Intelligent Irrigation System (IIS)

A Report submitted in partial fulfillment for the degree of,

Bachelor of Technology in

Data Science & Artificial Intelligence

By,

Rechu Vivek Reddy (21STUCHH010365), P. Kirthi Tejesh Reddy (21STUCHH010360) & Pinipe Sai Yaswanth (21STUCHH010362)

Pursued in,

Department of Data Science & Artificial Intelligence

IcfaiTech (Deemed to be University)

HYDERABAD

April 2024

CERTIFICATE

This is to certify that the project report entitled **Intelligent Irrigation System (IIS)** submitted by **Pinipe Sai Yaswanth** to the IcfaiTech Faculty of Science and Technology, Hyderabad, in partial fulfillment for the award of the degree of **B. Tech in (Data Science & Artificial Intelligence)** is a *bona fide* record of project work carried out by him under my supervision. The contents of this report, in full or in parts, have not been submitted to any other Institution or University for the award of any degree or diploma.

Dr. Sirisha Potluri

Supervisor

Dept. of DSAI

Counter signature of HOD with seal

Hyderabad,

April 2024

DECLARATION

I declare that this project report titled **Intelligent Irrigation System (IIS)** submitted in partial fulfillment of the degree of **B. Tech in (Data Science & Artificial Intelligence)** is a record of original work carried out by me under the supervision of **Dr. Sirisha Potluri**, and has not formed the basis for the award of any other degree or diploma, in this or any other Institution or University. In keeping with the ethical practice in reporting scientific information, due acknowledgements have been made wherever the findings of others have been cited.

Pinipe Sai Yaswanth

21STUCHH010362

Hyderabad,

April 2024

TABLE OF CONTENTS

ABSTRACT

This group project aims to develop an Intelligent Irrigation System (IIS) designed to enhance agricultural practices through the integration of sensor technologies and data-driven decision-making. The system involves strategically placed soil moisture sensors in a uniform layout across the field, coupled with climate data specific to the farm's location. By incorporating information about the crop type, a dynamic and intelligent model is created.

The proposed solution includes a user-friendly mobile application for farmers, offering both automatic and manual modes. In the automatic mode, real-time data on climate and soil moisture is leveraged to make informed decisions about when and how much water to supply to the field. The manual mode provides farmers with hands-on control over irrigation.

Furthermore, the application provides additional features such as a timer and customizable water supply duration, empowering farmers with flexibility and control. This holistic approach to intelligent irrigation aims to optimize water usage, increase crop yield, and contribute to sustainable farming practices. The integration of technology into traditional agriculture holds the potential to revolutionize farming methods, making them more efficient and environmentally friendly.

LITERATURE REVIEW

- Researchers like Chen et al. (2019) optimized tomato water use using fuzzy neural networks and genetic algorithms. Guha et al. (2021) predicted irrigation volumes based on crop growth and greenhouse effects. Rodríguez et al. (2019) improved tomato yields by treating salinity, while Zhai et al. (2015) proposed strategies like saline water use to enhance production.
- Shao et al. (2014) improved crop productivity with innovative irrigation techniques. Keswani et al. (2020), Maroli et al. (2021), and Mousavi et al. (2021) optimized tomato quality using rain shelters for irrigation management. Gil et al. (2019) employed IoT for precise water irrigation, integrating desalination and solar energy for enhanced cultivation.
- Krishna et al. (2017) introduced water-saving irrigation with Raspberry Pi and sensors. Qiu et al. (2020) analyzed tomato growth using neural networks. Mason et al. (2019) devised climate-specific irrigation for optimal tomato yield. Chen M. et al. (2021) applied smart irrigation with IoT and sensors.
- The agriculture sector relies on 4G/3G/NB-IoT networks for IoT-based smart device connectivity (Dell'Uomo and Scarrone, 2002). Challenges include limited space, rural coverage gaps, and QoS issues in 4G networks (Payaswini and Manjaiah, 2014; Payero et al., 2017). Short device lifespans and limited drone utility in remote fields hinder IoT farming progress. Enhancements are needed for efficient, cost-effective farming operations.

- Transitioning from 4G to 5G networks resolves limitations in wireless technology (Zhaogan et al., 2007). With 5G, farmers can remotely control drones over long distances for real-time data collection (Faraci et al., 2018; Sinha and Dhanalakshmi, 2022). They can access high-definition video streams and sensory data, enhancing efficiency (Bhattacharya and De, 2021).
- Thilakarathne et al. (2023) underscore smart agriculture's value with a low-cost IoT platform for monitoring and automating tomato cultivation indoors. Their study aims to enhance productivity, quality, and sustainability in agriculture.
- Usman et al. (2022) propose using 6G to monitor plant health with a THz-based sensor and communication system. Precision agriculture aims to optimize resource use by monitoring plant health and water levels at micro and macro scales. THz technology assesses plant health at a cellular level, enhancing decision-making in agriculture
- Dinesh Kumar, Pramod, and Sravani from Vignan Institute of Technology & Science, India, offer an intelligent irrigation system for Indian agriculture, addressing challenges posed by erratic monsoons. Their microcontroller-based system conserves water by detecting soil moisture levels and optimizes plant health with automated pesticide spraying
- Yomna Gamal et al. from Nile University, Egypt, provide a succinct review on intelligent irrigation control and monitoring strategies for enhanced efficiency in smart agriculture
- Debabrata Singh et al. present an IoT-based smart farming strategy for increasing efficiency and output while reducing water consumption in

tomato cultivation. Farmers utilize IoT devices to monitor crop health, predict moisture levels, and automate irrigation systems

- Vandana Sharma and Ravi Tiwari present a paper exploring the transformative potential of Internet of Things (IoT) technology in computing.
- Kamal Gulati, Raja Sarath Kumar Boddu, Dhiraj Kapila, Sunil L. Bangare, Neeraj Chandnani, and G. Saravanan discuss the rapid evolution of wireless systems based on IoT technology across various sectors. The Internet of Things (IoT) facilitates communication among physical devices, sensors, and objects without human intervention.
- Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal explore the significance of wireless sensor networks (WSNs) in remote environmental monitoring and target tracking. They highlight the evolution of WSNs enabled by smaller, cost-effective, and intelligent sensors equipped with wireless interfaces.
- Ferdoush and Xinrong Li present a wireless sensor network system designed with Arduino, Raspberry Pi, XBee, and open-source software packages for environmental monitoring applications. The system boasts low-cost, scalability, ease of customization, deployment, and maintenance.
- Dharmaraj and Vijayanand discuss the potential of Artificial Intelligence (AI) in agriculture to address the challenges posed by a growing global population and limited land resources.

Chapter-1: INTRODUCTION

The project at hand combines machine learning (ML), precision irrigation systems, and the Internet of Things (IoT), three cutting-edge technologies that have the potential to completely transform agriculture. Combining these technologies presents a promising way to maximize water use, boost crop output, and support ecologically friendly farming methods in a time when sustainable farming practices are essential to meeting the demands of a growing global population.

There has never been a greater need for effective water management because irrigation uses around 85% of the freshwater resources on Earth. Conventional farming practices frequently lead to excessive water use and inadequate crop yields. Nevertheless, we aim to revolutionize traditional farming paradigms by utilizing IoT sensors, machine learning algorithms, and precise irrigation systems.

Our project's goal is to create an intelligent irrigation system that can dynamically modify water usage in response to crop needs and current environmental circumstances. Our system will use machine learning (ML) algorithms to assess data collected from IoT sensors on crop health, weather patterns, and soil moisture levels. Based on this analysis, choices will be made on water distribution and irrigation schedule.

Farmers will be able to take control of the irrigation process and obtain actionable insights by integrating an intuitive smartphone application. Our technology aims to maximize crop output, promote sustainable farming practices, and optimize

resource consumption by providing farmers with real-time data and predictive analytics.

Components of the System:

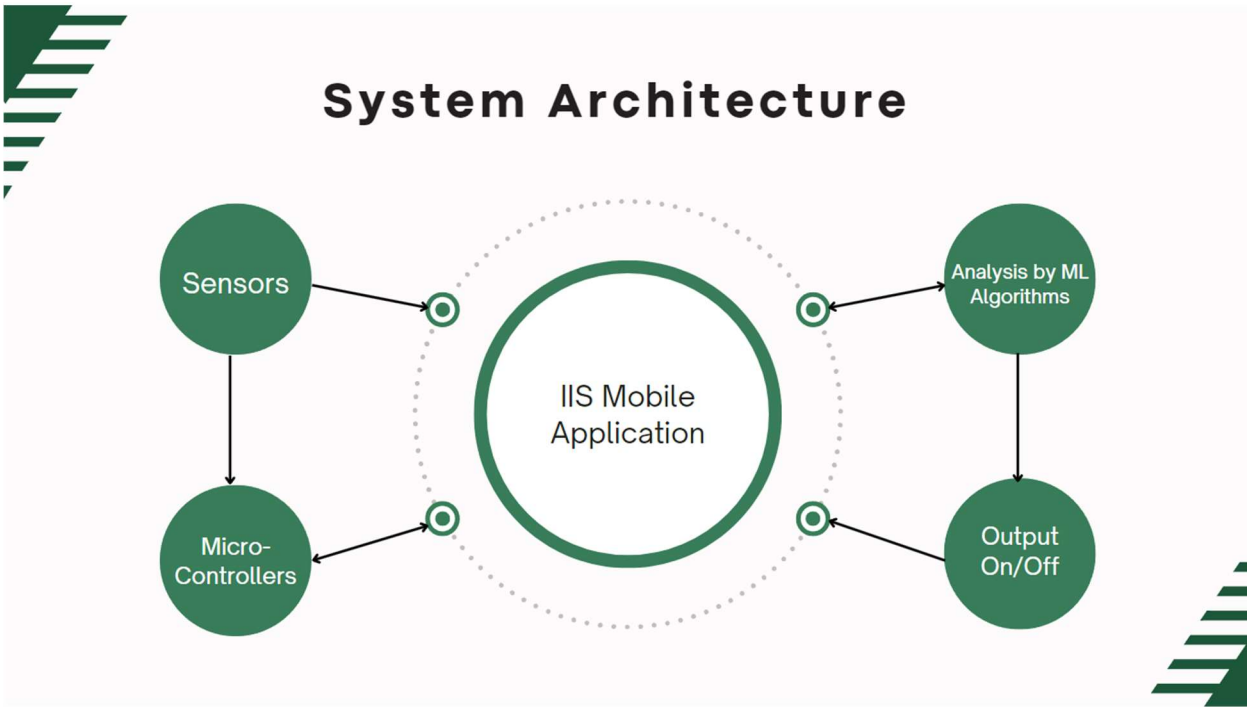
1. Physical Components:
    - Moisture Sensors
    - Arduino and Relay module
    - ESP32 Microcontrollers
    - Connecting wires and batteries
    - Water pumping motor
2. Software Components:
    - Arduino IDE 2
    - Android studio/ Vscode-Flutter
    - Gretel(Data Generation)
    - Firebase
    - Kotlin Gradle
    - Google Colab

With a proper integration of both the software and hardware components we will be able to create a working Intelligent Irrigation system.

We provide a thorough summary of the project in this report, together with information on its goals, implementation strategy, specifics, and prospects for the future. We hope to clear the path for a more productive, efficient, and sustainable future in agriculture by utilizing precision irrigation, machine learning, and the Internet of Things.

Chapter-2: SYSTEM ARCHITECTURE

In this chapter, we delve into the intricacies of the system architecture underpinning our smart irrigation project. Our system is designed to efficiently manage water usage and enhance crop yields through the integration of various components, including a mobile application, sensors, microcontrollers, and a machine learning model.



2.1 Interface

The mobile application interface, created using Android Studio, Kotlin Gradle, and Flutter in VS Code, is the central component of our system. Users interact

primarily with this user-friendly interface, which gives them access to a variety of functions and functionalities.

Flutter: With Flutter, developers can create natively built apps for desktop, web, and mobile platforms all from the same codebase. Flutter is an open-source UI toolkit from Google. It offers a wide range of ready-made widgets and tools for quickly and easily building aesthetically pleasing user interfaces.

Android Studio: Google's official integrated development environment (IDE) for creating Android apps is called Android Studio. It provides an extensive feature set and tool set designed especially for creating Android applications. Developers can efficiently build, debug, and test their apps with Android Studio.

2.2 Sensors

The system uses moisture sensors to measure the amount of moisture in the soil. It also retrieves location data from an API for temperature and meteorological information related to the crop or garden. These sensors are essential for showing the moisture, temperature and weather details in the application and for making decisions for motor (on/off) automatically by inputting them into a machine learning model.

2.3 Micro-controller Integration

Arduino and ESP32 microcontrollers are employed to interface between the sensors and the mobile application. These microcontrollers receive data from the sensors and transmit it to the mobile application via a local Wi-Fi network. Additionally, they facilitate manual control of the irrigation system by allowing users to activate the motor manually. In the manual mode, users have the flexibility to manually control the motor and access additional features such as a timer. In automatic mode the motor decision made by the machine learning

model is sent to the micro controller which will then operate the motor as required.

2.4 Machine Learning Model

A machine learning model is integrated into the system to automate irrigation decisions in the automatic mode. This model processes data from the sensors, including moisture levels, temperature, and climate information, to determine whether to activate the motor for irrigation. The integration of the machine learning model enhances the system's efficiency and autonomy. The machine learning model is seamlessly integrated into the Flutter application, allowing users to switch between automatic and manual modes with ease.

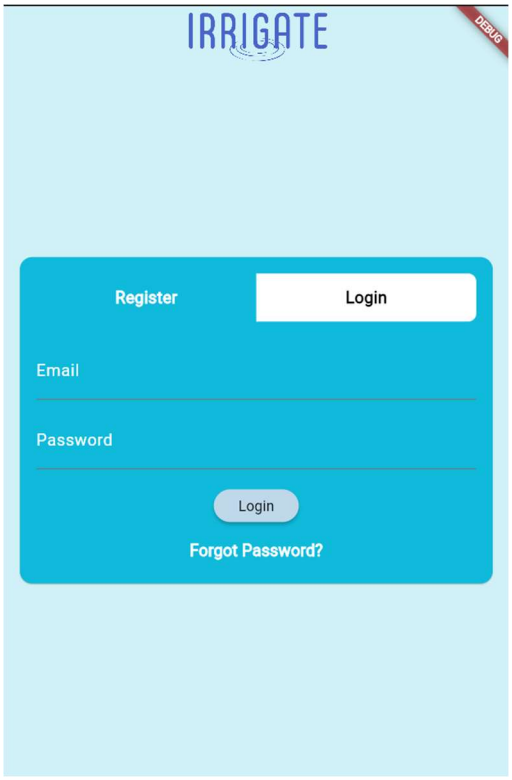
Chapter-3: FLUTTER APPLICATION

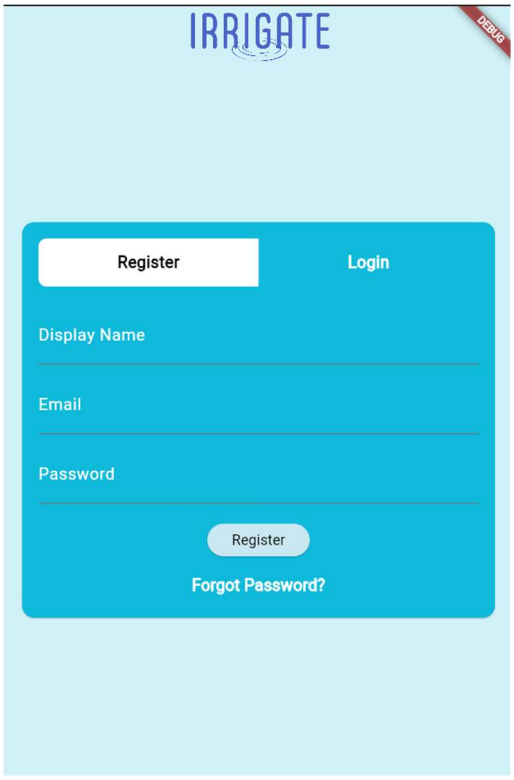
In this chapter, we outline the development of a Flutter application tailored for smart irrigation management, featuring multiple screens for various agricultural tasks. We highlight its integration with Firebase for user authentication, real-time data synchronization, and cloud storage. The application interfaces with microcontrollers like Arduino and ESP32 for remote monitoring and control of irrigation systems. Additionally, we discuss the integration of a machine learning model for predictive insights and automated decision-making. By combining Firebase, microcontrollers, and machine learning, the application offers a user-centric solution for optimizing resource utilization and enhancing crop yield in smart agriculture.

3.1 User Interface

Login/Register Page:

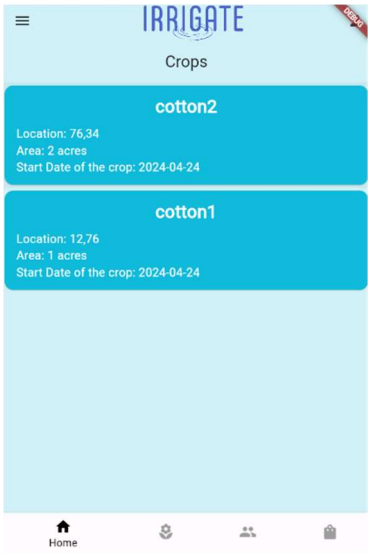
The user interface of our application begins with a login page, facilitating access for registered users. For new users, the option to create an account is available through the registration page. All authentication credentials and user data are securely stored and managed using Firebase, seamlessly integrated with our Flutter application. We will delve into the details of Firebase integration later in this report.

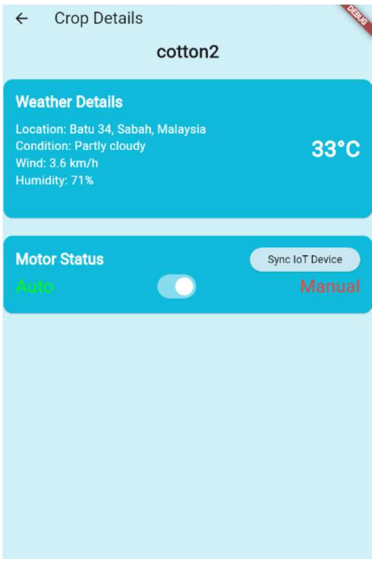


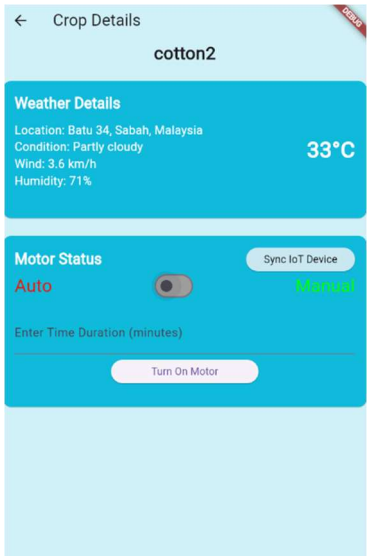


Home Page:

The home page of our application features a comprehensive overview of multiple crops added by the user. Each crop is represented as a clickable element, enabling users to access detailed information and control functionalities. Upon selecting an individual crop, users can regulate the motor and access real-time data such as weather conditions, temperature, and humidity. Additionally, users can monitor the pump status (on/off) and seamlessly switch between manual and auto modes. In auto mode, the system leverages a machine learning model for intelligent decision-making, while manual mode offers a customizable timer duration for motor activation. Furthermore, a dedicated "Turn On Motor" button facilitates manual motor control. For device synchronization, users can initiate a sync process with the IoT server of the microcontroller by clicking the "Sync IoT Device" button.

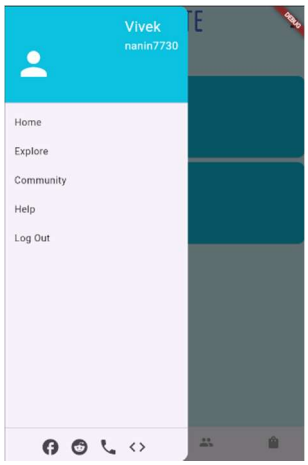






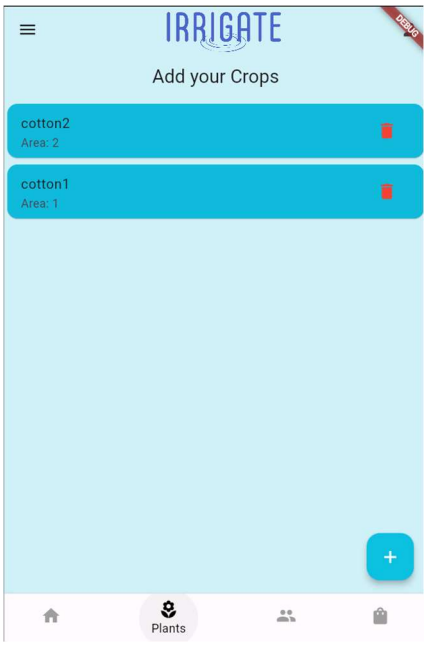
Hamburger:

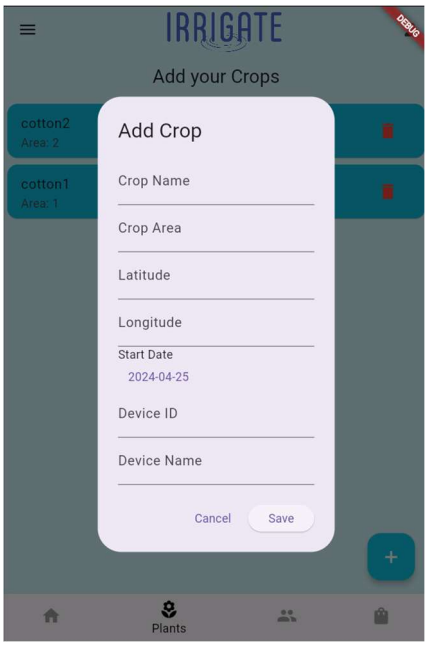
In addition to giving complete information on the user profile, the hamburger page offers easy access to all of the application's capabilities. It provides a handy menu for navigating, making it easy for users to explore the different features and areas of the application. A customized and individualized experience is ensured by the users' ability to access their profile information, which includes personal information, preferences, and settings. With an easy-to-use layout that facilitates smooth navigation and feature interaction, the hamburger page improves usability and user engagement.



Crops Page:

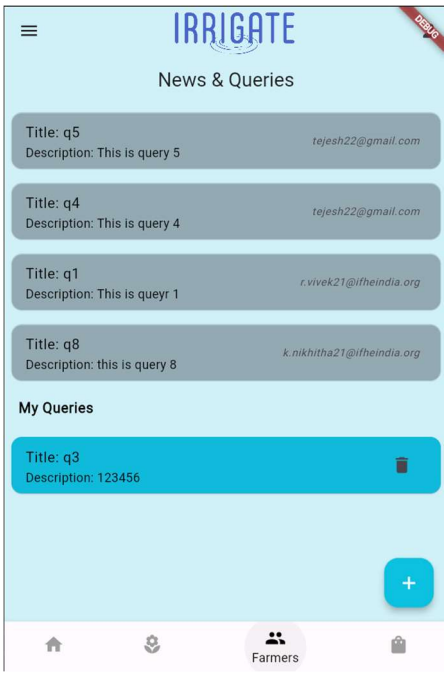
Users may add new crops to their crop list or remove existing ones using the crops page, which streamlines the crop management process. Information about each crop, including name, kind, planting date, and any other remarks or facts pertinent to its cultivation, may be entered by users with ease.

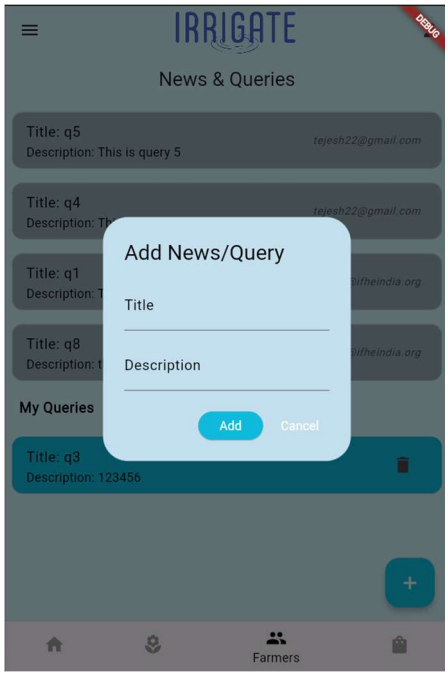




Farmers Page:

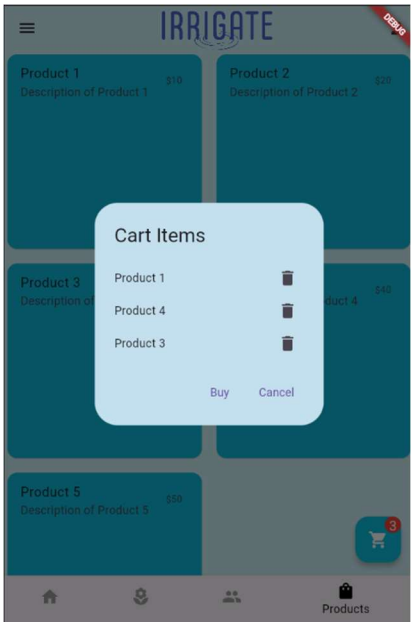
The farmers page provides a forum for farmers to participate in information exchange and community engagement, encouraging cooperation and support among the farming community. Users may post questions to ask other farmers for guidance or information, and they can also examine other people's questions to get insightful knowledge.





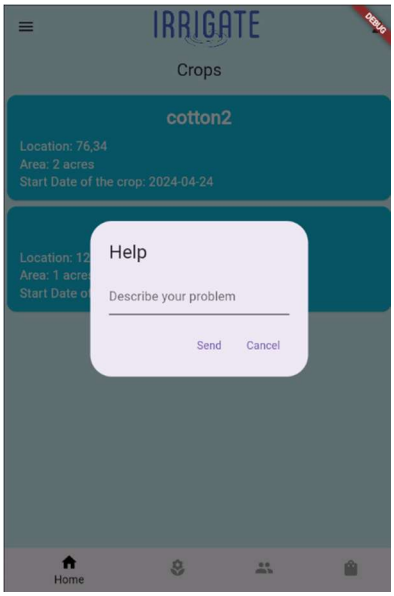
Products Page:

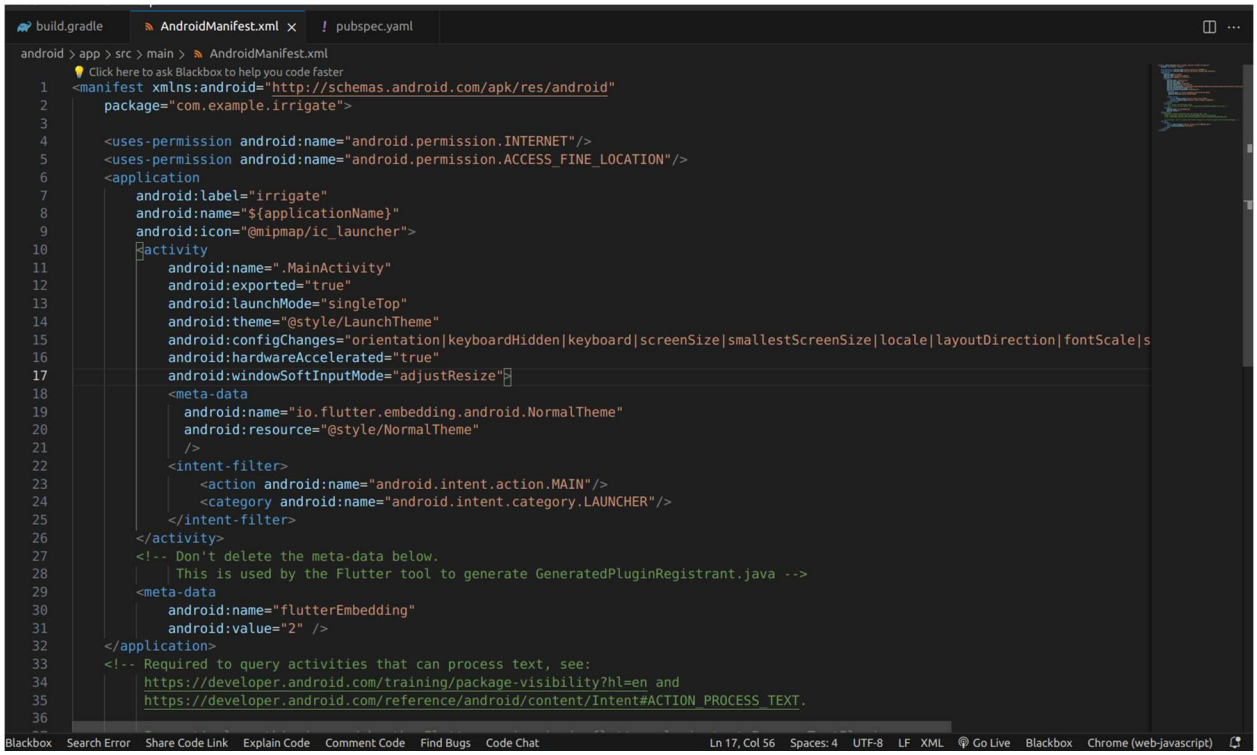
The products page offers a curated selection of essential farming products, including fertilizers, pesticides, and other agricultural supplies, providing farmers with convenient access to necessary resources for their farming needs. Users can browse through the product listings, exploring detailed descriptions and specifications to make informed purchasing decisions. The interactive interface enables users to add desired products to their cart for easy ordering and removal, ensuring a seamless shopping experience. Additionally, the platform may feature special promotions or discounts on select products, further enhancing the value proposition for farmers seeking quality agricultural inputs.



Help & Support:

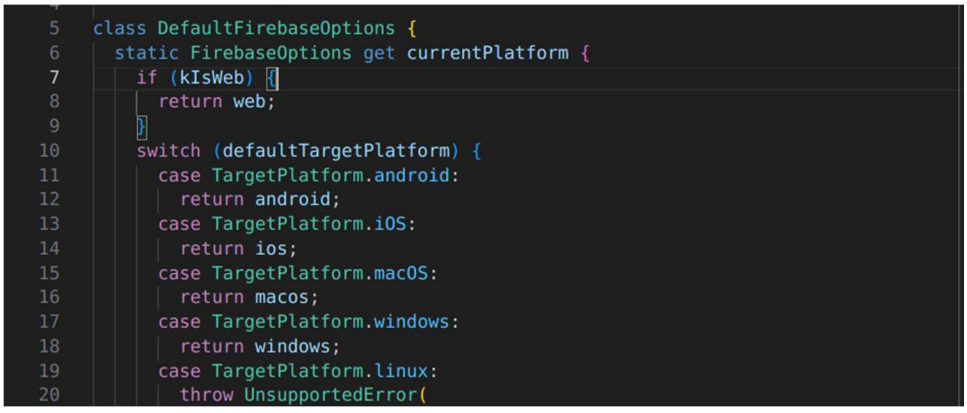
The help option provides users with a direct avenue to articulate their concerns or issues, enabling them to submit detailed descriptions of their problems or queries.





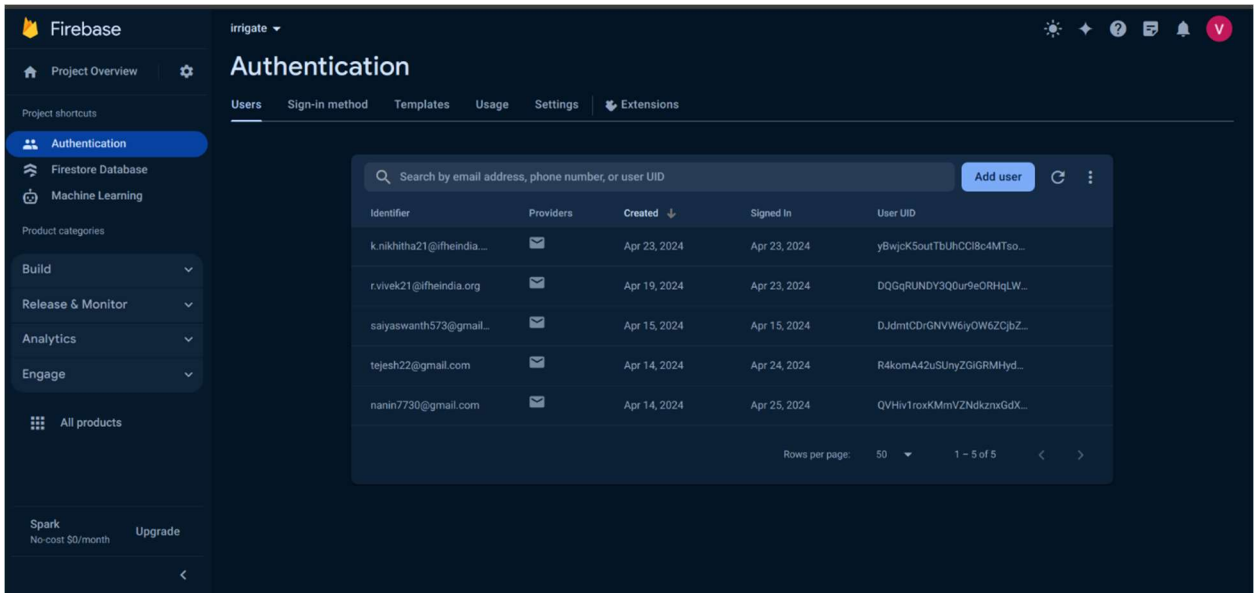
3.2 Firebase Integration

Firebase integration in our Flutter application provides a secure backend for user authentication and data storage. With Firebase Authentication, users can securely register and log in to the app. Firebase Realtime Database or Cloud Firestore stores dynamic data like crop details and user queries, ensuring real-time updates across devices. Cloud Functions automate backend processes, enhancing scalability. This integration ensures a reliable, secure, and scalable foundation for our app.



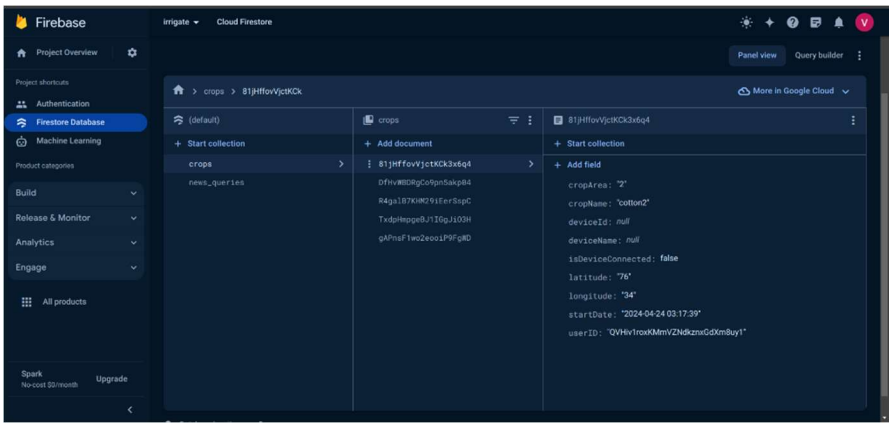
Authentication:

Firebase Authentication handles account creation, login, and secure storage of user credentials. This feature ensures that user accounts are securely managed, allowing for smooth registration and authentication processes within the Flutter application.



Crop and Queries:

Firebase Firestore is utilized for storing crop data and user queries within the Flutter application. This integration enables efficient and scalable management of agricultural information, facilitating seamless access and manipulation of crop details and user-generated queries.



3.3 Iot Integration

The Flutter application is integrated with IoT devices for real-time communication and control. This integration allows the application to send and receive data to and from IoT devices, enabling remote monitoring and control of agricultural equipment such as microcontrollers.

```dart
class _CropDetailsScreenState extends State<CropDetailsScreen> {
  void turnOnRelay() async {
    var response = await http.get(Uri.parse('http://$esp32Ip/relay-on'));
    print('Response: ${response.body}');
  }

  void turnOffRelay() async {
    var response = await http.get(Uri.parse('http://$esp32Ip/relay-off'));
    print('Response: ${response.body}');
  }

  void turnAuto() async {
    var response = await http.get(Uri.parse('http://$esp32Ip/mode-auto'));
    print('Response: ${response.body}');
  }

  void turnManual() async {
    var response = await http.get(Uri.parse('http://$esp32Ip/mode-manual'));
    print('Response: ${response.body}');
  }

  void syncIoTDevice() {
    // Implement logic to sync with IoT device here
    // For demonstration purposes, set isDeviceSynced to true after syncing
    setState(() {
      isDeviceSynced = true;
    });
```

3.4 ML Model Integration

The Flutter application integrates a machine learning (ML) model for intelligent decision-making in crop irrigation. Using TensorFlow Lite, the ML model is seamlessly integrated into the application, enabling real-time prediction of optimal irrigation schedules based on environmental data such as moisture levels, temperature, and weather conditions. This integration enhances the application's capabilities by providing automated and data-driven irrigation management for improved crop yield and resource efficiency.

In conclusion, the Flutter application represents a comprehensive solution for smart agriculture, leveraging advanced technologies such as machine learning, Firebase, and IoT integration. By seamlessly integrating these technologies, the application provides farmers with powerful tools for crop management, irrigation optimization, and community interaction.

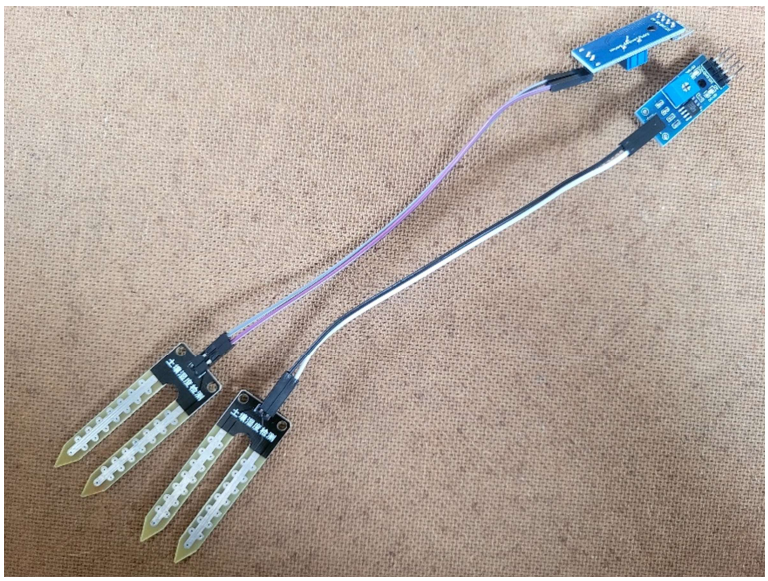
Chapter-4: HARDWARE ENIRONMENT

The hardware environment for the smart agriculture system comprises essential components such as moisture sensors, an ESP32 microcontroller, and a relay module. Together, these hardware components form the backbone of the smart agriculture system.

4.1 Components

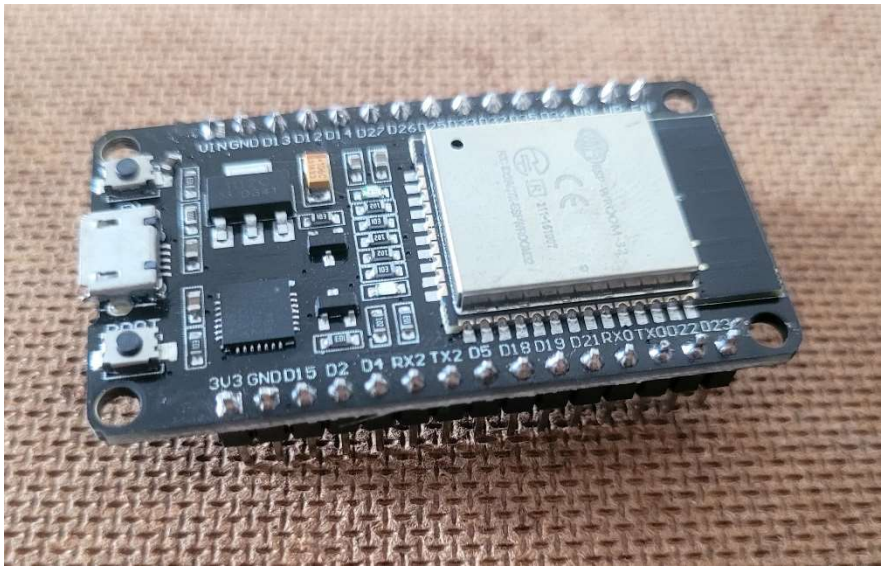
Moisture sensors:

A moisture sensor is a device used to measure the moisture content of soil. It typically consists of two electrodes inserted into the soil, measuring the electrical conductivity or resistance between them, which varies with soil moisture. These sensors play a crucial role in smart agriculture by providing real-time data on soil moisture levels, enabling efficient irrigation management to optimize crop health and yield.



ESP32 Micro-controller:

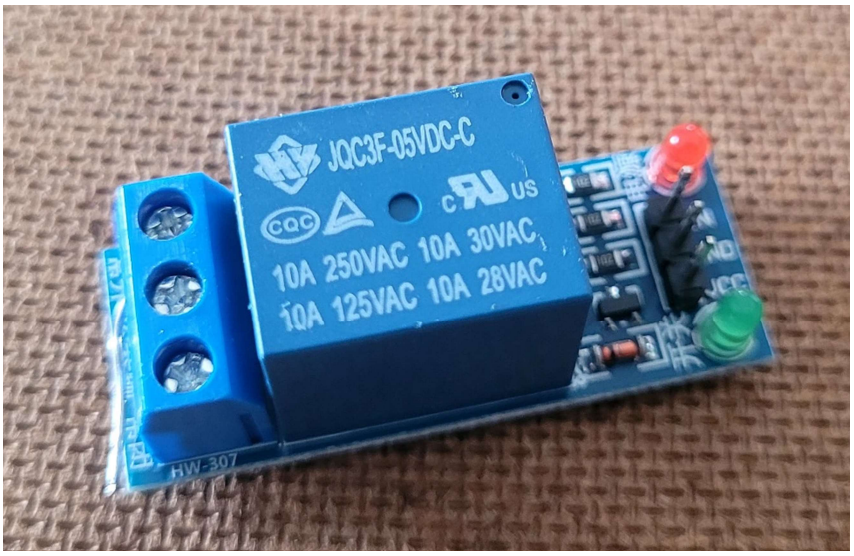
The ESP32 microcontroller is a powerful and versatile embedded system-on-chip (SoC) developed by Espressif Systems. It features a dual-core processor, Wi-Fi and Bluetooth connectivity, ample I/O pins, and various built-in peripherals, making it ideal for a wide range of IoT applications. In smart agriculture, the ESP32 is commonly used to interface with sensors, control actuators, and communicate data wirelessly to cloud platforms or mobile devices. Its low power consumption, small form factor, and support for various programming languages make it well-suited for deploying IoT solutions in agricultural environments.



Relay Modeule:

A relay module is an electromechanical switch that controls the flow of electricity in a circuit. It consists of a coil and one or more sets of contacts. When the coil is energized, it creates a magnetic field that attracts the contacts, closing or opening the circuit depending on the relay type. In smart agriculture applications, relay modules are commonly used to control high-power devices such as motors, pumps,

or lights based on signals received from microcontrollers or other control systems. They provide isolation between the control circuit and the high-voltage load, ensuring safe and reliable operation in agricultural environments.



Water Pumping Motor:

A water pumping motor is an essential component in irrigation systems used for agricultural purposes. It is typically an electric motor coupled with a pump that draws water from a water source such as a well, reservoir, or storage tank and delivers it to the fields through pipelines or irrigation channels. These motors are designed to provide the necessary pressure and flow rate to distribute water evenly across the cultivated area, ensuring optimal moisture levels for crop growth. In smart agriculture systems, water pumping motors are often controlled remotely using microcontrollers or IoT devices, allowing farmers to automate irrigation schedules and conserve water resources more efficiently.

4.2 Arduino IDE2

The ESP32 and other microcontrollers may be programmed using the Arduino IDE 2, an integrated development environment. Writing, building, and uploading code to the ESP32 microcontroller board is made easier with its user-friendly interface. It is simple for developers to add several capabilities including sensor readings, data processing, and control algorithms using the Arduino IDE 2 since it provides easy access to a large library and example programs. The development and debugging process is made easier by the IDE's capabilities, which include code completion, syntax highlighting, and serial monitoring. Additionally, it facilitates the simple deployment and connection of ESP32-based projects by supporting seamless interaction with well-known IoT platforms and services. All things considered, the Arduino IDE 2 is a potent tool that both novice and expert developers may use to construct cutting-edge microcontroller applications.

Here is the code for implementing Iot on Hradware components,

```
#include <WiFi.h>
#include <WebServer.h>

const char* ssid = "NANI";
const char* password = "Nani@7730";
#define soil_moisture_pin 32
#define relay_in 5

WebServer server(80);
bool automaticMode = false; // Flag to indicate automatic mode

void handleRoot() {
  server.sendHeader("Access-Control-Allow-Origin", "*");
  server.send(200, "text/plain", "Hello from ESP32!");
}

void handleRelayOn() {
  server.sendHeader("Access-Control-Allow-Origin", "*");
  server.send(200, "text/plain", "Relay turned ON");
  digitalWrite(relay_in, LOW); // Turn on the relay
}

void handleRelayOff() {
  server.sendHeader("Access-Control-Allow-Origin", "*");
```

```
  server.send(200, "text/plain", "Relay turned OFF");
  digitalWrite(relay_in, HIGH); // Turn off the relay
}

void handleModeAuto() {
  pinMode(soil_moisture_pin, INPUT);
  automaticMode = true;
  server.sendHeader("Access-Control-Allow-Origin", "*");
  server.send(200, "text/plain", "Automatic mode activated");
}

void handleModeManual() {
  automaticMode = false;
  server.sendHeader("Access-Control-Allow-Origin", "*");
  server.send(200, "text/plain", "Manual mode activated");
}

void enableCORS() {
  server.sendHeader("Access-Control-Allow-Origin", "*");
  server.sendHeader("Access-Control-Max-Age", "10000");
  server.sendHeader("Access-Control-Allow-Methods", "POST, GET, OPTIONS");
  server.sendHeader("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");
}

void setup() {
  Serial.begin(115200);

  // Connect to WiFi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
    // Add timeout handling or retry logic if needed
  }

  // Print ESP32 IP address
  Serial.print("Connected to WiFi. IP address: ");
  Serial.println(WiFi.localIP());

  // Route setup
  server.on("/", HTTP_GET, handleRoot);
  server.on("/relay-on", HTTP_GET, handleRelayOn);
  server.on("/relay-off", HTTP_GET, handleRelayOff);
  server.on("/mode-auto", HTTP_GET, handleModeAuto); // New route for automatic mode
  server.on("/mode-manual", HTTP_GET, handleModeManual); // New route for manual mode
  server.onNotFound(enableCORS);

  // Set relay pin as output and initially turn off the relay
  pinMode(relay_in, OUTPUT);
  digitalWrite(relay_in, HIGH);

  // Set soil moisture pin as input
  pinMode(soil_moisture_pin, INPUT);

  Serial.println("Setup completed.");
```

```
  // Start server
  server.begin();
  Serial.println("HTTP server started");
}

void loop() {
  server.handleClient();  // Handle incoming client requests

  if (automaticMode) {
    pinMode(soil_moisture_pin, INPUT);
    int moistureValue = analogRead(soil_moisture_pin);
    Serial.println(moistureValue);

    if (moistureValue > 2000) {
      digitalWrite(relay_in, LOW); // Turn on the relay
      Serial.println("Water Pump is ON");
    } else {
      digitalWrite(relay_in, HIGH); // Turn off the relay
      Serial.println("Water Pump is OFF");
    }
  }

  delay(1000); // Adjust delay as needed for your application
}
```

Integration of the ESP32 microcontroller, moisture sensors, relay modules, and water pumping motor has enabled the development of a sophisticated smart farming system. Leveraging Arduino IDE 2 for ESP32 programming, we've achieved seamless communication between hardware components, allowing for real-time data acquisition and control. The moisture sensors provide accurate measurements of soil moisture levels, which are crucial for efficient irrigation management. The relay modules facilitate the control of the water pumping motor, enabling automated irrigation based on sensor readings and user-defined parameters. This integration enhances agricultural productivity by ensuring optimal water usage and crop health. Overall, the hardware environment plays a critical role in enabling smart farming practices, offering farmers the tools they need to make informed decisions and optimize resource utilization for sustainable agriculture.

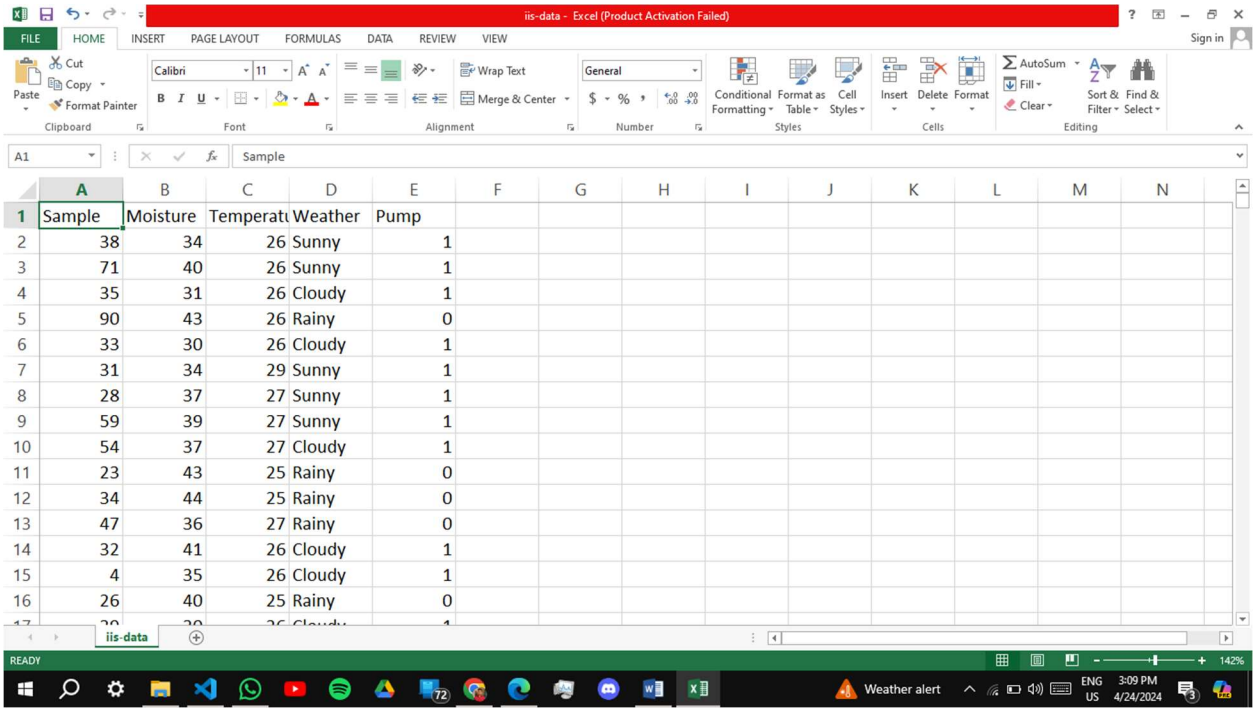
Chapter-5: AUTOMATED MODEL

The model developed is a Multilayer Perceptron (MLP) classifier, implemented using scikit-learn library in Python. This type of model is commonly used for classification tasks in machine learning.

5.1 Data Generation

The data set consist of three attributes or feature vectors moisture, temperature and weather and a class label which indicates the pump status(on/off).

50 of the samples are recorded manually and from these 50 samples another 500 samples are generated using a synthetic data generator(Gretel.ai)



Sample	Moisture	Temperatu	Weather	Pump
38	34	26	Sunny	1
71	40	26	Sunny	1
35	31	26	Cloudy	1
90	43	26	Rainy	0
33	30	26	Cloudy	1
31	34	29	Sunny	1
28	37	27	Sunny	1
59	39	27	Sunny	1
54	37	27	Cloudy	1
23	43	25	Rainy	0
34	44	25	Rainy	0
47	36	27	Rainy	0
32	41	26	Cloudy	1
4	35	26	Cloudy	1
26	40	25	Rainy	0

The model is then trained on this dataset for classifying the pump status in binary values(0/1).

5.2 Model Architecture

Model Type: Multilayer Perceptron (MLP) Classifier

An MLP is a type of feedforward artificial neural network that consists of multiple layers of nodes (neurons), each connected to the next layer. It is a supervised learning algorithm used for classification and regression tasks.

The model architecture includes an input layer, one or more hidden layers, and an output layer.

```
# Step 1: Prepare the data
import pandas as pd
# Load the dataset
data = pd.read_csv('/content/iis-data.csv')
# Convert Weather Condition to categorical variables
data['Weather'] = pd.Categorical(data['Weather'])
data['Weather'] = data['Weather'].cat.codes
# Define features (X) and target (y)
X = data[['Moisture', 'Temperature', 'Weather']]
y = data['Pump']
# Step 2: Split the dataset into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Step 3: Define the model (Neural Network)
from sklearn.neural_network import MLPClassifier
# Define the neural network model
model = MLPClassifier(hidden_layer_sizes=(100, 50), activation='relu', solver='adam', alpha= 0.0001, learning_rate = 'constant' , random_state=42)
# Step 4: Train the model
model.fit(X_train, y_train)
# Step 5: Evaluate the model
accuracy = model.score(X_test, y_test)
print("Accuracy:", accuracy)
# Step 6: Predict pump status for custom input
```

Training Algorithm:

The model uses the Adam optimizer, which is an efficient gradient-based optimization algorithm. The loss function used is binary cross-entropy, suitable for binary classification problems. The model is trained iteratively using backpropagation, adjusting the weights to minimize the loss function.

Hyperparameters:

Hidden Layer Sizes: (100, 50)

The model has two hidden layers with 100 and 50 neurons respectively.

Activation Function: ReLU (Rectified Linear Unit)

ReLU is used as the activation function in the hidden layers, allowing the model to learn complex patterns in the data.

Learning Rate: Constant

The learning rate determines the step size taken during optimization.

Regularization: L2 (alpha=0.0001)

L2 regularization is applied to prevent overfitting by penalizing large weight values.

Evaluation:

The model's performance is evaluated using accuracy, which measures the proportion of correctly classified instances in the test dataset.

Accuracy is calculated by comparing the model's predictions with the true labels in the test dataset.

Integration with flutter :

The integration of TensorFlow with Flutter enables the deployment of machine learning models directly within mobile applications developed using the Flutter framework.

```
import tensorflow as tf
# Convert scikit-learn MLPClassifier to TensorFlow model
tf_model = tf.keras.Sequential([
    tf.keras.layers.Input(shape=(3,)),  # Input layer with 3 features
    tf.keras.layers.Dense(100, activation='relu'),  # Hidden layer 1
    tf.keras.layers.Dense(50, activation='relu'),  # Hidden layer 2
    tf.keras.layers.Dense(1, activation='sigmoid')  # Output layer
])
# Compile the model
tf_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
# Transfer weights from scikit-learn model to TensorFlow model
for i, layer in enumerate(model.coefs_):
    tf_model.layers[i].set_weights([layer, model.intercepts_[i]])
# Save the TensorFlow model
tf_model.save("pump_model_tf")
converter = tf.lite.TFLiteConverter.from_saved_model('pump_model_tf')
converter.optimizations = [tf.lite.Optimize.DEFAULT]
tflite_model = converter.convert()
# Save the converted model to a file
with open('pump_model.tflite', 'wb') as f:
    f.write(tflite_model)
```

5.3 Model Development

TensorFlow is used to develop machine learning models, including neural networks, decision trees, and other algorithms, for various tasks such as classification, regression, and clustering.

Models are trained using TensorFlow's high-level APIs or custom implementations, leveraging its extensive set of tools and libraries for building and training models.

TensorFlow Lite:

TensorFlow Lite is a lightweight version of TensorFlow designed for mobile and embedded devices.

Models trained with TensorFlow are converted to TensorFlow Lite format, which is optimized for efficient execution on mobile devices with limited computational resources.

TensorFlow Model Creation:

A TensorFlow Sequential model is created using tf.keras.Sequential.

The model architecture consists of an input layer with 3 features, followed by two hidden layers with 100 and 50 neurons, respectively, and an output layer with a single neuron and sigmoid activation function.

Model Compilation: The TensorFlow model is compiled using compile(), specifying the optimizer ('adam') and loss function ('binary_crossentropy') for binary classification. Metrics for evaluation, such as accuracy, are also specified.

Transfer Weights: The weights and biases of the scikit-learn MLPClassifier model are transferred to the corresponding layers of the TensorFlow model. This

ensures that the TensorFlow model inherits the learned parameters from the scikit-learn model.

Model Saving: The TensorFlow model is saved in the SavedModel format using tf_model.save().

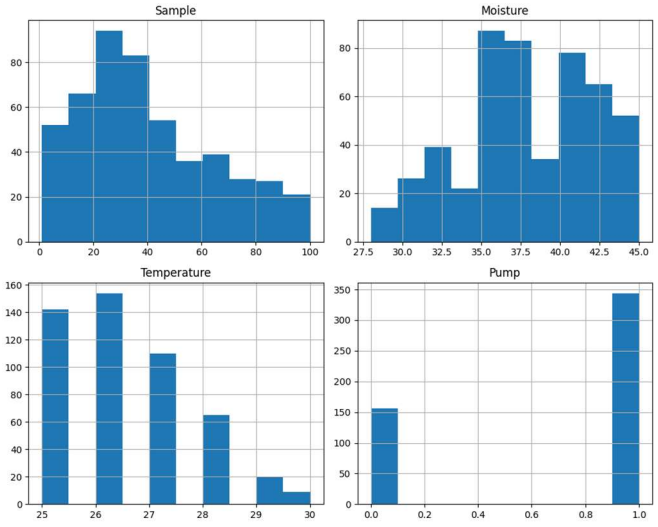
Conversion to TensorFlow Lite: The TensorFlow SavedModel is converted to TensorFlow Lite format using tf.lite.TFLiteConverter.from_saved_model(). Optimization options are applied to improve the efficiency and performance of the converted model.

Saving the Converted Model: The converted TensorFlow Lite model is saved to a file ('pump_model.tflite') using the write() method.

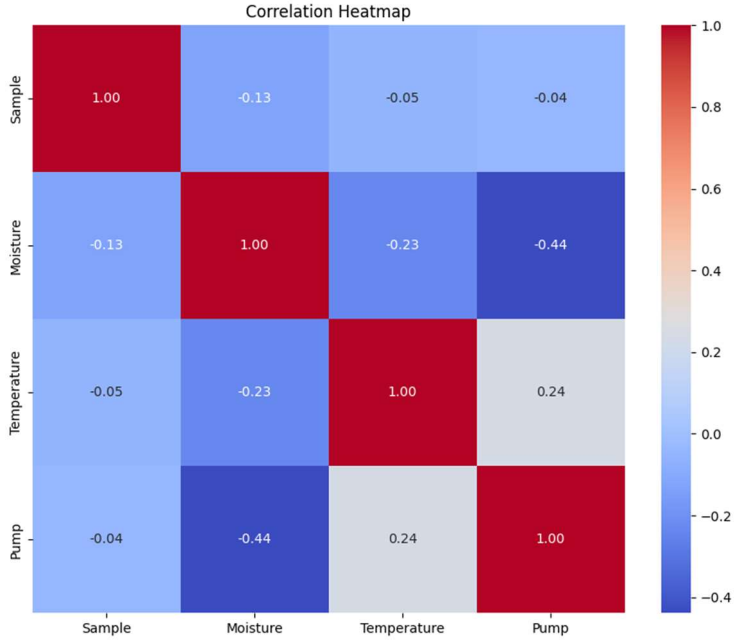
Overall, the integration of TensorFlow with Flutter empowers developers to build powerful mobile applications that leverage machine learning for tasks such as image recognition, natural language processing, predictive analytics, and more, enhancing the user experience and functionality of the app.

5.4 Analysis & Plots

Histograms:



Heatmap:


Correlation Heatmap
Sample
Moisture
Temperature
Pump
1.00
-0.13
-0.05
-0.04
-0.13
1.00
-0.23
-0.44
-0.05
-0.23
1.00
0.24
-0.04
-0.44
0.24
1.00
Sample
Moisture
Temperature
Pump
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

Chapter-6: IMPLEMENTATION

The assembly and testing phase of our smart farming system involve physically integrating the hardware components and validating their functionality. Firstly, we connect the moisture sensors to the ESP32 microcontroller, ensuring proper wiring and placement in the soil for accurate readings. The relay modules are then connected to the ESP32 to control the water pumping motor based on moisture sensor data.

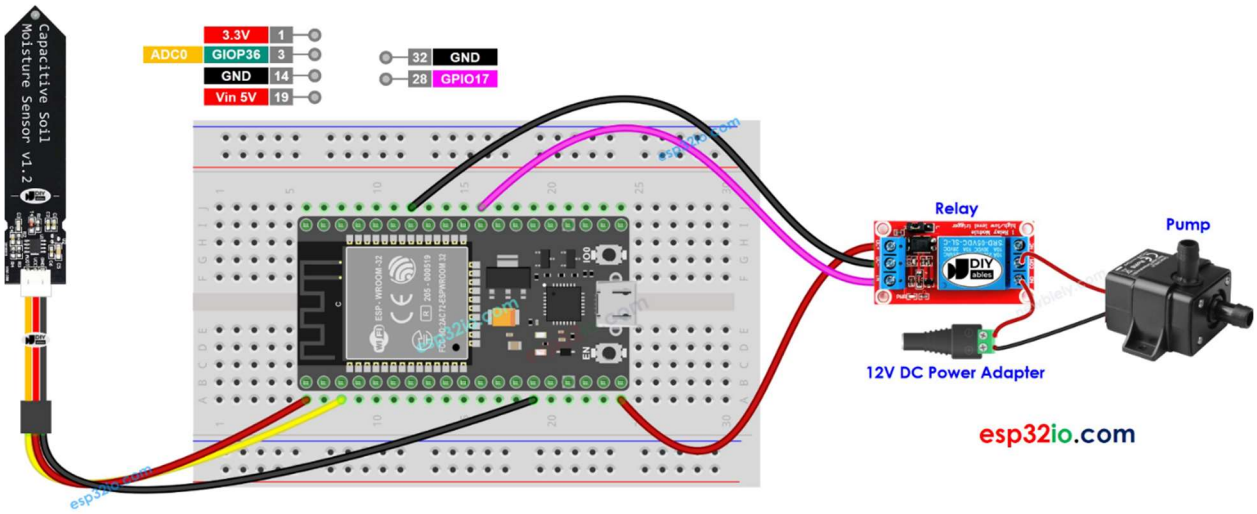
Once the hardware setup is complete, we proceed to test each component individually to verify its operation. We check the moisture sensors by placing them in different soil conditions and observing the readings obtained. Similarly, we test the relay modules by activating them to control the water pumping motor and verify that it operates as expected.

After confirming the functionality of each hardware component independently, we integrate them into the overall system architecture. This involves connecting the ESP32 microcontroller to the mobile application via an IoT connection, allowing for remote monitoring and control of the irrigation system.

Throughout the testing phase, we conduct various scenarios to ensure the robustness and reliability of the system under different conditions. We simulate changes in soil moisture levels, weather conditions, and user inputs to validate the responsiveness and accuracy of the smart farming system.

6.1 Circuit

In our circuit assembly, we connected the ESP32 microcontroller to the relay module, moisture sensors, and power supply to create an integrated smart farming system. The connections are as follows:



1. ESP32 Microcontroller:

- Connected to the relay module to control the water pumping motor.
- Interfaced with the moisture sensors to read soil moisture levels.
- Powered by the power supply unit to ensure continuous operation.

2. Relay Module:

- Connected to the ESP32 microcontroller to receive control signals for activating/deactivating the water pumping motor.
- Connected to the water pumping motor to control its operation based on moisture sensor readings.

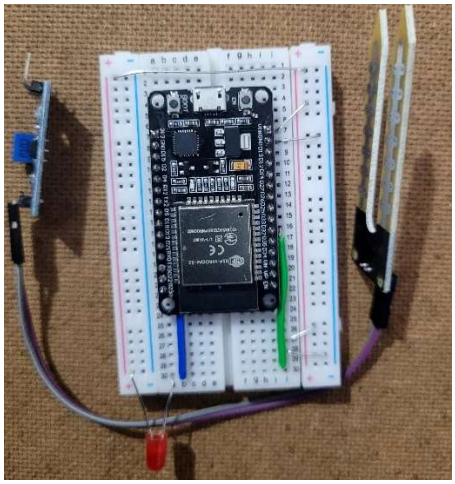
3. Moisture Sensors:

- Connected to the ESP32 microcontroller to provide soil moisture data.

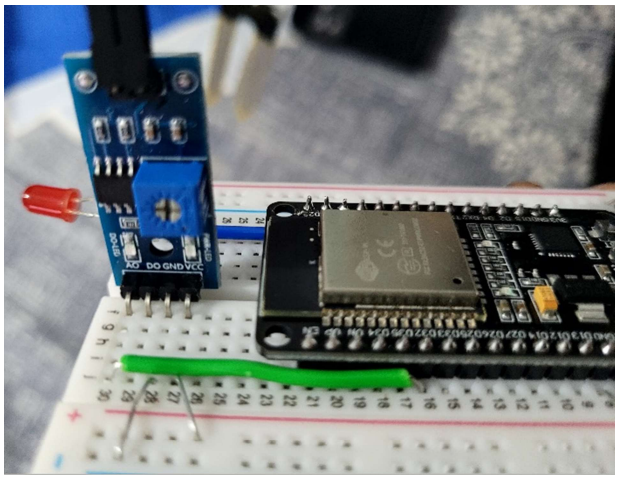
- Positioned in the soil to accurately measure moisture levels.

4. Power Supply:

- Connected to the ESP32 microcontroller, relay module, and moisture sensors to provide electrical power for their operation.
- Ensures stable and sufficient power delivery to all components of the smart farming system.



(Basic Circuit before connection)



(Moisture sensor connection)

By establishing these connections, we enable the ESP32 microcontroller to receive sensor data from the moisture sensors, make decisions based on pre-defined logic or machine learning models, and control the relay module to manage the water pumping motor accordingly. This integrated circuit forms the backbone of our smart farming solution, enabling automated irrigation based on real-time soil moisture measurements.

6.2 Connecting to Iot Sever

After making all the necessary connections of the components we connect the micro controller to our local computer through a data transfer cable and connect them to a single Wifi network. After a successful connection, the hardware components are connected to the Iot server and an IP address is provided which will be used in the application.

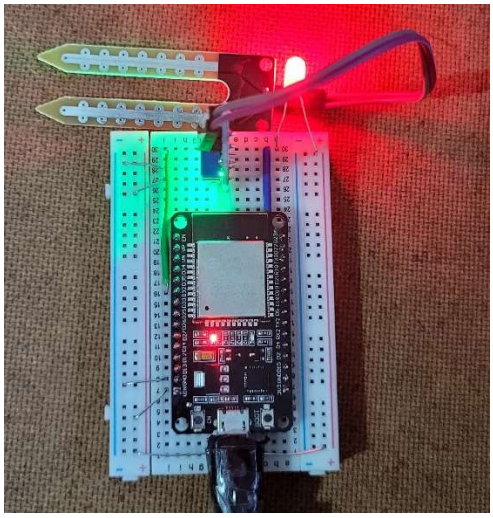
This connection helps in the real time data transfer of the moisture sensor to the application which is further used for analysis and decision making for the water pump.

```
17:36:43.953 -> Connecting to WiFi...
17:36:44.947 -> Connecting to WiFi...
17:36:45.942 -> Connecting to WiFi...
17:36:46.937 -> Connecting to WiFi...
17:36:46.937 -> Connected to WiFi. IP address: 192.168.1.7
17:36:46.970 -> Setup completed.
17:36:46.970 -> HTTP server started
17:43:49.099 -> Connecting to WiFi...
17:43:50.093 -> Connecting to WiFi...
17:43:51.089 -> Connecting to WiFi...
17:43:52.083 -> Connecting to WiFi...
17:43:53.078 -> Connecting to WiFi...
17:43:53.078 -> Connected to WiFi. IP address: 192.168.1.7
17:43:53.112 -> Setup completed.
17:43:53.112 -> HTTP server started
```

(Initializing the Iot Server)

```
17:46:53.235 -> 4095
17:46:53.235 -> Water Pump is ON
17:46:54.230 -> 4028
17:46:54.230 -> Water Pump is ON
17:46:55.258 -> 1347
17:46:55.258 -> Water Pump is OFF
17:46:56.253 -> 4095
17:46:56.253 -> Water Pump is ON
17:46:57.247 -> 4095
17:46:57.247 -> Water Pump is ON
```

(Pump Status after connection)



(Circuit after connecting to Iot server)

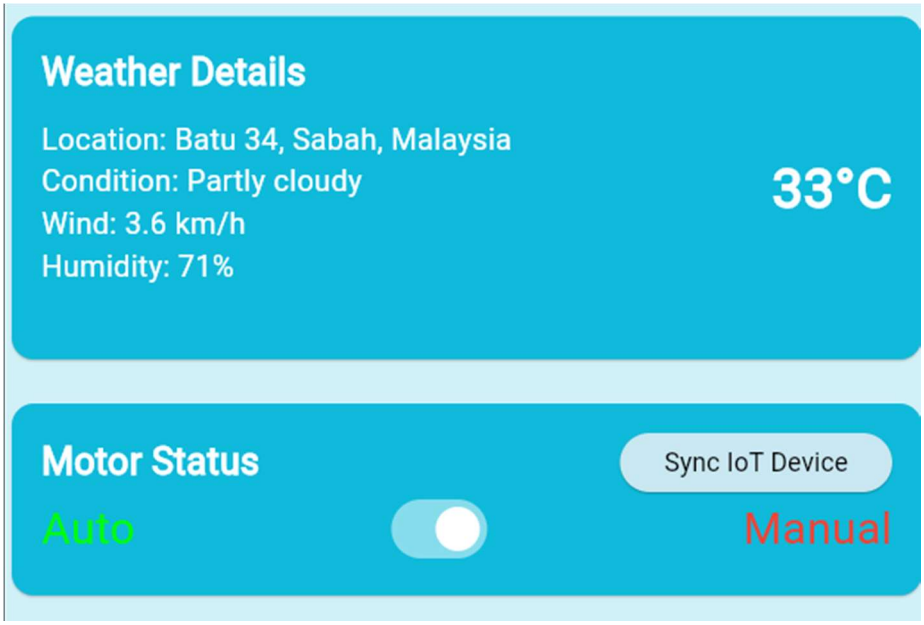
*The red LED light resembles the water pump, the bright red color of LED indicates the ON status of the pump while the same LED with light red color (not completely turned off) indicates the OFF status of pump.

6.3 Testing in Automatic Mode

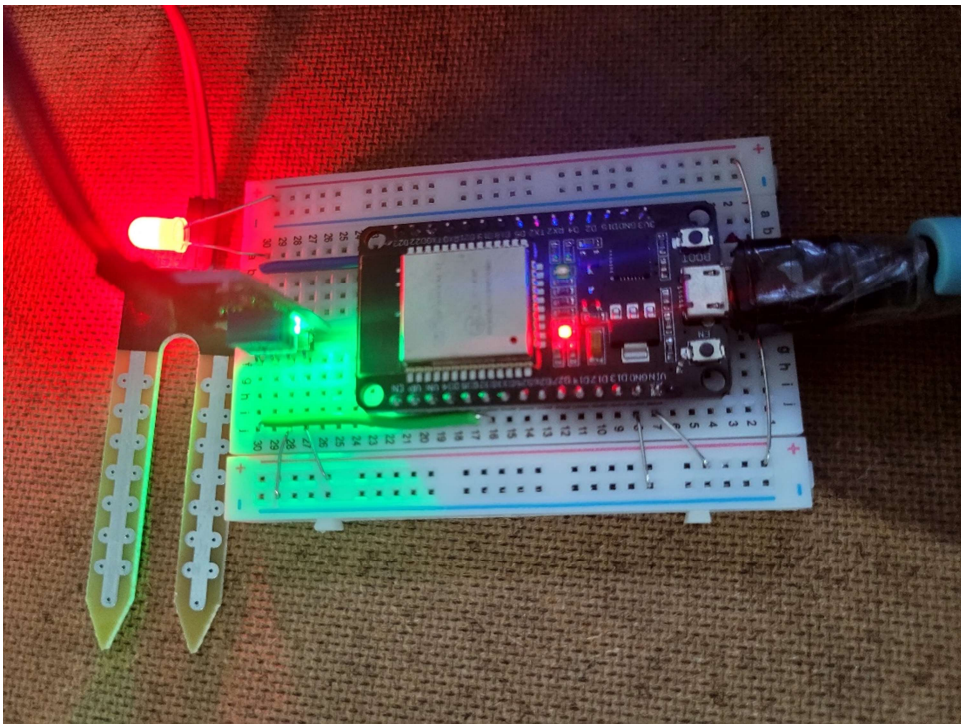
Here in automatic mode we consider the moisture readings of the soil and then based on these readings and the other information like, temperature and weather we make the decision of turning the motor ON or OFF.

As this is a demo model we implement and test this mode with a glass of water which has maximum moisture possible.

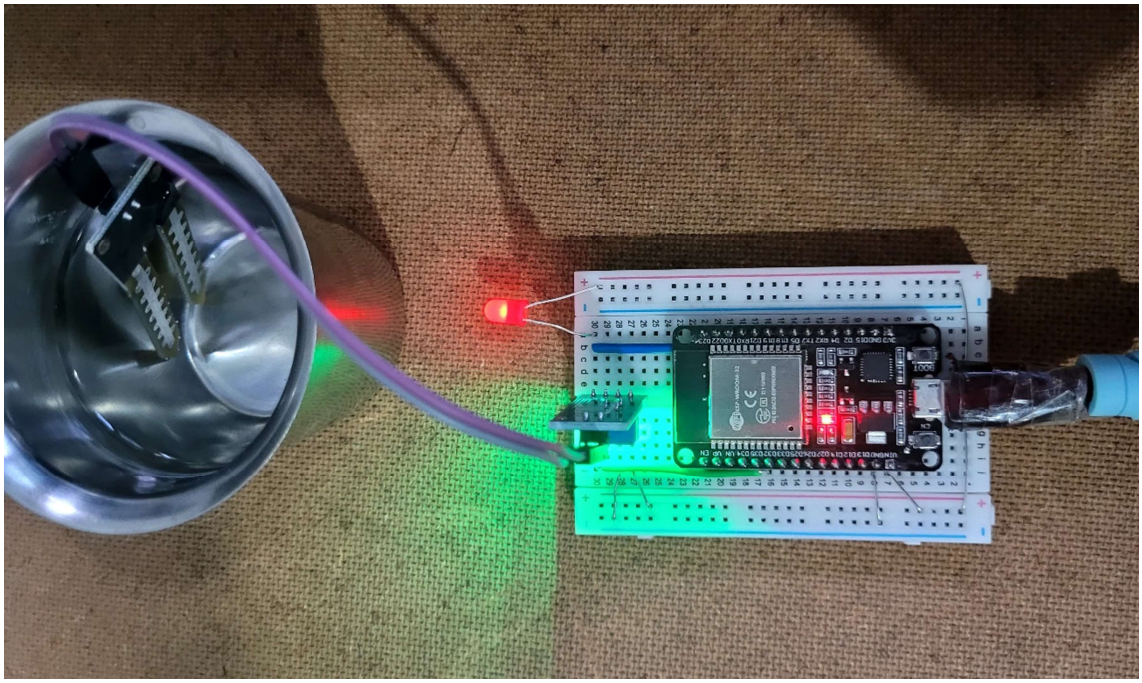
After all the necessary connections, we simply switch to the automatic mode in the application.



Test the output by placing the moisture sensor in the water and then removing it, the expected output should be the LED light being bright when the moisture sensor is removed out of the water indicating the pump to turn ON and dim when sensor is placed in the water indicating the pump to be turned OFF.



(LED/Pump: ON as the sensor is not in the water- Indicates that moisture is less and the pump should be turned on.)

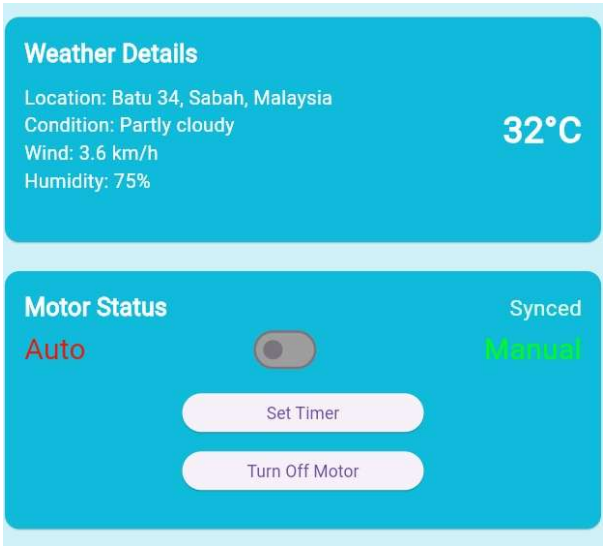


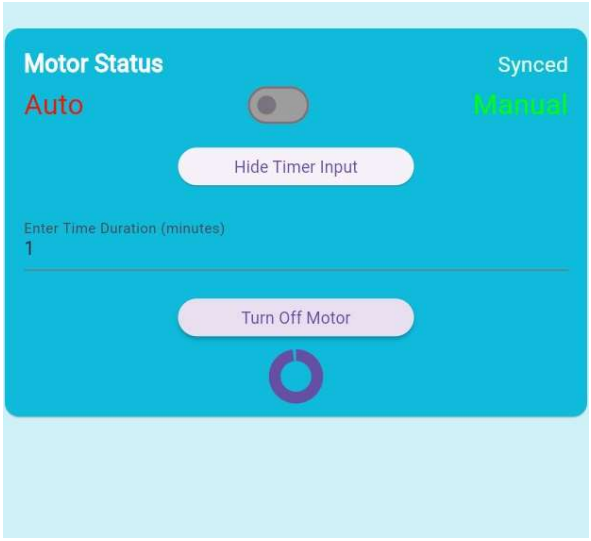
(LED/Pump: OFF as the sensor is placed in the water- Indicates that pump should be turned off.)

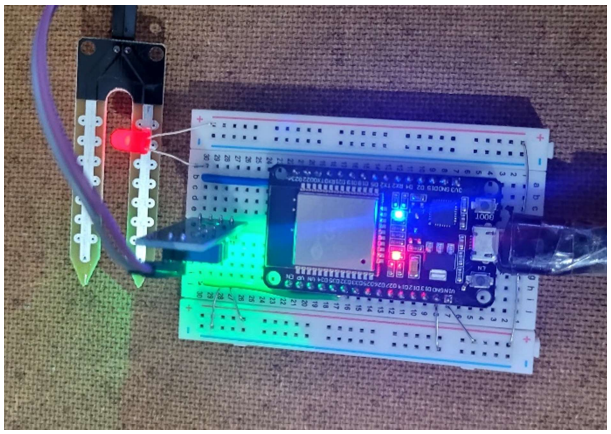
6.4 Testing in Manual Mode

Unlike the automatic mode, in this mode we do not need water for testing as the user will manually tell the system to turn the motor ON/OFF as they intend to.

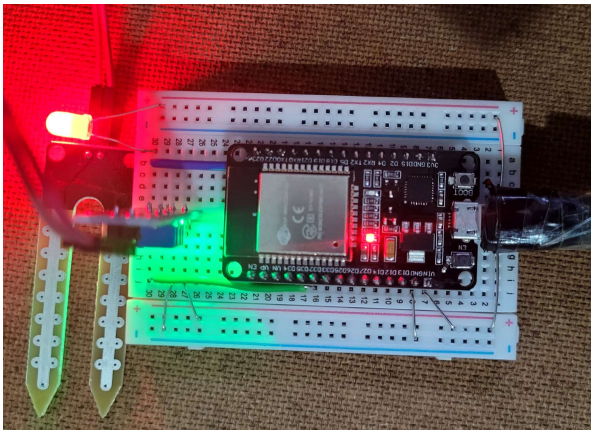
We can just click a button for controlling the motor and with the timer we can set the duration for which the motor should be ON.







(LED/Pump: OFF-Manually)



(LED/Pump: ON-Manually)

CONCLUSION

All things considered, our smart farming initiative is a big step toward using technology to transform conventional agricultural methods. We have created a system that not only monitors environmental conditions but also automates irrigation procedures and maximizes resource usage in crop production by seamlessly combining IoT, machine learning, and hardware components.

Although we are still in the demonstration phase of our project, there is still plenty we can do to better the mobile application that goes along with it and its overall user experience. Subsequent versions of the application may concentrate on optimizing the user interface, simplifying the navigation, and integrating more user-friendly controls to ensure smooth communication with the intelligent farming system. Personalized suggestions, real-time warnings, and remote monitoring capabilities are further features that would enable farmers to effectively manage their crops and make well-informed decisions.

Looking ahead, our project's future course entails a number of crucial elements. First and foremost, we want to expand the system's capacity to handle bigger farming operations and a greater variety of crop types. Furthermore, we want to include cutting-edge data analytics methods to extract valuable insights from the gathered agricultural data, facilitating farmers' decision-making and predictive modeling.

In addition to our technical endeavors, our passion for farming and innovation serves as a driving force behind this project. We are deeply committed to leveraging technology to address the challenges facing modern agriculture and to empower

farmers with the tools and knowledge needed to thrive in an increasingly complex and unpredictable environment.

In conclusion, our journey in developing this Intelligent Irrigation System has been both challenging and rewarding. As we look to the future, we remain committed to pushing the boundaries of what is possible in agriculture, driven by a shared vision of harnessing technology to create a more sustainable, efficient, and equitable food system. Together, let us cultivate a future where technology and agriculture work hand in hand to nourish our planet and empower those who feed the world.

REFERENCES

- https://ieeexplore.ieee.org/Xplore/home.jsp
- https://www.sciencedirect.com/
- https://pubmed.ncbi.nlm.nih.gov/
- https://scholar.google.com/
- https://www.researchgate.net/
- https://link.springer.com/
- https://onlinelibrary.wiley.com/
- https://www.tandfonline.com/