IOT-BASED SOIL HEALTH MONITORING AND CROP RECOMMENDATION SYSTEM

A

MINI PROJECT REPORT

submitted by

KARTHIK S	**TCR19EC035**
RAHUL R NAIR	**TCR19EC044**
SIVAPRASAD A S	**TCR19EC051**

to
APJ Abdul Kalam Technological University
in partial fulfillment of the requirements for the award of the Minor Degree
of
Bachelor of Technology
in
Computer Science and Engineering (Networking)

Department of Computer Science and Engineering
Government Engineering College Thrissur
January 2023

DECLARATION

We undersigned hereby declare that the project report **IoT-BASED SOIL HEALTH MONITORING AND CROP RECOMMENDATION SYSTEM** submitted for partial fulfillment of the requirements for the award of minor degree of bachelor of Technology of the APJ Abdul Kalam Technological University, Kerala, is a bonafide work done by **Karthik S (TCR19EC035)**, **Rahul R Nair (TCR19EC044)**, **Sivaprasad A S (TCR19EC051)** under supervision of **Prof. Umasree A K**. This submission represents our ideas in our own words and where ideas or words of others have been included, we have adequately and accurately cited and referenced the original sources. We also declare that we have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

KARTHIK S

Place: **Thrissur**

Date: **January 07,2023**

RAHUL R NAIR

SIVAPRASAD A S

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Government Engineering College Thrissur

CERTIFICATE

This is to certify that the report entitled **IoT-BASED SOIL HEALTH MONITORING AND CROP RECOMMENDATION SYSTEM** is submitted by **KARTHIK S (TCR19EC035) , RAHUL R NAIR(TCR19EC044) , SIVAPRASAD A S (TCR19EC051)** to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the **Minor Degree of Bachelor of Technology in Computer Science and Engineering (Networking)** is a bonafide record of the project work carried out by him/her under my/our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Internal Supervisor	Project Coordinator	Head of Department

ACKNOWLEDGMENT

It gives us great pleasure to present our project report on “**IoT-BASED SOIL HEALTH MONITORING AND CROP RECOMMENDATION SYSTEM** ”. No work, however big or small, has ever been done without the contribution of others. So these words of acknowledgement come as a small gesture of gratitude towards all those people, without whom the successful completion of this report would not have been possible. We are extremely grateful to **Dr. Ranjini Bhattathiripad T**, Principal, Government Engineering College Thrissur and **Dr. Shibily Joseph,**, Head of the Department, Computer Science Engineering, for providing all the required resources for the successful completion of my project phase 1. We would like to express our gratitude towards project coordinators **Prof. Umasree A K , Prof. Manu G Thomas** , Department of Computer Science And Engineering and our guide **Prof. Umasree A K** who gave us their valuable suggestions, reviews, motivation and direction. Last but not the least we would like to thank all our friends, who supported us with their valuable criticism, advice and support.

KARTHIK S

RAHUL R NAIR

SIVAPRASAD A S

ABSTRACT

Soil is the base of agriculture. Soil provides nutrients that increase the growth of a crop. Some chemical and physical properties of soil, such as its moisture, temperature and its pH, heavily affect the yield of a crop. These properties can be sensed by the open-source hardware, and they can be used in the field. In this project, we intend to build a IOT based soil health monitoring system in which a farmer will be able to monitor soil moisture, soil temperature and soil pH in his android smart-phone. Based on the sensed value and climate data fetched from the database, we also provide a recommendation about the crop which is most suitable to cultivate according to current conditions with the help of a mobile app. This will make the farmers and youngsters who are heading to agriculture to choose the best crop for cultivation ,thereby obtaining maximum yield.

CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 CURRENT SYSTEM

The conventional method of selecting crops for cultivation is based on analyzing the soil and climatic conditions manually. Basically, the most suited crops were selected based on the experience of the elders. From that experience gained through interacting with agriculture fields they selected the most suited for the current condition.

1.2 PROPOSED SYSTEM

In the proposed system, we intend to build a IoT based soil health monitoring system in which a farmer will be able to monitor soil moisture, soil temperature and soil pH in his android smart-phone. Based on the sensed value and climate data fetched from the database, we also provide a recommendation about the crop which is most suitable to cultivate according to current conditions with the help of a mobile app. This will make the farmers and youngsters who are heading to agriculture to choose the best crop for cultivation ,thereby obtaining maximum yield.

1.3 FEASIBILITY

1.3.1 Technical Feasibility

The designed system can provide real-time data with high temporal resolution (readings every seconds when data changed) and is equipped with providing recommendation in any time. User friendly mobile Apps can be created using existing technologies to complement the hardware part and facilitate recommendation systems. These are all technically feasible and scalable for future use cases.

1.3.2 Economical & Financial Feasibility

A cloud server powered by Google Firebase and hardware from the college were used in the prototype development process to save money. The application will be set up in the free MIT App inventor for production.

1.4 PROCESS MODEL

1.4.1 Possible Process Models

The following were the models that were considered our project.

- ML based model
- Data based model
- Sensor based network model

1.4.2 Selected Model

We used to select sensor based network model since we are planning to build low cost and less complicated IoT device. Less complicated IoT devices can provide better comfort for users.

Chapter 2

REQUIREMENT ANALYSIS

The process of refining, modelling, and specifying the already identified user needs is known as requirements analysis and validation. This stage of development includes the systematic use of tried-and-true ideas, techniques, languages, and tools for the efficient analysis, documenting, and ongoing evolution of user demands as well as the definition of a system's external behaviour to meet those needs. This chapter explains the technique used for requirement elicitation and the user requirements that were as a result acquired. Following validation using an appropriate approach, the criteria are also completed.

2.1 METHOD OF REQUIREMENT ELICITATION

To elicit the requirements, we adopted certain techniques which helped us to identify the expectations from users, requirements at the system level, and expected behavior of various subsystems. The techniques we employed were :

2.1.1 Group discussion

Examined several solutions to the issue description. Discussed the advantages and disadvantages of existing options. Each group member addressed the topic from a different angle and gathered pertinent data from potential product end users by conducting interviews, brief discussions, etc. Based on it, the group members engage in discussion.

2.1.2 Brainstorming

Through brainstorming meetings, the potential solutions to the requirements were chosen. At this point, the best potential answer is chosen. One problem shouldn't lead to another once it has been solved. Therefore, the development of a basic sensor-based IoT system is proposed.

2.1.3 Literature Survey

In order to reinforce the prototype design and improve execution, we looked at projects that were similar to the proposed application. We looked at a number of scholarly articles that explained how different methods may be used to create crop recommendations.

In the year 2018, [1] a prototype model of distributed monitoring system of different properties of aquaculture water quality, such as pH, temperature and availability of oxygen. The proposed prototype model is useful for improving environmental control, reduction of production cost, etc. Authors used Arduino sensors, Zigbee technology, Arduino module, web services, mobile application and database to design the system. [2]Using the Arduino module, sensors sense pH, temperature and availability of oxygen and sensed value which are transmitted to external cloud with the help of database and different web services. Smart-phones are connected through the cloud by which these sensed data can be visualized. As the future work, alarm conditions are to be applied in the system with help of artificial intelligence.

In the year 2009, a prototype for calibrating textile-based wearable sensor for sports performance was developed. These sensors are comfortable in wearing in that manner a person can do exercise and can play a sport. To analyse the rehydration, the fabric-based pH sensor is used. This pH sensor used sweat of the body to sense the rehydration. The above-mentioned pH sensor is calibrated with the help of artificial sweat of range 4–8 pH. Authors calibrated three times and illustrated that this pH sensor is sensitive to changes in pH of less 0.2 units and that the response is repeatable.LilyPad Arduino is used to control the sensors attached to the body.

Yamaguchi et al. [8] presented a sensor network-based agriculture project called as e-kakashi project using different sensors such as camera, Zigbee module, temperature and humidity sensor. It was basically designed for improvement of productivity in farming system and reduction of vermin disease. This application was designed for iOS operating system. Lee et al. [9] presented IoT-designed agricultural production system for stabilizing the demand and supply of agricultural related products while developing the required sensors and prediction system for the growth and production amount of crops. The authors used HSDPA/Dual CDMA, TCP/UDP/IP/ICMP Protocol Support, 2.4 GHz IEEE 802.15.4 compatible, temperature sensor, humidity sensor, soil EC senor and pH sensor.

2.2 USER REQUIREMENTS

Some of the requirements that is expected by the users are:-

- Easy to carry
- Easy to understand results
- Easy to interface

2.3 PROJECT REQUIREMENTS

So by considering user requirements, the project is trying to achieve requirements such as - ease of monitoring your local weather conditions in real-time from anywhere in the world, Provide better crop recommendations according to the condition, Improve productivity, Integrating IoT as a part of agriculture.

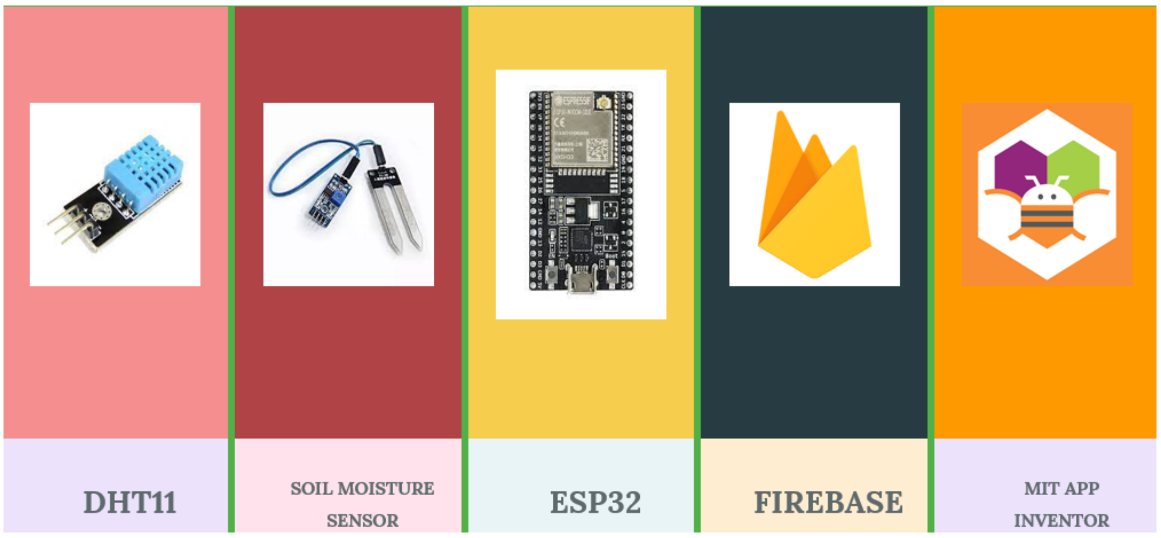


Figure 2.1: Project Requirements

Chapter 3

DESIGN AND IMPLEMENTATION

3.1 ARCHITECTURAL DESCRIPTION

An architectural description is a document or set of documents that describe the architecture of a software system. It provides a high-level overview of the system's structure, components, and interactions, and helps to ensure that the system meets its functional and non-functional requirements.An architectural description typically includes an overview of the system and its purpose, and outlines the scope of the architectural description, section describes the overall vision for the system's architecture, including its goals and objectives, and how it will meet the system's requirements, section describes the design patterns and architectural styles that are used in the system, and how they are implemented etc.By providing a clear and comprehensive overview of the system's architecture, an architectural description helps to ensure that the system meets its requirements and can be easily understood and maintained over time.

Here the system consists of a capacitive soil moisture sensor and a DHT-11 humidity sensor coupled to an ESP32 microcontroller. The built-in WiFi module in this micro-controller is used to communicate the data gathered from these sensors to a real-time database. Based on database server processing, recommendations for crops are obtained via a mobile application.

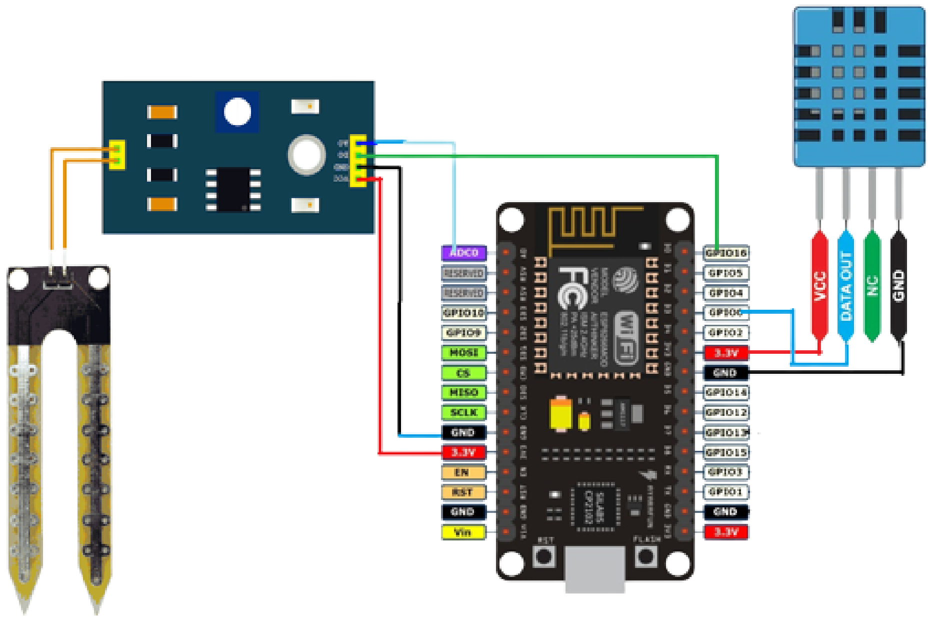


Figure 3.1: ESP32 Architecture

3.1.1 Decomposition Description

The System is divided into 4 module based on the functionality of the system, that is it is divided into module for each of the functionality the system would provide. The Parts are :

- **Humidity and temperature measurement module** - A basic, extremely affordable digital temperature and humidity sensor is the DHT11. It measures the humidity in the air using a thermistor and a capacitive humidity sensor, and it outputs a digital signal on the data pin (no analogue input pins needed). Although reasonably easy to operate, data collection needs precise timing.
- **Soil Moisture measurement module** - A tool used to gauge the moisture content of soil is a soil moisture measuring module. It generally includes of sensors that detect the electrical conductivity of the soil, which is directly connected to its moisture content, and a probe that is placed into the soil. The controller receives the sensor readings and displays the soil moisture level in many ways, including as a percentage, volumetric water content, or relative water content.Modules for measuring soil moisture might be analogue or digital. Digital modules employ capacitive sensors to detect the capacitance of the soil, whereas analogue modules often use a resistive sensor that changes resistance when the soil moisture level varies.Both types of modules have their advantages and disadvantages, and the choice between them depends on the specific application and requirements.
- **Real time database** - Google's Firebase platform offers a cloud-based NoSQL database called Firebase Real-Time Database. As a result, when data is updated on one client, changes are instantaneously reflected on all other clients. It enables developers to store and sync data across numerous clients in real-time.Data is saved as documents and collections rather than tables and rows

since the Firebase Real-Time Database is a document-oriented database. Developers may access the database directly from their client-side code since the data is set up in a tree structure like JSON. One of the main advantages of the Firebase Real-Time Database is that it expands automatically to manage high volumes of traffic and handles data synchronisation across clients automatically. As well as providing real-time updates for clients even when the device is offline, it also has an integrated security mechanism that enables developers to manage access to the data.

- **Mobile Application** - A software programme known as an IoT (Internet of Things) mobile application runs on a mobile device and communicates with IoT devices. An IoT mobile application's primary function is to offer a user-friendly interface for controlling, monitoring, and displaying the data gathered by IoT devices. User experience, security, and scalability are critical considerations when creating an IoT mobile application. The software should be simple to use, secure, and able to manage a lot of data and traffic. It should also be user-friendly. It should also be created to be scalable, allowing for simple expansion and adaptation when new IoT ecosystem components are introduced.

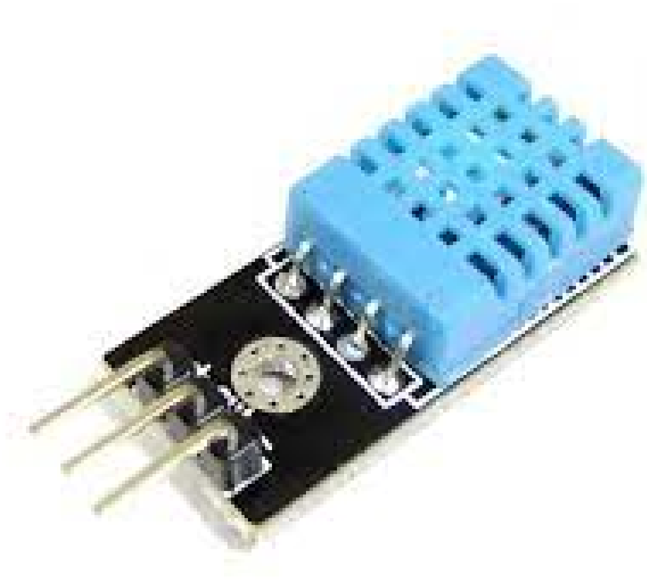


Figure 3.2: Temperature and humidity sensor module

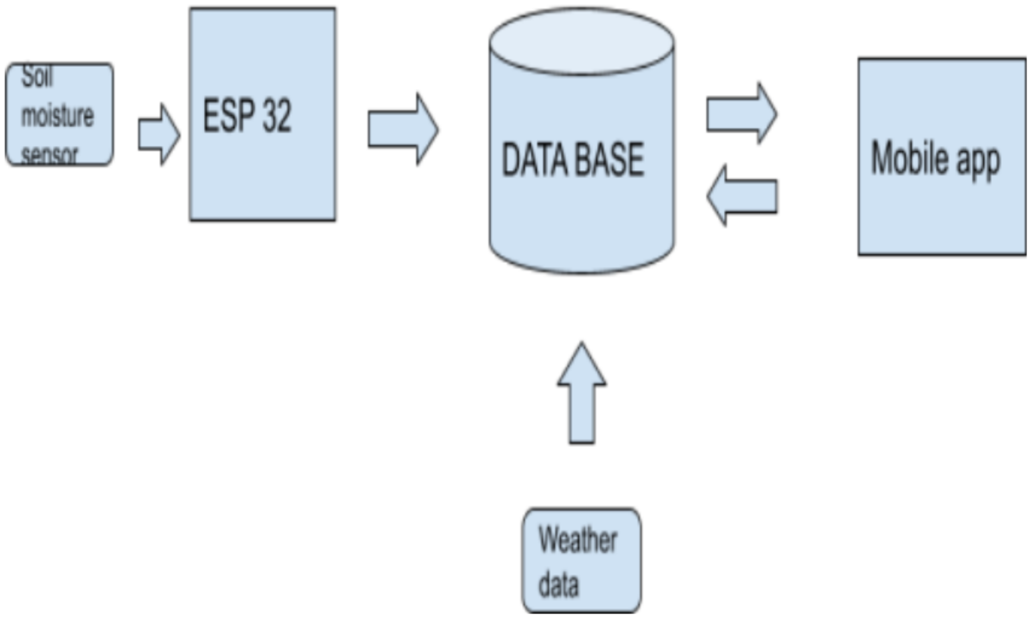


Figure 3.3: System Block Diagram

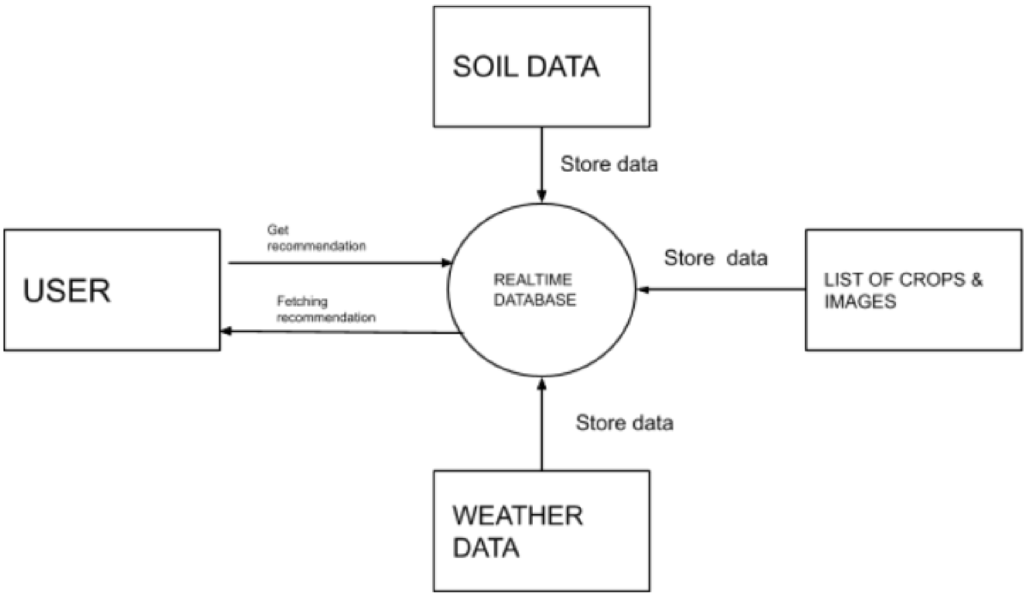


Figure 3.4: Intermodule Dependency Diagram

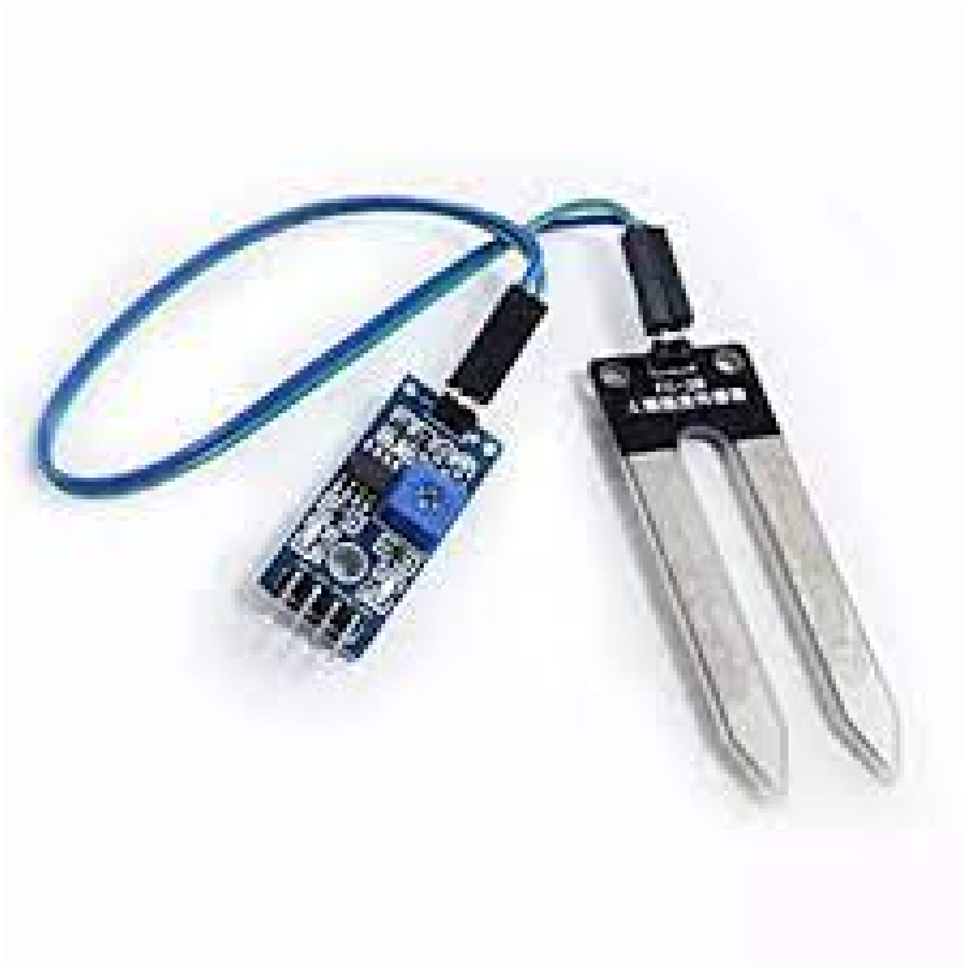


Figure 3.5: Soil moisture sesnor module

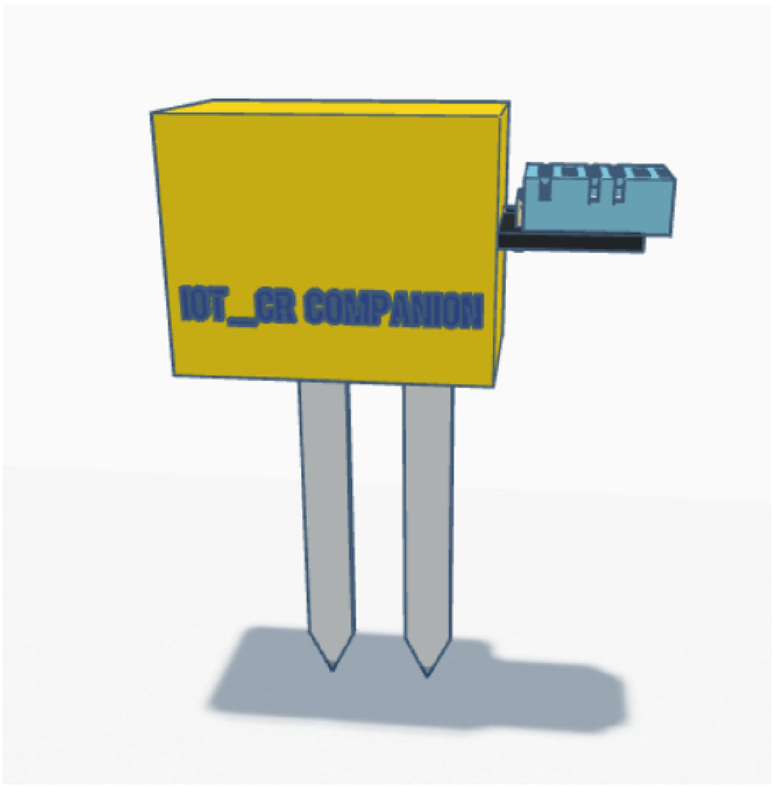


Figure 3.6: Overall IoT module

3.1.2 Dependency Description

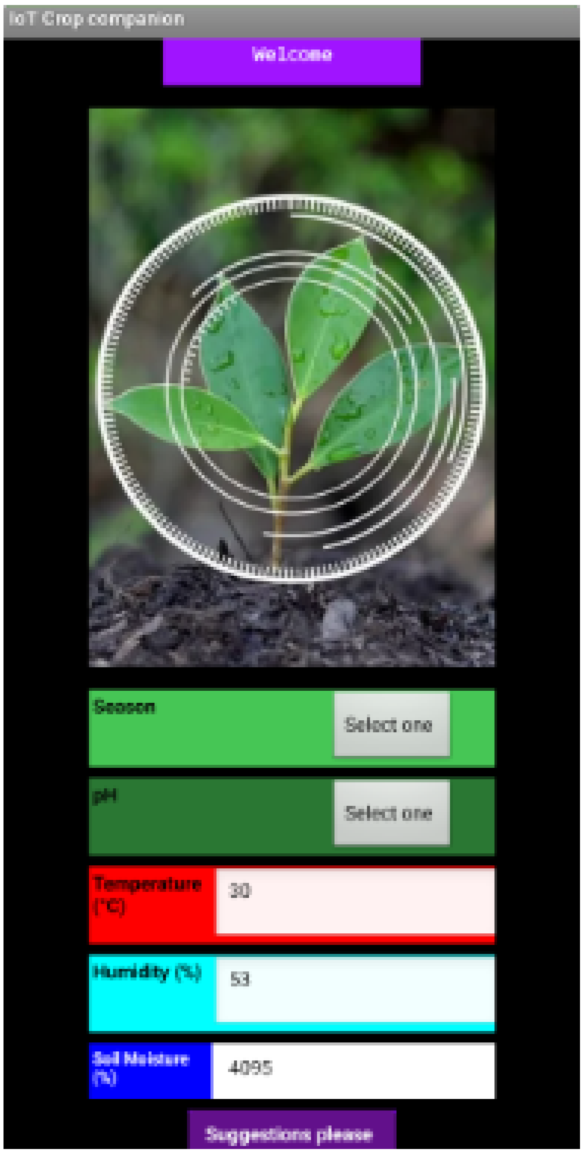
The Dependencies between the modules is expressed as the Design diagram shown below

- User connect with IoT companion using mobile hotspot
- The IoT device collect real time temperature,humidity and soil moisture data using respective sensors attached with the device
- Collected data send to real time database
- Realtime database process the collected data
- Based on the processed results it will fetch details of recommended crops
- The fetched details are displayed to end user using a mobile application

3.1.3 User Interface Design

User interface design includes mobile application screen which helps user to understand and analyze crop recommendation easily. It includes three screens:-

- **Home screen**- This is the opening screen of mobile application where users have the option to see the data collected from sensors and provide manual inputs. A ”Suggestion Please” button is placed at the bottom to initiate recommendation.
- **Recommendation Screen**- It shows list of crops that is recommended to cultivate according to the condition. Different condition produces different set of crops in the screen
- **Default Screen**-This screen is showed when none of the crop is matched to data collected from IoT device and manual inputs of users in home screen.



Season Select one

pH Select one

Temperature (°C) 30

Humidity (%) 53

Soil Moisture (%) 4095

Figure 3.7: Home screen

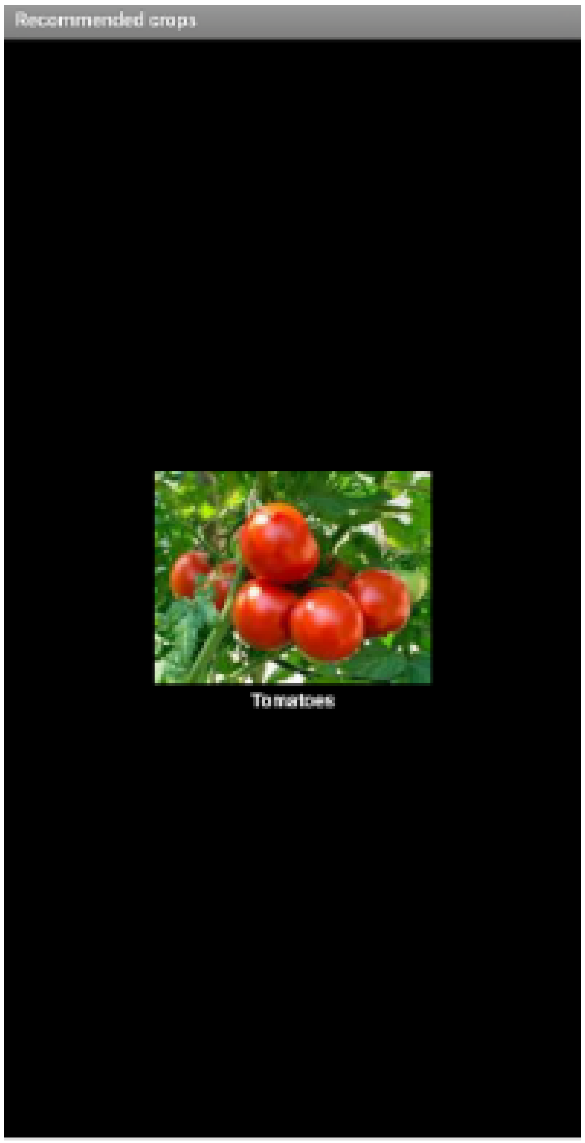


Figure 3.8: Recommendation screen

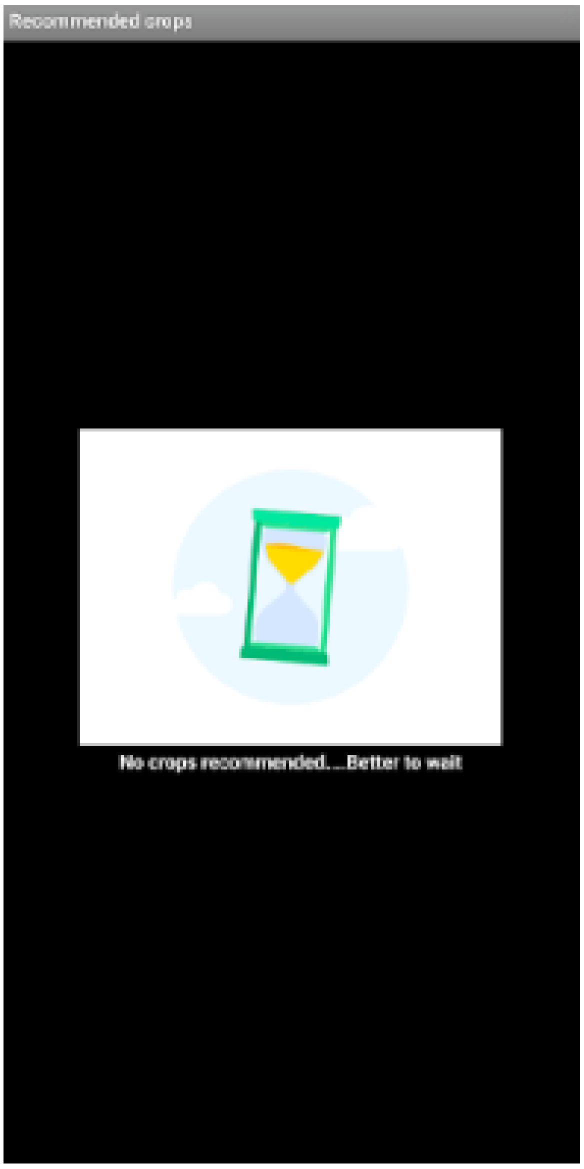


Figure 3.9: Default screen

3.2 TEST CASE DESIGN

Test case design is a systematic approach to creating and documenting test cases for a software application or system. The goal of test case design is to ensure that all relevant functionality, functionality requirements, and edge cases are tested thoroughly, thereby increasing the overall quality and reliability of the software. Here the testing involves checking various crops are recommended according to various condition.

3.2.1 Requirement Based Testing

The first requirement is to connect the hardware device with internet. It is done with the help of mobile hotspot. The second requirement is collecting real time temperature,humidity and soil moisture.It is done with the help of DHT-11 sensor and soil moisture sensor. Then we want to satisfy the requirement of processing collected data.It is done with the help of realtime database in firebase. Finally the end results are shown to user using a mobile application interface. The overall device is tested by varying conditions according to user requirements.

Chapter 4

CODING

Section 4.1 gives a general overview of the coding of the application. The next section (Section 4.2) gives an overview of the organization of the packages. The important code snippets and their working is described in section 4.3

4.1 OVERVIEW OF THE CODE

The coding consist of three parts such as coding ESP32 microcontroller, configuring real time database and coding mobile application to visualize results. The following section consists of the important code snippets implemented in the hardware and mobile application.

4.2 ORGANIZATION OF PACKAGES

The organization of packages in ESP32 typically follows the standard file structure for an embedded systems project, with the following components:

1. **Source files**: This folder contains the source code for the project, including C/C++ files, header files, and any assembly code.
2. **Include files**: This folder contains the header files that provide declarations for functions, variables, and data structures used in the project.
3. **Library files**: This folder contains pre-compiled libraries, either provided by the ESP32 SDK or external libraries that are required for the project.

4. **Build files**: This folder contains the makefiles and other configuration files used to build the project, including the project configuration, linker scripts, and build scripts.

5. **Documentation**: This folder contains the project documentation, including user manuals, API reference guides, and any other relevant information.

The specific organization of packages in ESP32 can vary based on the development tools and environment being used, but the general structure described above is a common starting point for most projects.

In MIT App Inventor, the organization of packages is handled through the use of ”Projects”. A project in MIT App Inventor is a collection of blocks, components, and other assets that make up a single app. Each project consists of the following components:

1. **Components**: This is the main building block of an App Inventor project, where you can drag and drop components such as buttons, text boxes, and images to create the user interface for your app.

2. **Blocks Editor**: This is where you write the logic for your app using blocks, similar to a flowchart. You can connect blocks to create event handlers, loops, and conditions.

3. **Assets**: This is where you can store all the assets you need for your app, such as images, sounds, and other media files.

4. **Design View**: This is where you can see a visual representation of your app, including the components and the layout.

5. **Blocks View**: This is where you can see the blocks that make up the logic for your app.

The organization of packages in MIT App Inventor is centered around projects, which consist of components, blocks, assets, design view, and blocks view.

The organization of packages in Firebase follows a hierarchical structure, which includes:

1. **Projects**: A Firebase project is the highest level of organization in Firebase and acts as a container for all the resources and services you want to use in your app. You can have multiple projects in your Firebase account, and you can use them to manage different apps or parts of the same app.

2. **Services**: Firebase provides a number of services that you can use in your app, such as authentication, cloud storage, and databases. Each service has its own configuration and settings, and you can use them independently or in combination with other services.

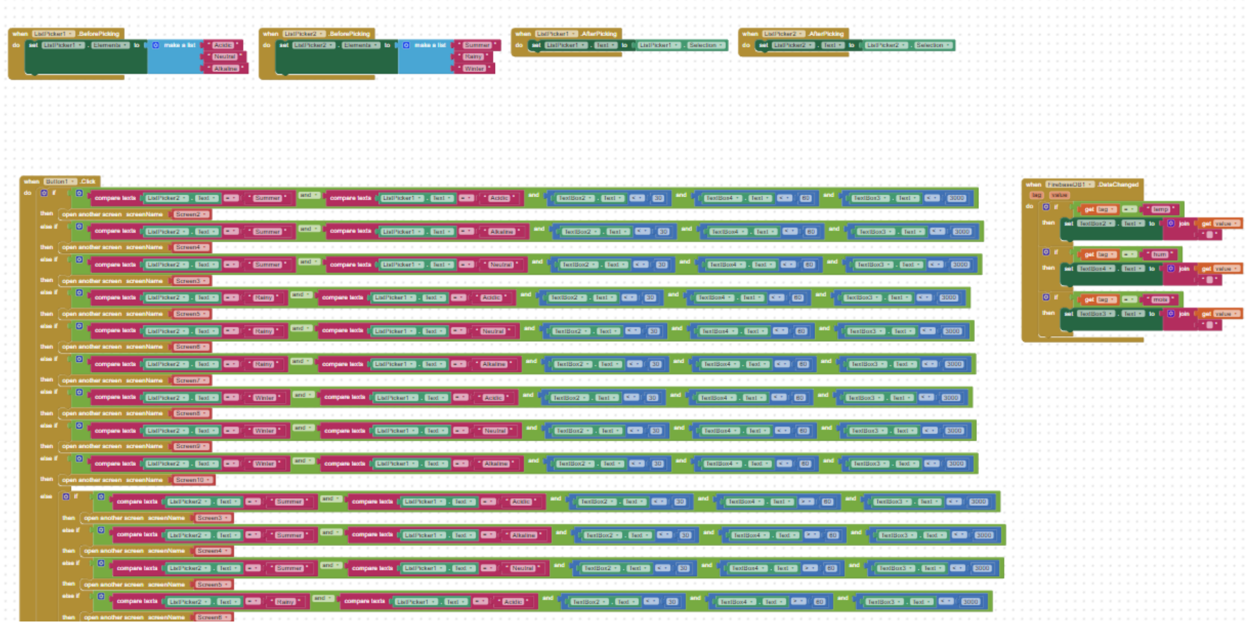
3. **Databases**: Firebase provides two types of databases: Realtime Database and Cloud Firestore. Each database has its own data structure and rules, and you can use them to store and retrieve data for your app.

4. **Functions**: Firebase Functions is a service that lets you run backend code in response to events triggered by Firebase services, such as changes to data in the Realtime Database or Firestore.

5. **Hosting**: Firebase Hosting is a service that lets you deploy and host static web assets, such as HTML, CSS, and JavaScript files, on a global content delivery network.

6. **Storage**: Firebase Cloud Storage is a service that lets you store and retrieve binary data, such as images and videos, for your app.

The organization of packages in Firebase is structured around projects, which contain databases, functions, hosting, and storage, each with its own configurations.



```
when ListPicker1 .BeforePicking
do  set ListPicker1 . Elements to  make a list  " Acidic "
                                                " Neutral "
                                                " Alkaline "

when ListPicker2 .BeforePicking
do  set ListPicker2 . Elements to  make a list  " Summer "
                                                " Rainy "
                                                " Winter "

when ListPicker1 .AfterPicking
do  set ListPicker1 . Text to  ListPicker1 . Selection

when ListPicker2 .AfterPicking
do  set ListPicker2 . Text to  ListPicker2 . Selection
```

```
when Button1 .Click
do  if       compare texts ListPicker2 . Text = " Summer " and compare texts ListPicker1 . Text = " Acidic " and TextBox2 . Text < 30 and TextBox4 . Text < 60 and TextBox3 . Text < 3000
    then     open another screen screenName Screen2
    else if  compare texts ListPicker2 . Text = " Summer " and compare texts ListPicker1 . Text = " Alkaline " and TextBox2 . Text < 30 and TextBox4 . Text < 60 and TextBox3 . Text < 3000
    then     open another screen screenName Screen4
    else if  compare texts ListPicker2 . Text = " Summer " and compare texts ListPicker1 . Text = " Neutral " and TextBox2 . Text < 30 and TextBox4 . Text < 60 and TextBox3 . Text < 3000
    then     open another screen screenName Screen3
    else if  compare texts ListPicker2 . Text = " Rainy " and compare texts ListPicker1 . Text = " Acidic " and TextBox2 . Text < 30 and TextBox4 . Text < 60 and TextBox3 . Text < 3000
    then     open another screen screenName Screen5
    else if  compare texts ListPicker2 . Text = " Rainy " and compare texts ListPicker1 . Text = " Neutral " and TextBox2 . Text < 30 and TextBox4 . Text < 60 and TextBox3 . Text < 3000
    then     open another screen screenName Screen6
    else if  compare texts ListPicker2 . Text = " Rainy " and compare texts ListPicker1 . Text = " Alkaline " and TextBox2 . Text < 30 and TextBox4 . Text < 60 and TextBox3 . Text < 3000
    then     open another screen screenName Screen7
    else if  compare texts ListPicker2 . Text = " Winter " and compare texts ListPicker1 . Text = " Acidic " and TextBox2 . Text < 30 and TextBox4 . Text < 60 and TextBox3 . Text < 3000
    then     open another screen screenName Screen8
    else if  compare texts ListPicker2 . Text = " Winter " and compare texts ListPicker1 . Text = " Neutral " and TextBox2 . Text < 30 and TextBox4 . Text < 60 and TextBox3 . Text < 3000
    then     open another screen screenName Screen9
    else if  compare texts ListPicker2 . Text = " Winter " and compare texts ListPicker1 . Text = " Alkaline " and TextBox2 . Text < 30 and TextBox4 . Text < 60 and TextBox3 . Text < 3000
    then     open another screen screenName Screen10
    else     if       compare texts ListPicker2 . Text = " Summer " and compare texts ListPicker1 . Text = " Acidic " and TextBox2 . Text < 30 and TextBox4 . Text > 60 and TextBox3 . Text < 3000
             then     open another screen screenName Screen3
             else if  compare texts ListPicker2 . Text = " Summer " and compare texts ListPicker1 . Text = " Alkaline " and TextBox2 . Text < 30 and TextBox4 . Text > 60 and TextBox3 . Text < 3000
             then     open another screen screenName Screen4
             else if  compare texts ListPicker2 . Text = " Summer " and compare texts ListPicker1 . Text = " Neutral " and TextBox2 . Text < 30 and TextBox4 . Text > 60 and TextBox3 . Text < 3000
             then     open another screen screenName Screen5
             else if  compare texts ListPicker2 . Text = " Rainy " and compare texts ListPicker1 . Text = " Acidic " and TextBox2 . Text < 30 and TextBox4 . Text > 60 and TextBox3 . Text < 3000
             then     open another screen screenName Screen6
```

```
when FirebaseDB1 .DataChanged
  tag  value
do  if    get tag = " temp "
    then  set TextBox2 . Text to  join  get value
                                        " [illegible] "
    if    get tag = " hum "
    then  set TextBox4 . Text to  join  get value
                                        " [illegible] "
    if    get tag = " mois "
    then  set TextBox3 . Text to  join  get value
                                        " [illegible] "
```

Figure 4.1: mit app inventor example code

4.3 CODE SNIPPETS

4.3.1 library functions used

```
#if defined(ESP32)
#include <WiFi.h>
#include <FirebaseESP32.h>
#elif defined(ESP8266)
#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>
#endif
#include <addons/TokenHelper.h>
#include <addons/RTDBHelper.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>
```

4.3.2 Configuring wireless pairing and database

```
#define WIFI_SSID "IN_Note1"
#define WIFI_PASSWORD "in@kerala"
#define API_KEY "AIzaSyASBEcCxE30Ou4UXacGRSwWWzNgj2L6DVQ"
#define DATABASE_URL "iotcrop-46409-default-rtdb.firebaseio.com"
#define USER_EMAIL "testiotcrop@gmail.com"
#define USER_PASSWORD "09122022"

FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;
```

4.3.3 Code for wireless and database connection

```
void setup()
{
  pinMode(soilPin ,INPUT);
  Serial.begin(115200);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(300);
  }
  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());
  Serial.println();
  Serial.printf("Firebase Client v%s\n\n",FIREBASE_CLIENT_VERSION);

config.api_key = API_KEY;
  auth.user.email = USER_EMAIL;
  auth.user.password = USER_PASSWORD;
  config.database_url = DATABASE_URL;
  config.token_status_callback = tokenStatusCallback;

  Firebase.begin(&config, &auth);
  Firebase.reconnectWiFi(true);
  Firebase.setDoubleDigits(5);
  sensor_t sensor;
```

4.3.4 Code to read sensor data

```
soil_val=analogRead(soilPin);


  // Get temperature event and print its value.
sensors_event_t event;
  dht.temperature().getEvent(&event);
  temp=event.temperature;


  // Get humidity event and print its value.
  dht.humidity().getEvent(&event);
  hum=event.relative_humidity;
```

4.3.5 Code for sending collected data to firebase

```
 if (Firebase.ready() &&
 (millis() - sendDataPrevMillis > 5000 || sendDataPrevMillis == 0)){

    sendDataPrevMillis = millis();

    Serial.printf("Set int... %s\n",
    Firebase.setInt(fbdo, F("/hum"), hum) ? "ok" : fbdo.errorReason().c_str());
    Serial.printf("Set int... %s\n",
    Firebase.setInt(fbdo, F("/mois"),soil_val)?"ok": fbdo.errorReason().c_str());
    Serial.printf("Set int... %s\n",
    Firebase.setInt(fbdo, F("/temp"), temp) ? "ok" : fbdo.errorReason().c_str());
    }
}
```

Chapter 5

PERFORMANCE EVALUATION

5.1 EVALUATION SCENARIO

5.1.1 Scenario based testing

For obtain real world scenario, the IoT device is placed over soil having different nature to verify different crops are recommended according to varying condition.Temperature and humidity variation is also considered by testing the device in various climate.

5.1.2 Load Testing

The load test was conducted by keeping the device running for 10 minutes, taking sensor readings every second and converting it to digital data. The application performed smoothly, without any effect on the latency of sensors.

5.1.3 Install/Uninstall testing

Installing the dependencies in the backend requires a single command. On average, it takes 30 seconds for all the dependencies to get installed and connect to the database. No errors will be thrown due to the package-lock file that specifies the exact version of dependencies to be installed. The hardware is implemented in a plug and play manner which would take a few minutes to get it up and running. The ESP32 is connected to the mobile using mobile hotspot and data collected from the microcontroller based device is sent through internet to real-time database.

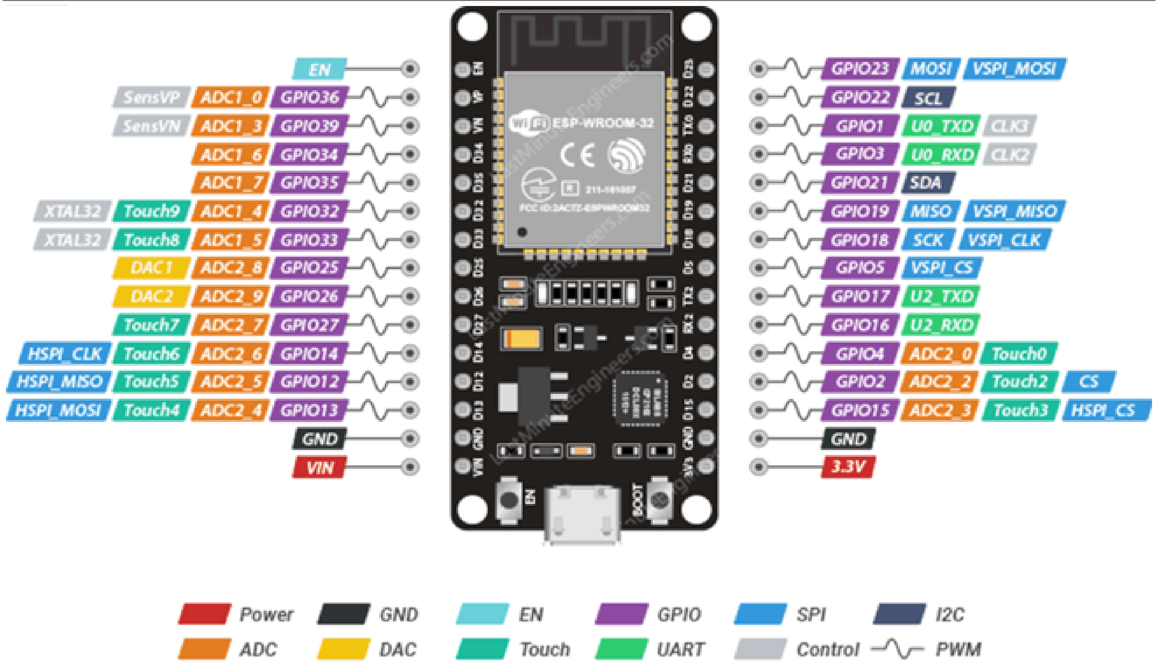
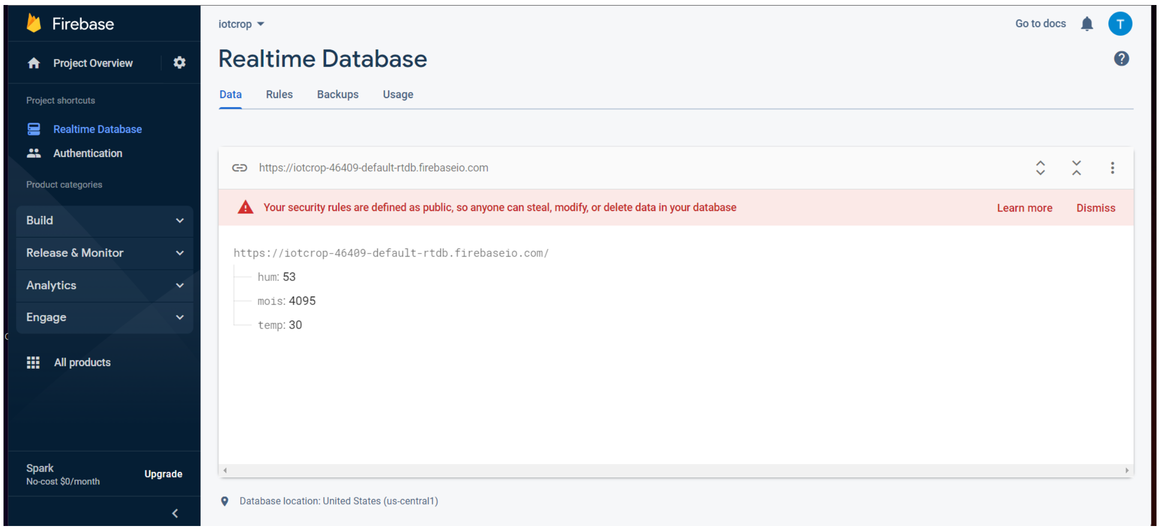


Figure 5.1: ESP32 Pinout

5.2 RESULTS

When the device is turned on, it will begin to establish a connection with a nearby mobile hotspot owned by a confirmed user. Then it will begin transmitting sensor data to a mobile application via a real-time database. The user can manually input certain data depending on their understanding in addition to sensor output. The data that was currently being gathered is examined after pressing the suggestion button, and the appropriate result is produced. That result is shown on a new screen. Every time the suggestion button is clicked, the procedure is repeated.



https://iotcrop-46409-default-rtdb.firebaseio.com

Your security rules are defined as public, so anyone can steal, modify, or delete data in your database

```
https://iotcrop-46409-default-rtdb.firebaseio.com/
  hum: 53
  mois: 4095
  temp: 30
```

Database location: United States (us-central1)

Figure 5.2: realtime database using firebase

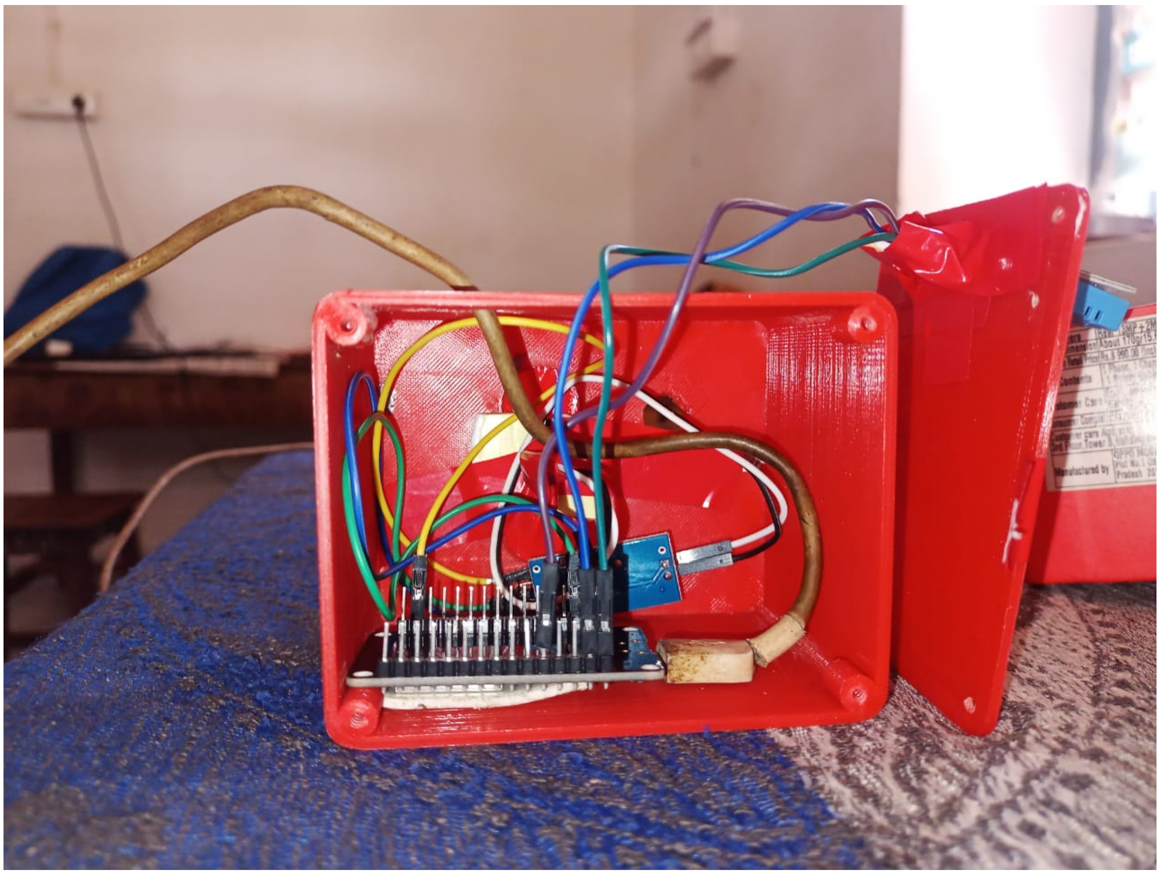


Figure 5.3: Finished product interior

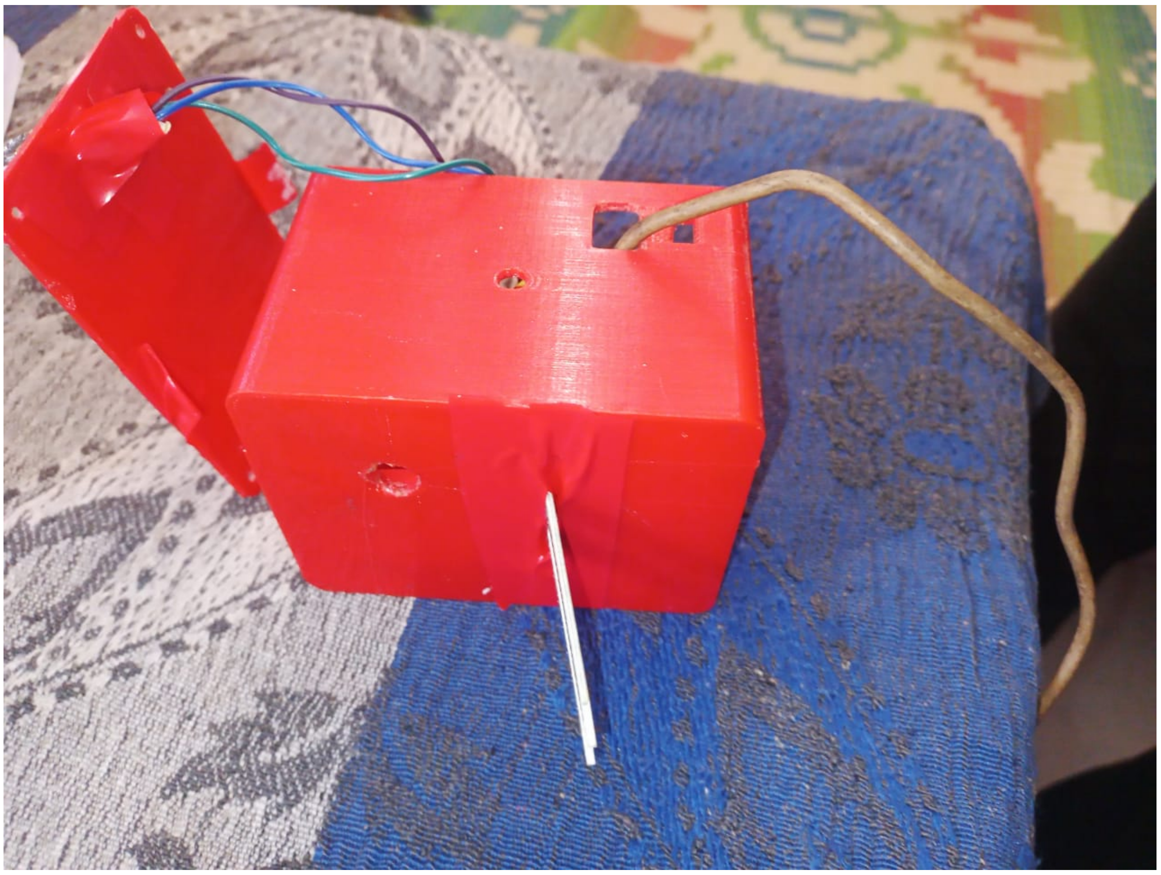


Figure 5.4: Finished product exterior

Chapter 6

DOCUMENTATION

This chapter is meant as a documentation to any who who wishes to use or contribute to the application through sections 6.2, 6.3. Section 6.4 consists of the means to contact the developers for any queries.

6.1 INTRODUCTION

The hardware side of the system is implemented using ESP32 and C with various libraries are used to take, pre-process and convert analog data into digital. The client side of the mobile application contains modules for the consumer. The front end uses MIT app inventor framework. For the back-end,realtime database is implemented using firebase.

6.2 WORKING WITH THE PRODUCT

Git is used for version control throughout the project. All code is also hosted on GitHub. Production code is maintained on the 'main' branch. Development and testing is to be done in 'dev' or other branches if created to solve any specific issue. To contribute we recommend creating an issue. To contribute to the project, one raise an issue whether it's

1. Reporting an issue
2. Discussing the current state of the code
3. Submitting a fix

4. Proposing new features

All changes to the production code happen through pull requests, only after the approval of the collaborators.

(i) Making Pull Requests

1. Fork the repository and create your branch (usually named ’patch-[the number of pull requests you’ve already made or your name - [shorthand for issue your working on]]) from staging.
2. If you’ve added code that should be tested, add some test examples.
3. Ensure to describe your pull request.

6.3 CONTACT

For any queries, contact the developers.

1. Karthik S - tcr19ec035@gectcr.ac.in
2. Rahul R Nair - tcr19ec044@gectcr.ac.in
3. Sivaprasad A S - tcr19ec051@gectcr.ac.in

Chapter 7

CONCLUSION AND FUTURE WORK

7.1 CONCLUSION

This project presented an IoT-based method for crop recommendation. The implemented system consists of a microcontroller (ESP32) as the main processing unit for the entire system and all the sensors(DHT-11, Soil moisture sensor) and devices can be connected to the microcontroller. The microcontroller can operate the sensors to retrieve the data from them and it processes the analysis with the sensor data and updates it to the internet-required information on a mobile application i.e suitable crops which can be cultivated according to the sensed conditions. An IoT-based crop recommendation system is a valuable tool for farmers that can help them improve yields, reduce waste, and increase profitability. As technology continues to evolve, we can expect to see even more advanced and sophisticated systems that can provide even more personalized and accurate recommendations for farmers.

7.2 ADVANTAGES

1. Ease of monitoring your local weather conditions in real-time from anywhere in the world.
2. Provide better crop recommendations according to the condition
3. Improve productivity
4. Less power consumption

7.3 LIMITATIONS & FUTURE EXPANSIONS

7.3.1 Limitations

1. Lack of network connectivity

2. The use of the mobile application should not require a technical background.

3. Applicable for short-range.

4. Lack of prior knowledge about seasons

7.3.2 Future Scope

1. Can add charging port and rechargeable power supply

2. Can integrate more sensors for more accuracy

3. Can be integrate with ML algorithms for better recommendation

REFERENCES

[1] **Lakshmi. N, Priya.M, Sahana Shetty, Manjunath C. R**,Crop Recommendation System for Precision Agriculture, vol. 6 Reading, IND: International Journal for Research in Applied Science & Engineering Technology, 2018.

[2] **Haedong Lee and Aekyung Moon**, "Development of Yield Prediction System Based on Real-time Agricultural Meteorological Information", 16th International Conference on Advanced Communication Technology, 2019.

[3] **T.R. Lekhaa**,"Efficient Crop Yield and Pesticide Prediction for Improving Agricultural Economy using Data Mining Techniques", International Journal of Modern Trends in Engineering and Science (IJMTES), 2016, Volume 03, Issue 10.

[4] **Vaibhav Bhatnagar**Internet of Things and Analytics for Agriculture, Volume 2

[5] **Thilakarathne, N.N.; Bakar, M.S.A.; Abas, P.E.; Yassin, H.**, A Cloud Enabled Crop Recommendation Platform for Machine Learning-Driven Precision Farming. Sensors 2022, 22, 6299.

[6] **CAIXIA SONG AND HAOYU DONG**, Application of Intelligent Recommendation for Agricultural Information: A Systematic Literature Review

[7] **Badamasi, Y.A**, The working principle of an Arduino. In: 2014 11th International Conference on Electronics, Computer and Computation (ICECCO). IEEE (2014)

[8] **Yamaguchi, N., et al.**, E-kakashi project, an agri sensor network using ad hoc network technology. In: 2011 Proceedings of SICE Annual Conference (SICE). IEEE (2011)

[9] **Lee, M., Hwang, J., Yoe, H.**, Agricultural production system based on IoT. In: 2013 IEEE 16th International Conference on Computational Science and Engineering (CSE). IEEE (2013)

[10] **Tianlong, N.**, Application of single bus sensor DHT11 in temperature humidity measure and control system. Microcontrollers Embed. Syst. 6, 026 (2010)

[11] **Pahuja, Ritika, Kumar, Narender**, Androidmobile phone controlled bluetooth robot using 8051 microcontroller. Int. J. Sci. Eng. Res. 2(7), 14–17 (2014)