A PROJECT REPORT ON

REAL-TIME PRECISION FARMING USING AI ANALYTICS AND IOT SENSOR NETWORKS

Submitted in partial fulfillment of the requirements
for the award of the degree of
Bachelor of Technology
in
ELECTRONICS & COMMUNICATION ENGINEERING

Submitted by

P CHANUKYA	**321129512048**
A ANIL KUMAR REDDY	**321129512002**
K MAHESH	**321129512031**
P AKHIL KUMAR	**321129512047**

Under the esteemed guidance of

Mrs. VADDADI NIRMALA M. Tech

Assistant Professor

Department of ECE, WISTM Engineering College

DEPARTMENT OF ELECTRONICS &COMMUNICATION ENGINEERING
WELLFARE INSTITUTE OF SCIENCE TECHNOLOGY AND MANAGEMENT
Approved by the AICTE, New Delhi & affiliated to ANDHRA UNIVERSITY
Pinagadi (V), Pendurthi(M), Visakhapatnam-531 173

2021-2025

WELLFARE INSTITUTE OF SCIENCE TECHNOLOGY &MANAGEMENT

Approved by the AICTE, New Delhi & affiliated to ANDHRA UNIVERSITY

Pinagadi(V), Pendurthi(M), Visakhapatnam-531 173

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that this project entitled "**REAL-TIME PRECISION FARMING USING AI ANALYTICS AND IOT SENSORE NETWORKS**", *is a* bonafide record of the work done **P. CHANUKYA (321129512048), A. ANIL KUMAR REDDY (321129512002), K. MAHESH (321129512031) P. AKHIL KUMAR (321129512047)** *in the department of* **ELECTRONICS & COMMUNICATION ENGINEERING, WELLFARE INSTITUTE OF SCIENCE, TECHNOLOGY & MANAGEMENT ENGINEERING COLLEGE,VISAKHAPATNAM** *during the academic* year 2021-2025, in a partial fulfilment of the requirements for the Award of the degree of bachelor of engineering in Electronics &Communication Engineering.

INTERNAL GUIDE

Mrs. VADDADI NIRMALA, M. Tech

Assistant Professor

Dept. of ECE

WISTM, VISAKHAPATNAM

HEAD OF THE DEPARTMENT

Dr. ANANDBABU GOPATOTI, M. Tech, Ph.D

Professor, Head of the Department

Dept. of ECE

WISTM, VISAKHAPATNAM

EXTERNAL EXMANIER

DECLARATION

We are the student of 8th semester in the bachelor of technology in **ELECTRONICS AND COMMUNICATION ENGINEERING** OF **WELLFARE INSTITUTE OF SCIENCE TECHNOLOGY AND MANAGEMENT,** Visakhapatnam.

Hereby declare that the project work entitled "**REAL-TIME PRECISION FARMING USING AI ANALYTICS AND IOT SENSORE NETWORKS**" submitted to **WELLFARE INSTITUTE OF SCIENCE TECHNOLOGY AND MANAGEMENT**, affiliated to **ANDHRA UNIVERSITY**,
in partial fulfillment for the award of B.Tech degree in **ELECTRONICS & COMMUNICATION ENGINEERING.**

It is a record of an original project work done by our team, under the guidance of **Mrs.VADDADI NIRMALA M.Tech** Assistant professor. And was not a duplication of work done by someone else. We hold the responsibility of the originality of the work incorporated into the thesis.

It is record of the original project work done by our team, under the supervision and the support of **Dr. ANANDBABU GOPATOTI .M.Tech,Ph.D** Professor, Head of the Department of **ELECTRONICS & COMMUNICATION ENGINEERING, WELLFARE INSTITUTE OF SCIENCE TECHNOLOGY & MANAGEMENT,** Visakhapatnam

PROJECT ASSOCIATIVES

P CHANUKYA	**321129512048**
A ANIL KUMAR REDDY	**321129512002**
K MAHESH	**321129512031**
P AKHIL KUMAR	**321129512047**

ACKNOWELEDGEMENT

A Project is a golden opportunity for learning and self-development. We consider ourselves blessed and privileged to express our gratefulness and deep sense of gratitude and to our guide of **Mrs.VADDADI NIRMALA, M.Tech** Assistant Professor, Department of Electronics & Communication Engineering for her stimulating suggestions and encouragement that helped us at every stage of our project work, which made this project successful.

We would like to give special thank to our **Dr. ANANDBABU GOPATOTI**, **M.Tech Ph.D.** Professor, **Head Of The Department** of ELECTRONICS & COMMUNICATION ENGINEERING for the permissions to use all the required equipment and the necessary materials and his encouragement throughout the project.

We also express our grateful gratitude to **our Dr. JOSHUA ARUMBAKAN .**Principal, **WELLFARE INSTITUE OF SCIENCE, TECHNOLOGY & MANAGEMENT** for his support and encouragement throughout the project.

We would like to extend our sincere thanks to **Ms. MALLA ANUSHA**, Managing Director, Wellfare Group Of Companies for her encouragement and facilities provided during project.

We would also like to extend our sincere thanks to **Dr. MALLA VIJAY PRASAD**, Managing Director, Wellfare Group Of Companies for her encouragement and facilities provided during project

PROJECT ASSOCIATIVES

P CHANUKYA	**321129512048**
A ANIL KUMAR REDDY	**321129512002**
K MAHESH	**321129512031**
P AKHIL KUMAR	**321129512047**

ABSTRACT

This project introduces a smart farming system that uses artificial intelligence (AI) and machine learning (ML) to support more efficient and productive agricultural practices. By placing sensors like the soil moisture sensor and DHT22 in the field, the system constantly monitors temperature, humidity, and soil conditions in real-time. These readings help farmers better understand their crop environment and make more informed decisions about when to water, fertilize, or take protective measures, all while reducing manual effort. One of the key strengths of this system is its ability to predict upcoming weather changes and irrigation needs. By training machine learning models on historical weather data and live sensor inputs, the system can anticipate future conditions and recommend timely actions. To keep farmers informed, it also sends automated SMS alerts through the Twilio API, ensuring they receive important updates wherever they are, even if they don't have internet access.

The system includes a simple web dashboard where farmers can view live sensor data, manage settings, and receive AI-based suggestions. It's designed to be flexible and scalable, so it can be used with different crops and adapted for farms of various sizes and locations. Overall, the goal is to support sustainable agriculture by using data and technology to make smart, resource-efficient choices that improve crop health and yield while minimizing waste.

KEY WORDS

Smart Farming, **Precision Agriculture**, **Artificial Intelligence (AI)**, **Machine Learning (ML)**, Real-Time Monitoring, Soil Moisture Sensor, DHT22 Sensor, Predictive Analytics, Weather Forecasting, **Irrigation Management**, Twilio API, SMS Alerts, **IoT in Agriculture**

CONTENTS

LIST OF FIGURES

CHAPTER-1

INTRODUCTION

CHAPTER-1
INTRODUCTION

1.1 Objective

Agriculture is one of the most essential sectors supporting human life and economic growth worldwide. However, traditional agricultural methods face increasing challenges due to population growth, climate change, labor shortages, and inefficient use of resources. Farmers often struggle with unpredictable weather patterns, overuse of water, and a lack of real-time data, which affects crop yield and profitability. In many rural regions, there's still heavy reliance on manual labor and experience-based decisions, leading to suboptimal use of fertilizers, pesticides, and irrigation. To overcome these limitations, it is essential to implement modern, intelligent solutions that integrate automation, data analytics, and predictive systems to support more accurate and sustainable farming practices.

This project sets out to build a Smart Agriculture System that uses a combination of Internet of Things (IoT), Artificial Intelligence (AI), and Machine Learning (ML) to achieve precision farming. Sensors such as DHT22 and soil moisture detectors are deployed across the field to monitor key environmental conditions like temperature, humidity, and soil moisture levels in real time. The collected data is processed by trained ML models to predict future irrigation needs, identify environmental trends, and recommend timely actions. To improve safety and responsiveness, Twilio's SMS gateway is used to send automatic alerts to farmers, notifying them about critical changes in the environment, optimal irrigation timing, and weather forecasts. The system is built on low-cost microcontrollers such as NodeMCU (ESP8266), making it both economical and suitable for rural and large-scale deployment.

Furthermore, a web-based dashboard is integrated into the system to provide users with a simple interface to visualize field data, remotely control irrigation or other devices, and receive smart recommendations. The architecture is modular, scalable, and crop-independent—meaning it can be customized based on the type of crop, geographical location, and farm size. Additional features such as AI-based crop health diagnosis, real-time alerts for pests or diseases, and cloud data storage can be incorporated for future expansion. The ultimate goal of this system is to reduce resource wastage, increase crop

yield, minimize labor dependency, and promote environmentally sustainable agricultural practices. By empowering farmers with data and automation, this project envisions a future where technology and farming go hand in hand to feed the growing global population.

MOTIVATION FOR PROJECT

The motivation behind developing a Smart Agriculture System for Precision Farming is to address the pressing challenges faced by the agricultural sector, particularly in regions where traditional farming methods are still dominant. With the global population on the rise and natural resources like water and fertile soil becoming increasingly scarce, there is an urgent need to optimize agricultural practices. Many farmers, especially in rural and developing areas, lack access to timely information about weather patterns, soil health, and irrigation requirements, which leads to poor crop yields, resource wastage, and economic instability.

By leveraging the power of IoT sensors and artificial intelligence, this system aims to bring modern technological solutions directly to the fields. Real-time data from sensors such as DHT22 for temperature and humidity, and soil moisture sensors, can help farmers make informed decisions about irrigation, fertilization, and crop care. Additionally, by using machine learning algorithms to analyze this data, the system can predict environmental conditions and suggest optimal farming practices tailored to specific crops and soil types. This not only reduces guesswork but also maximizes productivity while conserving resources.

Furthermore, the integration of SMS alerts through platforms like Twilio ensures that farmers receive timely updates and recommendations, even if they are away from their farms or lack access to the internet. A user-friendly web dashboard complements this system by offering remote monitoring and control features. Overall, the motivation behind this project is to empower farmers with intelligent tools that promote sustainable agriculture, increase crop yields, reduce manual workload, and improve overall farm management—ultimately contributing to food security and environmental conservation.

IMPLEMENTATION OF THE PROJECT:

The implementation of a Smart Agriculture System using IoT, AI, and machine learning involves a structured and multi-phase approach, aimed at improving crop yield, optimizing resource usage, and ensuring sustainable farming practices. The system primarily depends on real-time environmental data acquisition, intelligent analysis, remote access, and proactive decision-making support.

Step-by-Step Implementation

1. **Sensor Deployment and Data Acquisition**
    - **Soil Moisture Sensors** are installed in the farmland to continuously monitor soil water levels. This data helps determine the optimal irrigation schedule.
    - **DHT22 Sensors** are deployed to monitor **temperature** and **humidity** levels, which directly influence plant health and pest activity.
    - **Light intensity sensors (optional)** can be added to monitor sunlight exposure and adjust greenhouse covers or suggest suitable planting times.
    - These sensors are connected to a **microcontroller unit** (such as NodeMCU or Arduino), which acts as the local data collector.
2. **Data Transmission and Cloud Integration**
    - The sensor data is transmitted to a **cloud server** via Wi-Fi using NodeMCU/ESP32 modules.
    - **Firebase** or other IoT platforms (e.g., ThingSpeak, AWS IoT, or Blynk) are used for storing, processing, and retrieving the sensor data.
    - The system operates in near real-time, ensuring that any abnormality or requirement (like low moisture) is flagged immediately.
3. **Machine Learning and Predictive Analytics**
    - Collected data is analyzed using **trained ML models** to predict:
        - **Weather conditions**
        - **Irrigation requirements**
        - **Crop health risks**
    - These models are trained on historical weather, soil, and crop yield datasets to forecast patterns and recommend actions.

4. **User Notification and Automation**
    - When a threshold is crossed (e.g., low moisture level), an **SMS alert is triggered via the Twilio API** to inform the farmer about the need for irrigation.
    - If automation is included, the system can also **automatically turn on the irrigation motor** using a relay switch.
    - Alerts can also include **fertilizer recommendations** based on soil data and crop stage.
5. **Power Management and Scalability**
    - The entire system is powered using **solar panels or battery-powered modules** to ensure continuous operation in remote fields.
    - The modular nature of the system allows adding more sensors or covering larger fields by simply replicating sensor units and linking them to the same cloud backend.

1.1.1 Opensource internet of things platforms

There are two possible ways for developing or concretely using an IoT system dedicated to smart agriculture: acquiring a technology developed ad hoc or tapping into the countless opportunities that the open source world offers.

While the first way, certainly more professional, involves major investments and therefore it is a prerogative of medium and large companies that can afford them, the second is an option that even small companies with staff with a strong aptitude for “Do It Yourself” could activate.

The three blocks such as measurement, decision and actuation can be as simple as the use case just described, but they can also be very complex. The measurement block might monitor multispectral or hyperspectral data. It might work on single points or *spatially resolved data*. Typically, as reported in the previous paragraph, these data never reach the decision-maker alone, but they are 'conditioned' by a series of other data (metadata) that characterize them (e.g., georeferencing, technical characteristics of measuring instruments, type of calibration, sensor behaviour curves, status information, etc.).

The DSS might make simple decisions (e.g., activate the solenoid valve if the humidity is below a threshold), might implement complex decision algorithms, might use real-time data

and historical data, or might access to local data or data provided by web services (e.g., weather data, historical data, statistical data, etc.).

1.1.2 Evolution of Sensor Nodes in Smart Agriculture

The history of sensing technologies in agriculture dates back to basic observational tools used by farmers to track weather patterns and soil conditions. Over time, these manual techniques have evolved into sophisticated sensor networks that provide precise, real-time data critical for improving yield, optimizing resource usage, and ensuring sustainable farming practices.

Early Developments in Agricultural Sensor Networks

The initial application of sensors in agriculture focused on remote weather stations and soil moisture meters, which provided limited but essential information. These standalone sensors were often manually read and lacked integration with decision-support systems. The transformation began with the advent of wireless communication technologies, enabling multiple sensors to form networks and relay data to centralized systems.

Rise of Wireless Sensor Networks (WSNs) in Farming

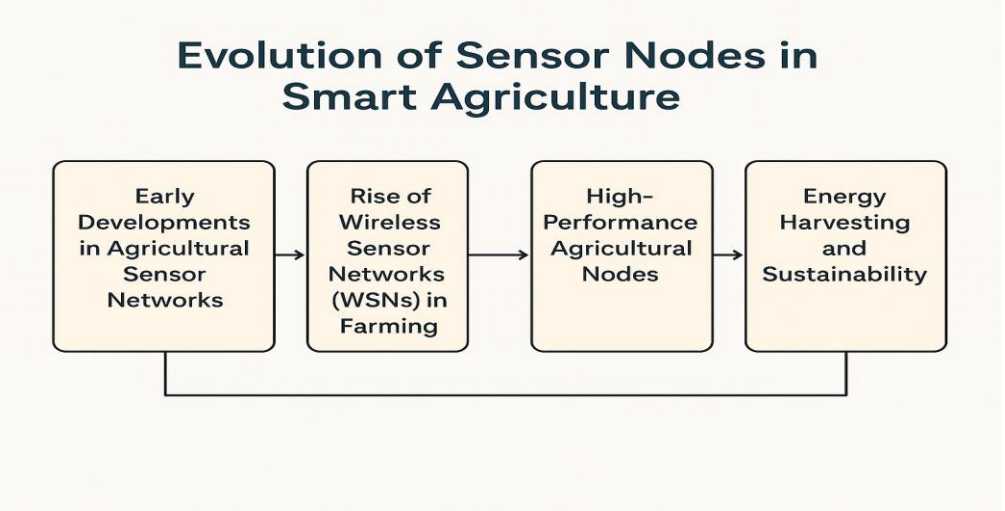
In the early 2000s, **Low Power Wireless Integrated Micro Systems (LWIM)** emerged, supporting short-range communication at low power, ideal for agricultural fields. These were soon followed by **Wireless Integrated Network Sensors (WINS)** that combined sensing, computation, and wireless communication capabilities. Such devices allowed farmers to monitor environmental factors like soil temperature, humidity, moisture, and light intensity across large fields.

The introduction of **motes** from UC Berkeley, such as **Mica, Mica2, and MicaZ**, marked a significant milestone. These devices used microcontrollers like the **ATmega128L**, featuring low-power consumption and radio modules supporting ZigBee protocols. Motes could be deployed throughout farms to gather granular data, enabling variable-rate irrigation, fertigation, and pest control — the core of precision farming..

1.1.3 Modern Agricultural Sensor Platforms

Today, sensor platforms include commercial products like **Ember ZigBee modules**, **Pluto motes**, and **LoRa-based IoT nodes** designed specifically for agriculture. These platforms support long-range communication, are weather-resistant, and are capable of connecting to cloud-based systems for real-time analytics. Integrated with AI and machine learning, they help farmers forecast yields, detect anomalies, and automate responses to climatic or soil changes.

From soil probes and DHT22 humidity sensors to drone-mounted multispectral cameras, the evolution of sensor nodes continues to revolutionize agriculture, making farming more efficient, data-driven, and environmentally friendly.



1.2 ADVANTAGES OF PROJECT

The Smart Agriculture System for Precision Farming is a revolutionary approach to modern-day farming that leverages advanced technologies such as Artificial Intelligence (AI), Machine Learning (ML), and the Internet of Things (IoT). The primary advantage of this system lies in its ability to transform conventional farming into a highly data-driven, automated, and sustainable process, addressing many of the challenges that traditional farmers face today.

One of the core benefits of this system is the significant improvement in crop yield. By utilizing sensors like the DHT22 and soil moisture probes, the system can accurately monitor and analyze critical environmental factors such as temperature, humidity, and soil conditions. This real-time data allows the system to optimize irrigation schedules, suggest ideal farming

practices, and ensure that crops are grown in optimal conditions. The integration of machine learning models further enhances this process by predicting irrigation needs and weather patterns, helping farmers make smarter decisions and plan their activities more effectively.

Another key advantage is the efficient use of essential farming resources. The system eliminates guesswork by providing precise data, which helps in reducing the excessive use of water, fertilizers, and pesticides. This not only minimizes costs for the farmer but also contributes to eco-friendly and sustainable farming by lowering the environmental footprint. By automating processes and reducing human error, the system ensures consistency and accuracy in farm operations.

Additionally, the system enhances the responsiveness of farmers to changing agricultural conditions. With real-time alerts sent via the Twilio API, farmers are instantly notified about critical changes in the field, such as unexpected weather conditions, temperature fluctuations, or low soil moisture. This level of responsiveness ensures that timely actions can be taken to avoid crop damage and losses.

The system also includes a user-friendly web dashboard, which serves as a centralized platform for farmers to access all the data, control devices remotely, and receive AI-driven insights and recommendations. This feature is particularly beneficial for farmers managing large or multiple farms, as it reduces the need for physical presence in the field. It allows remote monitoring and control through any internet-connected device, bringing farming operations into the digital age.

Furthermore, the project is designed with scalability and adaptability in mind. Its modular structure allows it to be easily expanded or modified to suit different crop types, climates, and farm sizes. Whether it’s a small garden or a large agricultural enterprise, the system can be tailored to meet specific needs without major redesigns.

In conclusion, the Smart Agriculture System provides a comprehensive, intelligent, and sustainable solution for the future of farming. By increasing productivity, optimizing resource usage, and enabling informed decision-making, it empowers farmers to meet growing food demands while preserving the environment. Its practical benefits, cost-effectiveness, and potential to revolutionize agriculture make it an invaluable asset in the transition to smart, precision-based farming practices.

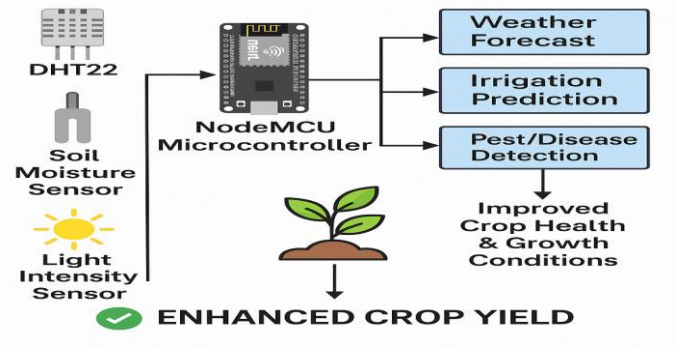
1.2.1 Enhanced Crop Yield

Enhanced crop yield is one of the most significant outcomes of adopting smart agriculture technologies. In traditional farming, crop productivity often suffers due to unpredictable weather, irregular irrigation, inconsistent soil quality monitoring, and lack of timely interventions. The Smart Agriculture System addresses these issues by integrating IoT sensors, AI-based analytics, and automated control systems to create an optimized growing environment for crops.

The system uses sensors like DHT22 and soil moisture detectors to gather real-time data about environmental parameters such as temperature, humidity, and soil moisture levels. This continuous stream of data is processed by machine learning algorithms that assess crop needs and growth patterns. Based on this analysis, the system provides recommendations and can even automate tasks like irrigation or send alerts to the farmer via SMS when attention is needed.

Over time, historical data collected from the farm can be used to improve predictions and fine-tune farming practices for even better outcomes in future growing seasons.

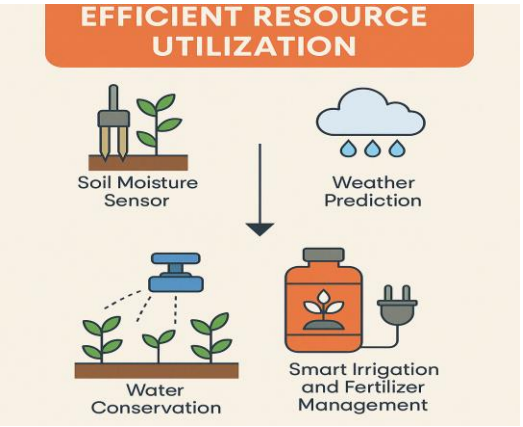
1.2.2 Efficient Resource Utilization



Efficient resource utilization is one of the key benefits of implementing smart agriculture and precision farming techniques. By integrating IoT-based sensors, machine learning algorithms, and data analytics, farmers can optimize the use of critical resources such as water, fertilizers, pesticides, and energy. This targeted approach reduces waste, saves costs, and minimizes the environmental impact of farming activities.

For example, soil moisture sensors and weather prediction models help determine the precise amount of water needed for crops, preventing over-irrigation and conserving water. Similarly, smart nutrient management systems analyze soil data and plant health to suggest accurate fertilizer quantities and application timings. This prevents nutrient runoff and ensures that crops get exactly what they need for healthy growth.

Additionally, automated irrigation systems and energy-efficient equipment reduce manual labor and power consumption, making the overall process more sustainable. As a result, smart farming promotes responsible stewardship of natural resources, while enhancing productivity and supporting long-term agricultural sustainability.



1.2.3 Real-Time Monitoring & Alerts

Real-time monitoring and alerts are critical components of smart agriculture systems, enabling farmers to maintain constant oversight of their crops and environmental conditions. By using IoT-based sensors—such as soil moisture sensors, DHT22 temperature-humidity sensors, and light sensors—data is continuously collected from the field. This data is processed and analyzed on cloud-based platforms or local servers to track vital parameters that affect crop health and yield.

When any parameter exceeds the optimal range—for instance, when soil moisture drops below a certain threshold or temperatures become too high—automated alerts are generated and sent to the farmer via SMS, email, or mobile apps. These notifications ensure quick responses to

potential threats such as drought stress, pest outbreaks, or extreme weather conditions, helping prevent crop loss and ensure timely interventions.

1.2.4 Remote Accessibility

Remote accessibility is one of the most powerful features of modern smart agriculture systems. It allows farmers and agricultural managers to monitor, manage, and control farm operations from virtually anywhere using internet-connected devices like smartphones, tablets, or computers. Through a centralized web dashboard or mobile application, users can access real-time data from sensors installed in the field, including information on soil moisture, temperature, humidity, and plant health.

This capability not only saves time and reduces the need for constant physical presence on the field but also enables faster responses to any irregularities or issues detected by the system. For example, if the soil moisture drops below a critical threshold, farmers can be alerted immediately and even activate irrigation systems remotely through automated controls.

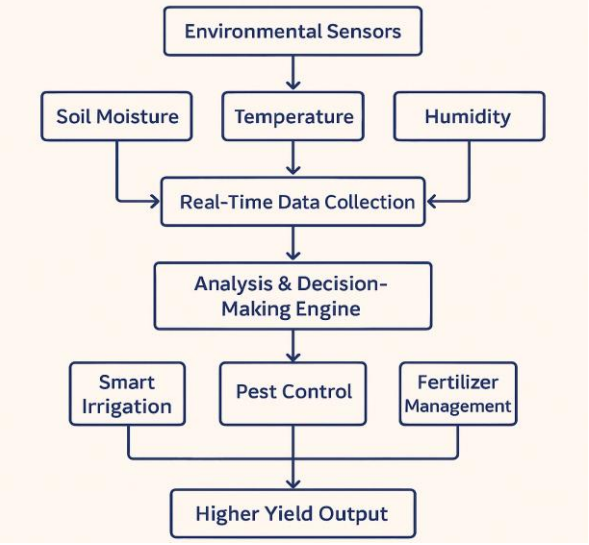
Remote accessibility is especially valuable for large-scale farms, farms located in remote regions, or for farmers managing multiple plots of land. It enhances operational efficiency, reduces labor costs, and provides greater flexibility and control over agricultural activities, all contributing to improved productivity and better resource management.

1.3 Precision Crop Management

Precision Crop Management is a key component of modern smart farming techniques that focuses on managing crops more accurately and efficiently using data-driven insights. In the Smart Agriculture System, precision crop management is achieved through the integration of IoT-based sensors, data analytics, and automated control mechanisms that enable real-time monitoring and management of crops at a micro level.

Through the deployment of environmental sensors such as DHT22 (for temperature and humidity), soil moisture sensors, and light intensity sensors, the system gathers detailed data about the current growing conditions of each field section. This granular data allows farmers to make more informed decisions regarding irrigation, fertilization, pest control, and harvesting schedules.

For instance, if the system detects that a specific area of the field has lower soil moisture levels, it can trigger irrigation only in that region rather than watering the entire field uniformly. Similarly, temperature and humidity data can help in predicting disease outbreaks and taking preventive measures in time.



1.3.1 Real-Time Alerts and Notifications

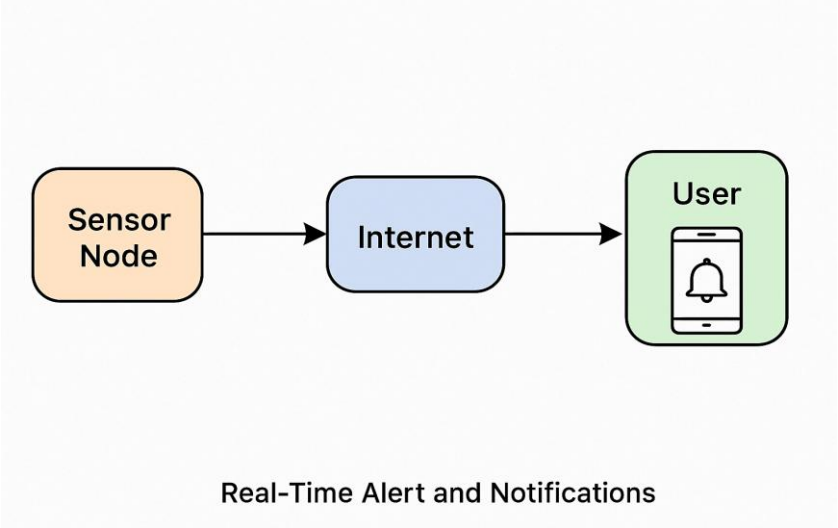
In the context of smart agriculture and precision farming, real-time alerts and notifications serve as a core feature that enables timely, data-driven decision-making. By integrating sensors, IoT devices, data analytics, and communication systems, the farming process becomes more

responsive to dynamic field conditions. This module continuously monitors environmental, soil, and crop parameters and delivers instant alerts to the farmer's device when thresholds are crossed or anomalies are detected.

The implementation of this system drastically reduces the time between issue detection and corrective action. For instance, if a drop in soil moisture is detected, the farmer is immediately notified to irrigate the field, thereby preventing crop stress. Similarly, predictive alerts about potential disease outbreaks—based on environmental conditions and sensor data—enable preventive treatment, minimizing yield loss.

The alert system is supported by a cloud-based backend that aggregates sensor data and applies machine learning algorithms to detect patterns and forecast potential risks.

Figure 1: Block Diagram



CHAPTER-2
LITERATURE SURVEY

CHAPTER-2

LITERATURE SURVEY

Smart agriculture, powered by precision farming technologies, has become a focal point of research in recent years due to the increasing demand for sustainable food production. With global climate change, unpredictable weather patterns, and limited resources, traditional farming methods are proving inefficient. To address these challenges, researchers and technologists have explored the integration of Internet of Things (IoT), Artificial Intelligence (AI), and embedded systems to create intelligent, real-time agricultural monitoring and decision-making platforms.

The ESP32 microcontroller has emerged as a popular platform in smart agriculture due to its low cost, Wi-Fi/Bluetooth connectivity, and energy efficiency. It allows for seamless integration with cloud platforms and mobile apps, enabling farmers to monitor and control their farms remotely. Research by Sharma et al. (2022) illustrated the use of ESP32 with DHT11 and capacitive soil moisture sensors to create an autonomous farm monitoring system with a mobile-based dashboard.

Weather API integration is another critical component of modern smart farming solutions. By accessing real-time weather forecasts (temperature, rainfall, wind speed), smart systems can anticipate environmental changes and adapt farming operations accordingly. In a 2020 study by Mehta and Patel, an IoT-based system connected to weather APIs showed improved crop resilience by dynamically adjusting irrigation schedules based on upcoming weather conditions.

Artificial Intelligence and Machine Learning have further elevated the capabilities of smart agriculture. Algorithms trained on historical and real-time data can predict crop diseases, recommend fertilization schedules, and even estimate yields. For example, a research paper published in *Elsevier's Computers and Electronics in Agriculture* journal (2021) discussed a

convolutional neural network (CNN) model for identifying leaf diseases with over 92% accuracy, enabling early intervention and reducing crop loss.

Precision farming also benefits from data visualization platforms and mobile applications, which allow farmers to interact with the system intuitively. Dashboards built with platforms like ThingSpeak, Firebase, or custom Android apps offer live updates, alerts, and actionable insights to the user.

In conclusion, the literature supports the use of integrated smart systems in agriculture to increase efficiency, conserve resources, and ensure better crop management. This project builds upon these established concepts by deploying a smart agriculture system using ESP32, multiple sensors, AI-based data analysis, and real-time weather forecasting to offer an adaptive and intelligent farming solution suitable for diverse crops and regions.

2.1 PROPOSED SYSTEM:

The proposed system aims to implement a smart agriculture solution using precision farming techniques, leveraging real-time data, IoT sensors, AI-based analysis, and cloud connectivity to optimize crop management and resource usage. The core objective is to develop an intelligent system that can automatically monitor environmental conditions, provide actionable insights, and control irrigation and fertilization processes with minimal human intervention.

This system utilizes the ESP32 microcontroller as the central processing unit due to its high performance, integrated Wi-Fi/Bluetooth capabilities, and compatibility with various sensors. It continuously gathers real-time data from a range of environmental sensors, including:

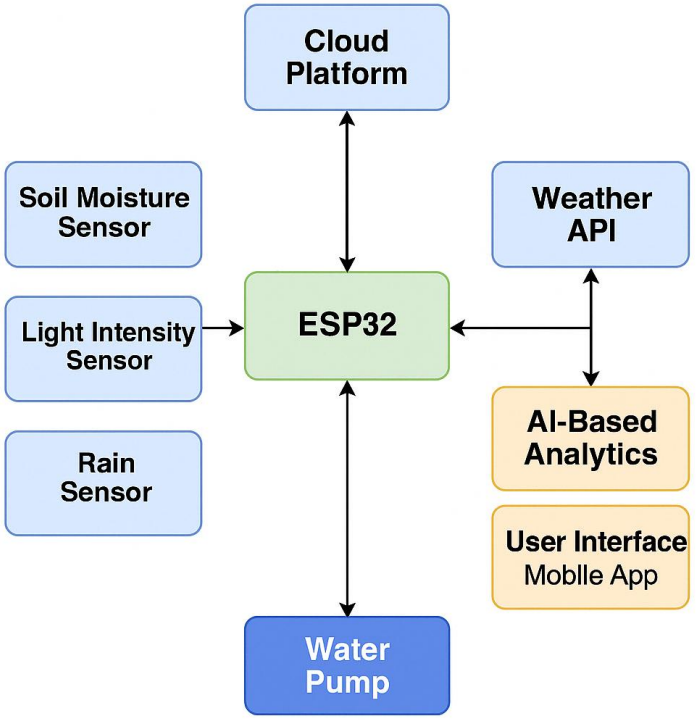
- **Soil Moisture Sensor** – for detecting the water content in the soil.
- **DHT11/DHT22 Sensor** – for measuring temperature and humidity.
- **Light Intensity Sensor (LDR)** – to monitor sunlight availability.
- **Rain Sensor** – to detect rainfall and prevent over-irrigation.

The collected data is processed locally and periodically uploaded to a cloud platform or mobile application via Wi-Fi. Simultaneously, the system fetches weather data through a Weather API to anticipate climatic changes and adapt decisions accordingly. For example, if rainfall is predicted, the system will automatically suspend irrigation to conserve water.

An AI algorithm embedded in the system or supported via cloud services analyzes the trends in sensor data over time to optimize irrigation schedules, forecast crop needs, and even detect potential threats such as disease onset or nutrient deficiency. Based on the data, the system can autonomously trigger actuators such as water pumps or solenoid valves to irrigate the crop field when soil moisture falls below a defined threshold.

The system also features a user interface, either via a web dashboard or mobile app, allowing farmers to monitor field conditions remotely, receive alerts, and override automation if needed. This remote accessibility ensures that users can make informed decisions anytime and from anywhere.

In summary, the proposed smart agriculture system provides a comprehensive, automated solution for precision farming. It supports adaptive irrigation, climate-responsive planning, and intelligent crop management. Designed to be scalable and cost-effective, the system is ideal for both small-scale farmers and large agricultural setups, with the flexibility to adapt to different crop types and regional conditions.



CHAPTER-3
ESP 32

CHAPTER-3
ESP 32

3.1 INTRODUCTION TO ESP 32

The ESP32 is a highly versatile and powerful microcontroller developed by Espressif Systems, specifically designed to address the growing needs of connected devices and Internet of Things (IoT) applications. As the successor to the widely adopted ESP8266, the ESP32 expands upon its capabilities by offering dual-core processing, integrated Wi-Fi and Bluetooth functionality, improved energy efficiency, and a broad array of peripheral interfaces. Due to its powerful performance, low power consumption, and cost-effectiveness, ESP32 has become the microcontroller of choice in numerous modern embedded applications—including smart homes, wearable devices, industrial automation, and notably, smart agriculture.

In the context of precision farming and smart agriculture, the ESP32 serves as the central control unit that manages sensor data collection, communication, local processing, and system automation. It plays a vital role in ensuring intelligent farm management by enabling real-time monitoring and automated control of irrigation, temperature regulation, humidity monitoring, and other key agricultural processes.

The ESP32 is powered by a dual-core Tensilica Xtensa LX6 processor with a clock speed of up to 240 MHz, which allows it to handle multiple tasks simultaneously, such as sensor data processing and wireless communication. It comes with 520 KB of SRAM, and most development boards include between 4MB to 16MB of flash memory, which is sufficient for running complex firmware, storing logs, and interfacing with cloud-based platforms. The ESP32's built-in Wi-Fi (802.11 b/g/n) and Bluetooth/Bluetooth Low Energy (BLE) features eliminate the need for additional network hardware, reducing system complexity and cost.

Another key aspect of the ESP32 in smart agriculture is its connectivity with cloud platforms and weather APIs. The microcontroller can fetch external environmental data such as rainfall forecasts, humidity trends, or UV index, and make intelligent decisions to adjust watering or nutrient delivery. Combined with machine learning or AI algorithms, it can also predict optimal watering times or detect anomalies in plant growth conditions.

In summary, the ESP32 is a compact yet highly capable microcontroller that brings together the essential features required for implementing a smart, connected, and autonomous agricultural solution. Its robust wireless communication, powerful processing capabilities, rich I/O support, and energy-efficient operation make it an indispensable component in the development of next-generation precision farming systems. With the ESP32 at its core, this project aims to bridge traditional agriculture with modern technology, enhancing productivity, resource management, and sustainability in farming practices.

3.1.1 History of the ESP32

The ESP32 is a powerful microcontroller developed by Espressif Systems, a Shanghai-based semiconductor company, and it represents a significant evolution in the world of Internet of Things (IoT) devices. It was introduced in 2016 as the successor to the highly popular ESP8266, which had gained widespread adoption due to its affordability and Wi-Fi capabilities. While the ESP8266 was limited in terms of processing power, memory, and functionality, the ESP32 addressed these shortcomings by incorporating a dual-core 32-bit processor, and adding both Wi-Fi and Bluetooth (classic and BLE) support, making it a far more versatile solution for modern IoT applications. This leap in capabilities opened the door for more complex projects, such as industrial automation, wearables, and precision agriculture, which required more processing power, memory, and connectivity options.

Alongside the hardware improvements, Espressif Systems also introduced a comprehensive development ecosystem, including the ESP-IDF (Espressif IoT Development Framework), and support for the Arduino IDE, which allowed both professional engineers and hobbyists to quickly develop applications for the ESP32. Over time, Espressif expanded the range of available variants, offering specialized models with different features to cater to various market needs, from low-power solutions to more feature-rich versions. The ESP32's combination of processing power, wireless connectivity, and scalability made it an ideal choice for a wide variety of applications, including home automation, sensor networks, and smart agriculture systems. As of today, the ESP32 continues to be one of the most popular microcontrollers used in the IoT space, widely appreciated for its cost-effectiveness, robust features, and developer-friendly ecosystem.

3.1.2 Why ESP32 for Smart Agriculture?

In the realm of **smart agriculture and precision farming**, the **ESP32 microcontroller** stands out as an essential enabler of automation, connectivity, and real-time decision-making. Acting as the **central brain** of the system, the ESP32 seamlessly integrates multiple agricultural components—**sensors, actuators, communication modules, and cloud platforms**—into a unified, intelligent ecosystem.

The ESP32 is responsible for **collecting real-time environmental data** from various field-deployed sensors such as:

- **Soil Moisture Sensors** – to monitor soil hydration levels.
- **Temperature and Humidity Sensors (e.g., DHT11/DHT22)** – to track atmospheric conditions..

With its **dual-core processing capability**, the ESP32 processes this data locally and makes real-time decisions. It can **trigger irrigation systems**, **send alerts to farmers**, or **log data to cloud servers** via its built-in **Wi-Fi and Bluetooth** capabilities. This wireless connectivity allows seamless communication with **mobile apps**, **web dashboards**, or **IoT cloud services** for monitoring and control.

A key advantage of the ESP32 in agricultural environments is its **energy efficiency**. With support for multiple **low-power modes**, it is particularly well-suited for deployment in **remote or off-grid areas**, where it can be powered by batteries or small solar panels.

Another powerful feature is its ability to **integrate with weather APIs**. The ESP32 can fetch **live weather forecasts** and dynamically adjust its operations. For instance, if rainfall is expected, it can delay irrigation to conserve water and optimize resource usage. When combined with **AI or machine learning models**, the ESP32 enables intelligent prediction and adaptive behavior based on historical trends and live inputs.

Moreover, the ESP32 supports **Over-the-Air (OTA) firmware updates**, allowing developers or system integrators to **remotely upgrade or patch the device** without needing physical access—critical for large-scale deployments across expansive fields.

3.1.3 Key Benefits of ESP32 in Smart Agriculture:

- **Real-time Sensor Data Acquisition and Processing:**

The ESP32 excels in processing and acquiring real-time data

from a wide variety of sensors used in smart agriculture systems. These sensors can monitor environmental conditions such as soil moisture, temperature, humidity, light intensity, and air quality. The ability to acquire this data in real-time ensures that farmers can make timely and informed decisions based on the current state of their fields. For instance, real-time monitoring of soil moisture can trigger automatic irrigation systems when the soil moisture falls below a certain threshold, optimizing water usage and ensuring healthy crops. The ESP32's high processing power allows for local computation of sensor data, reducing latency and the need for constant cloud communication, which is particularly beneficial in remote areas with limited internet connectivity.

- **Wireless Communication via Built-in Wi-Fi/Bluetooth:**

One of the most important features of the ESP32 is its built-in Wi-Fi and Bluetooth capabilities. This enables the device to wirelessly transmit sensor data to a central server or cloud-based platform for analysis, storage, and reporting. For farmers operating in large or remote areas, this wireless communication reduces the need for extensive wiring infrastructure, which can be costly and cumbersome to maintain. The Wi-Fi feature allows for high-speed data transmission, while Bluetooth can be used for short-range communication between devices. This wireless flexibility makes the ESP32 ideal for a wide range of agricultural applications, such as monitoring multiple fields, greenhouse environments, and livestock tracking, all in real-time.

- **Support for AI and Weather API Integration:**

The ESP32 can be integrated with Artificial Intelligence (AI) models to analyze the collected sensor data and derive actionable insights. For example, machine learning models can predict crop yield, detect early signs of plant diseases, or optimize irrigation schedules based on historical data and weather patterns. The ESP32's support for weather API integration further enhances its functionality by allowing farmers to access up-to-date weather forecasts. By linking real-time weather data with sensor data, farmers can receive automatic alerts about potential adverse conditions like heavy rainfall, frost, or high winds, enabling them to take proactive measures to protect crops and optimize their farming practices.

- **Low Power Consumption for Remote Operation:**

Designed with energy efficiency in mind, the ESP32 is highly suited for long-term deployment in remote or off-grid locations, where access to power may be limited. Its low power consumption allows the device to operate on battery or solar power for extended periods, reducing the need for frequent recharging or maintenance. This is particularly important in agricultural environments where constant monitoring is essential, but access to electricity may be scarce. Additionally, the ESP32 includes power-saving modes, such as deep sleep mode, which significantly extends the battery life without sacrificing performance. This low power consumption ensures that remote farming operations can run continuously, collecting data and providing insights without interruptions.

- **Scalable and Cost-Effective:**

The ESP32 is a highly scalable solution, making it an ideal choice for both small-scale and large-scale agricultural operations. Its compact size, low cost, and high functionality allow for the deployment of a large number of devices across a wide area, without breaking the bank. Farmers can easily scale up their monitoring system by adding more sensor nodes as their needs grow, whether it's monitoring a single greenhouse or an entire farm. This cost-effectiveness extends beyond the hardware itself, as the ESP32's open-source nature and widespread community support mean that development costs are minimized. Additionally, the availability of numerous libraries and frameworks for agricultural applications reduces the time and effort required to bring a system online.

3.2 PINS OF ESP 32:

The ESP32 is a versatile microcontroller with a wide range of pins, which can be used for various applications in smart agriculture. These pins offer different functionalities that can be used to interface with sensors, actuators, communication modules, and other peripherals necessary for precision farming. Below is a detailed overview of the ESP32 pins, their general functions, and their relevance to your smart agriculture.

1. GPIO Pins (General Purpose Input/Output)

The ESP32 features **34 GPIO pins**, which are used to interface with a variety of digital components like sensors, relays, and actuators. These pins are highly versatile, supporting different input/output modes:

- **Digital Input/Output**: Used to read data from sensors (e.g., soil moisture, temperature) and control relays or actuators (e.g., turning on irrigation systems, controlling ventilation).
- PWM Output: Some GPIO pins support Pulse Width Modulation (PWM) output, which can be used for controlling motors, pumps, or dimming lights.
- Interrupts: These pins can trigger interrupts for time-sensitive tasks, such as detecting motion or environmental changes, essential for real-time monitoring in agriculture.

Pin Example:

- **GPIO 4**: Often used as a digital I/O pin for sensors or as a PWM output pin for controlling actuators.

2. Analog Input Pins

The ESP32 has **18 ADC channels** (Analog-to-Digital Converter), which allow the board to read analog sensor values, such as soil moisture or light levels. These pins can be used to convert analog voltages into digital values that can be processed by the microcontroller.

- **ADC1 and ADC2**: These two ADCs on the ESP32 allow reading of analog sensor values. ADC1 is typically used for lower-speed applications, while ADC2 supports faster readings but with certain limitations on usage during Wi-Fi operations.

Pin Example:

- **GPIO 34 (ADC1)**: Used for reading analog signals like soil moisture levels.
- **GPIO 36 (ADC1)**: A common pin for connecting sensors like temperature or humidity sensors

3. DAC Pins (Digital-to-Analog Converter)

The ESP32 features **2 DACs**, which are used to generate analog signals from digital data. These can be useful in applications where you need to output analog signals, such as controlling a variable speed motor in a pump or creating analog audio signals for notifications

.

Pin Example:

- **GPIO 25**: Can be used as a DAC output pin, potentially for controlling a pump speed or irrigation system via analog signals.

4. I2C Pins (Inter-Integrated Circuit)

I2C is a communication protocol often used for connecting multiple low-speed devices like environmental sensors (e.g., temperature and humidity sensors). The ESP32 has dedicated pins for **I2C communication**:

- **SDA (Data Line) and SCL (Clock Line):** These are the two essential lines used for I2C communication, and the ESP32 allows you to configure any GPIO pin for I2C functionality.

Pin Example:

- **GPIO 21 (SDA):** Commonly used for I2C data communication.

5. SPI Pins (Serial Peripheral Interface)

The ESP32 supports the SPI communication protocol, which is used for high-speed data transfer between devices, such as sensors, displays, and SD cards. This is particularly useful when dealing with more complex sensors, such as GPS modules or high-speed environmental sensors.

- **MISO** (Master In Slave Out), **MOSI** (Master Out Slave In), **SCK** (Clock), and **CS** (Chip Select) are the primary pins involved in SPI communication.

Pin Example:

- **GPIO 23 (MOSI):** Used for sending data to a sensor or display.
-

6. UART Pins (Universal Asynchronous Receiver/Transmitter)

The ESP32 offers **2 UART interfaces** for serial communication, often used to communicate with external devices like GPS modules, modems, and other peripherals.

- **TX (Transmit)** and **RX (Receive)** pins allow for bidirectional communication between the ESP32 and other devices

Pin Example:

- **GPIO 1 (TX)**: Typically used for transmitting data.
- **GPIO 3 (RX)**: Typically used for receiving data.

7. Touch Pins

The ESP32 also features **capacitive touch** pins, which are used for touch-sensitive interfaces. These pins are sensitive to touch, and they can be used for controlling irrigation systems or providing feedback for user interactions with the system.

Pin Example:

- **GPIO 4**: Often used as a touch input pin, which could be used to manually trigger an irrigation system.

8. PWM Pins (Pulse Width Modulation)

The ESP32 supports **PWM output** on several GPIO pins, which is used for controlling the speed of motors or pumps, dimming lights, or controlling servos.

9. External Interrupt Pins

Certain GPIO pins on the ESP32 are configured to trigger **external interrupts**, allowing real-time responses to changes in the environment, such as detecting motion or sudden changes in sensor values.

Pin Example:

- **GPIO 34 (Interrupt)**: Can be used to detect environmental changes such as the presence of livestock or changes in soil moisture.

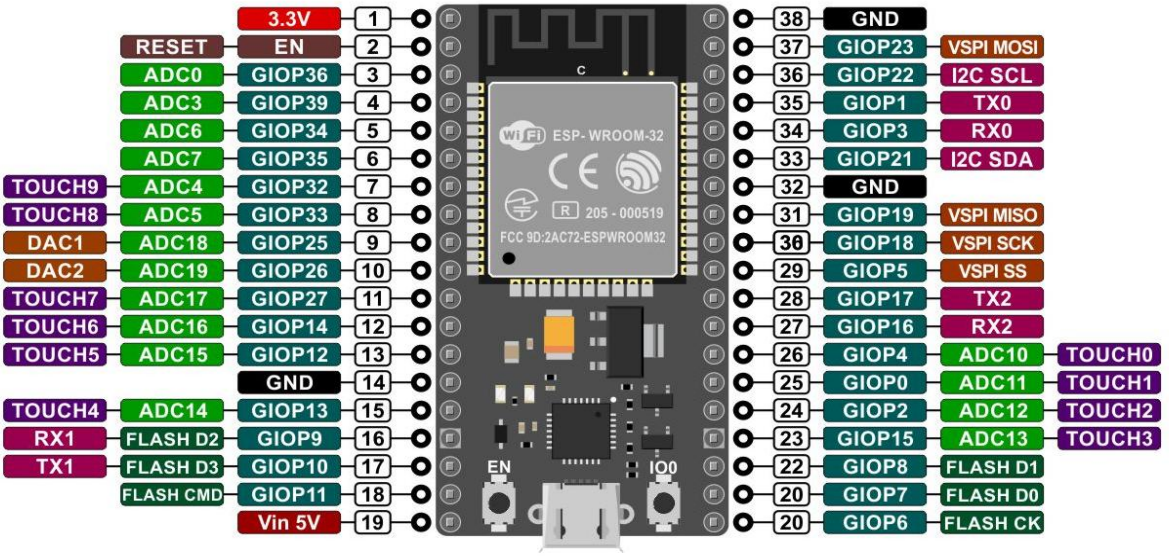
10. Power and Ground Pins

The ESP32 has various power and ground pins for providing stable power to the system and connected sensors.

- **3.3V**: Power supply for the board and low-power sensors.
- **GND**: Ground connection for completing the circuit.

Pin Example:

- **3.3V Pin**: Used to power low-voltage sensors such as soil moisture or temperature sensors.
- **GND Pin**: Common ground connection for all sensors and devices.

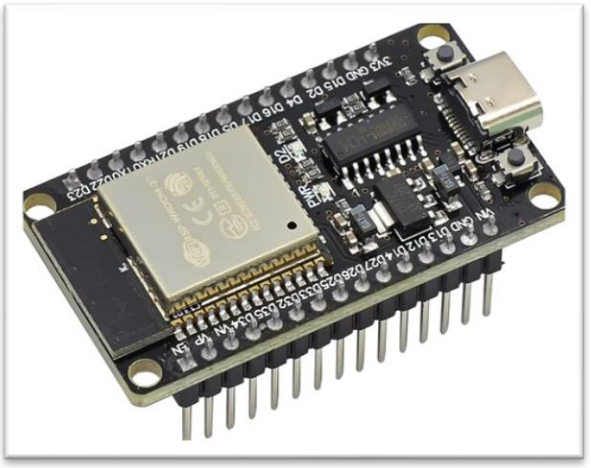


3.2.1 Figure 1: ESP32 PINS

CHAPTER-4
INTERFACING DEVICES

CHAPTER-4
INTERFACING DEVICES

4.1 ESP 32:



Interfacing Overview of ESP32 in Smart Agriculture

In the era of smart and sustainable farming, the **ESP32 microcontroller** stands out as a cornerstone of embedded intelligence and automation. With its **dual-core processing**, integrated **Wi-Fi and Bluetooth** capabilities, and rich peripheral support—including **GPIOs, ADCs, DACs, UART, I2C, SPI**, and **PWM**—the ESP32 becomes an ideal choice for interfacing a wide range of agricultural sensors and actuators.

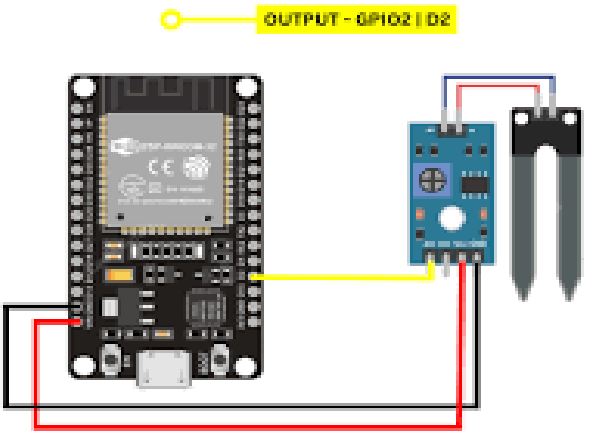
This versatility allows ESP32 to serve as both the **brain and communicator** of the precision farming system, enabling seamless integration of real-time sensor data acquisition, automated control of irrigation systems, environmental monitoring, and wireless data transmission to cloud platforms. Its ability to operate in **low-power modes** also makes it suitable for **remote field deployments** where energy efficiency is crucial.

In smart agriculture, this microcontroller plays a pivotal role in achieving high-efficiency farming by connecting the physical environment to the digital realm—collecting data from the field, processing it locally, making informed decisions, and executing real-time actions—all without manual intervention.

4.2 SOIL MOISTER SENSOR:

- **Soil Moisture Sensor Interfacing with ESP32:**

In any smart agriculture system, soil moisture monitoring is a fundamental requirement for achieving efficient irrigation and water conservation. The Soil Moisture Sensor is one of the key components interfaced with the ESP32 microcontroller to enable real-time measurement of soil water content, ensuring crops receive the right amount of water at the right time.



Specifications of Soil Moisture Sensor:

- **Operating Voltage:** 3.3V to 5V
- **Output Type:** Analog & Digital
- **Detection Range:** Varies depending on soil type
- **Response Time:** Fast (real-time detection)
- **Operating Temperature:** -10°C to 60°C
- **Interface:** 2-pin or 3-pin connectors (VCC, GND, A0/D0)

Working Principle:

The soil moisture sensor typically consists of two probes that act as conductors. When inserted into the soil, the resistance between the probes changes depending on the water content:

- Wet soil conducts electricity better, resulting in lower resistance and higher analog voltage output.

- Dry soil has higher resistance, producing a lower voltage output.

The ESP32 reads this analog voltage using its built-in ADC (Analog-to-Digital Converter), allowing it to determine the moisture level of the soil in real-time.

4.2.1 Importance in Precision Farming:

In precision agriculture, overwatering and underwatering can significantly affect crop yield. The integration of the soil moisture sensor with ESP32 allows:

- Automated irrigation when soil moisture falls below a defined threshold.
- Conservation of water resources by preventing unnecessary watering.
- Healthier crops through optimized soil hydration.
- Remote monitoring and alerting via Wi-Fi or cloud-based IoT dashboards.

Interfacing with ESP32:

Below is a simple guide to interface the soil moisture sensor with ESP32:

1. **VCC** of the sensor is connected to 3.3V or 5V pin of ESP32 (depending on sensor model).
2. **GND** is connected to the ground (GND) of ESP32.
3. **Analog Output** (A0) is connected to one of the ADC-enabled GPIO pins (e.g., GPIO 34, 35, 36).

4.3 Relay and Motor Interfacing with ESP32

In smart agriculture systems, **automation of irrigation** is one of the most vital applications. For this purpose, **relays** and **DC water pump motors** are interfaced with the ESP32 microcontroller to control water flow based on environmental sensor data, such as soil moisture, temperature, or scheduled timings.

4.3.1 Relay Module:

A **relay** is an electrically operated switch that allows the ESP32, which operates at low voltage (3.3V), to control **high-voltage devices** like **12V or 220V AC water pumps**. This provides an essential layer of **electrical isolation and safety** between the control logic and the high-power components.

Specifications of Relay Module:

- **Operating Voltage**: 5V
- **Trigger Voltage**: 3.3V – compatible with ESP32
- **Current Rating**: Up to 10A @ 250V AC / 30V DC
- **Type**: Single Channel / Dual Channel / 4 Channel
- **Isolation**: Optocoupler-based isolation (in most modules)

4.3.2 DC Water Pump Motor:

The **DC motor** or **mini water pump** is used for delivering water to crops through pipes or drip systems. It can be operated by switching the relay ON or OFF based on soil conditions.

Specifications of DC Pump Motor:

- **Operating Voltage**: 6V to 12V DC
- **Rated Speed**: ~200 RPM
- **Flow Rate**: Varies depending on the motor and setup
- **Current Consumption**: ~0.5A to 2A

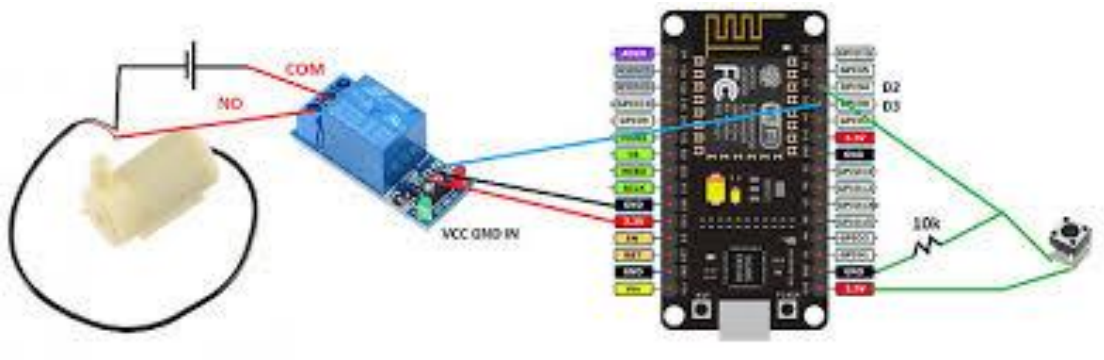
Working and Automation:

- When the **soil moisture sensor** detects dry soil (below threshold), ESP32 **triggers the relay**.
- The relay **activates the water pump**, supplying water to crops.
- Once the moisture reaches a sufficient level, ESP32 **deactivates the relay**, turning the pump off.
- This creates a **fully automatic irrigation system**, minimizing water wastage and maximizing crop health.

4.3.3 Interfacing Circuit with ESP32:

1. **Relay VCC** → 5V pin of ESP32 (or external 5V supply if needed)
2. **Relay GND** → GND of ESP32
3. **Relay IN** → Any GPIO pin (e.g., GPIO 23)
4. **Motor VCC** → Connected to **NO (Normally Open)** terminal of relay

5. **Common terminal of relay (COM)** → Connected to external 12V power supply



6. **Motor GND** → Connected to the ground of power supply

4.4 Open Weather API Integration with ESP32

In modern smart agriculture systems, integrating **external weather intelligence** is just as important as collecting on-field sensor data. The **Open Weather API**, a popular cloud-based service, provides **real-time and forecasted weather data**, which can be accessed using the ESP32's built-in **Wi-Fi** module. This integration enhances the decision-making capability of smart farming systems by predicting environmental changes and optimizing operations accordingly.

Why Use Open Weather in Precision Farming?

While sensors can measure current conditions like temperature, humidity, and rainfall at a specific location, Open Weather adds a **broader context** by delivering:

- **Weather forecasts** (hourly/daily)
- **Rain predictions**
- **Temperature and humidity trends**
- **Wind speed and direction**
- **Sunrise and sunset timings**

By leveraging this data, farmers can:

- Avoid unnecessary irrigation when rainfall is predicted
- Schedule pesticide spraying when wind is low
- Take preventive action before storms or extreme weather
- Analyze historical weather trends to plan future crops

4.4.1 How It Works with ESP32 and OpenWeather API Integration

The **ESP32** microcontroller, with its built-in **Wi-Fi** and **Bluetooth** capabilities, serves as a versatile device for gathering **real-time weather data** from external sources, such as the **OpenWeather API**. In a precision farming system, **weather intelligence** is crucial for **predicting environmental conditions** and **automating farming operations** based on forecasted weather patterns.

Operational Workflow:

1. **Wi-Fi Configuration and Connection:**
    - The ESP32 is first connected to a local Wi-Fi network (via SSID and password configuration in the code). This gives the ESP32 access to the internet and the ability to send and receive data over HTTP.
2. **Sending the HTTP Request to OpenWeather API:**
    - Once the ESP32 is connected to the internet, it sends a **GET request** to the OpenWeather API. This request is sent over **HTTP** to retrieve current weather information for a specified location, such as a city or geographic coordinates.
    - The request URL includes:
        - The **API key** (unique to each user)
        - The **location query** (city name or geographical coordinates)
        - The desired **unit system** (metric for Celsius, Fahrenheit, etc.)

Example API Request:

s=metrichttps://api.openweathermap.org/data/2.5/weather?q=London&appid=YOUR_API_KEY&unit

3. **Receiving Data from OpenWeather API:**
    - The ESP32 receives the weather data in **JSON format**, which contains detailed information about the weather, such as:
        - **Temperature** (main.temp)
        - **Humidity** (main.humidity)
        - **Weather condition description** (weather[0].description)
        - **Wind speed** (wind.speed)
        - **Rain data** (rain.1h or rain.3h)
4. **Parsing and Storing the Data:**

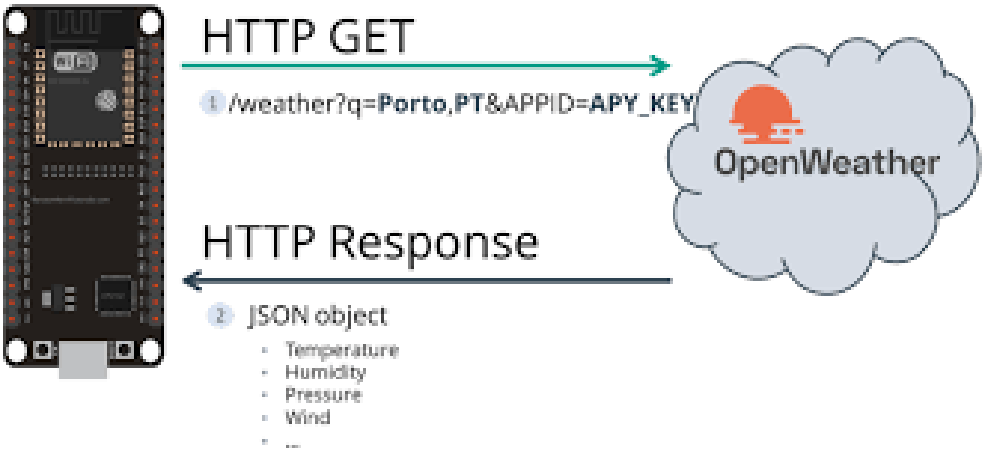
- Using an **ArduinoJson** library or similar, the ESP32 processes the incoming JSON data. It extracts and stores key values (e.g., temperature, humidity, rain probability) in variables that can be used to make smart farming decisions.
- For example, if the forecast indicates rain, the system might prevent irrigation from activating or delay scheduled operations.

5. **Implementing Automated Decision Logic:**
   - Based on the received data, the ESP32 can trigger specific actions in the farming system:
     - **Automatic Irrigation Control:** If **rain** is expected, the system could turn off irrigation, saving water and preventing over-watering.
     - **Alerts and Notifications:** Real-time weather changes can be sent as **alerts** to the farmer's mobile app, email, or web dashboard for further action.
6. **Scheduling and Frequency of Updates:**
   - The weather data retrieval process can be set to run at **regular intervals** (e.g., every 30 minutes or hourly), ensuring that the farming system remains up-to-date with the latest weather conditions.
   - In addition, the ESP32 can **store previous weather data**, which could be useful for trend analysis, historical forecasting, or optimization of future farming actions.

An alert portable tele-medical monitor system (AMON) is proposed in [9], aiming to provide continuous monitoring for high-risk cardiac/respiratory patients. The system collects multiple vital signs, detects multi-parameter medical emergencies and is connected to a cellular telemedicine center (TMC). This system uses a wrist worn device, to monitor vital parameters of patients in order to provide an integrated picture of their health condition.

4.4.2 Environmental Applications

Environmental applications that demand continuous monitoring of ambient conditions at hostile and remote areas can be improved with the utilization of WSNs. In **Figure 8**, the main subcategories of environmental applications of WSNs, namely water monitoring, air monitoring, and emergency alerting, are depicted along with the types of sensors that are typically used in them. WSNs that have been developed for these types of environmental applications are studied in the following subsectio.



4.5 GROQ LLM Interfacing with ESP32

The interfacing of **ESP32 with GROQ LLM (Large Language Model)** marks a transformative step in the implementation of intelligent and autonomous Smart Agriculture systems. This integration bridges real-time sensor data collected at the field level with cloud-based AI processing, enabling the system to make human-like decisions, provide predictive insights, and offer interactive support to farmers.

Overview of Interfacing

The ESP32 microcontroller acts as the **edge device**, interfacing with sensors and actuators in the field. It collects environmental data (soil moisture, temperature, humidity, etc.) and transmits it to a **cloud server or database** via Wi-Fi. This data is then accessed and processed by the **GROQ LLM**, which runs on a high-speed, cloud-based AI engine. The model interprets this data and sends back human-readable recommendations, decisions, or commands, which are received by the ESP32 and used to automate or inform actions.

4.5.1 Architecture of the Interfacing:

ESP32 ↔ Cloud Server/API ↔ GROQ LLM

- **Step 1:** ESP32 reads real-time sensor data.
- **Step 2:** Data is sent via HTTP POST or MQTT protocol to a cloud platform.
- **Step 3:** GROQ LLM accesses this data through an API, analyzes it, and generates intelligent insights or decisions.
- **Step 4:** The processed response is sent back to ESP32 or a user dashboard for action or alerting.

Use Case Example:

Let's consider an example where the ESP32 gathers the following data:

- Soil Moisture: 14%
- Temperature: 35°C
- Humidity: 40%
- Forecast: No rain in next 48 hours (from OpenWeather API)

ESP32 sends this data to the cloud where GROQ LLM processes the input. The model responds:

"Low soil moisture detected and high temperature expected. No rain predicted. Initiate irrigation for 20 minutes in Zone A to protect crop yield."

The ESP32 receives this response and triggers the relay module to activate the irrigation motor.

4.5.2 Advantages of GROQ LLM Interfacing with ESP32:

- **AI-Driven Farming**: Integrates AI for decision support.
- **Human-Like Interaction**: Farmers can ask questions in natural language.
- **Real-Time Action**: Sensors → Cloud → Decision → Action in seconds.
- **Localized Recommendations**: Tailored to weather, crop type, and soil condition.
- **Multi-Device Coordination**: Can control multiple farm devices from one AI brain.

CHAPTER-5
RADIO FREQUENCY COMMUNICATION

CHAPTER-5
RADIO FREQUENCY COMMUNICATION

5.1 Introduction to Protocols in Communication of ESP32 with AI

(for Real-Time Precision farming using AI Analytics and IOT sensor Networks)

In modern smart agriculture, integrating AI with IoT-enabled microcontrollers like the ESP32 has revolutionized how farms are monitored and managed. At the heart of this integration lies a set of communication protocols that enable seamless data transfer between hardware (e.g., sensors, actuators) and AI systems (e.g., GROQ LLM, cloud platforms).

These protocols ensure that data collected from the field—such as soil moisture, humidity, and temperature—is reliably transmitted to AI engines, which then analyze and respond with intelligent recommendations or automatic actions in real time.

Why Communication Protocols Matter in Smart Farming:

Feature	Description
Real-Time Monitoring	Enables live data collection and updates for dynamic farm environments.
AI-Driven Decision Making	Allows sensor data to be analyzed instantly by AI models like GROQ LLM.
Secure Data Transmission	Ensures reliable and encrypted communication across devices and the cloud.
Remote Accessibility	Farmers can monitor and control devices from any location via the Internet.
Efficient Automation	Facilitates instant responses such as irrigation control or alert generation.

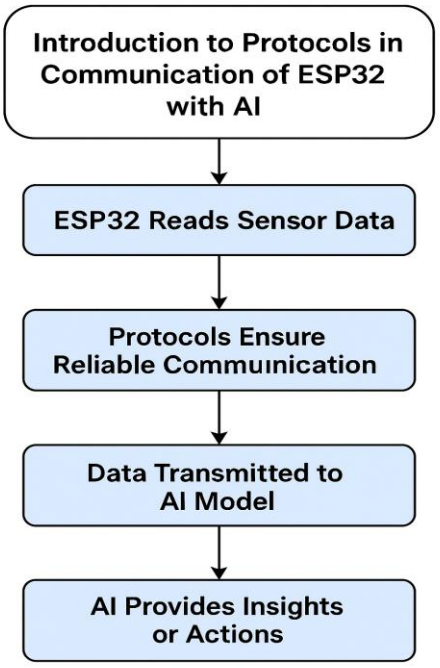
5.1.1 How ESP32 Communicates with AI

The ESP32 uses a combination of wired and wireless communication protocols such as:

- Wi-Fi / HTTP – For REST API calls to AI models and weather APIs.

- MQTT – For lightweight, real-time messaging between ESP32 and cloud services.
- UART / I2C / SPI – For local communication with onboard sensors and actuators.
- JSON Parsing – To extract structured information from AI and weather API responses.

5.1.2 Flowchart for Protocols in Communication of ESP32 with AI:



1. Wi-Fi

The **ESP32** is equipped with both Wi-Fi and Bluetooth capabilities, but Wi-Fi is most commonly used in IoT applications for its speed, range, and availability. Wi-Fi provides the necessary connectivity to link the ESP32 with the AI processing system, whether it's cloud-based or edge computing.

- **Usage in Smart Agriculture:** The ESP32 will send sensor data (e.g., temperature, humidity, soil moisture, etc.) to the cloud or local server, where the AI system performs analytics and decision-making.
- **Advantages:** High-speed data transfer, reliable connectivity, and ability to work over long ranges (depending on infrastructure).
- **Protocol Details:** You can use HTTP, MQTT, or WebSockets over Wi-Fi, depending on the application's needs. For real-time communication, **MQTT** is preferred for its lightweight, efficient message-passing architecture.

2. HTTP/HTTPS (Hypertext Transfer Protocol/Secure)

HTTP is widely used for communication over the web. It enables the ESP32 to communicate with cloud services or a centralized server where the AI model can process data.

- **Usage in Smart Agriculture:** The ESP32 sends sensor readings to a server or cloud platform via **HTTP/HTTPS** requests (using **GET**, **POST**, or **PUT** methods). The server processes the data using AI algorithms and returns actionable insights to the ESP32 or other connected devices.
- **Advantages:** Universally supported, easy to implement, and suitable for both real-time and batch data transmission.
- **Protocol Details:** The ESP32 can use **RESTful APIs** to communicate with the cloud or server. With **HTTPS**, data is securely transmitted, ensuring privacy and integrity, which is crucial for sensitive agricultural data

3. Data Encryption and Security Protocols

Communication between the ESP32 and AI must be secure to ensure the confidentiality and integrity of the data transmitted, especially when dealing with sensitive agricultural information.

- **SSL/TLS (Secure Sockets Layer/Transport Layer Security):** Secure communication between the ESP32 and cloud servers or AI systems should be ensured using **SSL/TLS** encryption, particularly for HTTP/HTTPS connections.
- **AES (Advanced Encryption Standard):** For additional security in data transmission, especially in low-power networks like LoRa or BLE, **AES** encryption can be employed to secure sensor data.

4. Edge AI Processing

In some applications, especially in large farms with limited connectivity, **edge computing** is used. Here, **AI models** are deployed directly on local servers or edge devices to process data in real-time, reducing latency and reliance on cloud services.

- **ESP32 + Edge AI:** The ESP32 could be used to send sensor data to an edge AI processor (such as a **Jetson Nano** or **Raspberry Pi with AI capabilities**). These devices process data locally and send actionable insights back to the ESP32 or other devices on the farm.

CHAPTER-6

SOFTWARE TOOLS

CHAPTER-6
SOFTWARE TOOLS

6.1 INTRODUCTION TO ARDUINO IDE

The Arduino Integrated Development Environment (IDE) is an open-source software platform used for writing, compiling, and uploading code to microcontrollers, including the ESP32 used in our Smart Agriculture system. It provides a user-friendly interface, extensive community support, and cross-platform compatibility, making it an essential tool for embedded system development and IoT applications.

Key Features of Arduino IDE:

- **Cross-Platform Support:** Runs on Windows, macOS, and Linux.
- **Sketch-Based Programming:** Programs are called "sketches", written in a simplified version of C/C++.
- **Built-in Libraries**: Comes with a wide array of libraries for sensors, communication protocols (e.g., Wi-Fi, MQTT), and hardware modules.
- **Board Manager:** Allows easy installation of support packages for additional boards like ESP32, ESP8266, etc.
- **Serial Monitor & Plotter:** Offers tools to debug and visualize data in real time from the connected device.
- **OTA (Over-The-Air) Programming:** Supports wireless code uploading on boards like ESP32.

Relevance in Smart Agriculture:

In the context of our Smart Agriculture system, the Arduino IDE serves as the primary platform for:

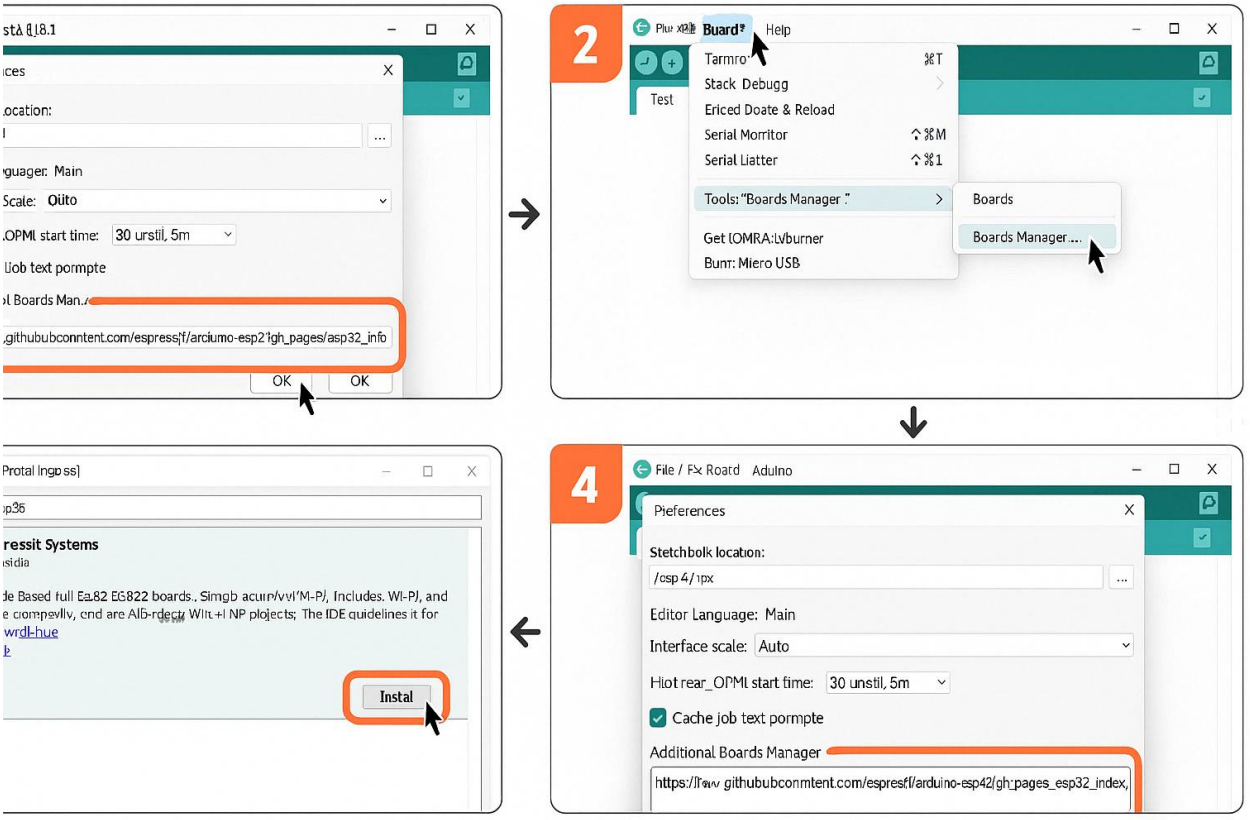
- Programming the **ESP32** to collect sensor data (e.g., temperature, humidity, soil moisture).

- Setting up Wi-Fi and communication protocols such as MQTT or HTTP for data transmission to AI/cloud systems.
- Controlling actuators like irrigation motors or alarms based on AI predictions or sensor thresholds.
- Debugging and Monitoring the system in real-time using the serial monitor.

Advantages of Using Arduino IDE:

- Ease of Use: Minimal setup required; suitable for both beginners and experts.
- Community Support: Large ecosystem of tutorials, forums, and libraries.
- Extensibility: Supports custom libraries and third-party boards like the ESP32 through the Board Manager**.**

6.1.1 Figure 1: Arduino IDE SetUp



6.2 LIBRARIES

In a Smart Agriculture system for Precision Farming, software libraries play a crucial role in enabling efficient development, sensor integration, data communication, and AI interaction. These libraries are used across different layers of the system — from embedded firmware on the ESP32 to cloud APIs and AI engines.

1. Arduino & ESP32 Firmware Libraries

These libraries are used directly in the ESP32 firmware, typically written and uploaded using the **Arduino IDE** or **PlatformIO**.

2. Cloud & AI Integration Libraries

When integrating with AI or cloud platforms, these libraries or SDKs are commonly used (either on ESP32, edge processors, or cloud services):

Library/SDK	Platform	Functionality
Firebase ESP32 Library	Firebase (Google)	Real-time database, user auth, data sync
FastAPI / Flask	Python backend	REST API for receiving sensor data and returning AI decisions

3. Security and Communication Libraries

Security is vital when transmitting agricultural data over the internet. These libraries ensure data integrity and confidentiality.

Library	Purpose
WiFiClientSecure	HTTPS communication on ESP32
mbedtls	Lightweight SSL/TLS for embedded devices

Library	Purpose
OAuth2 / FirebaseAuth	User authentication with cloud services

In a precision farming system, the software ecosystem spans **microcontrollers**, **edge devices**, and **cloud platforms**. Libraries simplify interactions between sensors, AI engines, and user interfaces — enabling faster development and more intelligent decision-making.

By combining libraries from hardware-level (like WiFi.h or DHT.h) with cloud & AI SDKs (like TensorFlow Lite or Firebase), the system becomes modular, efficient, and scalable.

6.3 BOOT LOADER

In embedded systems like the Smart Agriculture System for Precision Farming, a bootloader is a crucial component that ensures the microcontroller starts correctly, loads the main application (firmware), and optionally supports firmware updates. For the ESP32, which is the central controller in our system, the bootloader manages how the firmware is uploaded, stored, and executed—making it an essential part of the software development and deployment lifecycle.

6.3.1 What is a Bootloader?

A **bootloader** is a small program that resides in a reserved section of the microcontroller's memory. It executes immediately after the device is powered on or reset. Its primary purpose is to:

- Initialize hardware peripherals.
- Verify and load the main firmware (user application) from non-volatile memory.
- Enable flashing (uploading) of new firmware via serial, USB, or Over-The-Air (OTA).
- Provide recovery mechanisms in case of firmware corruption.

In the context of a Smart Agriculture system, where devices are often deployed remotely in fields, **bootloader functionality** is critical for ensuring reliability and supporting remote updates.

6.3.2 Key Features of ESP32 Bootloader

- **Serial Flashing Support:** Uses UART to upload firmware through USB.
- **OTA Firmware Updates:** Enables wireless firmware upgrades, essential for large-scale field deployments.
- **Partition Table Support:** Organizes flash memory into sections like app, OTA, SPIFFS, etc.
- **Fallback Logic:** Reverts to a previous firmware version if the new update fails.

6.3.3 Flashing Firmware via Bootloader

Automatic Mode (Using Arduino IDE):

When uploading a sketch via Arduino IDE:

- It automatically sets ESP32 into boot mode.
- Uploads the firmware through the serial port.
- Resets the ESP32 to start the new application.

Manual Mode:

If automatic mode fails:

1. Hold down the **BOOT** button on the ESP32 board.
2. Click the **Upload** button in Arduino IDE.
3. Release **BOOT** when the uploading starts.

6.4 SOFTWARE COMPONENTS

Firebase Realtime Database, **Groq AI API**, **Twilio API**, and a custom-built **Kodular application** to deliver a seamless and intelligent user experience. The **Kodular app** serves as the front-end interface, enabling users to interact effortlessly with the system. Firebase Realtime Database is used for real-time data storage and synchronization across devices. The **Groq AI API** provides fast and efficient AI-driven responses and decision-making, enhancing automation and user interaction. Additionally, the **Twilio API** is utilized to send instant SMS alerts and notifications, ensuring real-time communication with users..

6.4.1 Firebase Realtime Database

A control flag (mode) is stored in the **Firebase Realtime Database** as either:

“auto” → AI-driven control

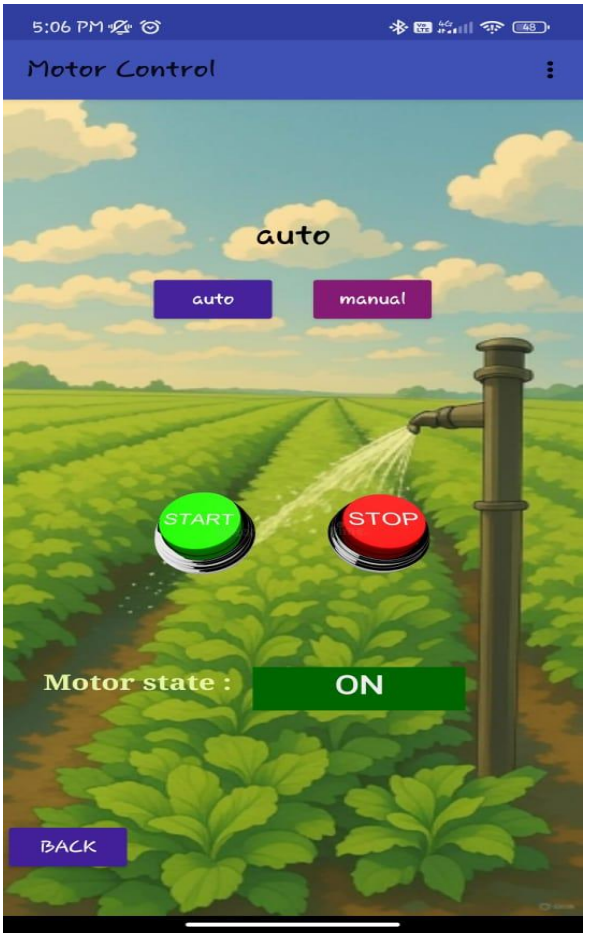
"manual" → User control via APP

Auto Mode (AI-Controlled)

- System uses **Groq AI decisions** based on:
    - Soil moisture
    - Weather forecast
- If irrigation is needed, **relay turns ON automatically**.

Ideal for **hands-free operation** in smart farming

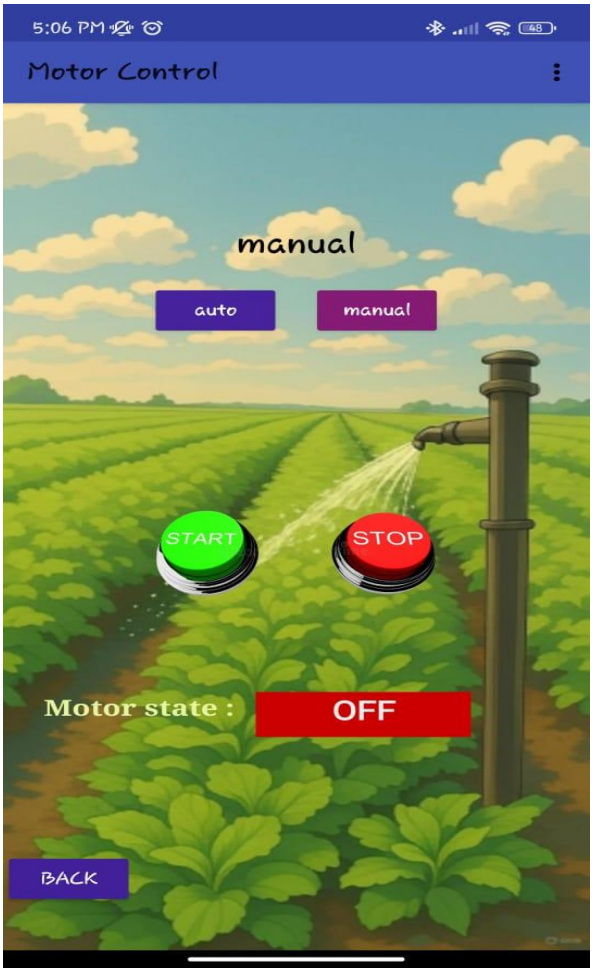
Figure 2: Motor Control Auto Mode



Manual Mode (User-Controlled)

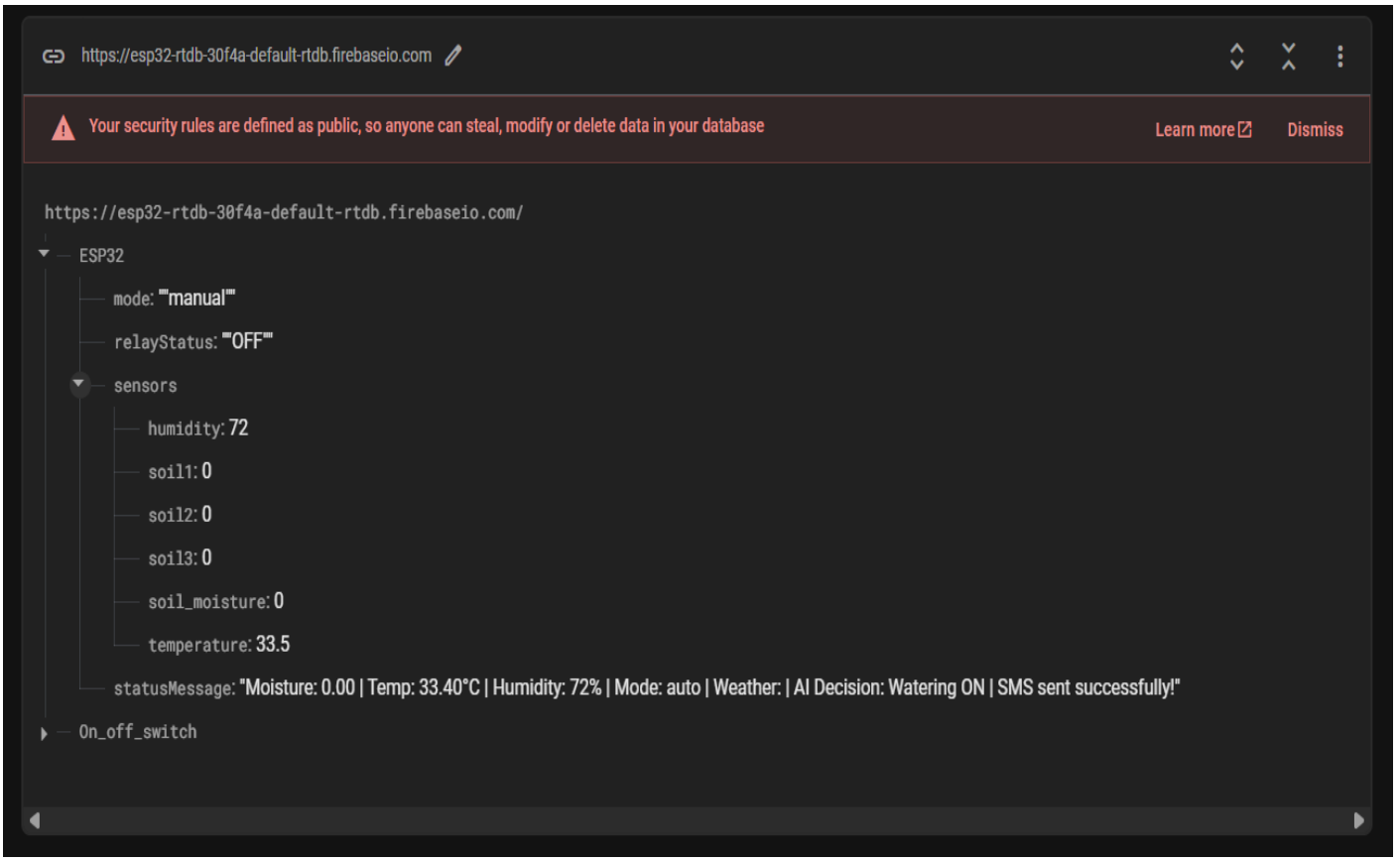
- Farmers can manually turn ON/OFF the motor via:
    - Mobile app (Kodular)
- Useful when the farmer wants to override AI decisions

Figure 3: Motor Control Manual Mode



This Kodular app is designed to provide real-time monitoring of environmental conditions such as temperature, humidity, and soil moisture. It utilizes built-in sensors or connected external modules to display live weather and soil data, making it especially useful for smart agriculture and home automation. By offering a user-friendly interface and accurate data display, the app helps users make informed decisions to improve efficiency and sustainability.

Figure 4: Firebase Database Structure



6.4.2 Groq AI API

1. Sensor & Weather Data Sent

ESP32 sends soil moisture and weather data (temperature, humidity, rain forecast, etc.) to a http protocal .Groq AI Integration

The HTTP protocal forwards this data to the Groq AI API, which processes the information using a trained model

2 .Decision Logic

The AI evaluates:

- Soil moisture levels
- Rain possibility
- Weather conditions

3 .Irrigation Decision

Based on the AI analysis, it sends back a decision like:

"Irrigation Required" or "No Irrigation Needed".

4 .Action Trigger

If irrigation is required and system is in auto mode, ESP32 activates the relay to turn ON the motor**.**

6.4.3 Twilio SMS Alerts

Real-Time Irrigation Notifications

- System sends automated SMS alerts to the farmer's mobile using the Twilio API.

When are SMS Alerts Triggered?

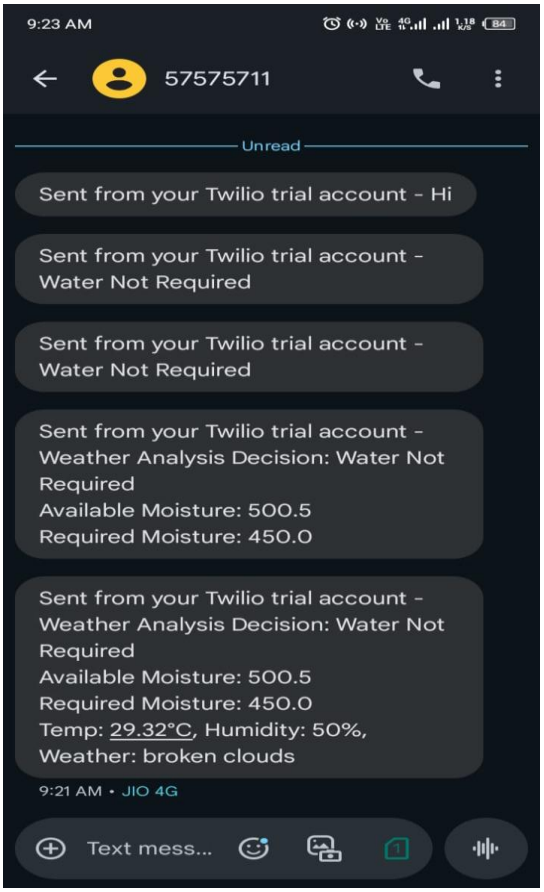
- When AI (Groq) predicts **irrigation is required**
- When soil moisture is too low and auto mode is active
- When **manual control is activated** via app

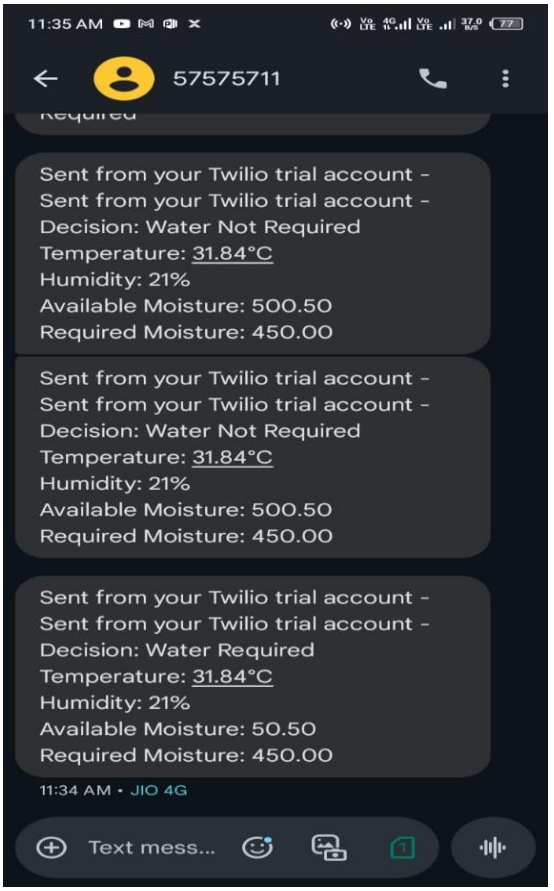
Why Twilio?

- Twilio allows you to send real-time SMS updates to farmers about weather conditions, irrigation status, and sensor readings.
- It helps deliver critical alerts when soil moisture is too low or weather changes, so farmers can take quick action.
- Twilio is a reliable platform that ensures message delivery, even in rural or low-network areas.
- Farmers don't need smartphones or internet access—just a basic mobile phone to receive SMS alerts.
- It supports automated messaging, so updates can be sent without manual effort.

- AI-based decisions from the Groq model can be communicated instantly via SMS, making the system smarter and proactive.

.





6.4.4 Kodular App

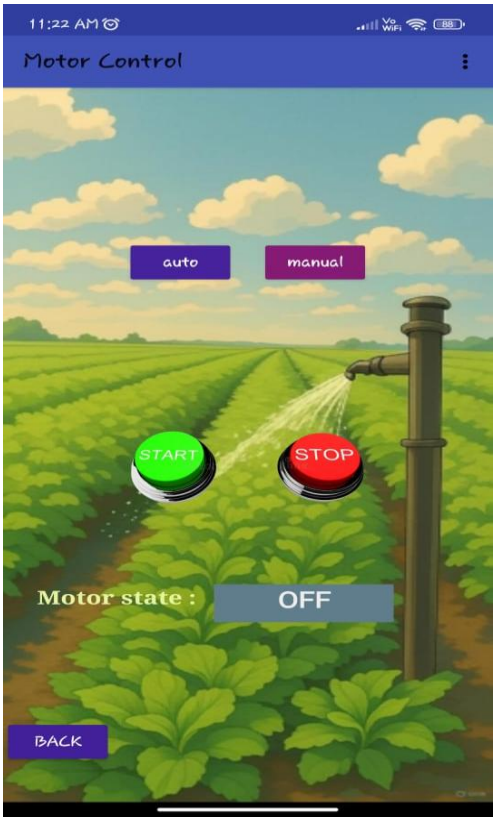
User-friendly mobile app created using Kodular

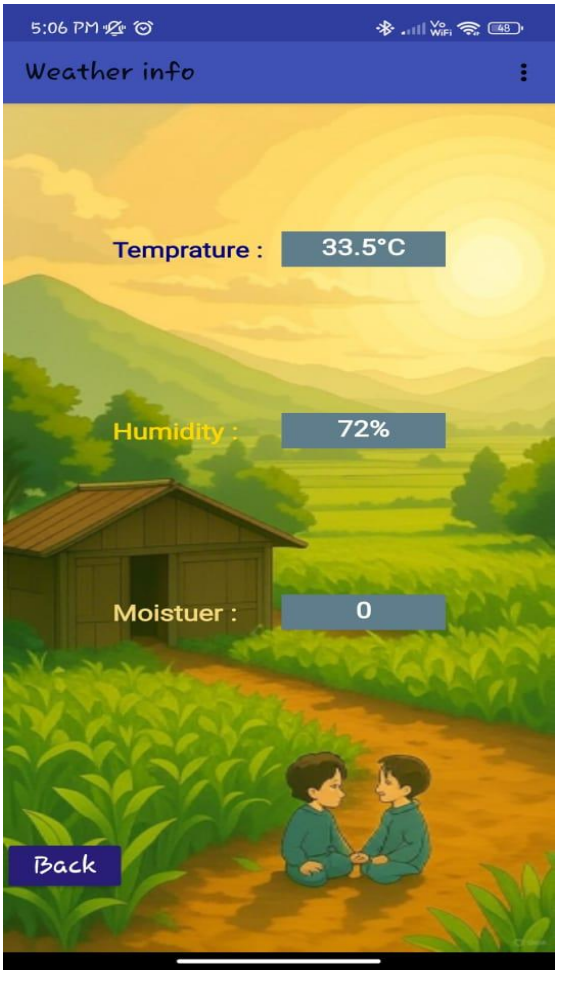
Displays real-time sensor data (temperature, humidity, soil moisture)

Allows farmers to manually control the motor (ON/OFF)

Kodular is a powerful no-code platform that empowers users to create fully functional Android apps without writing a single line of code. Using a drag-and-drop interface and a block-based logic system, it simplifies app development, making it accessible even to beginners. It supports Material Design, offers monetization through ad integration, and includes an extensive library of components like sensors, maps, and web APIs. Hosted on Google Cloud, Kodular ensures secure and scalable development, with live testing and real-time debugging tools. Its active community and growing ecosystem make it an ideal choice for startups, students, and hobbyists alike.


5:06 PM
Home
Smart Agriculture system for precision Farming
motor control
weather info


11:22 AM
Motor Control
auto
manual
START
STOP
Motor state :
OFF
BACK


5:06 PM
Weather info
Temprature :
33.5°C
Humidity :
72%
Moistuer :
0
Back

6.5 CODE

The **ESP32** microcontroller serves as the heart of the Smart Agriculture System. It collects data from environmental sensors, processes it, and transmits it to cloud servers or AI engines for further analysis. In this section, we provide the complete code flow used to control and monitor the system using the Arduino IDE.

The code consists of the following functionalities:

- Initialization of sensors and modules.
- Reading sensor data (e.g., soil moisture, temperature, humidity).
- Sending data via Wi-Fi using HTTP or MQTT.
- (Optional) Receiving AI-based decisions from the cloud.
- Controlling actuators like water pumps or fans.
- OTA firmware updates.

6.5.1 Code Used For this project

Required Libraries

```
#include <WiFi.h>

#include <HTTPClient.h>

#include <ArduinoJson.h>

#include <mbedtls/base64.h>

#include <DHT.h>

// Firebase

#include <Firebase_ESP_Client.h>

#include <addons/TokenHelper.h>

#include <addons/RTDBHelper.h>

// Wi-Fi Credentials
```

```
const char* ssid = "***********";

const char* password = "**********";

// OpenWeatherMap

const char* weatherApiKey = "06ea9d51a37babf63dcf0f165ea8f592";

const char* weatherBaseUrl = "https://api.openweathermap.org/data/2.5/weather";

// Groq AI

const char* groqApiKey = "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";

const char* groqApiUrl = "https://api.groq.com/openai/v1/chat/completions";

// Twilio

const char* twilioSID = "AC73a915c2b4244700612c37a98ba1761d";

const char* twilioAuthToken = "2432a6275a42d200232df920a297489a";

const char* twilioFrom = "+12706068368";

const char* twilioTo = "+919398612747";

// Firebase

#define API_KEY "AIzaSyC7lhw3aFo_iINX440iYnho05_xdvLisXo"

#define DATABASE_URL "https://esp32-rtdb-30f4a-default-rtdb.firebaseio.com/"

#define USER_EMAIL "saspf @gmail.com"

#define USER_PASSWORD "saspf@123"

// Location

float lat = 12.9716;

float lon = 77.5946;
```

Define Sensor and Pin Configuration

```
// Sensor Pins

#define SOIL_SENSOR_1 34

#define SOIL_SENSOR_2 35

#define SOIL_SENSOR_3 32
```

```
#define RELAY_PIN 27

#define DHTPIN 21

#define DHTTYPE DHT22

// Globals

float availableMoisture = 0.0;

float requiredMoisture = 450.0;

float temperature = 0.0;

int humidityValue = 0;

// Firebase objects

FirebaseData fbdo;

FirebaseAuth auth;

FirebaseConfig config;

// DHT object

DHT dht(DHTPIN, DHTTYPE);
```

Wi-Fi Connection Setup

```
void setup() {

  Serial.begin(115200);

  pinMode(RELAY_PIN, OUTPUT);

  digitalWrite(RELAY_PIN, LOW); // Default OFF

  dht.begin();

  WiFi.begin(ssid, password);

  Serial.print("Connecting to Wi-Fi");

  while (WiFi.status() != WL_CONNECTED) {

    delay(500); Serial.print(".");

  }

  Serial.println("\nConnected to Wi-Fi");

  // Firebase setup
```

```
  config.api_key = API_KEY;
  config.database_url = DATABASE_URL;
  auth.user.email = USER_EMAIL;
  auth.user.password = USER_PASSWORD;
  Firebase.begin(&config, &auth);
  Firebase.reconnectWiFi(true);
  while (auth.token.uid == "") {
    Serial.print("."); delay(500);
  }
  Serial.println("\nFirebase is ready!");
  fetchSensorData(); // Initial read
}
void loop() {
  delay(60000);  // Every 1 minute
  fetchSensorData();
}
void fetchSensorData() {
  int s1 = 4095 - analogRead(SOIL_SENSOR_1);
  int s2 = 4095 - analogRead(SOIL_SENSOR_2);
  int s3 = 4095 - analogRead(SOIL_SENSOR_3);
  availableMoisture = (s1 + s2 + s3) / 3.0;
  temperature = dht.readTemperature();
  humidityValue = dht.readHumidity();
  if (isnan(temperature) || isnan(humidityValue)) {
    Serial.println("Failed to read from DHT22!");
    return;
  }
```

```
  Serial.printf("Moisture: %.2f | Temp: %.2f°C | Humidity: %d%%\n", availableMoisture, temperature, humidityValue);
```

Send Data to Cloud

```
  // Upload to Firebase
  String path = "/ESP32/sensors";
  Firebase.RTDB.setFloat(&fbdo, path + "/temperature", temperature);
  Firebase.RTDB.setFloat(&fbdo, path + "/humidity", humidityValue);
  Firebase.RTDB.setFloat(&fbdo, path + "/soil_moisture", availableMoisture);
  Firebase.RTDB.setInt(&fbdo, path + "/soil1", s1);
  Firebase.RTDB.setInt(&fbdo, path + "/soil2", s2);
  Firebase.RTDB.setInt(&fbdo, path + "/soil3", s3);
  checkModeAndControl();
}
void checkModeAndControl() {
  if (Firebase.RTDB.getString(&fbdo, "/ESP32/mode")) {
    String mode = fbdo.stringData();
    // Clean unwanted characters
    mode.trim();                    // Removes leading/trailing whitespace
    mode.replace("\"", "");         // Remove double quotes
    mode.replace("\\", "");         // Remove backslashes
    Serial.println("Mode: " + mode); // Clean output: Mode: manual / Mode: auto
    if (mode == "manual") {
      if (Firebase.RTDB.getString(&fbdo, "/ESP32/relayStatus")) {
        String relayState = fbdo.stringData();
        // Clean relayState as well
        relayState.trim();
        relayState.replace("\"", "");
```

```
      relayState.replace("\\", "");


      if (relayState == "ON") {
        digitalWrite(RELAY_PIN, HIGH);
        Serial.println("Manual Mode: Relay ON");
      } else {
        digitalWrite(RELAY_PIN, LOW);
        Serial.println("Manual Mode: Relay OFF");
      }
    }
    return;  // ☑ Exit here to skip AI control
  }
 }
 // If not manual, go to AI logic
 fetchWeatherData();
}
void fetchWeatherData() {
 HTTPClient http;
 String url = String(weatherBaseUrl) + "?lat=" + String(lat) + "&lon=" + String(lon) +
              "&appid=" + String(weatherApiKey) + "&units=metric";
 http.begin(url);
 int httpCode = http.GET();
 if (httpCode > 0) {
   String response = http.getString();
   Serial.println("Weather Response:");
   Serial.println(response);
   DynamicJsonDocument doc(2048);
```

```
  deserializeJson(doc, response);
  String description = doc["weather"][0]["description"].as<String>();
  String prompt = "{ \"temperature\": " + String(temperature) +
                  ", \"humidity\": " + String(humidityValue) +
                  ", \"weather\": \"" + description +
                  "\", \"available_moisture\": " + String(availableMoisture) +
                  ", \"required_moisture\": " + String(requiredMoisture) + " }";
  analyzeWithGroq(prompt);
 } else {
  Serial.printf("Weather API Error: %s\n", http.errorToString(httpCode).c_str());
 }
 http.end();
}
void analyzeWithGroq(String data) {
 HTTPClient http;
 http.begin(groqApiUrl);
 http.addHeader("Content-Type", "application/json");
 http.addHeader("Authorization", "Bearer " + String(groqApiKey));
 DynamicJsonDocument req(2048);
 req["model"] = "llama-3.3-70b-versatile";
 JsonArray messages = req.createNestedArray("messages");
 // Add system message
 JsonObject systemMessage = messages.createNestedObject();
 systemMessage["role"] = "system";
 systemMessage["content"] = "Analyze the weather and moisture data. Reply ONLY with 'Water Required' or 'Water Not Required'.";
 // Add user message
 JsonObject userMessage = messages.createNestedObject();
```

```
  userMessage["role"] = "user";
  userMessage["content"] = data;
  String requestBody;
  serializeJson(req, requestBody);
  int code = http.POST(requestBody);
  if (code > 0) {
    String response = http.getString();
    Serial.println("Groq Response:");
    Serial.println(response);
    DynamicJsonDocument res(2048);
    deserializeJson(res, response);
    String decision = res["choices"][0]["message"]["content"].as<String>();
    sendSMS(decision);
  } else {
    Serial.printf("Groq API Error: %s\n", http.errorToString(code).c_str());
  }
  http.end();
}
String base64Encode(const String& input) {
  size_t outputLen;
  unsigned char encoded[1024];
  mbedtls_base64_encode(encoded, sizeof(encoded), &outputLen, (const unsigned char*)input.c_str(), input.length());
  return String((char*)encoded);
}
```

Actuator Control Logic

```
void sendSMS(String decision) {
```

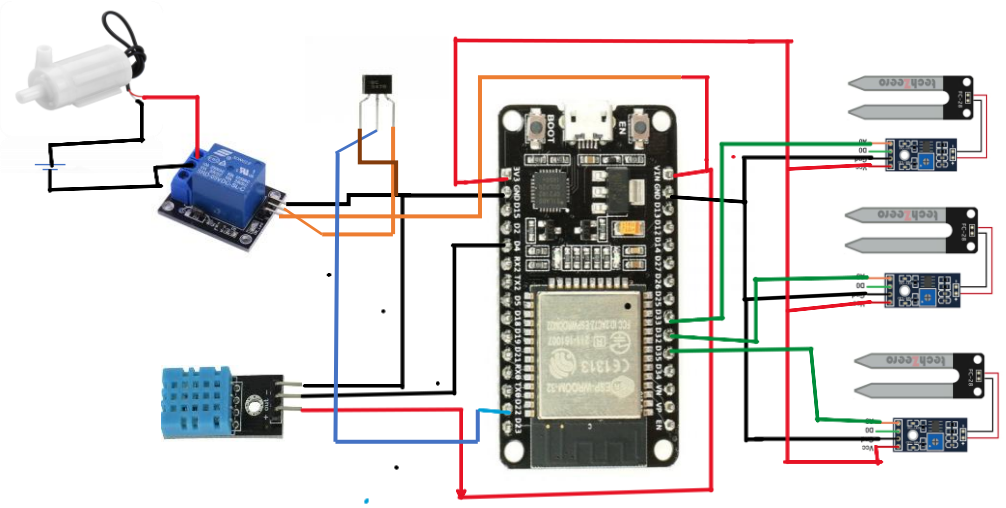
```
if (decision.indexOf("Water Required") >= 0) {
  digitalWrite(RELAY_PIN, HIGH);
  Serial.println("AI Decision: Watering ON");
} else {
  digitalWrite(RELAY_PIN, LOW);
  Serial.println("AI Decision: Watering OFF");
}
// --- Twilio SMS ---
HTTPClient http;
String msg = "Temp: " + String(temperature) + "°C, Humidity: " + String(humidityValue) +
             "%, Moisture: " + String(availableMoisture) + ", AI: " + decision;
String auth = base64Encode(String(twilioSID) + ":" + String(twilioAuthToken));
http.begin("https://api.twilio.com/2010-04-01/Accounts/" + String(twilioSID) + "/Messages.json");
http.addHeader("Authorization", "Basic " + auth);
http.addHeader("Content-Type", "application/x-www-form-urlencoded");
String body = "From=" + String(twilioFrom) + "&To=" + String(twilioTo) + "&Body=" + msg;
int httpCode = http.POST(body);
if (httpCode > 0) {
  Serial.println("SMS sent successfully!");
} else {
  Serial.printf("SMS sending failed: %s\n", http.errorToString(httpCode).c_str());
}
http.end();
// Log to Firebase
Firebase.RTDB.setString(&fbdo, "/ESP32/relayStatus", decision);
Firebase.RTDB.setString(&fbdo, "/ESP32/relayStatus", digitalRead(RELAY_PIN) == HIGH ? "ON" :
"OFF");
}
```

CHAPTER-7
RESULTS

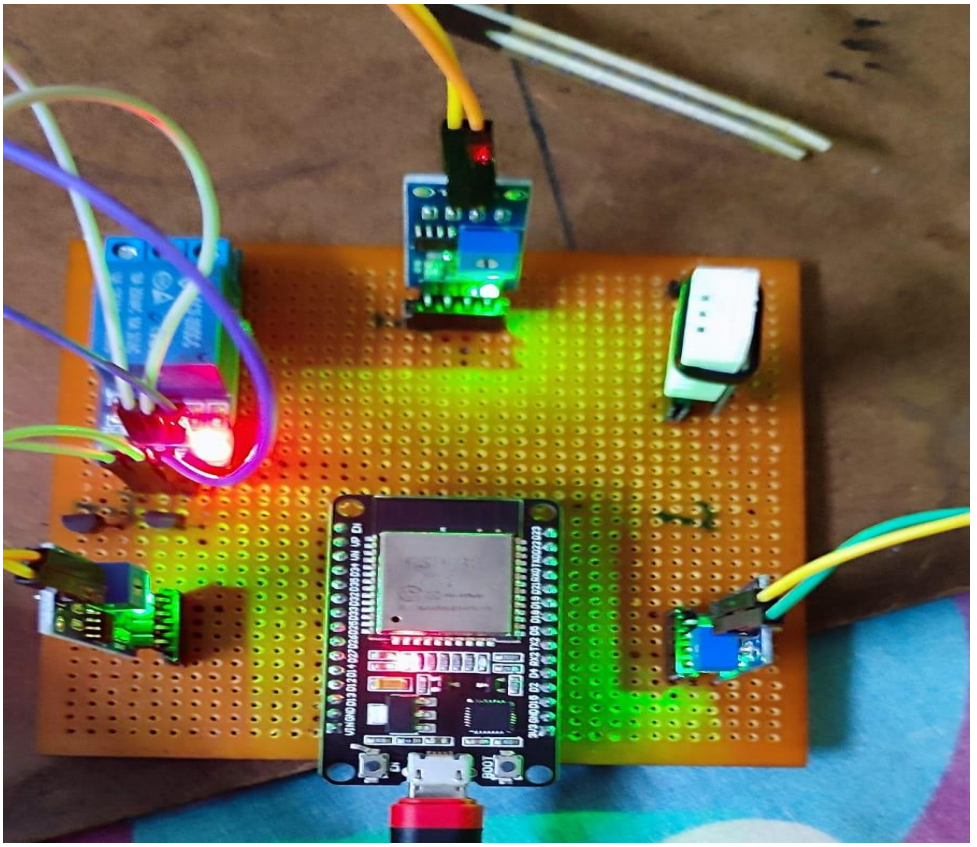
CHAPTER-7

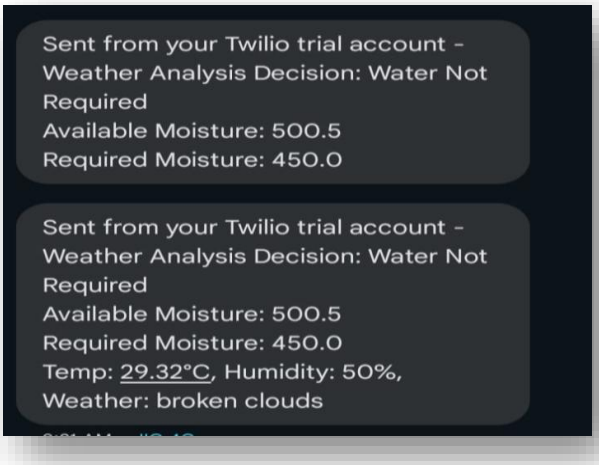
RESULTS

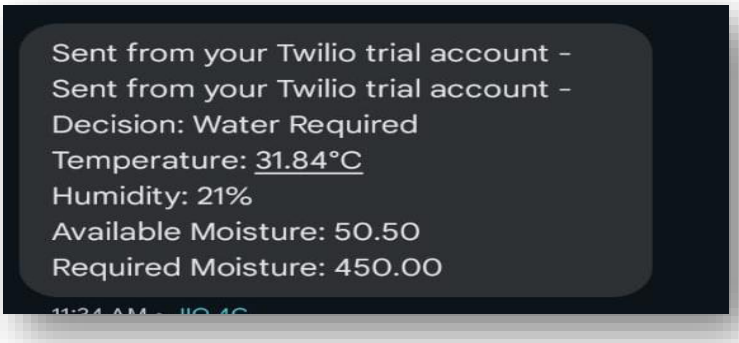
7.1 CIRCUIT DIAGRAM

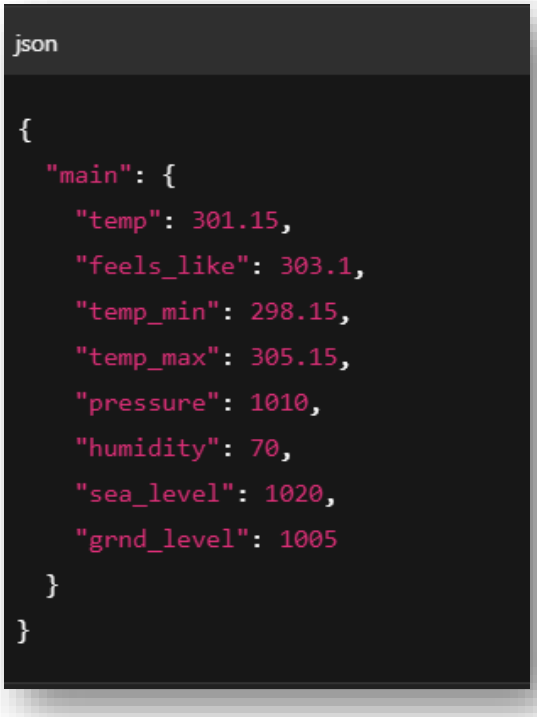


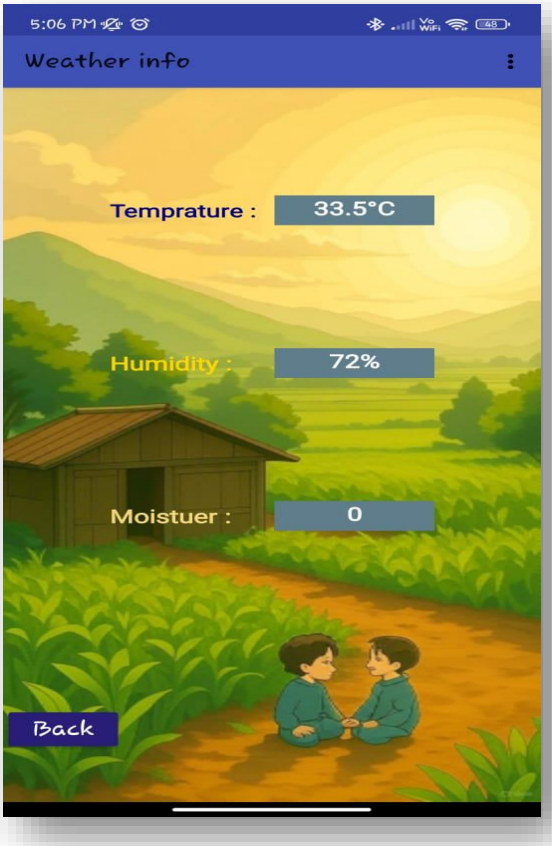
7.2.1 KIT SETUP




Sent from your Twilio trial account -
Weather Analysis Decision: Water Not Required
Available Moisture: 500.5
Required Moisture: 450.0
Sent from your Twilio trial account -
Weather Analysis Decision: Water Not Required
Available Moisture: 500.5
Required Moisture: 450.0
Temp: 29.32°C, Humidity: 50%,
Weather: broken clouds


Sent from your Twilio trial account -
Sent from your Twilio trial account -
Decision: Water Required
Temperature: 31.84°C
Humidity: 21%
Available Moisture: 50.50
Required Moisture: 450.00


json
{
"main": {
"temp": 301.15,
"feels_like": 303.1,
"temp_min": 298.15,
"temp_max": 305.15,
"pressure": 1010,
"humidity": 70,
"sea_level": 1020,
"grnd_level": 1005
}
}


5:06 PM
Weather info
Temprature :
33.5°C
Humidity :
72%
Moistuer :
0
Back


Firebase is ready!
Moisture: 0.00 | Temp: 33.30°C | Humidity: 70%
Mode: auto
Weather Response:
{"coord":{"lon":77.5946,"lat":12.9716},"weather":[{"id":800,"main":"Clear","description":"clear sky","icon":"01d"}],"base":"stations","main":{"temp":32.35,"feels_like":30.51,"temp_min":32.3
Groq Response:
{"id":"chatcmpl-e6ac59fd-8ae2-408c-b493-e1bb3fab22b7","object":"chat.completion","created":1744880317,"model":"llama-3.3-70b-versatile","choices":[{"index":0,"message":{"role":"assistant","
AI Decision: Watering ON
SMS sent successfully!

CHAPTER-8
CONCLUSION

CHAPTER-8
CONCLUSION

8.1 FUTURE SCOPE

The Smart Agriculture System for Precision Farming, built on cutting-edge technologies like ESP32, Groq AI, Twilio API, Kodular, and Firebase, is well-positioned for scalable and impactful growth. The following future enhancements can greatly expand its capabilities and effectiveness**:**

1.Advanced AI-Powered Crop Advisory (Groq AI)

Expand Groq AI integration to deliver personalized crop recommendations, pest prediction, yield forecasting, and disease detection using real-time sensor data and historical patterns stored in Firebase**.**

2.Real-Time Multilingual Alerts via Twilio

Enhance the Twilio API integration to send voice or SMS alerts in local languages, providing farmers with critical updates on irrigation needs, weather alerts, or system malfunctions — even on feature phones.

3.Smartphone App Enhancements (Kodular)

Upgrade the Kodular-built app with dynamic dashboards, offline data syncing, weather-integrated planning tools, and interactive tutorials, making it a complete control and advisory hub for farmers.

4.Firebase-Driven Community Intelligence

Enable crowd-sourced data features using Firebase, where multiple farms share data anonymously,

5. Weather API + AI-Driven Forecast Models

Integrate advanced weather APIs and use AI to give microclimate-based decision support, such as best time for fertilization, irrigation, and harvesting.

CHAPTER-9
REFERENCES

CHAPTER-9

REFERENCES

REFERENCES

[1] Firebase, "Firebase Realtime Database Documentation," Google Developers, [Online]. Available: https://firebase.google.com/docs/database. [Accessed: Apr. 18, 2025].

(Chapter 4, Section 4.1, "ESP32")

[2] Twilio Inc., "Twilio SMS API – Send & Receive Text Messages," Twilio, [Online]. Available: https://www.twilio.com/docs/sms. [Accessed: Apr. 18, 2025].

(Chapter 4, Section 4.5, "Groq LLM")

[3] OpenWeather, "OpenWeatherMap API Documentation," [Online]. Available: https://openweathermap.org/api. [Accessed: Apr. 18, 2025].

(Chapter 4, Section 4.4, "Open Weather")

[4] R. Fielding et al., "RFC 2616: Hypertext Transfer Protocol – HTTP/1.1," IETF, 1999. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc2616. [Accessed: Apr. 18, 2025].

(Chapter 5, Section 5.1, "Introduction to Protocols in Communication")

[5] WeatherAPI.com, "Weather API Documentation," [Online]. Available: https://www.weatherapi.com/docs/. [Accessed: Apr. 18, 2025].

(Chapter 4, Section 4.4, "Open Weather")

[6] Tomorrow.io, "Weather API for Developers," [Online]. Available: https://www.tomorrow.io/weather-api/. [Accessed: Apr. 18, 2025].

(Chapter 4, Section 4.4, "Open Weather")

[7] Groq Inc., "Groq AI Accelerator Platform," [Online]. Available: https://www.groq.com/. [Accessed: Apr. 18, 2025].

(Chapter 4, Section 4.5, "Groq LLM")

[8] Google Developers, "Firebase Authentication Documentation," [Online]. Available: https://firebase.google.com/docs/auth. [Accessed: Apr. 18, 2025].

(Chapter 4, Section 4.1, "ESP32")

[9] Espressif Systems, "ESP32 Series Datasheet," [Online]. Available: https://www.espressif.com/en/products/socs/esp32. [Accessed: Apr. 18, 2025].

(Chapter 3, Section 3.1, "Introduction to ESP32")

[10] JSON.org, "JavaScript Object Notation (JSON)," [Online]. Available: https://www.json.org/json-en.html. [Accessed: Apr. 18, 2025].

(Chapter 5, Section 5.1, "Introduction to Protocols in Communication")