execution for getting temperature and humidity data

The time stamp of the particular instance is taken using the date and time function. And then time sleep(5) is used to provide a 5 second delay to replicate a real world scenario so that the data is sent into the code in a delay of five seconds.

Set values are used for every other parameter to check whether the particulars are crossed. Animal intrusion test is set to 'Animal intrusion at' and then message is appended when the particular thresholds are crossed.

And then if animal intrusion is found to be set then the message is sent using send sms function.

```
        value[2]=rand_prox_location_3[i]
    if rand_prox_location_4[i] > constrain_location_4:
        temp[3]=1
        value[3]=rand_prox_location_4[i]
    for j in range(len(temp)):
        if temp[j]==1:
            animal_intrusion_set=1
            animal_intrusion_text += ' location '+str(j)+ ' with a distance from sensor of ' + str(valu
    trigger=0
    loc_list = [rand_prox_location_1[i],rand_prox_location_2[i],rand_prox_location_3[i],rand_prox_locat

    if animal_intrusion_set==1:
        temporary_text+=animal_intrusion_text
        trigger = 1
    else:
        if temporary_text != '':
            trigger = 1
    update_sheet(loc_list,temporary_text , str(time_stamp),str(soil_humidity[i]) ,str(current_humidity
    if trigger ==1:
        print(temporary_text,username)
        print("------------------------------------------------------------------")
        print("updating db")
        print("------------------------------------------------------------------")
```

Fig 7.6 Program execution for Animal Intrusion

And then update sheet is used to update the required values into the sheets using the sheets api.

```
        trigger = 1
    else:
        if temporary_text != '':
            trigger = 1
    update_sheet(loc_list,temporary_text , str(time_stamp),str(soil_humidity[i]) ,str(current_humidity
    if trigger ==1:
        print(temporary_text,username)
        print("-------------------------------------------------------------------")
        print("updating db")
        print("-------------------------------------------------------------------")
        print("sending sms")
        print("-------------------------------------------------------------------")
        time.sleep(30)
        send_sms(temporary_text,username)
        print("sms sent")
        print("-------------------------------------------------------------------")
```

```
1000
total_length 10

-------------------------------------------------------------------
sending sms
-------------------------------------------------------------------
```

Fig 7.7 Sending SMS to the user

Send sms function:

```
In [71]: def send_sms(username,text):
             SID= 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
             auth_token='xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'

             cl = Client(SID, auth_token)
             cl.messages.create(body="Hey,"+username+text,from_='+1xxxxxxxxxxxxxxx',to='xxxxxxxxxxxxxxxx' )
```

Fig 7.8 SMS function

This send sms function is used to connect to the Twilio API and uses the SID and auth token given to send a message from the number given in the Twilio dashboard to the required number to which we should notify.

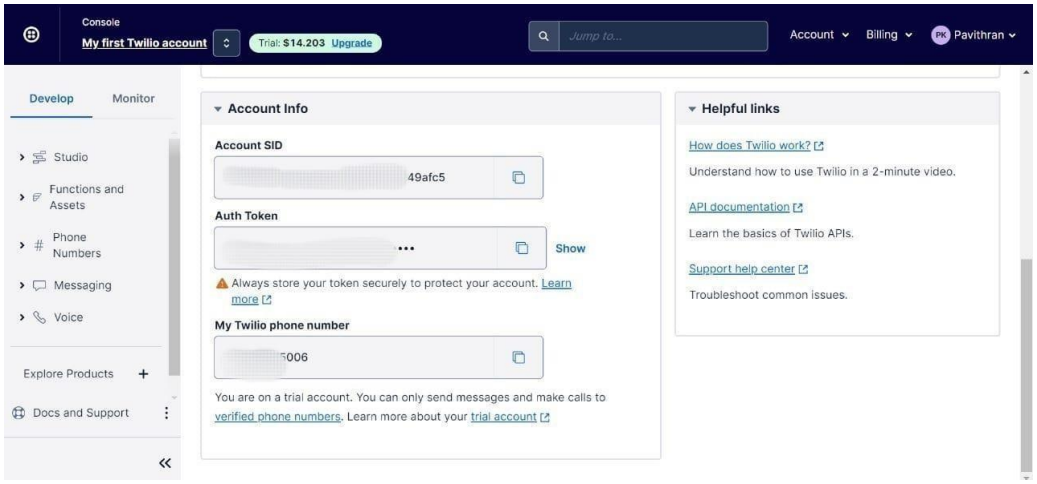


Fig 7.9 Twilio Dashboard

The above figure shows the Account SID, Auth token and the my Twilio phone number from which the messages are sent.

Free version of Twilio gives a trial credit of $15 and allows you to send a message until the credits get over. Also it notifies in each message that the Twilio trial version is used. which can be seen in the figure below.

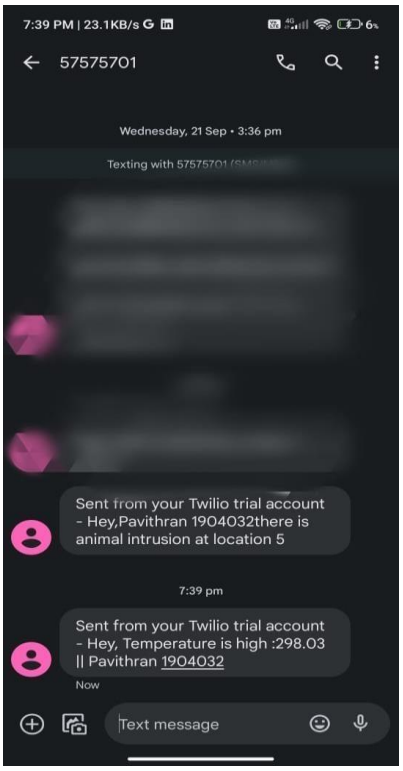


Fig 7.10 Message Output

The logs are saved in the dashboard for further reference. The from and to numbers are mentioned below.

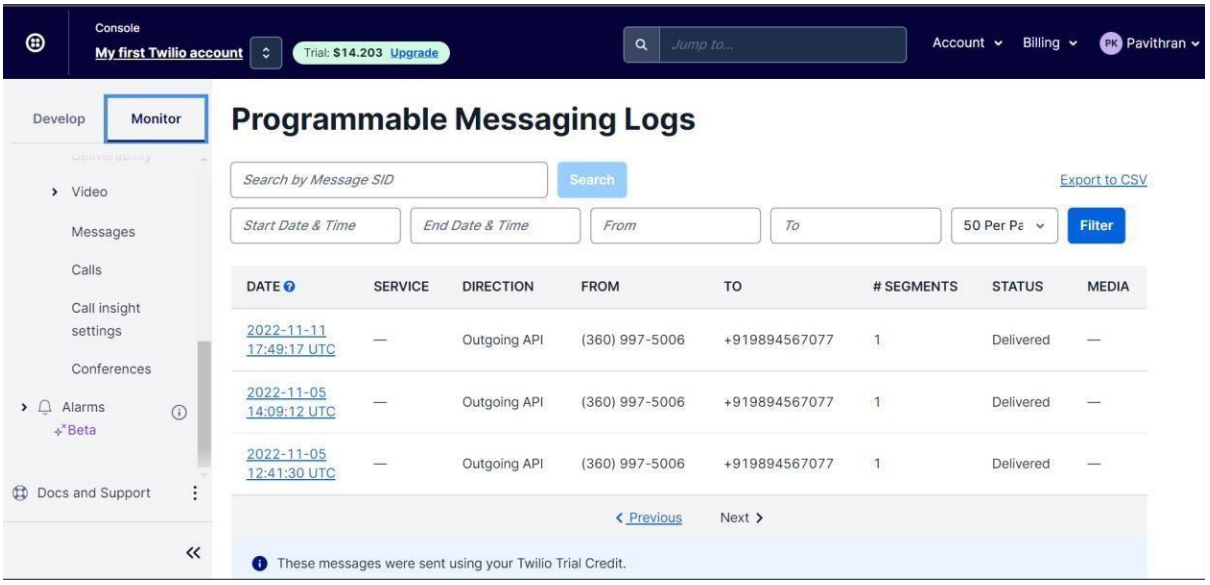


Fig 7.11 Twilio logs

Open weather API:

Fig 7.12 Open weather API

Open Weather Map is an online service, owned by OpenWeather Ltd, that provides global weather data via API, including current weather data, forecasts, nowcasts and historical weather data for any geographical location. The company provides a minute-by-minute hyperlocal precipitation forecast for any location.

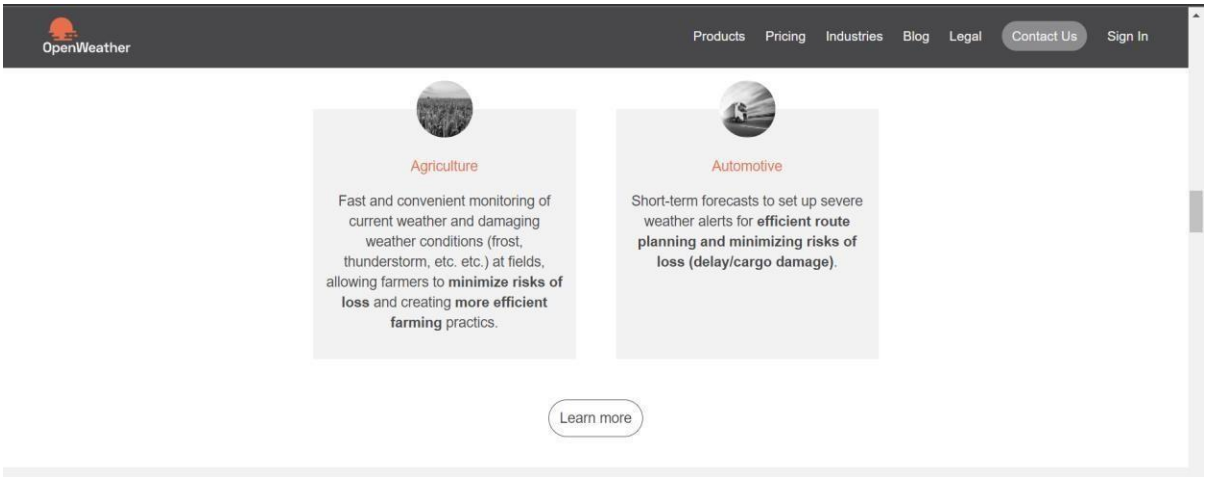


Fig 7.13 Open Weather Website

Open weather api is used because this is one of the most reliable weather tracking API among all others and agriculture is the top most sector the open weather api is used. This API also provides a free tier with about 60 API calls per minute that is one api call for every second which is more than enough for our use case.

Getting weather data using API:

Get_whether function api key and base url of the open weather map upi is used to fetch the weather data and parameters such as temperature, humidity, pressure and a small weather description.

This function returns the weather data to the main function where the parameters are checked with their thresholds before calling the send sms function.

```
In [20]: def get_whether():
             api_key = "xxxxxxxxxxxxxxxxxxxxxxxx"
             base_url = "http://api.openweathermap.org/data/2.5/weather?"
             city_name = "Coimbatore"
             complete_url = base_url + "appid=" + api_key + "&q=" + city_name
             response = requests.get(complete_url)
             x = response.json()
         #     print(x)
             if x["cod"] != "404":
                 y = x["main"]
                 current_temperature = y["temp"]
                 current_pressure = y["pressure"]
                 current_humidity = y["humidity"]
                 z = x["weather"]
                 weather_description = z[0]["description"]
         #         print(" Temperature (in kelvin unit) = " + str(current_temperature))
         #         print(" Atmospheric pressure (in hPa unit) = " +str(current_pressure))
         #         print(" Humidity (in percentage) = " + str(current_humidity))
         #         print(" Description = " + str(weather_description))
                 return(current_temperature,current_pressure,current_humidity,weather_description)
             else:
                 print(" City Not Found ")
```

Fig 7.14 Get Weather function

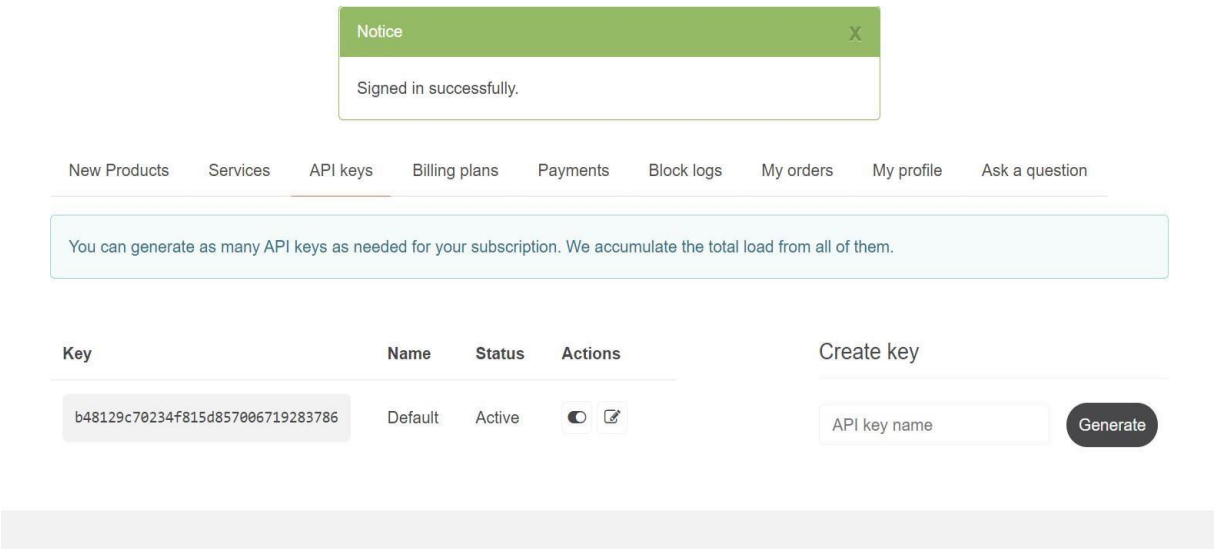


Fig 7.15 Key values and the status of the open weather API

Google sheets and drive API for cloud DB:

Fig 7.16 Google sheets for developers

The Google Sheets API lets you read, write, and format Google Sheets data with your preferred programming language, including Java, JavaScript, and Python.

Why You Should Use Google Sheets as a Database?

Google sheets are not the complete solution for the database management of your business. But, it works wonders for small businesses and projects to handle their business financial operations efficiently.
In this case the information that needed to be stored in the cloud are in strings and integers for which google sheets database is more than sufficient.
The following are the features that google sheets api provides

Access Control:

It has full access to edit, delete, and share it with others with just a click on google sheets. Here, you can revoke granted permissions. You can provide different levels of access to users, such as edit, view, and command.

Connectivity:

Google sheets is a web application, and your complete data is stored in a cloud-based database. Hence you no need to worry about losing your data; you can access your data online.

Pricing:

Google sheets provide unlimited free access to its users. Unlike other Microsoft Excel spreadsheets, they will ask you to buy a license or subscription for it after a certain period. Thus, it will reduce the overall cost spent on database management system tools.

Visualisation:

Like other spreadsheets, Google Sheets also has many built-in features to customise the overall visualisation and analyse it. You can add, delete or update new features based on your preference.

Google Cloud Console:

A new project is created in the google cloud (named IBM-crop-protection) and google drive and sheets api are enabled from there.
Google drive api is used because it is a necessity to use Google Sheets api to integrate the online database with the program offline.

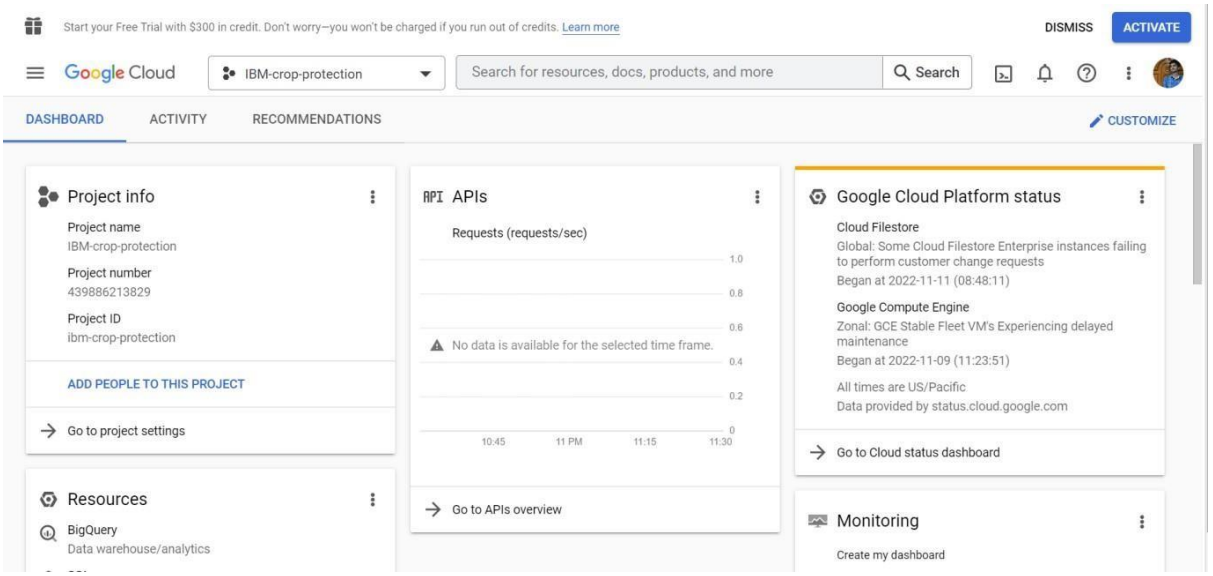


Fig 7.17 Google cloud Dashboard

How it is used in this program:

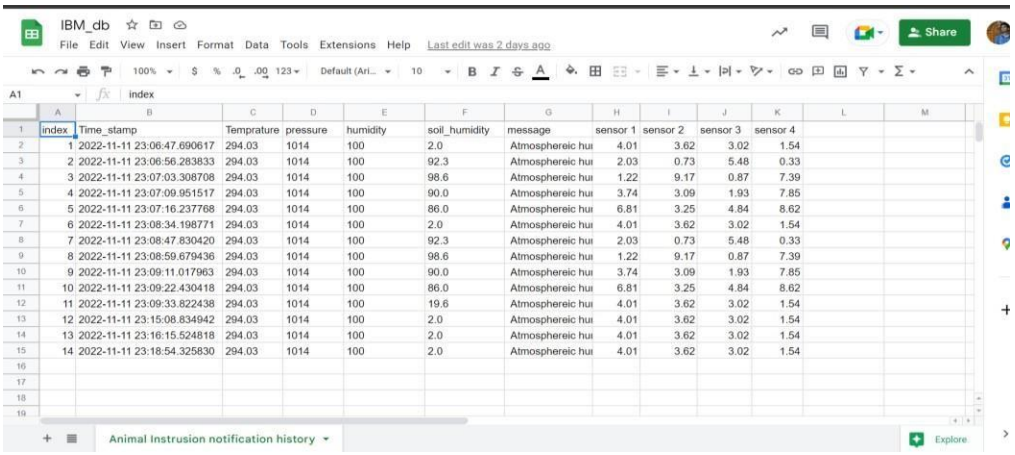
```
In [5]: sa= gspread.service_account(filename=r"C:\Users\K B PAVITHRAN\ibm-crop-protection-9837998227d9.json")
        sh = sa.open("IBM_db")
        whs =sh.worksheet("Animal Instrusion notification history")
```

```
In [71]: def update_sheet(loc_list,message,time_stamp,soil_humidity,current_humidity,current_pressure,current_te
         #     update_sheet(loc_list,temporary_text , str(time_stamp),str(soil_humidity[i]) ,str(current_humidit
             index = 'A'
             time_stamp_index = 'B'
             current_temperature_index = 'C'
             current_pressure_index = 'D'
             current_humidity_index= 'E'
             soil_humidity_index = 'F'
             message_index = 'G'
             location_1_index='H'
             location_2_index='I'
             location_3_index='J'
             location_4_index='K'
             TEMP_LEN = len(whs.get_all_values())
             if TEMP_LEN ==0:
                 TEMP_LEN =1
             print(whs.get_all_values())
```

Fig 7.18 Update sheet function

Gspread.service_account is used to access the JSON file which has the required credentials to access the sheets from the sheets api.

Indexes of various kinds are declared as ‘A’ to ‘K’ each representing the column numbers in the sheet, which can be seen in the figure below.



index	Time_stamp	Temprature	pressure	humidity	soil_humidity	message	sensor 1	sensor 2	sensor 3	sensor 4
1	2022-11-11 23:06:47.690617	294.03	1014	100	2.0	Atmosphereic hu	4.01	3.62	3.02	1.54
2	2022-11-11 23:06:56.283833	294.03	1014	100	92.3	Atmosphereic hu	2.03	0.73	5.48	0.33
3	2022-11-11 23:07:03.308708	294.03	1014	100	98.6	Atmosphereic hu	1.22	9.17	0.87	7.39
4	2022-11-11 23:07:09.951517	294.03	1014	100	90.0	Atmosphereic hu	3.74	3.09	1.93	7.85
5	2022-11-11 23:07:16.237768	294.03	1014	100	86.0	Atmosphereic hu	6.81	3.25	4.84	8.62
6	2022-11-11 23:08:34.198771	294.03	1014	100	2.0	Atmosphereic hu	4.01	3.62	3.02	1.54
7	2022-11-11 23:08:47.830420	294.03	1014	100	92.3	Atmosphereic hu	2.03	0.73	5.48	0.33
8	2022-11-11 23:08:59.679436	294.03	1014	100	98.6	Atmosphereic hu	1.22	9.17	0.87	7.39
9	2022-11-11 23:09:11.017963	294.03	1014	100	90.0	Atmosphereic hu	3.74	3.09	1.93	7.85
10	2022-11-11 23:09:22.430418	294.03	1014	100	86.0	Atmosphereic hu	6.81	3.25	4.84	8.62
11	2022-11-11 23:09:33.822438	294.03	1014	100	19.6	Atmosphereic hu	4.01	3.62	3.02	1.54
12	2022-11-11 23:15:08.834942	294.03	1014	100	2.0	Atmosphereic hu	4.01	3.62	3.02	1.54
13	2022-11-11 23:16:15.524818	294.03	1014	100	2.0	Atmosphereic hu	4.01	3.62	3.02	1.54
14	2022-11-11 23:18:54.325830	294.03	1014	100	2.0	Atmosphereic hu	4.01	3.62	3.02	1.54

Fig 7.19 Google Sheets Database

‘whs’ is the worksheet we are working and the indexes are parsed accordingly so that it can be fed into the function whs.update which takes in the parameters location of the cell and the values that need to be written in that particular cell given that both are in string format. The illustration below shows how the values are updated in the google sheets database

```
print(whs.get_all_values())
print(whs.row_count)
print("total_length",TEMP_LEN)

index += str(TEMP_LEN +1)
current_temperature_index +=str((TEMP_LEN) +1)
current_pressure_index +=str((TEMP_LEN) +1)


current_humidity_index +=str((TEMP_LEN) +1)
soil_humidity_index +=str((TEMP_LEN) +1)
message_index +=str((TEMP_LEN) +1)
time_stamp_index +=str((TEMP_LEN) +1)

location_1_index +=str((TEMP_LEN) +1)
location_2_index +=str((TEMP_LEN) +1)
location_3_index +=str((TEMP_LEN) +1)
location_4_index +=str((TEMP_LEN) +1)

whs.update(index,TEMP_LEN)
whs.update(message_index,message  )
whs.update(time_stamp_index,time_stamp)
whs.update(current_temperature_index,current_temperature)
whs.update(current_pressure_index,current_pressure)
whs.update(soil_humidity_index,soil_humidity)
whs.update(current_humidity_index,current_humidity)
whs.update(location_1_index,loc_list[0])
whs.update(location_2_index,loc_list[1])
whs.update(location_3_index,loc_list[2])
whs.update(location_4_index,loc_list[3])
```

Fig 7.20 Usage of whs.update function

Get the values out of the database:

To do this the get all values() function is used so that the values are printed.

```
def get_values():
    data=whs.get_all_values()
    for i in data:
        print(i )
```

Fig 7.21 Get values function

The values displayed are in the form of a list where the first list contains the name of each columns and the remaining are the subsequent rows, which contains the values of each sensor and the text message that is sent to the farmer.

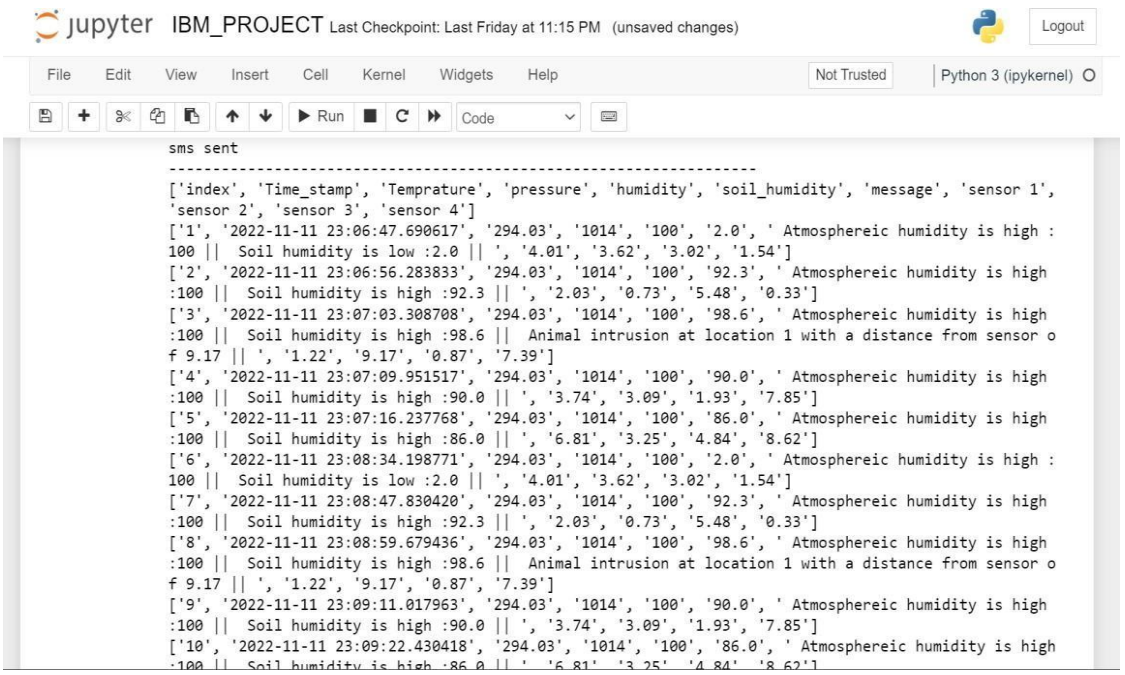


Fig 7.22 Program execution output

Activate Pump when soil humidity is low:

A sms is sent to a particular number when the soil humidity goes below a particular level, after which the local Arduino board will read the message and turn on the sprinkler pump system to activate the sprinkler for 10 minutes after which it stops and if the humidity is further low this process continues.

```
if soil_humidity[i] < constrain_soil_humidity_low:
    soil_humidity_set=1
    temporary_text += ' Soil humidity is low :'+str(soil_humidity[i]) + ' || '
    send_sms("activate the pump",username)
    print("The soil humidity is low turning on the pump")
```

Fig 7.23 Soil humidity code

Exception handling:

Try except blocks are used to handle exceptions:

The exceptions that are handled are

1. When weather api cannot find the location
2. When sms cannot be sent

```
        print("-----------------------------------------------------------------")
        time.sleep(30)
        try:
            send_sms(temporary_text,username)
            print("sms sent")
            print("-----------------------------------------------------------------")
        except:
            print("failed to send sms")
        get_values()
except:
    print("please enter a valid location")
```

Fig 7.24 Exception handling

Webpage:

Figure 7.25 shows the webpage which has the updated database and it is shown to the user and the user also gets the option to download or visit the database directly in a click of a button.

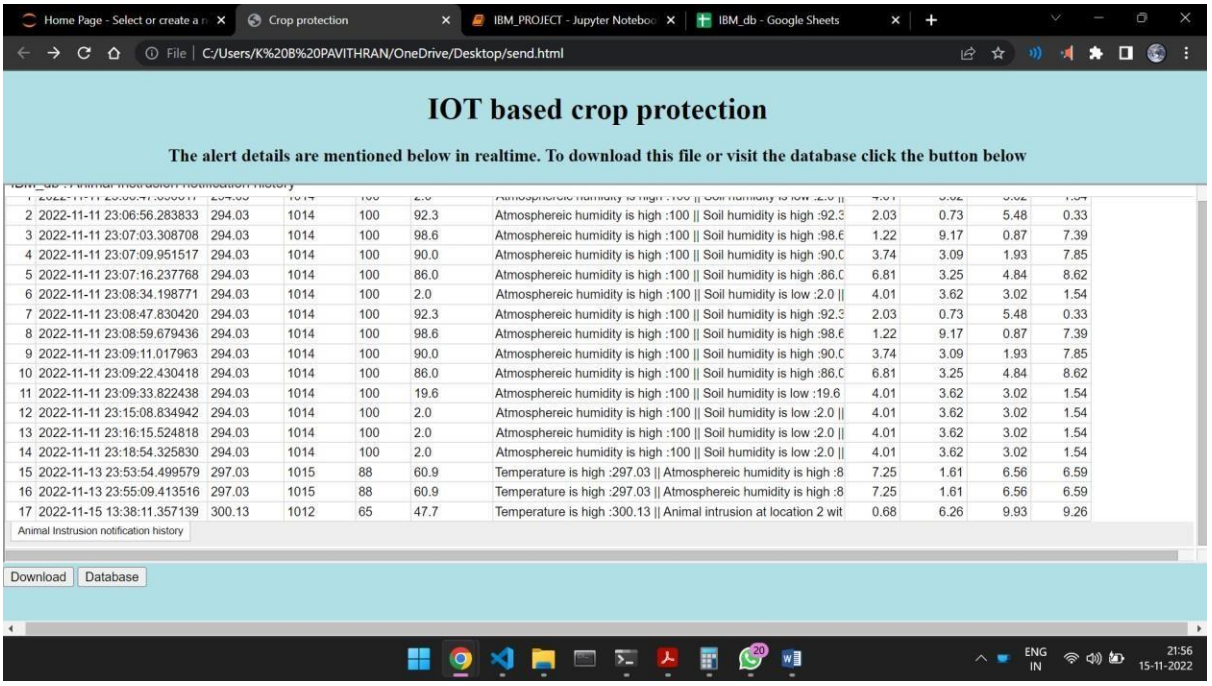


Fig 7.25 Webpage

Wokwi Simulation of the project:

ESP32 :

ESP32 is a series of low-cost, low-power system on chip microcontrollers with integrated Wi-Fi and dual-mode Bluetooth. The ESP32 series employs either a Ten silica Xtensa LX6 microprocessor in both dual-core and single-core variations, Xtensa LX7 dual-core microprocessor and a single-core RISC-V microprocessor and includes built-in antenna switches, RF balun, power amplifier, low-noise receive amplifier, filters, and power-management modules. ESP32 is created and developed by Espressif Systems, a Shanghai-based Chinese company, and is manufactured by TSMC using their 40 nm process. It is a successor to the ESP8266 microcontroller.

PIR sensor:

PIR sensors allow you to sense motion, usually 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 do not wear out. For that reason, they are commonly found in appliances and gadgets used in homes or businesses. They are often referred to as PIR, "Passive Infrared", "Pyroelectric", or "IR motion" sensors.

Working:

The figure below represents the simple block diagram of the setup using wokwi simulation. ESP32 is connected with a LED, Buzzer and a PIR sensor so that if the sensor detects any motion then it notifies to the ESP32 and then it can be visualised using a display.

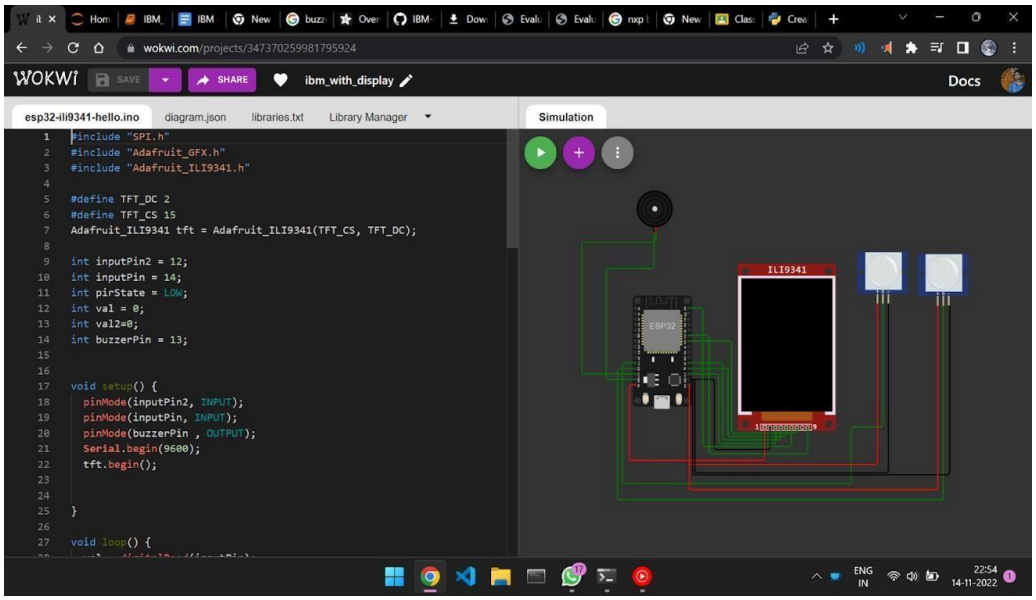


Fig 7.26 Wowki Simulation

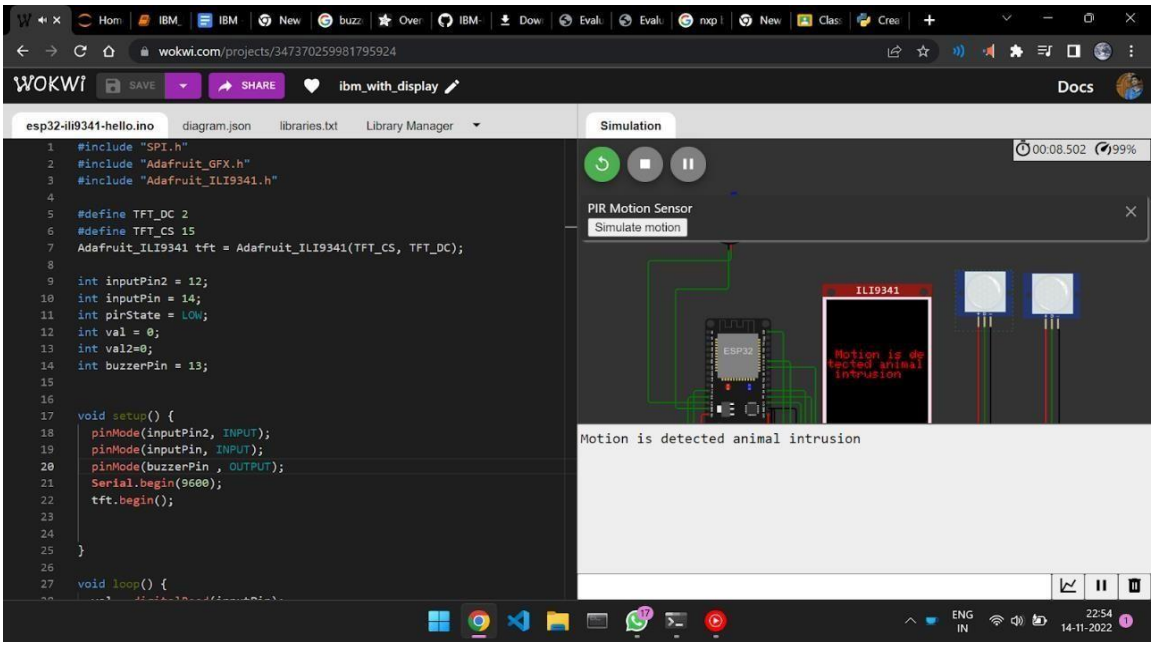


Fig 7.27 Wowki Simulation

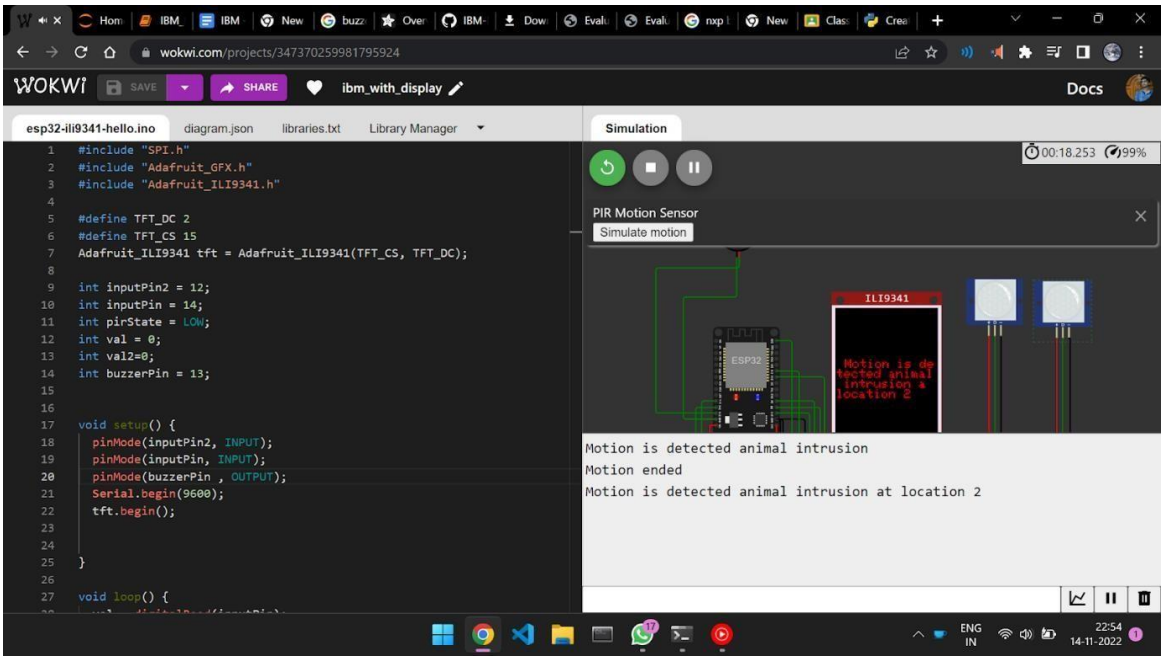


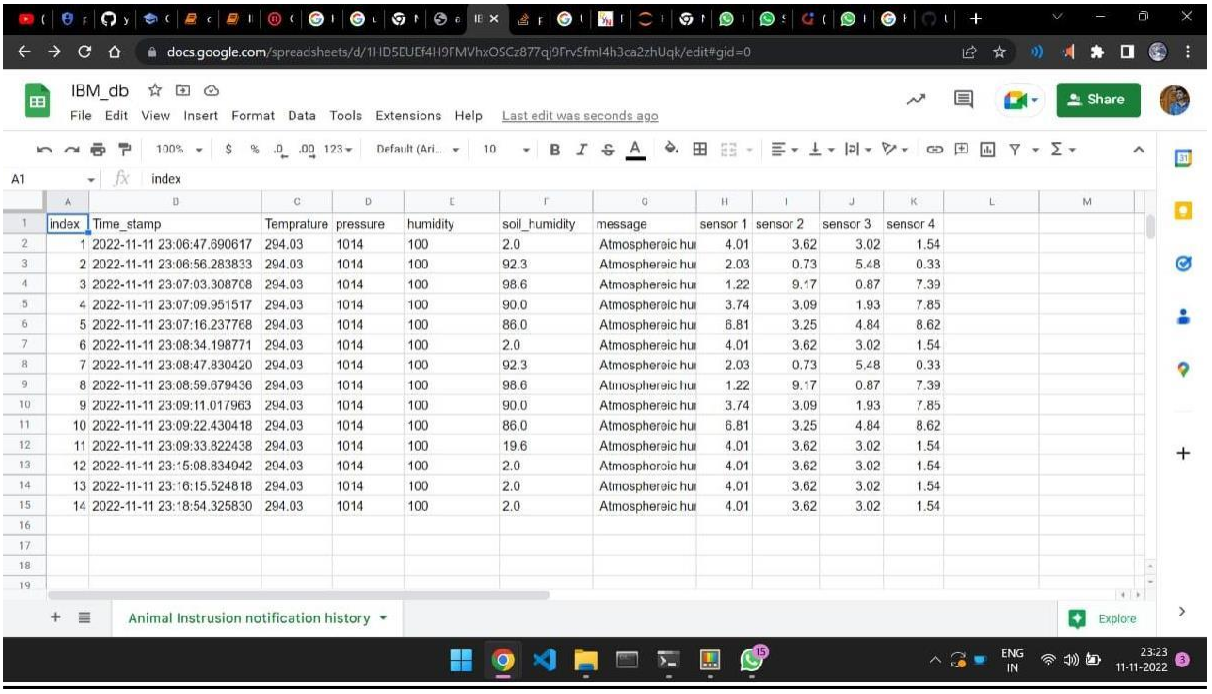
Fig 7.28 Wowki Simulation

CHAPTER 8

TESTING AND RESULTS

8.1 TEST CASES

From the below figure 8.1 and 8.2 it can be seen that the 14th index is the value , where the temperature is high which means the message should be sent to the farmer regarding the same This value is updated in the google sheets database so that it can be used for further analysis. Typical time taken for the message to reach the user after the detection of intrusion is 30 to 40 seconds, so the response time is average around 35 seconds



index	Time_stamp	Temprature	pressure	humidity	soil_humidity	message	sensor 1	sensor 2	sensor 3	sensor 4
1	2022-11-11 23:06:47.390617	294.03	1014	100	2.0	Atmospheric hu	4.01	3.62	3.02	1.54
2	2022-11-11 23:06:56.283833	294.03	1014	100	92.3	Atmospheric hu	2.03	0.73	5.48	0.33
3	2022-11-11 23:07:03.308708	294.03	1014	100	98.6	Atmospheric hu	1.22	9.17	0.87	7.39
4	2022-11-11 23:07:09.951517	294.03	1014	100	90.0	Atmospheric hu	3.74	3.09	1.93	7.85
5	2022-11-11 23:07:16.237768	294.03	1014	100	86.0	Atmospheric hu	6.81	3.25	4.84	8.62
6	2022-11-11 23:08:34.198771	294.03	1014	100	2.0	Atmospheric hu	4.01	3.62	3.02	1.54
7	2022-11-11 23:08:47.830420	294.03	1014	100	92.3	Atmospheric hu	2.03	0.73	5.48	0.33
8	2022-11-11 23:08:59.379436	294.03	1014	100	98.6	Atmospheric hu	1.22	9.17	0.87	7.39
9	2022-11-11 23:09:11.017963	294.03	1014	100	90.0	Atmospheric hu	3.74	3.09	1.93	7.85
10	2022-11-11 23:09:22.430418	294.03	1014	100	86.0	Atmospheric hu	6.81	3.25	4.84	8.62
11	2022-11-11 23:09:33.822438	294.03	1014	100	19.6	Atmospheric hu	4.01	3.62	3.02	1.54
12	2022-11-11 23:15:08.834042	294.03	1014	100	2.0	Atmospheric hu	4.01	3.62	3.02	1.54
13	2022-11-11 23:16:15.524818	294.03	1014	100	2.0	Atmospheric hu	4.01	3.62	3.02	1.54
14	2022-11-11 23:18:54.325830	294.03	1014	100	2.0	Atmospheric hu	4.01	3.62	3.02	1.54

Fig 8.1 Database update for testing

14	2022-11-11 23:18:54.325830	294.03	1014	100	2.0	Atmosphereic hu	4.01	3.62	3.02	1.54

Fig 8.2 The updated value

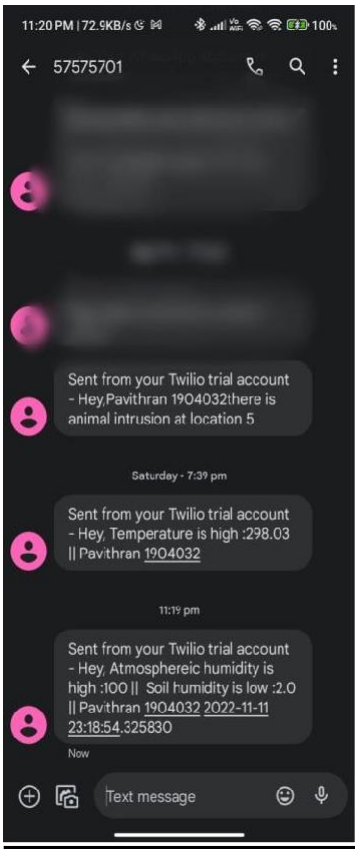
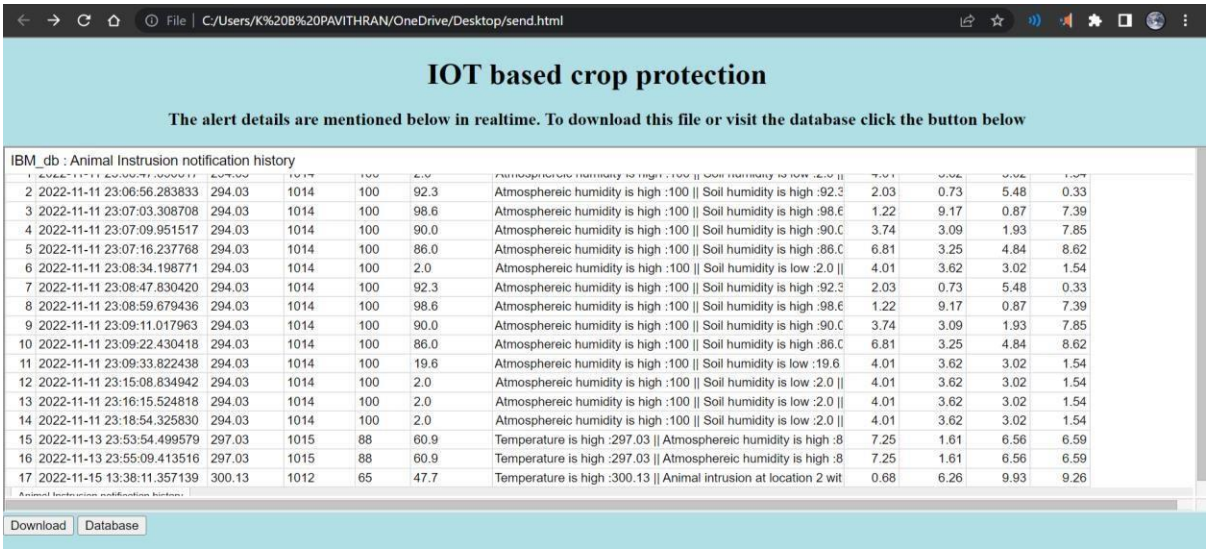


Fig 8.3 Message Output testcase

Fig 8.3 shows the text message the farmer receives regarding the high temperature and the low soil humidity with date and time. Fig 8.4 shows the website which is shown in the user side to view and download the database.



File | C:/Users/K%20B%20PAVITHRAN/OneDrive/Desktop/send.html

IOT based crop protection

The alert details are mentioned below in realtime. To download this file or visit the database click the button below

IBM_db : Animal Instrusion notification history

2	2022-11-11 23:06:56.283833	294.03	1014	100	92.3	Atmosphereic humidity is high :100 || Soil humidity is high :92.3	2.03	0.73	5.48	0.33
3	2022-11-11 23:07:03.308708	294.03	1014	100	98.6	Atmosphereic humidity is high :100 || Soil humidity is high :98.6	1.22	9.17	0.87	7.39
4	2022-11-11 23:07:09.951517	294.03	1014	100	90.0	Atmosphereic humidity is high :100 || Soil humidity is high :90.0	3.74	3.09	1.93	7.85
5	2022-11-11 23:07:16.237768	294.03	1014	100	86.0	Atmosphereic humidity is high :100 || Soil humidity is high :86.0	6.81	3.25	4.84	8.62
6	2022-11-11 23:08:34.198771	294.03	1014	100	2.0	Atmosphereic humidity is high :100 || Soil humidity is low :2.0 ||	4.01	3.62	3.02	1.54
7	2022-11-11 23:08:47.830420	294.03	1014	100	92.3	Atmosphereic humidity is high :100 || Soil humidity is high :92.3	2.03	0.73	5.48	0.33
8	2022-11-11 23:08:59.679436	294.03	1014	100	98.6	Atmosphereic humidity is high :100 || Soil humidity is high :98.6	1.22	9.17	0.87	7.39
9	2022-11-11 23:09:11.017963	294.03	1014	100	90.0	Atmosphereic humidity is high :100 || Soil humidity is high :90.0	3.74	3.09	1.93	7.85
10	2022-11-11 23:09:22.430418	294.03	1014	100	86.0	Atmosphereic humidity is high :100 || Soil humidity is high :86.0	6.81	3.25	4.84	8.62
11	2022-11-11 23:09:33.822438	294.03	1014	100	19.6	Atmosphereic humidity is high :100 || Soil humidity is low :19.6	4.01	3.62	3.02	1.54
12	2022-11-11 23:15:08.834942	294.03	1014	100	2.0	Atmosphereic humidity is high :100 || Soil humidity is low :2.0 ||	4.01	3.62	3.02	1.54
13	2022-11-11 23:16:15.524818	294.03	1014	100	2.0	Atmosphereic humidity is high :100 || Soil humidity is low :2.0 ||	4.01	3.62	3.02	1.54
14	2022-11-11 23:18:54.325830	294.03	1014	100	2.0	Atmosphereic humidity is high :100 || Soil humidity is low :2.0 ||	4.01	3.62	3.02	1.54
15	2022-11-13 23:53:54.499579	297.03	1015	88	60.9	Temperature is high :297.03 || Atmosphereic humidity is high :8	7.25	1.61	6.56	6.59
16	2022-11-13 23:55:09.413516	297.03	1015	88	60.9	Temperature is high :297.03 || Atmosphereic humidity is high :8	7.25	1.61	6.56	6.59
17	2022-11-15 13:38:11.357139	300.13	1012	65	47.7	Temperature is high :300.13 || Animal intrusion at location 2 wit	0.68	6.26	9.93	9.26

Download Database

Fig 8.4 Webpage

CHAPTER 9

9.1 PERFORMANCE METRICES

Fig 9.1 and 9.2 shows the traffic and error rate from the google sheet API. In addition, the median latency is also displayed.

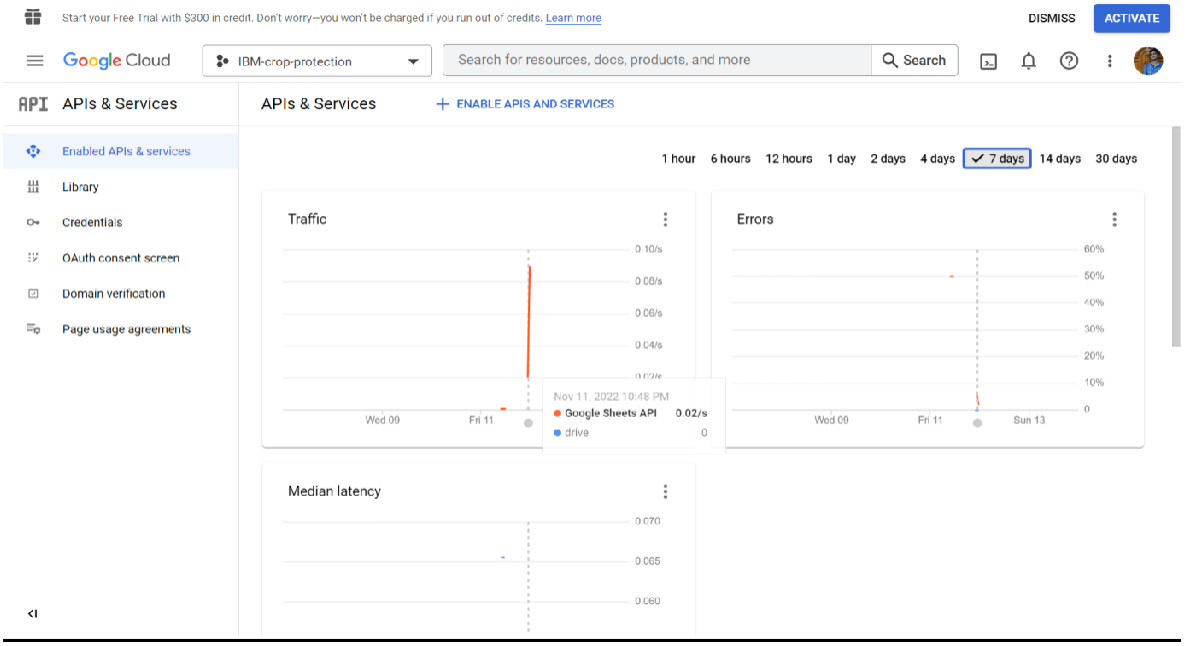


Fig 9.1 Perfomance metrics in Google Sheets API

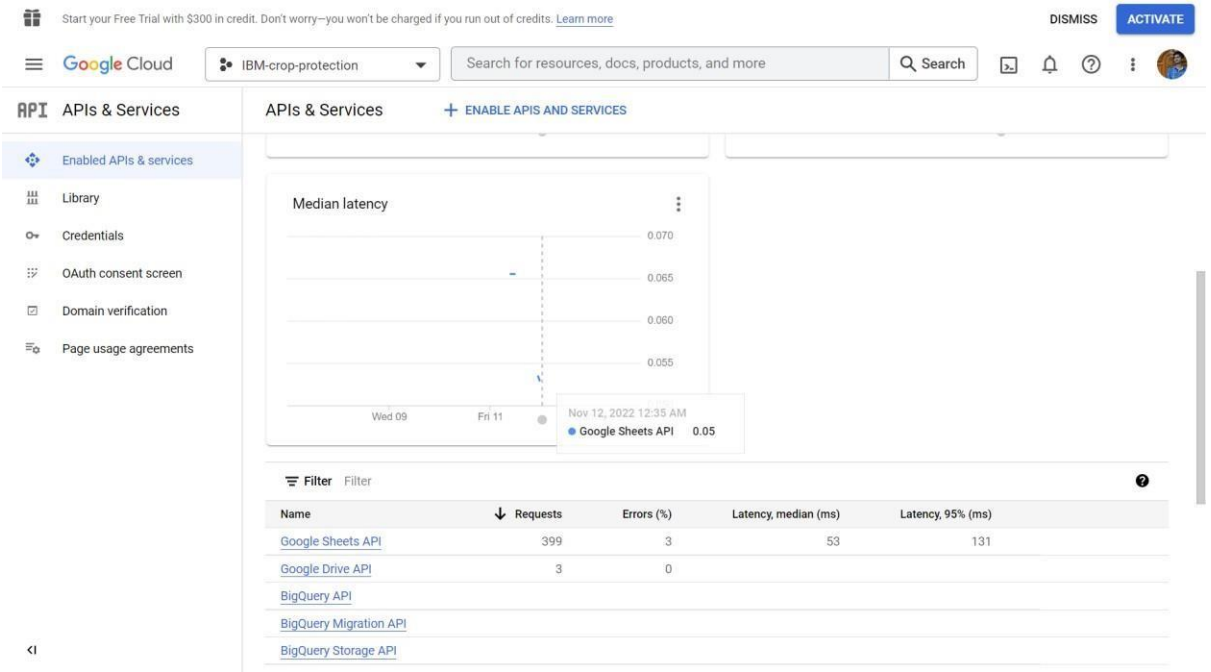


Fig 9.2 Latency in Google sheets API

RESPONSE TIME

Typical time taken for the message to reach the user after the detection of intrusion is 30 to 40 seconds .So the response time is average around 35 seconds.

CHAPTER 10

ADVANTAGES AND DISADVANTAGES

ADVANTAGES

This proposed work successfully designs a working prototype of an integrated IoT hardware with an platforms like Twilio that fulfils the following functions of timer-based irrigation control, real-time monitoring of farm data and prediction of the weather condition of the crops.

It also integrated with the function of warning the activity of animals and birds in the fields to the users. The real-time data from the farm such as soil moisture are collected from the environment using the sensors that is interfaced respectively.

Another novel feature added in this model is weather API that notifies the weather condition when it goes beyond the expected weather condition. It also comes up with the usage of Twilio ,a platform which is provides programmable communication tools for making and receiving phone calls ,sending and receiving messages and performing other communication functions using its web service APIs.

Here a message is sent to the user when animal is found in the crop field. This proposed prototype system uses the Open Weather map API which provides the real time weather condition for any geographical location.

This system uses the Google Sheets which acts as a Database system that collects the sensor data and store it for future reference to the users.

DISADVANTAGES

The use of technology in farming and agriculture making it smart agriculture, is of course, a good initiative and a much-needed one with the present increasing demand in the food supply.

But there is the chance where this proposed smart farming protection system will require certain skill sets in particular in order to understand and operate the equipment. In the case of equipment computer-based intelligence for running the devices, it is highly unlikely that a normal farmer will be able to possess this knowledge or even develop them.

Farmers are not used to these high-end technologies. They do not understand computer language or the artificial intelligence. For the smart agriculture, Internet of Things is essential which will require artificial intelligence and computer-based intelligence.

This cannot be balanced here. To overcome this challenge, the devices will have to be changed in a dramatic fashion so as to make it understandable for farmers.

This also means that the devices should be somewhere in between where the technology experts and farmers can both communicate about it.

And also this system needs availability of internet continuously. Rural part of most of the developing countries do not fulfil this requirement. Moreover internet connection is slower.

CHAPTER 11

UAT

Acceptance Testing

UAT Execution & Report Submission

Date	14 November 2022
Team ID	PNT2022TMID52774
Project Name	Project – IOT based smart crop protection using
Maximum Marks	4 Marks

1. Purpose of Document

Fig 11.1 UAT execution

The main Purpose of UAT is to validate end-to-end business flow. It does not focus on cosmetic errors, spelling mistakes or system testing. User Acceptance Testing is carried out in a separate testing environment with production-like data setup. It is kind of black box testing where two or more end-users will be involved.

UAT is performed by :

1. Client
2. End users

2. Defect Analysis

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

RESOLUTION	SEVERITY1	SEVERITY2	SEVERITY3	SEVERITY4	SUBTOTAL
By Design	3	2	1	0	9
Duplicate	2	1	2	0	5
External	4	1	1	1	7
Fixed	3	4	6	8	21
Not Reproduced	1	1	0	2	4
Skipped	1	0	1	1	3
Won't fix	1	2	0	0	3
Totals	15	11	11	12	49

3. Test Case Analysis

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

SECTION	TOTALCASES	NOTTESTED	FAIL	PASS
Web Page	10	0	0	10
Database	40	0	0	40
Twilio Dashboard	15	0	0	15
Sensor output value generation	10	0	0	10

CHAPTER 11
CONCLUSION

Smart farming is a modern farming management concept with IoT technology to increase the productivity in agriculture.

With the use of smart farming, users can effectively monitor the crop field the quality and quantity of their crops.

Users cannot be physically present on the field 24 hours a day. In addition, the farmers may not have the knowledge to use different tools to measure the ideal environmental conditions for their crops.

IoT provides them with the automated system, which can function without any human supervision and can notify them to make proper decision to deal with different kind of problems they may face during farming.

It has the capability to reach and notify the farmer even if farmer is not on the field, which can allow farmer to manage more farmland, thus improving their production.

Thus, we can conclude that this IoT based smart crop protection system will definitely help users in farmland to effectively monitor their crops with the user-friendly platforms and other alert means.

CHAPTER 12
FUTURE SCOPE

The proposed work system is a successful working prototype that fulfils to protect crops from the intrusion of animals and birds.

This system will helps the users to monitor the temperature and to notify the weather conditions.

This system assuredly assists the users to know about the soil moisture level. And the IoT based smart crop protection system implemented here brings a naval approach crop protection system from animals.

This assures the early detection and prevention of incurring losses due to the damage of crops.

The following suggestions may be carried out in future implementation of the system; the smart crop prediction may be also carried out by considering the various factors like NPK content of the soil, UV radiation along with the tracking of the crop field location using GPS module system. The automated pest traps also be introduced using image recognition techniques and neural networks in smart protection system

SOURCE CODE

PYTHON SCRIPT

```
import pandas as pd

import random

import requests, json

import time

from twilio.rest import Client

import gspread

from datetime import datetime



sa= gspread.service_account(filename=r"C:\Users\K B PAVITHRAN\ibm-crop-protection-
9837998227d9.json")

sh = sa.open("IBM_db")

whs =sh.worksheet("Animal Instrusion notification history")



def
update_sheet(loc_list,message,time_stamp,soil_humidity,current_humidity,current_pressure,
current_temperature):

#     update_sheet(loc_list,temporary_text , str(time_stamp),str(soil_humidity[i])
,str(current_humidity ),str(current_pressure),str(current_temperature))

    index = 'A'

    time_stamp_index = 'B'

    current_temperature_index = 'C'

    current_pressure_index = 'D'

    current_humidity_index= 'E'

    soil_humidity_index = 'F'

    message_index = 'G'
```

```
location_1_index='H'
location_2_index='I'
location_3_index='J'
location_4_index='K'
TEMP_LEN = len(whs.get_all_values())
if TEMP_LEN ==0:
    TEMP_LEN =1
print(whs.get_all_values())
print(whs.row_count)
print("total_length",TEMP_LEN)

index += str(TEMP_LEN +1)
current_temperature_index +=str((TEMP_LEN) +1)
current_pressure_index +=str((TEMP_LEN) +1)



current_humidity_index +=str((TEMP_LEN) +1)
soil_humidity_index +=str((TEMP_LEN) +1)
message_index +=str((TEMP_LEN) +1)
time_stamp_index +=str((TEMP_LEN) +1)

location_1_index +=str((TEMP_LEN) +1)
location_2_index +=str((TEMP_LEN) +1)
location_3_index +=str((TEMP_LEN) +1)
location_4_index +=str((TEMP_LEN) +1)

whs.update(index,TEMP_LEN)
```

```
    whs.update(message_index,message )
    whs.update(time_stamp_index,time_stamp)
    whs.update(current_temperature_index,current_temperature)
    whs.update(current_pressure_index,current_pressure)
    whs.update(soil_humidity_index,soil_humidity)
    whs.update(current_humidity_index,current_humidity)
    whs.update(location_1_index,loc_list[0])
    whs.update(location_2_index,loc_list[1])
    whs.update(location_3_index,loc_list[2])
    whs.update(location_4_index,loc_list[3])
def get_values():
    data=whs.get_all_values()
    for i in data:
        print(i )



df = pd.DataFrame()
username = "Pavithran 1904032 "



rand_prox_location_1=[]
constrain_location_1=9.7
rand_prox_location_2=[]
constrain_location_2=7
rand_prox_location_3=[]
constrain_location_3=9.5
```

```
rand_prox_location_4=[]
constrain_location_4=9.9
soil_humidity=[]
constrain_soil_humidity= 75
constrain_temp= 295
constrain_humidity= 80
constrain_pressure= 1030
constrain_soil_humidity_low =40
for i in range(1,100):
    rand_prox_location_1.append(round(random.uniform(0.25,10),2))
    rand_prox_location_2.append(round(random.uniform(0.25,10),2))
    rand_prox_location_3.append(round(random.uniform(0.25,10),2))
    rand_prox_location_4.append(round(random.uniform(0.25,10),2))
    soil_humidity.append(round(random.uniform(1,100),1))
print("          ",rand_prox_location_4)
print("          ",rand_prox_location_3)
print("          ",rand_prox_location_2)
print("          ",rand_prox_location_1)
print("         ",soil_humidity)


df['Location_1']=rand_prox_location_1
df['Location_2']=rand_prox_location_2
df['Location_3']=rand_prox_location_3
df['Location_4']=rand_prox_location_4
df['soil_humidity']=soil_humidity
df.to_excel('IBM_sensor_data.xlsx', index = False)
```

```
def send_sms(username,text):
    SID= 'AC34c7f1445fd65764ad5af8cfb249afc5'
    auth_token='9fc5715505bd02ac73be8429b116d7a8'


    cl = Client(SID, auth_token)
    cl.messages.create(body="Hey,"+username+text,from_='+13609975006',to='+9198945670
77' )

def get_whether():
    api_key = "b48129c70234f815d857006719283786"
    base_url = "http://api.openweathermap.org/data/2.5/weather?"
    city_name = "Coimbatore"
    complete_url = base_url + "appid=" + api_key + "&q=" + city_name
    response = requests.get(complete_url)
    x = response.json()
#     print(x)
    if x["cod"] != "404":
        y = x["main"]
        current_temperature = y["temp"]
        current_pressure = y["pressure"]
        current_humidity = y["humidity"]
        z = x["weather"]
        weather_description = z[0]["description"]
#         print(" Temperature (in kelvin unit) = " + str(current_temperature))
#         print(" Atmospheric pressure (in hPa unit) = " +str(current_pressure))
#         print(" Humidity (in percentage) = " + str(current_humidity))
#         print(" Description = " + str(weather_description))
        return(current_temperature,current_pressure,current_humidity,weather_description)
```

```
    else:
        print(" City Not Found ")
        return(None)
 for i in range(1):
    time_stamp = datetime.now()
    time.sleep(5)
    temp=[0,0,0,0]
    value=[0,0,0,0]
    animal_intrusion_set=0
    temp_set=0
    soil_humidity_set=0
    pressure_set=0
    atmos_humidity_set=0
    animal_intrusion_text = ' Animal intrusion at'
    temporary_text= ''
    try:
        current_temperature,current_pressure,current_humidity,weather_description =
get_whether()
    except:
        print("enter appropriate location")


    if current_temperature > constrain_temp:
        temp_set=1
        temporary_text += ' Temperature is high :'+str(current_temperature) + ' || '
    if current_pressure > constrain_pressure:
        pressure_set=1
        temporary_text += ' Pressure is high :'+str(current_pressure) + ' || '
```

```
if current_humidity > constrain_humidity:
    atmos_humidity_set=1
    temporary_text += ' Atmosphereic humidity is high :'+str(current_humidity ) + ' || '
if soil_humidity[i] > constrain_soil_humidity:
    soil_humidity_set=1
    temporary_text += ' Soil humidity is high :'+str(soil_humidity[i]) + ' || '
if soil_humidity[i] < constrain_soil_humidity_low:
    soil_humidity_set=1
    temporary_text += ' Soil humidity is low :'+str(soil_humidity[i]) + ' || '
    # send_sms("activate the pump",username)
    print("The soil humidity is low turning on the pump")
if rand_prox_location_1[i] > constrain_location_1:
    temp[0]=1
    value[0]=rand_prox_location_1[i]
if rand_prox_location_2[i] > constrain_location_2:
    temp[1]=1
    value[1]=rand_prox_location_2[i]
if rand_prox_location_3[i] > constrain_location_3:
    temp[2]=1
    value[2]=rand_prox_location_3[i]
if rand_prox_location_4[i] > constrain_location_4:
    temp[3]=1
    value[3]=rand_prox_location_4[i]
for j in range(len(temp)):
    if temp[j]==1:
        animal_intrusion_set=1
```

```
        animal_intrusion_text += ' location '+str(j)+ ' with a distance from sensor of ' +
str(value[j]) + ' || '
    trigger=0
    loc_list =
[rand_prox_location_1[i],rand_prox_location_2[i],rand_prox_location_3[i],rand_prox_locati
on_4[i]]


    if animal_intrusion_set==1:
        temporary_text+=animal_intrusion_text
        trigger = 1
    else:
        if temporary_text != '':
            trigger = 1
    update_sheet(loc_list,temporary_text , str(time_stamp),str(soil_humidity[i])
,str(current_humidity ),str(current_pressure),str(current_temperature))
    if trigger ==1:
        print(temporary_text,username)
        print("------------------------------------------------------------------------")
        print("updating db")
        print("------------------------------------------------------------------------")
        print("sending sms")
        print("------------------------------------------------------------------------")
        time.sleep(30)
        try:
            # send_sms(temporary_text,username)
            print("sms sent")
            print("------------------------------------------------------------------------")
        except:
```

```
        print("failed to send sms")
    get_values()
```

Arduino Script:

```
Hi #include "SPI.h"
#include  "Adafruit_GFX.h"
#include "Adafruit_ILI9341.h"

#define TFT_DC 2
#define TFT_CS 15
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);

int inputPin2 = 12;
int inputPin = 14;
int pirState = LOW;
int val = 0;
int val2=0;
int buzzerPin = 13;


void setup()  {
  pinMode(inputPin2, INPUT);
  pinMode(inputPin, INPUT);
  pinMode(buzzerPin , OUTPUT);
  Serial.begin(9600);
  tft.begin();
```

```
}

void loop() {
  val = digitalRead(inputPin);
  val2= digitalRead(inputPin2);
  if (val == HIGH) {
  tone(buzzerPin, 100);
  delay(1000);
  noTone(buzzerPin);
  delay(1000);
    tone(buzzerPin, 100, 1000);
    delay(2000);
    if (pirState == LOW) {
      Serial.println("Motion is detected animal intrusion");
      pirState = HIGH;
      tft.setCursor(20, 120);
      tft.setTextColor(ILI9341_RED);
      tft.setTextSize(3);
      tft.println("Motion is detected animal intrusion");
    }
  }
  else if (val2 == HIGH) {
    tone(buzzerPin, 100);
    delay(1000); +
    noTone(buzzerPin);
    delay(1000);
    tone(buzzerPin, 100, 1000);
```

```
    delay(2000);
    if (pirState == LOW) {
      Serial.println("Motion is detected animal intrusion at location 2");
      pirState = HIGH;
      tft.setCursor(20, 120);
      tft.setTextColor(ILI9341_RED);
      tft.setTextSize(3);
      tft.println("Motion is detected animal intrusion a location 2");
    }
  } else {
    if (pirState == HIGH) {

      Serial.println("Motion ended");

      pirState = LOW;
    }
  }
}
```

HTML SCRIPT:

```
<!DOCTYPE html>


<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Crop protection</title>
    <link rel="stylesheet" href="./style.css">
    <link rel="icon" href="./favicon.ico" type="image/x-icon">
  </head>
  <body style="background-color:powderblue;">


    <main>
      <centre><h1>IOT based crop protection</h1></centre>
      <style>
        h1 {
         text-align:center
        }
        </style>
        <!-- <h2>Temperature: 294.03  </h2>
        <h2>Pressure: 1014</h2>
        <h2>Humidity: 100</h2>
        <h2>Soil humidity: 2</h2>-->
        <h3> The alert details are mentioned below in realtime. To download this file or visit
the database click the button below</h2>
```

```
<!-- <img src="farmer.jpg" alt="Italian Trulli"float-left> -->

<iframe src="https://docs.google.com/spreadsheets/d/e/2PACX-1vRPXOExKJk73wJt7bv6B-VinbcOPrGtQWFtBUPH20dy2aNaOxAl3d0H1zvjtb55jlf9x9FIieIb6nhH/pubhtml?gid=0&amp;single=true&amp;widget=true&amp;headers=false"width="1280" height="400"></iframe>

<button type="submit" onclick="window.open('IBM_db - Animal Instrusion notification history.csv')">Download</button>

<style>

  h2 {

   text-align:center

  }

  </style>

  <style>

    h3 {

     text-align:center

    }

    </style>

<style> img {

    align-content: center;

  }

  </style>

  <button onclick="location.href = 'https:docs.google.com/spreadsheets/d/1HD5EUEf4H9FMVhxOSCz877qj9FrvSfmI4h3ca2zhUqk/edit#gid=0';" id="myButton" class="float-left submit-button" Margin: 0 auto>Database</button>

    </main>

  <script src="index.js"></script>

  </body>

</html>
```

GITHUB LINK: https://github.com/IBM-EPBL/IBM-Project-9706-1659068981

DEMO LINK:

https://drive.google.com/drive/folders/1xBCzg_wXeLSwOhbdwMKKBxUu7wzH3fhq?usp=sharing

REFERENCES

1. Nanda, Ipseeta&Chadalavada, Sahithi&Swathi, Medepalli&Khatua, Lizina. (2021). Implementation of IIoT based smart crop protection and irrigation system. Journal ofPhysics: Conference Series. 1804. 012206.10.1088/1742-6596/1804/1/012206.

2. P. Bhadani and V. Vashisht, "Soil Moisture, Temperature and Humidity MeasurementUsing Arduino," 2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence), 2019, pp. 567-571, doi:10.1109/CONFLUENCE.2019.8776973.

3. S. Yadahalli, A. Parmar and A. Deshpande, "Smart Intrusion Detection System for Crop ProtectionbyusingArduino,"2020SecondInternationalConferenceonInventiveResearch in Computing Applications (ICIRCA), 2020, pp. 405-408, doi: 10.1109/ICIRCA48905.2020.9182868**.**