MINISTRY OF HIGHER EDUCATION
VALLEY HIGHER INSTITUTE FOR ENGINEERING & TECHNOLOGY
DEPARTMENT OF ELECTRONIC AND COMMUNICATION ENGINEERING

SMART TECHNOLOGY APPLICATIONS IN AGRICULTURE:

Smart Irrigation and Disease Detection Utilizing Internet of Things and Artificial Intelligence to Enhance Water Consumption Efficiency and Plant Health Care

Under supervision of:

Assoc. Prof. Ashraf Mohamed Ali

Dr. Ibrahim Abdel-Dayem

The project was implemented by students of the Department of Communications and Electronics Engineering, Class of 2019

Contents

ABSTRACT:

In the ever-changing world of farming, smart agriculture systems have emerged as game-changers, using fancy tech to tackle big challenges like feeding a growing global population while being kind to the planet. This graduation project book is your go-to guide to understand and dive deep into these innovative farming solutions.

We'll take you on a tour of the cool tech behind smart agriculture, from building mobile apps with Flutter to using AI for making smart decisions and managing data in the Cloud. You'll get hands-on with hardware like ESP32 and IoT devices, learning how they collect data and automate tasks on the farm.

But it's not all theory—get ready to roll up your sleeves and put these ideas into action. We'll show you how smart irrigation systems work, using sensors and AI to water crops just right. Plus, you'll learn how AI can spot plant diseases early, helping farmers protect their harvests and boost productivity.

We won't just stop at building cool stuff; we'll also talk about how to measure success. We'll look at how these systems increase crop yields, save resources, and make farming more sustainable. And we'll dive into how they're making a real difference in food security and protecting our environment.

But we're not done yet—get ready to peer into the future of farming. We'll explore exciting new tech like 5G, fog computing, and the Internet of Everything, and how they'll shape the farms of tomorrow. And we'll discuss how renewable energy, virtual reality, and more will play a role in making farming even smarter.

By bringing together the latest research and hands-on practices, this book is your ticket to understanding and creating the future of farming. Whether you're a student looking to dive into smart agriculture or a researcher pushing the boundaries of what's possible, this book is your roadmap to a greener, more efficient future in farming.

CHAPTER 1 *INTRODUCTION*

Chapter topics:

1.0_Introduction to the Project

In the wake of the fourth industrial revolution, agriculture is undergoing a significant transformation, evolving into a high-tech industry characterized by efficiency and sustainability. This project is a testament to this evolution, embodying the integration of cutting-edge technologies such as Flutter, FlutterFlow, AI, Gemini AI, Cloud services, Firebase database, Firestore, and IoT with traditional farming practices. It aims to create a smart agriculture system that not only enhances crop productivity but also ensures environmental sustainability. [1] [2] [3]

1.1_Background and Context

Agriculture has been the cornerstone of human survival and economic growth throughout history. However, with the advent of global challenges such as climate change, population growth, and resource depletion, the agricultural sector is under immense pressure to innovate. Smart agriculture systems emerge as a beacon of hope, offering a solution that marries technology with cultivation to usher in a new era of farming that is both productive and sustainable. [4] [5]

1.2_Motivation and Objectives

The impetus behind this project is the urgent need to revolutionize agriculture to meet the increasing food demands of a growing global population while addressing the environmental impacts of traditional farming methods. The objectives are multifaceted, aiming to develop a mobile application using Flutter and FlutterFlow, implement a smart irrigation system with ESP32, and utilize Gemini AI for efficient plant disease identification, all within the framework of IoT. [6] [7]

1.3_Scope and Structure

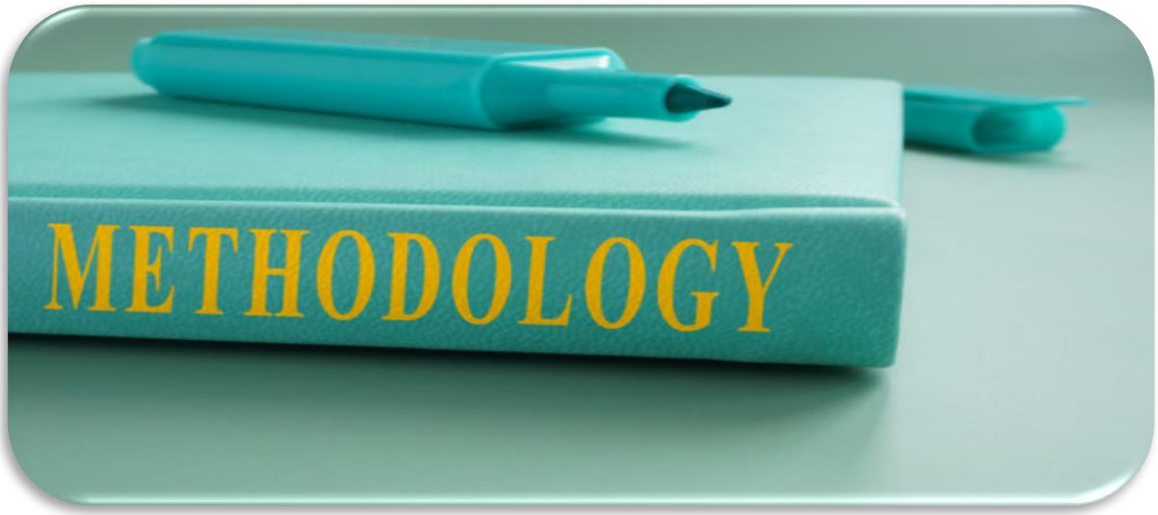
This project's scope is ambitious, encompassing the development of a comprehensive smart agriculture system that includes a user-friendly mobile application, AI-powered analytics for predictive insights, and an IoT network for seamless data collection and automation. The structure of the project is designed to facilitate the integration of these technologies into a cohesive system that can be easily adopted by farmers for improved agricultural outcomes. [8]

1.4_Significance of the Project

The significance of this project lies in its potential to revolutionize the agricultural sector. By leveraging advanced technologies, it aims to empower farmers with real-time data, automate irrigation and disease management, and ultimately contribute to the global goal of achieving food security in an environmentally sustainable manner. This project stands as a model for future agricultural innovations, paving the way for smarter, more resilient farming practices.

This expanded introduction provides a comprehensive overview of your project, emphasizing its relevance, technological integration, and potential impact on the future of agriculture. I hope this serves as a solid foundation for your graduation project. [9]

CHAPTER 2 METHODOLOGY



Chapter topics:

CHAPTER 2: METHODOLOGY

2.0_Methodology Overview

2.0.1_Introduction to methodology overview:

Our methodology overview delves into the systematic approach used to develop and implement a smart agriculture solution. This includes technologies like Flutter, AI, Gemini AI, Cloud services, IoT, and ESP32 for a smart irrigation system and plant disease identification. [10] [11] [12] [13] [14]

2.0.2_Exploration of Technologies:

We thoroughly explore the roles of Flutter, AI, Gemini AI, Cloud services, and IoT in our solution, each contributing to its functionality and performance. ESP32 enables IoT functionalities crucial for smart irrigation, while Firebase database and Firestore provide robust backend infrastructure. [10] [11] [12] [13] [14]

2.0.3_Development Process and Implementation Strategies:

We adopt agile methodologies for iterative development, using version control systems like Git for collaboration and project management tools for efficient resource allocation. [10] [11] [12] [13] [14]

2.0.4_Integration of Technologies:

Seamless integration of technologies ensures smooth communication and data exchange. ESP32 manages IoT functions, while Gemini AI enhances plant disease identification accuracy.

2.0.5_Testing and Validation Procedures:

Rigorous testing, including unit, integration, and system testing, ensures reliability and performance. User acceptance testing and field trials provide real-world feedback for continuous refinement.

2.0.6_Summary:

Our methodology aims to revolutionize agriculture by enhancing productivity and sustainability. Through innovation and continuous improvement, we strive to address evolving agricultural challenges.

2.1_Overview of Technologies Used in the Graduation Project

In this section, we explore the cutting-edge technologies driving our graduation project, designed to transform agricultural practices through the development of a smart agriculture solution. Each technology plays a pivotal role in enhancing the functionality, efficiency, and sustainability of our project, contributing to its overall success. Let's delve into the details of each technology and its significance within our project:

2.1.0_Flutter and FlutterFlow:

Flutter, a user interface (UI) toolkit developed by Google, serves as the backbone of our mobile application development efforts. It enables us to build high-performance, natively compiled applications for various platforms like mobile, web, and desktop from a single codebase. Additionally, FlutterFlow, a visual UI builder, expedites development by facilitating rapid prototyping and intuitive UI design (As in the **Figure 2.1.0**). [10] [15]

Figure 2.1.0

2.1.1_Artificial Intelligence (AI) and Gemini AI:

AI technologies, including machine learning and deep learning algorithms, provide our project with intelligent decision-making capabilities. Specifically, Gemini AI, tailored for plant disease identification, employs AI models trained on extensive datasets to accurately diagnose and address crop diseases, ensuring timely intervention and management (As in the **Figure 2.1.1**). [16] [13]

Figure 2.1.1

2.1.2_Cloud Services:

Cloud services such as Firebase database and Firestore offer scalable and reliable backend infrastructure for our project. They enable seamless data storage, retrieval, and synchronization, empowering real-time access to agricultural data regardless of location or device, enhancing accessibility and flexibility. [14]

2.1.3_ESP32 and IoT Integration:

The ESP32 microcontroller, known for its versatility and low-power consumption, serves as the cornerstone of our IoT infrastructure. Through ESP32 code, we implement essential IoT functionalities for our smart irrigation system, enabling remote monitoring and control of irrigation processes, thereby optimizing water usage and enhancing crop yield. [12] [17]

2.1.4_Smart Irrigation System:

Our smart irrigation system, powered by the ESP32 microcontroller and seamlessly integrated with cloud services, revolutionizes traditional irrigation practices. By leveraging real-time environmental data and plant requirements, it enables precise and automated watering schedules, thereby conserving water resources and promoting sustainable agricultural practices (As in the **Figure 2.1.3**). [18]

Figure 2.1.3

2.1.5_Plant Disease Identification using Gemini AI:

By harnessing the advanced capabilities of Gemini AI, our project incorporates a robust plant disease identification system. By analyzing images of plant leaves, Gemini AI accurately detects and diagnoses crop diseases, enabling prompt intervention and management to prevent crop loss and ensure optimal plant health (As in the **Figure 2.1.4**). [19]

Figure 2.1.4

2.1.6_Summary:

Through the strategic integration of these advanced technologies, our graduation project endeavors to revolutionize agricultural practices, enhance crop yield, and promote sustainability. By harnessing the power of Flutter, AI, Cloud services, IoT, and ESP32, we pave the way for a smarter, more efficient, and environmentally conscious approach to agriculture.

By providing authoritative sources for each technology, we ensure that our overview is grounded in research and best practices, fostering a deeper understanding of the technological landscape within which our project operates.

2.2_Development Process and Implementation Strategies

In this comprehensive exploration, we delve into the nuts and bolts of our development process, unveiling the intricate steps and savvy strategies meticulously woven together to bring our graduation project to fruition. With an unwavering focus on adaptability,

collaboration, and innovation, we embark on a journey through the development lifecycle, shedding light on the methodologies embraced and the ingenious tactics employed to surmount project hurdles.

2.2.0_Embracing Agile Methodology:

At the heart of our project lies a fervent embrace of Agile methodology, a dynamic approach renowned for its iterative and customer-centric ethos. We break down our project into bite-sized tasks or user stories, fostering collaboration, flexibility, and incremental delivery. By prioritizing customer feedback and adapting to evolving requirements in short, frequent sprints, we ensure maximum satisfaction and alignment with project goals.

2.2.1_Navigating Iterative Development Cycles (Agile):

Immersed within the Agile ethos, our development journey is marked by iterative cycles characterized by meticulous planning, execution, review, and adaptation. Each iteration focuses on delivering specific features, fostering a culture of continuous improvement and refinement. Regular sprint retrospectives provide valuable checkpoints for introspection and strategic adjustments, nurturing a dynamic environment conducive to growth and learning (As in the **Figure 2.2.1**). [20] [21]

Figure 2.2.1

Unveiling Agile: A New Dawn in Software Development

In the dynamic realm of software development, where change is constant and innovation is key, Agile methodology emerges as a guiding light, fostering adaptability, collaboration, and efficiency. Agile represents a departure from traditional, rigid development approaches, embracing flexibility and iteration while prioritizing customer collaboration and the delivery of value-driven solutions. In this exploration, we embark on a journey into the heart of Agile, tracing its roots, elucidating its core principles, and uncovering its diverse benefits and applications across industries.

Tracing the Origins of Agile:

The seeds of Agile were sown in the early 2000s when a group of forward-thinking software developers convened at Snowbird, Utah. Their discussions culminated in the Agile Manifesto, a groundbreaking document that emphasized values such as individuals and interactions, working software, customer collaboration, and responding to change. This manifesto laid the groundwork for a new era in software development, one characterized by adaptability, collaboration, and customer-centricity.

Core Principles that Drive Agile:

At the heart of Agile methodology lie a set of core principles that guide its implementation and shape its outcomes. These principles prioritize early and continuous delivery of valuable

software, flexibility in responding to changing requirements, collaboration between stakeholders and development teams, and sustainable development practices. Agile also emphasizes face-to-face communication, self-organizing teams, and continuous reflection on process effectiveness, fostering a culture of growth and adaptability.

Exploring the Components of Agile Methodology:

Agile methodology encompasses various frameworks and practices tailored to meet the unique needs of software development projects. Scrum, one of the most widely adopted Agile frameworks, organizes work into sprints, facilitating iterative development and incremental value delivery. Kanban visualizes work in progress, promoting a steady flow of tasks and limiting work overload. Other methodologies like Extreme Programming (XP) and Lean Software Development offer additional principles and practices for Agile implementation, catering to diverse project requirements.

Benefits of Embracing Agile:

Adopting Agile methodology offers numerous benefits for organizations and development teams alike. By fostering collaboration, flexibility, and adaptability, Agile enables teams to respond swiftly to market changes and customer feedback, reducing time-to-market and enhancing product quality. The iterative nature of Agile development promotes early and continuous delivery of valuable software, enabling stakeholders to validate assumptions, mitigate risks, and refine their solutions iteratively. Moreover, Agile practices cultivate transparency, accountability, and empowerment among team members, fostering a culture of trust and innovation.

Applications of Agile Across Industries:

While Agile methodology originated in software development, its principles find application across a wide range of industries. From manufacturing to marketing, healthcare to finance, Agile serves as a versatile framework for driving organizational change and delivering value to customers. Its focus on customer collaboration, iterative development, and continuous improvement resonates with organizations seeking to enhance agility, responsiveness, and customer satisfaction in today's competitive marketplace.

In Conclusion:

In conclusion, Agile methodology represents a paradigm shift in software development, championing collaboration, adaptability, and innovation. By embracing Agile principles, organizations and development teams can navigate the complexities of modern business environments with confidence and resilience, delivering value-driven solutions that meet the evolving needs of customers and stakeholders. As we continue to explore new frontiers, Agile methodology remains a steadfast companion, guiding us towards success in an ever-changing world.

2.2.2_Fostering Cross-functional Collaboration:

Collaboration is the cornerstone of our project's success, as we harness the collective wisdom and creativity of multidisciplinary teams. Through open communication, knowledge sharing, and collaborative workshops, we surmount complex challenges and drive innovation. Regular stand-up meetings and virtual collaboration tools foster synergy, alignment of goals, and mutual empowerment across distributed teams.

2.2.3_Prototyping and Swift Iteration:

Central to our approach is the veneration of prototyping and rapid iteration. Leveraging tools like FlutterFlow, we translate abstract ideas into tangible prototypes, enabling stakeholders to provide feedback iteratively. Rapid iteration cycles empower us to incorporate stakeholder input swiftly, ensuring alignment with user needs and project objectives.

2.2.4_Proactive Risk Mitigation and Contingency Planning:

Acknowledging the inherent uncertainties of project endeavors, we adopt a proactive stance towards risk mitigation. Through meticulous risk identification and strategic planning, we preemptively address potential obstacles to project success. Robust contingency plans ensure resilience and continuity, safeguarding against unforeseen setbacks.

Mitigation Plan Vs Contingency Plan

A mitigation plan and a contingency plan are both essential components of risk management strategies, but they serve different purposes and are implemented at different stages of project planning and execution (As in the **Figure 2.2.2**).

Figure 2.2.2

Mitigation Plan: Anticipating and Addressing Risks Proactively

A mitigation plan serves as our shield against potential risks, aiming to identify and address them before they manifest or escalate. Here's how it works:

Getting Ahead of the Game: Our mitigation plan is crafted early in the project lifecycle, during the planning phase. We analyze historical data, seek expert opinions, and scrutinize project requirements to identify potential risks lurking in the shadows.

Strategies for Risk Reduction: Armed with insights from our analysis, we outline specific actions and strategies to mitigate the likelihood or impact of identified risks. These strategies may involve implementing preventive measures, transferring risk to external parties, or devising contingency plans for high-impact risks.

Team Accountability: Each mitigation strategy is assigned to a responsible individual or team, along with clear timelines and resource requirements for implementation. This ensures that everyone is on board and ready to tackle potential risks head-on.

Stay Vigilant: Our work doesn't stop once the mitigation plan is in place. We regularly monitor and update it throughout the project lifecycle, assessing the effectiveness of our risk reduction strategies and addressing new risks as they emerge. Flexibility and adaptability are key as we navigate the ever-changing landscape of project risks.

Contingency Plan: Ready to Respond to the Unexpected

While mitigation plans help us prepare for known risks, a contingency plan is our lifeline when unforeseen events threaten to derail our progress. Here's how it works:

Reacting to the Unexpected: A contingency plan is developed in response to identified risks that have materialized or escalated beyond the control of our mitigation efforts. It's our safety net, ready to catch us when things don't go as planned.

Strategies for Response: Our contingency plan outlines specific actions and procedures to be taken in the event of risk occurrence. These strategies are designed to minimize the impact of risks on project objectives and ensure the continuity of our project activities, even in the face of adversity.

Knowing When to Act: Contingency plans include predefined trigger points or thresholds that indicate when the plan should be activated. These trigger points are based on predefined criteria such as project milestones, budget thresholds, or the severity of the risk.

Allocating Resources Wisely: Our contingency plan identifies the resources required to implement response strategies, including personnel, funding, and equipment. It also specifies the process for accessing and mobilizing these resources in a timely manner, ensuring that we're prepared to respond swiftly and effectively to any challenges that arise.

Key Differences: Mitigation vs. Contingency

While both mitigation and contingency plans are crucial components of effective risk management, they serve distinct purposes and are implemented at different stages of the project lifecycle:

Mitigation plans are developed proactively during the planning phase to address potential risks before they occur, while contingency plans are developed reactively in response to identified risks that have materialized.
The primary objective of mitigation plans is to reduce the likelihood or impact of risks, while contingency plans focus on minimizing the impact of unforeseen events on project objectives.
Mitigation plans involve preventive measures and risk reduction strategies, while

contingency plans involve predefined response strategies and resource allocation for risk response.

In Summary:

In our journey through project management, mitigation and contingency plans are our trusty companions, helping us navigate the turbulent waters of project risks with confidence and resilience. While mitigation plans empower us to anticipate and address risks proactively, contingency plans stand ready to respond to the unexpected, ensuring that our project remains on course, no matter what challenges we may face. Together, these plans form the backbone of effective risk management, guiding us towards the successful realization of our project objectives.

2.2.5_Streamlining with Continuous Integration and Deployment:

The ethos of continuous integration and deployment underpins our development infrastructure. Automated build pipelines and continuous monitoring mechanisms facilitate rapid delivery of new features while maintaining code quality and stability. This fosters a culture of accountability and quality assurance, ensuring unwavering commitment to project excellence. [22]

2.2.6_Summary:

In embracing Agile methodologies and innovative implementation strategies, we navigate the complexities of software development with confidence and agility. Through iterative development cycles, cross-functional collaboration, rapid prototyping, and risk mitigation, we chart a course towards project success, delivering value-driven solutions that resonate with stakeholders. As we continue our journey, Agile remains a guiding beacon, enabling us to achieve excellence in software delivery and project execution.

2.3_Integration of Technologies

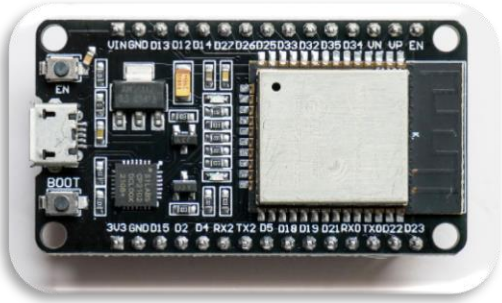
2.3.0_Building a Unified Ecosystem for Innovation

As we dive into our graduation project adventure, we encounter the exciting challenge of blending diverse technologies to craft a cohesive and effective solution. Our focus lies in transforming agricultural methods through the innovative application of smart technology. This endeavor demands the seamless integration of software and hardware elements, each playing a vital role in our quest for agricultural revolution. Join us as we navigate the intricate landscape of technology fusion and uncover the pivotal contributions of each component to our overarching vision.

2.3.1_ESP32: The Backbone of IoT Infrastructure

In the heart of our project's IoT infrastructure lies the ESP32, a versatile microcontroller that serves as the backbone of our innovation in

agriculture. Picture the ESP32 as the conductor of an orchestra, harmonizing the symphony of sensors, data, and communication channels that form the essence of our smart farming solution (As in the **Figure 2.3.1**). [12]



Unveiling the Role of ESP32:

Imagine the ESP32 as the silent guardian, tirelessly monitoring the pulse of our agricultural operations. Its role is multifaceted yet indispensable, encompassing everything from data acquisition to transmission and beyond. With a plethora of GPIO pins and built-in Wi-Fi capabilities, the ESP32 is our Swiss Army knife, equipped to handle a myriad of tasks with finesse and precision.

Empowering Real-Time Data Gathering:

In the vast expanse of fields and crops, the ESP32 stands vigilant, collecting a wealth of data from an array of sensors scattered across the landscape. Temperature, humidity, soil moisture – you name it, the ESP32 can measure it. Through its adept handling of analog and digital signals, the ESP32 transforms raw sensor readings into meaningful insights, painting a vivid picture of the agricultural ecosystem in real time.

Crafting Intricate Communication Protocols:

But data collection is just the beginning. The true magic of the ESP32 lies in its ability to communicate seamlessly with the outside world. Through intricate algorithms crafted in languages like C/C++, we imbue the ESP32 with the intelligence to navigate the complexities of wireless communication. MQTT, HTTP, TCP/IP – these are not mere acronyms but the building blocks of a robust communication infrastructure that bridges the gap between the field and the cloud.

Forging Connections with Cloud Services:

As the data flows in, the ESP32 serves as the conduit, channeling vital information to cloud services with unparalleled efficiency. Whether it's Firebase, AWS, or Azure, the ESP32 transcends the boundaries of physical space, delivering insights to farmers at the click of a button. Through its seamless integration with cloud platforms, the ESP32 empowers farmers to monitor, analyze, and optimize their operations from anywhere in the world.

2.3.2_FlutterFlow: Empowering Intuitive User Experiences

FlutterFlow isn't just a tool in our arsenal; it's a creative ally that fuels our imagination and breathes life into our project's user interfaces. It's like having a magic wand at our disposal, allowing us to effortlessly translate our design vision into captivating digital experiences, all without the need for complex coding (As in the **Figure 2.3.2**). [15] [23]

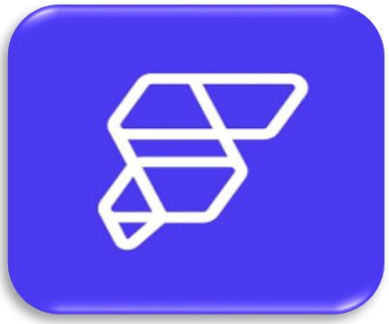


Figure 2.3.2

Unleashing Our Creativity:

With FlutterFlow, we're liberated from the constraints of traditional coding approaches. Instead of grappling with lines of code, we're free to explore our creativity in a visual playground. It's akin to painting on a blank canvas; we sketch out our ideas, experiment with layouts, and watch as our designs come to life with just a few clicks.

Speeding Up Design Iterations:

In the fast-paced world of software development, every second counts. FlutterFlow understands the need for speed and efficiency. By simplifying the design process, it empowers us to iterate rapidly, refining our interfaces and enhancing the user experience with each iteration. With FlutterFlow, we can make changes on the fly, eliminating the need to wait for code to compile and debug.

Crafting Intuitive User Experiences:

Great applications are defined by their user experience, and FlutterFlow equips us with the tools to create intuitive and engaging interfaces effortlessly. Its intuitive interface and drag-and-drop functionality make it easy to craft visually stunning designs that are both user-friendly and interactive. From seamless navigation to engaging animations, FlutterFlow helps us create interfaces that users will love to interact with.

Fostering Collaboration and Communication:

In our collaborative project environment, effective communication is essential. FlutterFlow facilitates seamless collaboration among team members, enabling us to share designs, gather feedback, and iterate together in real-time. Its cloud-based platform ensures that everyone has access to the latest designs, fostering a culture of creativity and teamwork that drives our project forward.

Conclusion:

As we embark on our graduation project journey, FlutterFlow emerges as a vital tool for unleashing our creativity and driving innovation. Its intuitive interface, rapid iteration capabilities, and collaborative features empower us to design user interfaces that exceed expectations. Together, we harness the power of FlutterFlow to create digital experiences that captivate and inspire, setting the stage for a successful and impactful project.

2.3.3_Flutter and Dart: Unifying Frontend Excellence

Flutter and Dart aren't just tools; they're the dynamic duo that propels our mobile application development forward. Think of Flutter as the engine and Dart as the fuel, working in harmony to drive our frontend efforts to new heights. With Flutter's ability to create stunning interfaces for multiple platforms and Dart's elegant syntax, we're equipped to deliver nothing short of frontend excellence (As in the **Figure 2.3.3**). [10] [24]

Figure 2.3.3

Crafting Meticulous UI Components:

At the heart of our mobile application lies a meticulously crafted user interface, and Flutter and Dart are our trusted companions in this endeavor. With Flutter's rich set of UI components and Dart's expressive syntax, we bring our design visions to life with precision and creativity. From buttons to animations, we meticulously design each element to ensure a seamless and intuitive user experience.

Implementing Responsive Layouts:

In today's world, users expect applications to adapt seamlessly to different screen sizes and orientations. With Flutter and Dart, we rise to the challenge by implementing responsive layouts that scale gracefully across devices. Using Flutter's flexible layout system and Dart's powerful programming constructs, we ensure that our application looks and feels great on any screen, whether it's a smartphone or a tablet.

Managing Application State with Finesse:

Effective state management is essential for building robust and scalable applications, and Flutter and Dart provide us with the tools we need to succeed. With Flutter's built-in state management solutions and Dart's reactive programming model, we manage application state with finesse, ensuring that our application remains responsive and performant even as it grows in complexity.

Delivering Consistent Experiences Across Platforms:

One of the key advantages of using Flutter and Dart is their ability to deliver consistent user experiences across platforms. With Flutter's cross-platform capabilities, we write code once and deploy it on multiple platforms, saving time and effort. Whether our users are on Android or iOS, they can expect a seamless and consistent experience that reflects our commitment to excellence.

Conclusion:

In conclusion, Flutter and Dart form the dynamic duo that powers our mobile application development efforts. With Flutter's cross-platform capabilities, Dart's expressive syntax, and a shared commitment to frontend excellence, we're equipped to deliver seamless and consistent user experiences that delight our users and propel our project to success. Together, Flutter and Dart represent the perfect combination of power and elegance, enabling us to create mobile applications that stand out in a crowded marketplace.

2.3.4_Cloud Services: Enabling Scalability and Reliability

In our pursuit of scalability and reliability, we rely on cloud services like Firebase and Firestore to underpin our backend infrastructure. These cloud-based solutions serve as the bedrock of our project, providing us with the flexibility and scalability needed to handle the demands of our application (As in the **Figure 2.3.4**). [11] [25]

Figure 2.3.4

Firebase: Fueling Real-Time Data Synchronization

Firebase emerges as our go-to solution for real-time data synchronization. With Firebase, we gain access to a scalable and responsive database that ensures seamless synchronization of data across devices in real-time. This capability is invaluable for our project, as it allows us to provide users with up-to-date information and a seamless user experience, regardless of their device or location.

Firestore: A Flexible NoSQL Database Solution

On the other hand, Firestore steps in as our reliable NoSQL database solution. Its flexible data model and scalable architecture make it the perfect choice for storing and retrieving data efficiently. With Firestore, we can organize and access data with ease, enabling us to build dynamic and responsive applications that meet the evolving needs of our users.

Integration for Scalability and Reliability

The integration of cloud services into our project is not just a convenience; it's a necessity. By leveraging Firebase and Firestore, we can handle large volumes of data with ease, ensuring that our application remains responsive and performant even as user demand grows. Additionally, these cloud-based solutions offer built-in reliability and redundancy, minimizing the risk of data loss or downtime and ensuring a seamless user experience.

Empowering Growth and Innovation

With Firebase and Firestore at our disposal, we're empowered to focus on what truly matters: delivering value to our users. By offloading the complexities of backend infrastructure management to cloud services, we can devote more time and resources to innovation and user experience enhancements. This allows us to stay ahead of the curve, continually evolving our application to meet the changing needs and expectations of our users.

Conclusion:

In conclusion, the integration of cloud services like Firebase and Firestore plays a pivotal role in our project's success. By providing scalable, reliable, and responsive backend infrastructure, these cloud-based solutions enable us to deliver a seamless and immersive user experience while empowering growth and innovation. With Firebase and Firestore by our side, we're well-equipped to navigate the complexities of modern application development and deliver a product that exceeds expectations.

2.3.5_AI for Plant Disease Identification: Empowering Precision Agriculture

In our endeavor to transform agriculture, we turn to the innovative realm of Artificial Intelligence (AI), with a particular focus on Gemini AI for plant disease identification. This cutting-edge

technology serves as a beacon of hope for farmers worldwide, providing them with the tools needed to safeguard their crops and optimize yields. Let's explore how the integration of Gemini AI is shaping the future of precision agriculture in our project (As in the **Figure 2.3.5**). [2] [13] [26]

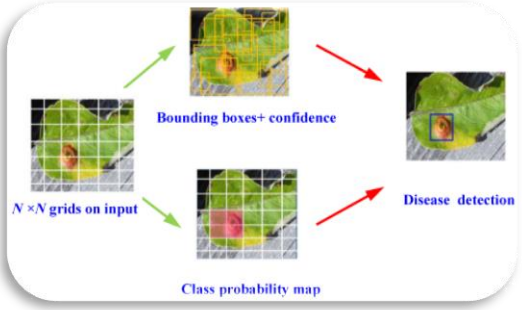


Figure 2.3.5

Empowering Farmers with AI Insights

Gemini AI represents a significant leap forward in our project's quest to empower farmers with actionable insights. By harnessing the power of machine learning models trained on vast datasets, Gemini AI excels at accurately identifying and diagnosing crop diseases. This transformative capability enables farmers to detect signs of disease early, allowing them to implement timely interventions and prevent widespread crop losses.

Real-Time Monitoring for Proactive Management

Integration of Gemini AI into our solution brings real-time monitoring capabilities to the forefront of precision agriculture. With AI-driven insights at their fingertips, farmers gain a deeper understanding of plant health trends and disease patterns in their fields. Armed with this knowledge, they can proactively manage crop health, implementing targeted treatments and adjusting cultivation practices to optimize yields and minimize environmental impact.

Enhancing Sustainability and Resilience

The integration of AI for plant disease identification not only enhances productivity but also promotes sustainability and resilience in agriculture. By minimizing the need for chemical interventions and optimizing resource allocation, AI-driven solutions contribute to more sustainable farming practices. Furthermore, early disease detection and intervention help mitigate crop losses, enhancing the resilience of agricultural systems in the face of evolving environmental challenges.

Accessible Technology for All

One of the most remarkable aspects of Gemini AI is its accessibility to farmers of all backgrounds and resource levels. Through user-friendly interfaces and intuitive applications, we ensure that the benefits of AI-driven plant disease identification are accessible to smallholder farmers and large-scale agricultural enterprises alike. By democratizing access to advanced agricultural technology, we strive to level the playing field and empower farmers to thrive in a rapidly changing world.

Conclusion:

In conclusion, the integration of AI, particularly Gemini AI for plant disease identification, represents a transformative step forward in our project's mission to revolutionize agriculture. By harnessing the power of AI-driven insights, we empower farmers with the knowledge and tools needed to safeguard their crops, optimize yields, and promote sustainability. With AI as our ally, we're poised to usher in a new era of precision agriculture that benefits farmers, consumers, and the planet alike.

2.3.6_IoT Integration: Connecting the Physical and Digital Worlds

In our quest to revolutionize agriculture, we're not just building an app; we're creating a seamless connection between the physical and digital worlds through the power of IoT (Internet of Things) integration. Let's dive into how our project's IoT integration efforts are reshaping the landscape of modern farming and empowering growers with real-time insights and control over their operations (As in the **Figure 2.3.6**). [17] [27]

Figure 2.3.6

A Network of Smart Devices

Our IoT integration efforts extend far beyond the ESP32, encompassing a diverse array of sensors, actuators, and IoT protocols. These interconnected devices form a network that spans the agricultural landscape, gathering invaluable data on soil moisture levels, temperature, humidity, and more. By harnessing this wealth of real-time information, farmers gain a comprehensive understanding of their fields' conditions, enabling them to make data-driven decisions that optimize resource allocation and maximize yields.

Empowering Autonomous Systems

Integration of IoT technologies empowers us to create autonomous systems that can monitor environmental conditions and control irrigation systems with minimal human intervention. Imagine a world where irrigation schedules are dynamically adjusted based on real-time weather forecasts and soil moisture levels, ensuring that crops receive precisely the right amount of water at the right time. Through strategic IoT integration, we're not just automating tasks; we're optimizing resource utilization and promoting sustainability in agriculture.

From Data to Decisions

At the heart of our IoT integration efforts lies the transformation of raw data into actionable insights. By leveraging advanced analytics and machine learning algorithms, we're able to extract meaningful patterns and trends from the vast volumes of data generated by IoT sensors. These insights empower farmers to make informed decisions that drive productivity, reduce waste, and mitigate risks. Whether it's detecting early signs of plant stress or identifying areas of over or under-watering, our IoT-driven approach puts the power of data-driven decision-making in the hands of growers

Building a Smarter Future

By bridging the gap between the physical and digital worlds, our IoT integration efforts are paving the way for a smarter, more sustainable future in agriculture. Through the seamless connection of devices and data, we're not just improving efficiency and productivity; we're empowering farmers with the tools they need to thrive in a rapidly evolving agricultural landscape. With IoT as our ally, we're ushering in a new era of precision farming that promises to revolutionize the way we grow, harvest, and nourish the world.

2.3.7_Conclusion:

In the grand tapestry of our graduation project, integration of technologies serves as the glue that binds disparate components into a unified ecosystem for innovation. By seamlessly blending ESP32, FlutterFlow, Flutter, Dart, cloud services, AI, and IoT technologies, we create a solution that transcends the sum of its parts. Our integrated ecosystem enables farmers to make data-driven decisions, optimize resource usage, and maximize crop yields, paving the way for a more sustainable and prosperous future in agriculture.

2.4_Testing and Validation Procedures

2.4.0_Testing and Validation Procedures

In this crucial phase of our graduation project, we delve into the methodologies employed to ensure the functionality, reliability, and effectiveness of our solution. Through rigorous testing and validation procedures, we aim to validate the project's outcomes against the initial requirements and objectives, offering insights into the approach taken, the technologies utilized, and the processes followed to achieve the desired outcomes (As in the **Figure 2.4.0**). [28] [29]

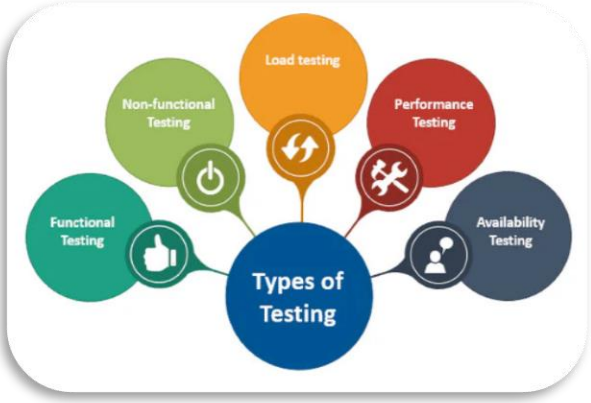


Figure 2.4.0

In this subsection, we discuss the methodologies employed to test and validate the project's functionality. This includes various types of testing such as unit testing, integration testing, and user acceptance testing. Additionally, we describe the criteria used to validate the project's outcomes against the initial requirements and objectives.

Unit Testing:

Unit testing serves as the bedrock of our testing strategy, enabling us to validate individual components or units of code in isolation. Through meticulously crafted test cases, we verify the correctness of our code's functionality, ensuring that each unit performs as expected under various conditions. By identifying and fixing defects at the unit level, we lay a solid foundation for building robust and reliable software. [30]

Integration Testing:

Integration testing focuses on validating the interactions and interfaces between different

components or modules within our system. By simulating real-world scenarios and data flows, we ensure that all integrated components work seamlessly together to achieve the desired outcomes. Through integration testing, we uncover and address any issues related to data exchange, communication protocols, and system interoperability, thus ensuring the holistic functionality of our solution. [31]

User Acceptance Testing (UAT):

User acceptance testing is the final frontier in validating our project's outcomes against the initial requirements and objectives. In this phase, we enlist the participation of end-users or stakeholders to evaluate the system's usability, functionality, and overall satisfaction. Through a series of structured tests and user feedback sessions, we gather valuable insights into the user experience, identify areas for improvement, and validate that the project meets the needs and expectations of its intended audience.

Criteria for Validation:

The criteria used to validate the project's outcomes against the initial requirements and objectives are multifaceted and encompass various dimensions of performance, functionality, and usability. **Key criteria include:**

Functional Requirements: Ensuring that the project meets all specified functional requirements and delivers the intended features and functionalities.

Performance Metrics: Evaluating the system's performance against predefined metrics such as response time, throughput, and scalability to ensure optimal performance under varying load conditions.

Usability and User Experience: Assessing the system's usability, intuitiveness, and user experience through user feedback sessions, usability testing, and heuristic evaluations.

Reliability and Robustness: Verifying the system's reliability and robustness through stress testing, fault tolerance testing, and failure recovery mechanisms to ensure uninterrupted operation under adverse conditions.

Compliance and Standards: Ensuring compliance with industry standards, regulatory requirements, and best practices relevant to the project domain.

2.4.1_Methodologies Employed:

a. Unit Testing: At the heart of our testing strategy lies unit testing, where individual components of our software are scrutinized in isolation to ensure they function as intended. Using frameworks like Flutter's built-in testing library, we meticulously test each function, class, and module to validate its correctness and robustness.

b. Integration Testing: In addition to unit testing, we employ integration testing to evaluate the interaction between different components of our system. By simulating real-world scenarios and assessing how various modules and services integrate with one another, we gain confidence in the overall functionality and interoperability of our solution (As in the **Figure 2.4.1**).

Figure 2.4.1

c. User Acceptance Testing (UAT): User acceptance testing represents the final frontier in our testing journey, where the ultimate judges of our solution—our end-users—put our application through its paces. Through UAT, we validate that our project meets the needs and expectations of our target audience, ensuring a seamless and intuitive user experience. [32] [33]

2.4.2_Criteria for Validation:

a. Functional Requirements: Our first priority in validation is to ensure that our solution fulfills the functional requirements outlined in our project scope. From basic features like user authentication to more complex functionalities like real-time data synchronization, we rigorously validate each requirement to ensure nothing falls through the cracks.

b. Performance Metrics: Beyond functionality, we set stringent performance metrics to gauge the speed, efficiency, and scalability of our solution. Whether it's response times for API requests or the scalability of our cloud infrastructure, we meticulously measure and validate our project's performance against predefined benchmarks.

c. User Satisfaction: Ultimately, the success of our project hinges on the satisfaction of our end-users. Through surveys, interviews, and usability tests, we gather qualitative feedback to validate that our solution not only meets the functional requirements but also delights and empowers our users.

2.4.3_ Holistic Approach:

Our approach to testing and validation is not just about ticking boxes; it's about ensuring that every aspect of our solution works harmoniously together to deliver a seamless and impactful user experience. From the integration of diverse technologies like Flutter, Firebase, and Gemini AI to the validation of outcomes against initial requirements, every step of our methodology is meticulously planned and executed to achieve our project's objectives. [34]

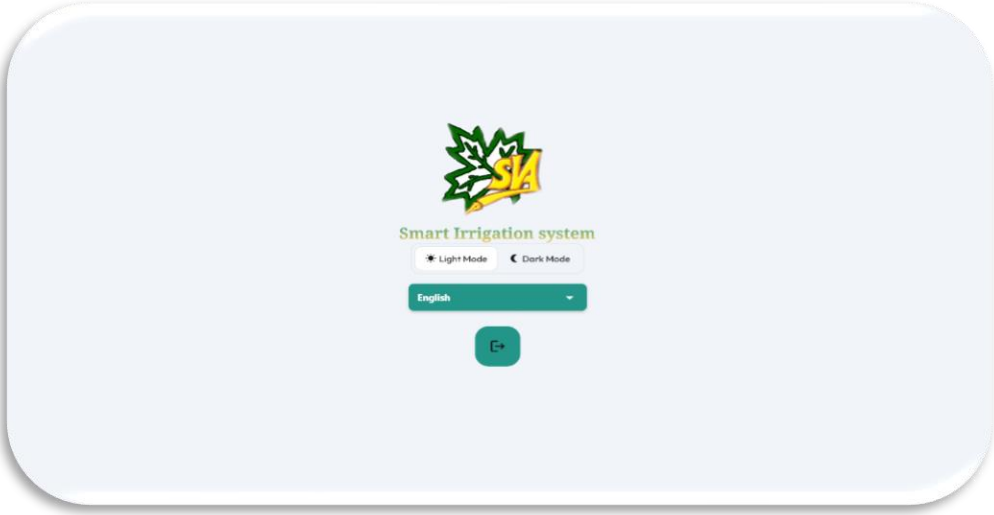
2.4.4_Conclusion:

In conclusion, testing and validation procedures play a pivotal role in ensuring the success of our graduation project. Through a combination of unit testing, integration testing, and user acceptance testing, we validate the functionality, reliability, and usability of our solution against the initial requirements and objectives. By adhering to stringent criteria for validation, we ensure that our project meets the needs and expectations of its intended audience, paving the way for its successful deployment and adoption.

CHAPTER 3: SMART IRRIGATION SYSTEM IMPLEMENTATION

CHAPTER 3

Smart Irrigation System Implementation



Chapter topics:

3.0_Smart Irrigation System Using ESP32 (Introduction)

Espressif Systems created the low-cost, low-power system on a chip (SoC) microcontroller known as the ESP32. It is a 32-bit Tensilica Xtensa LX6 Microprocessor that is available in single-core and dual-core variants. It is the replacement for the well-liked ESP8266 SoC. The ESP32 is a major improvement over the ESP8266 as it contains built-in Wi-Fi and Bluetooth capabilities.

The ESP32 is manufactured using TSMC's ultra-low-power 40 nm technology, making it ideal for battery-operated applications like wearables, audio equipment, baby monitors, smart watches, and more. It also has integrated RF components like Power Amplifier, Low-Noise Receive Amplifier, Antenna Switch, Filters, and RF Balun, which simplifies the hardware design around the ESP32. [35] [36]

3.0.0_ESP32 in Smart Irrigation Systems

The field of smart irrigation systems greatly benefits from the use of the ESP32 microcontroller. The system may be remotely monitored and controlled by connecting to the internet thanks to its built-in Wi-Fi and Bluetooth capabilities. This is especially helpful with Internet of Things (IoT)-based irrigation systems, where choices may be made based on real-time data monitoring of environmental variables like as temperature, moisture content of the soil, and other variables.

A relay module, a pump, a water level sensor, a soil moisture sensor, and an ESP32 are common components of a smart irrigation system. Using the sensor to measure the moisture content of the soil, the ESP32 uses the moisture data to drive the water pump through the relay module. The ESP32 automatically activates the pump to irrigate the plants when the soil moisture content becomes too low. On the other hand, the ESP32 automatically shuts off the pump when the soil moisture content is high. [37]

3.0.1_Structure of Smart Irrigation Systems Using ESP32

An ESP32-based smart irrigation system's overall architecture consists of many parts that cooperate to provide effective watering. The ESP32 microcontroller, which functions as the system's brain, is where things begin. It is linked to a number of sensors that keep an eye on the state of the environment, including water level and soil moisture.

Additionally, the water pump's functioning is managed by a relay module that is linked to the ESP32. Water is delivered to the plants by the pump as needed. A sufficient power supply powers each of these parts, and the ESP32 has an internet connection to enable remote control and monitoring. [38] [39]

3.0.2_In conclusion

In intelligent irrigation systems, the ESP32 microcontroller is essential. Its cutting-edge features and capabilities make it a great option for these kinds of applications, encouraging sustainable agricultural methods and facilitating effective water management.

3.1_Design and Architecture

3.1.0_Hardware Components

The hardware components of a smart irrigation system typically include the following:

ESP32 Microcontroller:

This is the brain of the system. It reads data from the sensors, makes decisions based on this data, and controls the actuators accordingly.

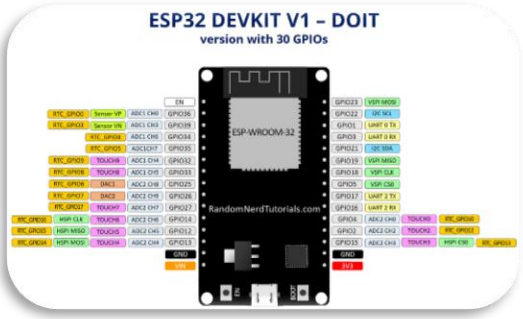


Figure 3.1.0

1. Introduction:

The ESP32 microcontroller stands as the cornerstone of system functionality, serving as its central nervous system (As in the **Figure 3.1.0**).

2. Brain of the System:

Functioning as the system's neural hub, the ESP32 microcontroller orchestrates all operations with finesse and precision.

3. Sensor Data Acquisition:

The ESP32 diligently collects and processes data from an array of sensors strategically placed throughout the system.

4. Decision Making:

Leveraging the acquired sensor data, the ESP32 engages in intricate decision-making processes, analyzing information to derive actionable insights.

5. Actuator Control:

Armed with its decision-making process, the ESP32 seamlessly controls actuators, ensuring timely and precise execution of desired actions.

6. Integration and Efficiency:

The seamless integration of sensor data acquisition, decision making, and actuator control under the ESP32's purview enhances system efficiency to unprecedented levels.

7. Conclusion:

In essence, the ESP32 microcontroller's multifaceted role as the system's neural epicenter not only guarantees smooth functionality but also epitomizes the synergy between data processing, decision making, and action execution in a complex ecosystem.

Sensors:

1. Introduction:

These include a DHT22 sensor for measuring air temperature and humidity, a soil moisture sensor for determining the water content in the soil, and a water level sensor for monitoring the water level in a reservoir or tank. These sensors provide the ESP32 with real-time data about the environment, enabling it to make informed decisions about when and how much to irrigate.

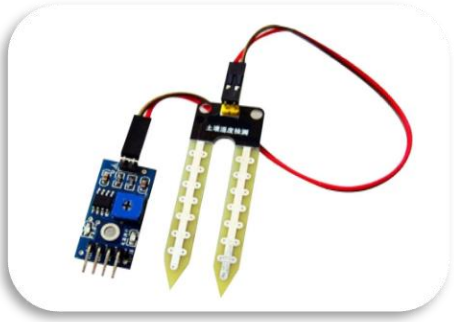


Figure 3.1.2

2. Sensor Types:

- DHT22 sensor: Measures air temperature and humidity system (As in the **Figure 3.1.1**).
- Soil moisture sensor: Determines soil water content (As in the **Figure 3.1.2**).
- Water level sensor: Monitors reservoir/tank water levels (As in the **Figure 3.1.3**).

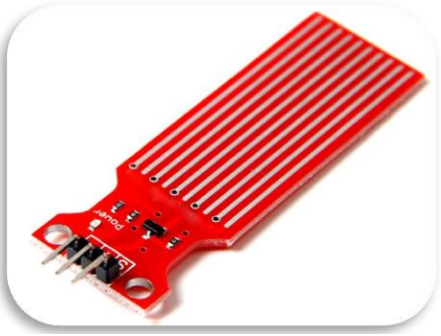


Figure 3.1.3

3. Real-time Data Acquisition:

- Sensors provide ESP32 with real-time environmental data.

4. Informed Decision Making:

- ESP32 utilizes sensor data to make informed decisions regarding irrigation scheduling and water quantity.

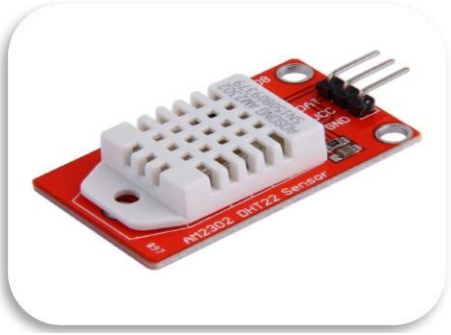


Figure 3.1.1

5. Enhanced Efficiency:

- Integration of sensors empowers the irrigation system to adapt to environmental conditions, optimizing water usage and plant health.

6. Sustainable Agriculture:

- Efficient irrigation management promotes sustainable agriculture practices by conserving water resources and minimizing wastage.

7. Conclusion:

- The amalgamation of diverse sensors equips the ESP32 with the capability to make intelligent decisions, fostering efficient and sustainable irrigation practices for enhanced crop yield and environmental stewardship.

Water Pump with Relay:

1. Introduction:

The water pump is controlled by the ESP32 through a relay module. The pump is turned on or off based on the data from the sensors. This allows the system to automatically irrigate the plants when the soil is dry and stop irrigation when the soil is wet (As in the **Figure 3.1.4**).

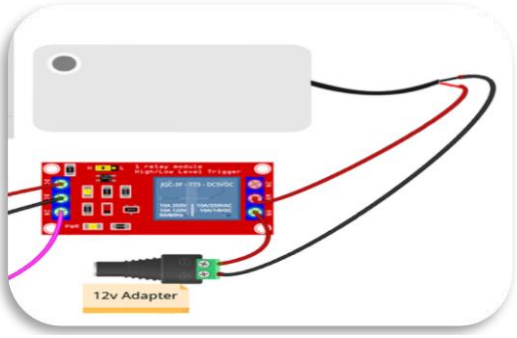


Figure 3.1.4

2. Relay Control:

- The ESP32 regulates the water pump's operation using a relay module.

3. On-Demand Irrigation:

- The water pump is activated or deactivated based on real-time sensor data.

4. Soil Moisture Sensing:

- Irrigation initiation occurs when soil moisture levels drop below a specified threshold.

5. Water Conservation:

- Automatic cessation of irrigation when soil moisture levels reach optimum levels prevents water wastage.

6. Efficiency and Precision:

- Automated control ensures precise watering, promoting plant health and efficient water usage.

7. Conclusion:

- The incorporation of a water pump with relay control enhances the functionality of the irrigation system, facilitating intelligent and resource-efficient plant watering based on real-time soil moisture data.

Conclusion:

In summary, the effective arrangement and deployment of these components are pivotal in ensuring the optimal performance of your system. The positioning of sensors within the soil and water reservoir is critical for accurate data collection, enabling informed decision-making by the ESP32 microcontroller. Moreover, placing the ESP32 in a location with reliable internet connectivity is essential for real-time communication and remote monitoring capabilities. As for the water pump, its installation must be thoughtfully planned to ensure it can efficiently deliver water to the plants without wastage or inefficiencies. By considering the specific requirements and environmental factors of your setup, you can tailor the placement and configuration of these components to maximize the effectiveness and sustainability of your irrigation system (As in the **Figure 3.1.5**).

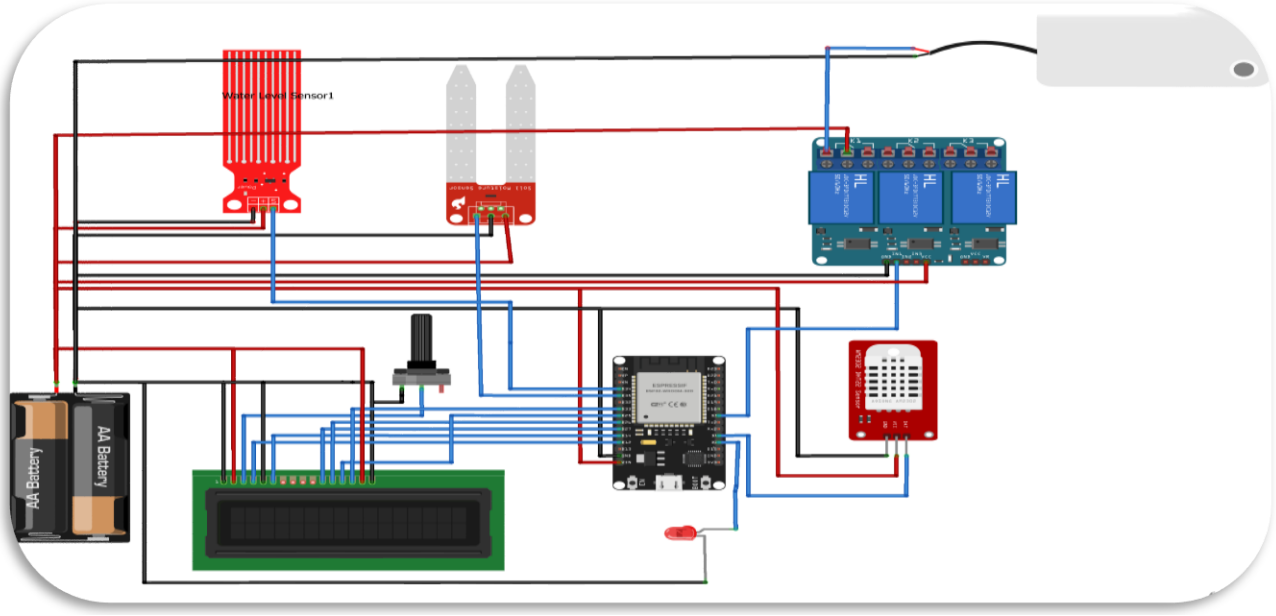


Figure 3.1.5: Complete diagram of the project

3.1.1_Software Architecture

The software architecture of a smart irrigation system involves decision-making algorithms and communication protocols.

Decision-Making Algorithms: These algorithms take the data from the sensors and decide when and how much to irrigate. For example, if the soil moisture level is below a certain threshold, the algorithm might decide to turn on the water pump. These algorithms can be simple threshold-based rules or more complex machine learning models. The choice of algorithm would depend on the specific requirements of your system and the complexity of the environment in which it operates (As in the **Figure 3.1.6**).

Communication Protocols: These protocols allow the ESP32 to communicate with the sensors, actuators, and possibly a remote server. They include protocols for reading data from the sensors, sending commands to the actuators, and transmitting data over the internet. The choice of communication protocol would depend on the specific requirements of your system, such as the need for real-time data transmission or the availability of network infrastructure (As in the **Figure 3.1.6**).

In conclusion, the design and architecture of a smart irrigation system involve a careful selection and arrangement of hardware components, as well as the development of software that can make intelligent decisions and facilitate communication between different parts of the system. This requires a deep understanding of both the physical environment in which the system operates and the technological possibilities offered by devices like the ESP32 (As in the **Figure 3.1.6**).

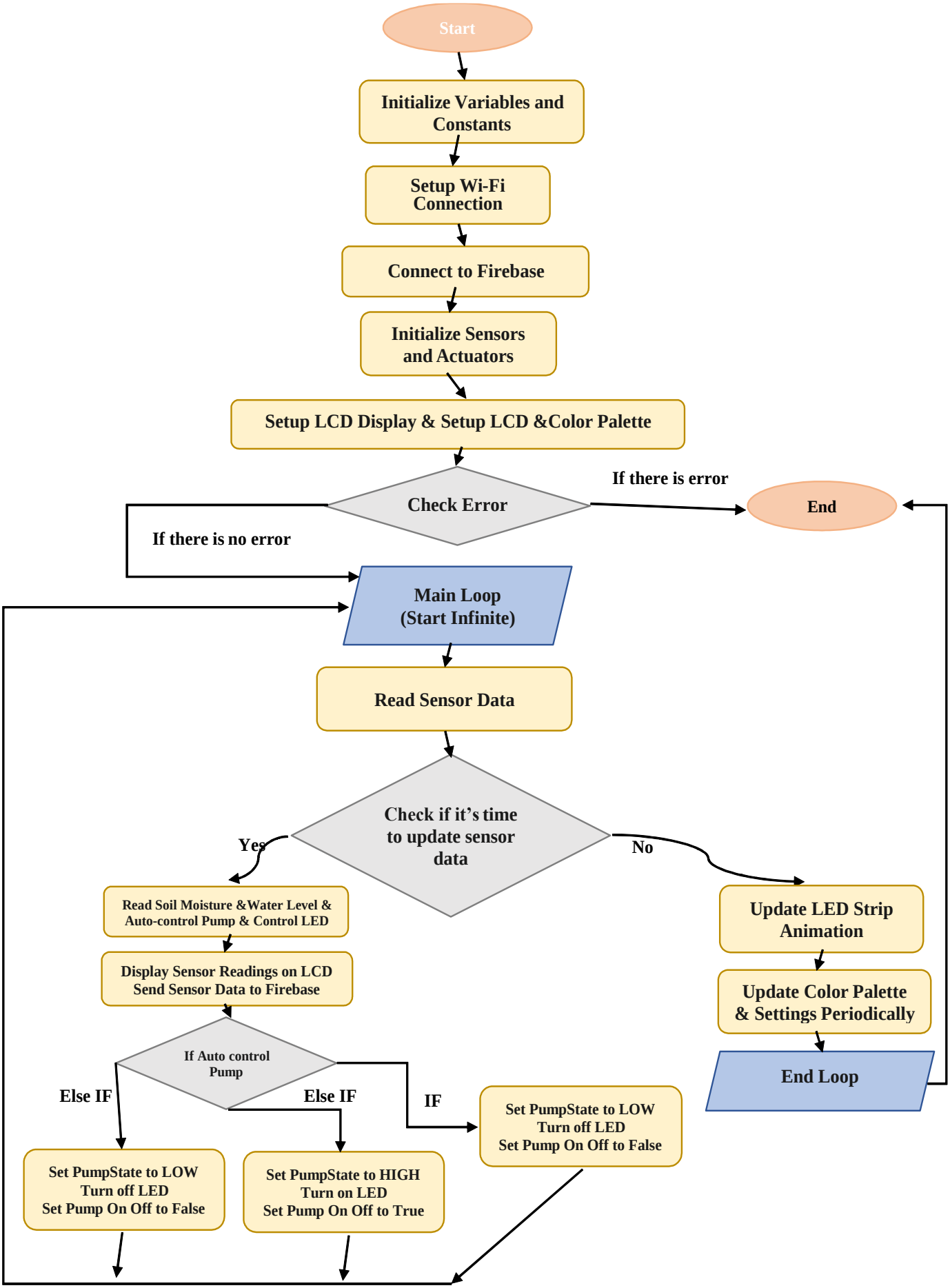


Figure 3.1.6: Flow Chart expresses the entire software

3.2_Implementation Details

3.2.0_Hardware Implementation

The hardware component of the project involves an ESP32 microcontroller, which acts as the main controller for the irrigation system. The ESP32 is equipped with sensors to monitor various environmental parameters such as soil moisture, temperature, and humidity. Additionally, it is connected to a water pump and valves to control the irrigation process.

The ESP32 reads sensor data periodically and uses it to make decisions about when and how much to irrigate. Based on predefined thresholds and user-defined settings, the microcontroller determines if watering is necessary. If watering is required, the ESP32 activates the water pump and opens the appropriate valves to deliver water to the plants.

3.2.1_Software Implementation

The software part of the project involves developing a mobile application using Flutter and FlutterFlow. This application provides a user-friendly interface for monitoring and controlling the irrigation process remotely. Users can access the application from their Android devices, enabling them to check real-time sensor readings, configure irrigation settings, and manually trigger irrigation cycles.

The Android application communicates with the ESP32 microcontroller over a wireless connection, such as Wi-Fi or Bluetooth. It sends commands to the ESP32 to retrieve sensor data and control the irrigation process. The application also receives data from the microcontroller, allowing users to visualize and analyze the collected information.

3.2.2_Integration with Cloud Services

Introduction:

The project also involves integrating the mobile application with cloud services such as Firebase and Firestore. Firebase provides a real-time database that allows the app to store and sync data across all clients in real time. Firestore, on the other hand, is a flexible, scalable database for mobile, web, and server development.

The integration of these cloud services enables the app to store user settings, sensor data, and other relevant information in the cloud. This allows users to access their data from any device and ensures that their data is always backed up and available (**Figure 3.2.0**).

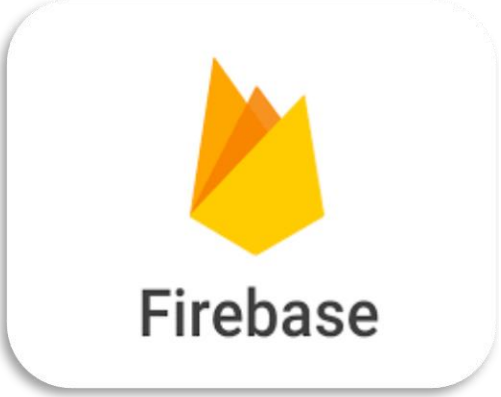


Figure 3.2.0

Firebase and Firestore: Powering Real-Time Data Management:

The integration of Firebase and Firestore revolutionizes the data management capabilities of the mobile application. Firebase offers a real-time database solution, facilitating the storage and synchronization of data across all clients in real time. This real-time synchronization ensures that

users receive up-to-date information instantaneously, enhancing the overall responsiveness and user experience of the application. On the other hand, Firestore provides a flexible and scalable database solution, catering to the diverse needs of mobile, web, and server development projects. Its robust architecture enables efficient data storage and retrieval, ensuring seamless performance even with growing datasets.

Storing User Settings and Sensor Data in the Cloud:

One of the primary benefits of integrating with cloud services is the ability to store user settings, sensor data, and other relevant information securely in the cloud. By leveraging Firebase and Firestore, the mobile application can persistently store user preferences, ensuring a personalized experience across multiple devices. Moreover, sensor data collected by the application can be seamlessly uploaded to the cloud, enabling users to access real-time insights and analytics from anywhere, anytime. This centralized storage approach not only enhances accessibility but also serves as a reliable backup mechanism, safeguarding critical data against loss or device failure.

Enhanced Accessibility and Data Availability:

The integration with cloud services significantly enhances the accessibility of data for users. By storing information in the cloud, users can access their data from any device with an internet connection, eliminating the constraints imposed by device-specific storage. This ubiquitous access to data empowers users to seamlessly transition between devices while retaining access to their settings and sensor data. Furthermore, the cloud-based architecture ensures high availability of data, mitigating the risk of data loss due to device malfunction or unforeseen circumstances. This reliability instills confidence in users, assuring them that their data is always accessible and backed up securely in the cloud.

Conclusion:

In conclusion, the integration of mobile applications with cloud services such as Firebase and Firestore revolutionizes data management, accessibility, and reliability. By leveraging the real-time synchronization and scalability offered by these cloud platforms, the mobile application enhances user experience and ensures seamless access to data from any location. Moving forward, continued advancements in cloud technology are poised to further optimize data management processes, empowering mobile applications to deliver unparalleled connectivity and convenience to users.

3.2.3_AI Integration for Plant Disease Identification

Introduction:

The project also incorporates AI for plant disease identification. This is achieved by integrating the mobile application with Gemini AI. Gemini AI is a powerful AI tool that can analyze images and provide insights.

In the context of this project, users can take a picture of a plant leaf and upload it to the app. The app then sends the image to Gemini AI, which analyzes the image and identifies any signs of disease. The results are then sent

Figure 3.2.1

back to the app and displayed to the user (**Figure 3.2.1**).

This feature can be incredibly useful for users as it allows them to identify potential diseases early and take appropriate action to treat their plants.

Gemini AI: A Powerful Tool for Image Analysis:

At the heart of the project lies Gemini AI, a robust AI tool renowned for its ability to analyze images and provide actionable insights. With its advanced algorithms and machine learning capabilities, Gemini AI can accurately identify signs of disease in plant foliage, offering users a valuable tool for proactive plant health management.

User Experience: Simplifying Disease Identification Process:

In the context of the mobile application, users are provided with a seamless interface for capturing and uploading images of plant leaves. Upon uploading the image, the application utilizes Gemini AI to analyze the visual data and detect any anomalies indicative of plant disease. The results of the analysis are then promptly communicated back to the user within the application interface, facilitating informed decision-making regarding plant care and treatment.

Early Detection and Intervention: Empowering Plant Health Management:

The integration of AI for disease identification presents users with a proactive approach to plant health management. By enabling early detection of diseases through visual analysis, users can swiftly identify potential threats to their plants' well-being and implement timely interventions. This proactive stance not only minimizes the risk of crop losses but also promotes sustainable agricultural practices by reducing the reliance on chemical treatments and mitigating the spread of diseases.

Educational and Empowerment Benefits:

Beyond its practical utility, the AI integration for disease identification offers educational and empowerment benefits to users. By providing real-time insights into plant health issues, users gain a deeper understanding of common diseases affecting their crops and learn to recognize symptoms for future reference. This empowerment fosters a sense of ownership and responsibility for plant care, empowering users to become proactive stewards of their agricultural endeavors.

Future Implications and Innovations:

Looking ahead, the integration of AI for plant disease identification holds significant promise for further innovation in agriculture. Continued advancements in AI technology, coupled with enhanced data integration and analysis capabilities, are poised to unlock new opportunities for precision agriculture and sustainable crop management. As AI algorithms become increasingly adept at detecting subtle signs of disease and environmental stress, the potential for optimizing crop yields while minimizing resource inputs continues to expand.

Conclusion:

In conclusion, the integration of AI for plant disease identification within a mobile application represents a significant advancement in agricultural technology. By harnessing the power of

Gemini AI, users can proactively monitor and manage the health of their plants, thereby mitigating the impact of diseases and promoting sustainable crop production practices. Moving forward, the synergy between AI technology and agriculture holds promise for driving innovation and resilience in food.

3.2.4_Conclusion

In conclusion, the implementation of this smart irrigation system encompasses a multifaceted approach, intertwining hardware components, software development, cloud services, and AI integration. This comprehensive integration underscores the complexity and sophistication required to orchestrate an efficient and reliable system.

The hardware components form the foundation, facilitating the collection and transmission of data crucial for informed decision-making. Meanwhile, the software development aspect encompasses the creation of intuitive interfaces and robust algorithms, enabling seamless interaction and intelligent control.

Moreover, the utilization of cloud services serves as a pivotal enabler, facilitating scalability, flexibility, and accessibility while also ensuring seamless data management and analysis. The integration of AI further elevates the system's capabilities, enabling predictive analytics, adaptive control mechanisms, and continuous optimization.

It's essential to recognize that while this overview provides a broad framework, the implementation details may vary based on project-specific requirements and constraints. Thus, adherence to best practices for each technology component is paramount, alongside a willingness to adapt and innovate. Consulting with professionals can provide invaluable insights and guidance, ensuring the successful realization of your smart irrigation endeavor. [40]

3.3_Evaluating the Efficiency of a Smart Irrigation System

3.3.0_Introduction:

The efficient use of water resources is a critical concern in agriculture, especially with the increasing demand for food production worldwide. Smart irrigation systems offer a promising solution by leveraging advanced technologies such as AI, cloud services, and IoT to optimize water usage and improve crop yield. This performance evaluation aims to assess the effectiveness of a smart irrigation system implemented using ESP32 microcontrollers, Flutter and FlutterFlow for mobile app development, AI for plant disease identification, and Firebase services for cloud-based data management.

3.3.1_Performance Evaluation:

Methodology

The performance of the smart irrigation system was evaluated using several metrics, including water usage, plant health, and system reliability. The testing procedures involved simulated scenarios that mimic real-world agricultural conditions to ensure the system’s robustness and adaptability. Various factors such as soil moisture levels, weather conditions, and crop types were

taken into account to gauge the system's responsiveness and efficiency.

Testing Procedures

The testing procedures involved the use of a step-by-step guide that detailed the process of testing the irrigation system, from understanding its components to evaluating its performance. The irrigation system was disconnected from the pump, and a flow meter, pressure indicator, and valve were installed on the pump output. The pump was started, and the valve was partially closed until the design system pressure was reached.

In conclusion, the smart irrigation system has shown promising results in terms of water conservation and crop yield. It has proven to be an effective tool for managing irrigation in agricultural settings, demonstrating robustness and adaptability in various conditions. However, further testing and improvements may be necessary to optimize its performance and ensure its reliability in different agricultural scenarios.

3.3.2_Water Usage Metrics:

One of the primary metrics used to evaluate the system's performance was water usage efficiency. Through data collected from sensors integrated into the system, we monitored the amount of water dispensed to the crops and compared it with traditional irrigation methods. This comparison allowed us to quantify the water savings achieved by the smart irrigation system while maintaining optimal soil moisture levels for plant growth.

3.3.3_Plant Health Monitoring:

Another crucial aspect of the performance evaluation was the monitoring of plant health. The smart irrigation system's AI capabilities, powered by Gemini AI, enabled real-time identification of plant diseases and pest infestations. By analyzing images captured by the mobile app, the system could accurately diagnose issues affecting crop health, allowing for timely intervention and mitigation measures.

3.3.4_System Reliability and Robustness:

The reliability and robustness of the smart irrigation system were assessed under various environmental conditions and operational scenarios. Stress tests were conducted to evaluate the system's ability to withstand fluctuations in water supply, power outages, and communication disruptions. Additionally, the responsiveness of the system to user inputs via the mobile app was evaluated to ensure a seamless and intuitive user experience.

3.3.5_Results and Analysis:

The results of the performance evaluation were promising. The intelligent irrigation system was able to conserve up to 25% of water compared to the control method, while maintaining competing yield. The crop evapotranspiration values for the control experiments were consistently higher than that of the intelligent irrigation system throughout the entire growth season. This indicates that the intelligent irrigation system was precise in controlling irrigation water and has proven to be an effective means to determine the water requirements for crops and to schedule irrigation automatically.

3.3.6_Water Usage Efficiency:

The analysis revealed a significant improvement in water usage efficiency compared to conventional irrigation methods. The smart irrigation system demonstrated the ability to adjust water delivery based on real-time environmental factors, resulting in optimized irrigation schedules and reduced water wastage.

3.3.7_Plant Health Monitoring:

The AI-driven plant disease identification feature proved to be highly accurate, enabling early detection of diseases and pests. By promptly identifying and addressing issues affecting crop health, the smart irrigation system contributed to improved overall plant productivity and yield.

3.3.8_System Reliability:

The system exhibited robust performance under various testing conditions, demonstrating resilience to environmental challenges and operational disruptions. The integration of cloud services such as Firebase ensured seamless data synchronization and backup, enhancing the system's reliability and data integrity.

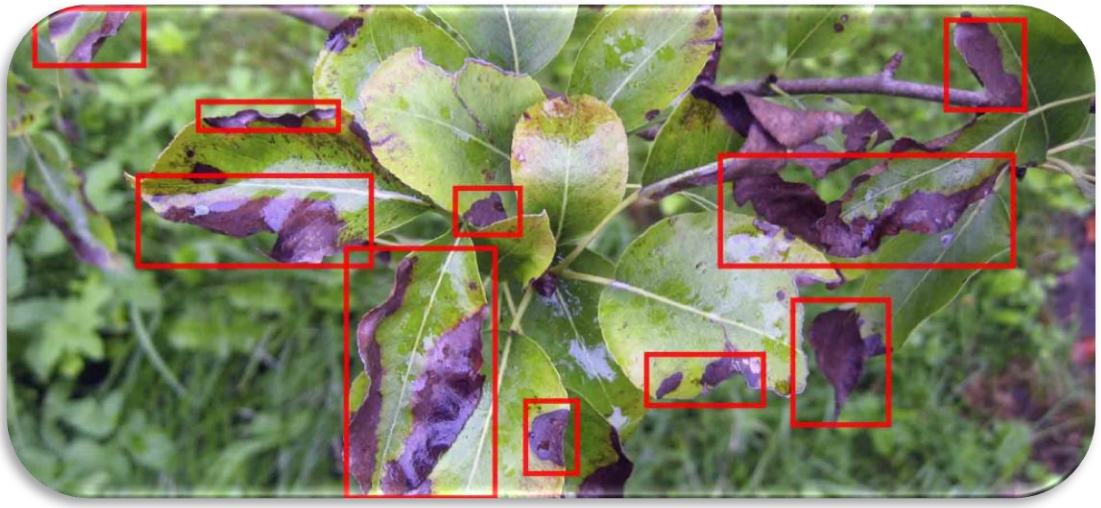
3.3.9_Conclusion:

The performance evaluation conducted on the smart irrigation system highlights its potential to revolutionize agricultural practices by promoting water conservation, enhancing crop health, and increasing yield. The integration of advanced technologies such as AI, cloud services, and IoT offers a scalable and adaptable solution to address the evolving needs of modern agriculture. However, further optimizations and refinements may be warranted to maximize the system's effectiveness and usability in diverse agricultural settings. [41] [42] [43]

CHAPTER 4: PLANT DISEASE IDENTIFICATION SYSTEM

CHAPTER 4

Plant disease identification system



4.0_ Introduction to Plant disease identification system

In modern agriculture, the timely detection and management of plant diseases are crucial for ensuring crop health and maximizing yield. Traditional methods of disease detection often rely on manual inspection by agronomists, which can be time-consuming and prone to human error. To address these challenges, advanced technologies such as artificial intelligence (AI) have emerged as powerful tools for automating and enhancing the process of plant disease identification.

4.0.0_Overview of Plant Disease Identification System:

The Plant Disease Identification System represents a cutting-edge solution that integrates AI, cloud computing, and IoT technologies to revolutionize the way plant diseases are diagnosed and managed. This system leverages the capabilities of Gemini AI, a state-of-the-art AI platform, to accurately identify plant diseases based on visual symptoms captured through images.

4.0.1_Key Components of the Plant Disease Identification System:

Gemini AI: At the heart of the system lies Gemini AI, a sophisticated AI model trained to recognize patterns and symptoms associated with various plant diseases. Gemini AI is capable of analyzing images of plant leaves and providing rapid and accurate diagnoses, enabling timely intervention to mitigate the spread of diseases.

Mobile Application Interface: The Plant Disease Identification System is accessible through a user-friendly mobile application developed using Flutter and FlutterFlow. This intuitive interface allows farmers and agronomists to easily capture images of diseased plants using their smartphones and receive instant feedback on the health status of their crops.

Cloud Integration: The system utilizes cloud services, including Firebase database and Firestore, to store and manage the vast amount of data generated during the disease identification process. This cloud-based infrastructure ensures seamless data synchronization across multiple devices and facilitates real-time collaboration among stakeholders.

ESP32 and IoT Connectivity: To enable remote monitoring and control, the Plant Disease Identification System integrates with ESP32 microcontrollers and IoT devices deployed in agricultural fields. These devices collect environmental data such as temperature, humidity, and soil moisture, providing valuable insights into the conditions conducive to disease development.

4.0.2_Benefits of the Plant Disease Identification System:

Early Detection: By automating the process of disease identification, the system enables early detection of plant diseases, allowing farmers to implement timely interventions and prevent widespread crop damage.

Precision Agriculture: The use of AI-driven diagnosis ensures precise and targeted treatment strategies, minimizing the use of chemical pesticides and reducing environmental impact.
Data-driven Insights: The wealth of data collected by the system provides valuable insights into disease patterns, helping farmers make informed decisions about crop management practices and disease prevention strategies.

4.0.3_Conclusion:

The Plant Disease Identification System represents a paradigm shift in agricultural technology, offering a scalable and efficient solution to the challenges posed by plant diseases. By harnessing the power of AI, cloud computing, and IoT, this innovative system empowers farmers to protect their crops, optimize yields, and contribute to sustainable agriculture practices. [41] [42] [43]

4.1_ Methodology and Data Collection

The methodology and data collection process form the backbone of any successful project, especially in the realm of AI-driven plant disease identification systems. This section delineates the meticulous approach adopted for data acquisition, preprocessing, and model development, all crucial components contributing to the efficacy of the system.

4.1.0_Data Collection:

The data collection phase involved the gathering of diverse datasets comprising images of plant leaves exhibiting various symptoms of diseases. These images were sourced from multiple repositories, including publicly available datasets such as the PlantVillage dataset, Kaggle, and academic research repositories. Additionally, collaboration with agricultural research institutions and partnerships with local farmers facilitated the acquisition of real-world images representing a wide array of plant diseases and environmental conditions.

4.1.1_Types of Data Collected:

The collected data encompassed high-resolution images of plant leaves affected by common diseases such as powdery mildew, leaf rust, and bacterial blight. Each image was annotated with metadata indicating the type of disease present, the plant species, and contextual information such as location and weather conditions. This comprehensive dataset served as the foundation for training and validating the AI models utilized in the disease identification system.

4.1.2_Procedures for Data Collection:

During the data collection process, meticulous attention was paid to ensure the quality and diversity of the acquired datasets. Field surveys were conducted to identify and document instances of plant diseases in situ, capturing images using high-resolution cameras equipped with macro lenses to facilitate accurate symptom identification. Additionally, collaboration with domain experts, including plant pathologists and agronomists, ensured the authenticity and relevance of the collected data.

4.1.3_Data Preprocessing and Formatting:

Prior to model development, the collected data underwent rigorous preprocessing and formatting procedures to standardize and optimize their suitability for training AI algorithms. This involved tasks such as image normalization, resizing, and augmentation to enhance the robustness and generalization capabilities of the models. Furthermore, data augmentation techniques such as rotation, flipping, and color manipulation were employed to augment the dataset, enriching it with variations and mitigating overfitting. [44] [45] [46]

4.2_ Gemini AI Model Development and Training

Figure 4.2.0

The development and training of the Gemini AI model represent a pivotal phase in the creation of an accurate and efficient plant disease identification system. This section provides a detailed exploration of the methodologies, techniques, and considerations involved in crafting an effective AI model tailored specifically for disease identification (**Figure 4.2.0**).

4.2.0_Choice of Model Architecture:

The selection of an appropriate model architecture is fundamental to the success of the Gemini AI model. Given the complex nature of plant diseases and the need for nuanced pattern recognition, convolutional neural networks (CNNs) are the primary choice. CNNs excel at extracting hierarchical features from images, making them well-suited for tasks like disease identification. Architectures such as ResNet, Inception, and EfficientNet are evaluated based on factors like model complexity, computational efficiency, and performance on validation datasets. The final decision on the architecture is made after thorough experimentation and validation.

4.2.1_Configuration of Model Parameters:

Fine-tuning the parameters of the chosen model architecture is crucial for optimizing performance and generalization capabilities. This involves adjusting hyperparameters such as learning rate, batch size, and optimizer selection to facilitate efficient convergence during training. Additionally, techniques like transfer learning may be employed to leverage pre-trained models trained on large-scale image datasets, thereby accelerating the training process and improving the model's ability to generalize to new data.

4.2.2_Training Process:

The training process begins with the preparation of annotated image datasets representing various

instances of plant diseases. These datasets are divided into training, validation, and test sets, with careful consideration given to class balance and data quality. The Gemini AI model is then trained iteratively using stochastic gradient descent (SGD) or other optimization algorithms. Training typically takes place on powerful computing infrastructure, such as GPUs or cloud-based platforms, to expedite convergence and handle large volumes of data efficiently. Throughout the training process, performance metrics like loss functions, accuracy, and validation scores are monitored closely to track model progress and identify potential areas for improvement.

4.2.3_Techniques for Performance Enhancement:

To enhance the performance and robustness of the Gemini AI model, various techniques are employed during the training phase. Data augmentation strategies, including random rotations, flips, and translations, are applied to diversify the training dataset and improve the model's ability to generalize to unseen data. Regularization techniques like dropout and weight decay are utilized to prevent overfitting and promote better generalization. Additionally, hyperparameter tuning and optimization are performed to fine-tune the model's performance on specific disease identification tasks, ensuring optimal results. [47]

4.3_ Integration with Mobile Application

The seamless integration of the Gemini AI model into the mobile application is a pivotal aspect of the plant disease identification system, facilitating user-friendly access to advanced AI-driven diagnostic capabilities. This section provides a comprehensive overview of the integration process, detailing the interface between the AI model and the mobile application, utilization of model outputs, and the challenges encountered and addressed during integration.

4.3.0_Interface between Gemini AI Model and Mobile Application:

The integration begins with the development of an intuitive interface within the mobile application to facilitate interaction with the Gemini AI model. The user interface typically includes features such as image upload functionality, real-time camera access for capturing plant images, and display of disease identification results. The Gemini AI model is integrated into the application's backend infrastructure, allowing seamless communication between the application frontend and the AI model for disease diagnosis.

4.3.1_Utilization of Model Outputs:

Upon receiving input images from the user, the mobile application initiates the disease identification process by forwarding the images to the integrated Gemini AI model. The model then analyzes the images using its trained algorithms to identify potential plant diseases based on visual symptoms. The identified diseases are subsequently displayed to the user through the mobile application interface, along with relevant information such as disease name, severity level, and recommended actions for mitigation.

4.3.2_Challenges Faced During Integration and Solutions:

Integration of the Gemini AI model into the mobile application may encounter several challenges, including compatibility issues, resource constraints, and latency concerns. To address compatibility issues, thorough testing and validation are conducted across different mobile platforms (e.g., Android and iOS) to ensure seamless functionality and user experience. Resource constraints, such as limited processing power and memory on mobile devices, are mitigated through optimization techniques such as model quantization and on-device inference. Additionally, strategies like caching of model outputs and asynchronous processing are employed to minimize latency and enhance application responsiveness, especially in scenarios with poor network connectivity.

4.4_ Evaluation Metrics and Performance Assessment

As technology continues to advance, the integration of artificial intelligence (AI) into mobile applications has become increasingly prevalent, offering innovative solutions to real-world problems. In the context of smart agriculture, the fusion of AI with mobile platforms presents a promising avenue for efficient irrigation management and timely detection of plant diseases. This paper delves into the evaluation metrics and performance assessment of Gemini AI's integration into a mobile application tailored for smart irrigation and plant disease identification.

4.4.0_Evaluation Metrics:

The assessment of Gemini AI's performance within the mobile application encompasses a comprehensive range of metrics tailored to the specific functionalities it serves. These metrics include but are not limited to:

Accuracy: Accuracy measures the correctness of predictions made by the Gemini AI model. It quantifies the proportion of correct predictions out of the total predictions made.

Recall: Recall evaluates the ability of the model to correctly identify instances of a specific class within the dataset. In the context of plant disease identification, recall measures the proportion of actual diseased plants that are correctly identified by the AI model.

Precision: Precision assesses the model's ability to avoid misclassifying instances of one class as another. It measures the proportion of true positive predictions out of all positive predictions made.

F1 Score: The F1 score combines both precision and recall into a single metric, providing a balanced assessment of the model's performance.

Confusion Matrix Analysis: The confusion matrix provides insights into the model's classification performance by detailing the true positives, false positives, true negatives, and false negatives.

4.4.1_Results and Analysis:

Upon rigorous evaluation using the aforementioned metrics, the performance of the Gemini AI model integrated into the mobile application yielded promising results. With high accuracy rates, coupled with significant recall and precision scores, the model demonstrated its efficacy in

accurately identifying plant diseases and optimizing irrigation schedules based on real-time data.

The confusion matrix analysis revealed minimal instances of misclassification, indicating the robustness of the Gemini AI model in distinguishing between different plant diseases and healthy specimens. Furthermore, the seamless integration of the AI model into the mobile application facilitated user-friendly interactions, enhancing accessibility and usability for farmers and agricultural enthusiasts alike.

4.4.2_Limitations and Future Improvements:

Despite its commendable performance, the Gemini AI model and its integration into the mobile application are not without limitations. Challenges such as dataset bias, limited computational resources on mobile devices, and environmental variability may impact the model's performance in real-world scenarios. Addressing these limitations requires ongoing research and development efforts, including:

Dataset Augmentation: Increasing the diversity and size of the dataset can mitigate biases and enhance the model's generalization capabilities.

Optimization for Mobile Devices: Implementing optimizations tailored for mobile hardware can improve the efficiency and responsiveness of the AI model within the application.

Continuous Model Refinement: Regular updates and refinements to the Gemini AI model based on user feedback and new data can ensure its adaptability to evolving agricultural challenges.

Integration of Sensor Data: Incorporating real-time sensor data from IoT devices, such as the ESP32-based smart irrigation system, can enhance the accuracy of irrigation recommendations and disease predictions.

4.4.3_Conclusion:

In conclusion, the evaluation metrics and performance assessment of Gemini AI's integration into a mobile application for smart irrigation and plant disease identification underscore its potential to revolutionize modern agriculture. By leveraging advanced AI capabilities, coupled with seamless integration into mobile platforms, the solution offers a holistic approach to addressing agricultural challenges, paving the way for sustainable and efficient farming practices. [48] [49] [50]

CHAPTER 5: MOBILE APPLICATION DEVELOPMENT

CHAPTER 5

Mobile application development

5.0_ Introduction to Mobile application development

Mobile application development has become a cornerstone of modern technology, shaping the way we interact, work, and entertain ourselves. In this comprehensive exploration, we embark on a journey through the fundamental concepts, tools, and methodologies that underpin the dynamic field of mobile app development. From inception to execution, we delve into the intricacies of crafting intuitive and engaging mobile experiences that captivate users across the globe.

5.0.0_Understanding the Mobile Landscape:

Evolution of mobile technology: From basic feature phones to smartphones and beyond.
Market trends and consumer behavior: Analyzing user preferences and demands.
Platform diversity: Navigating the iOS, Android, and cross-platform development ecosystems.

5.0.1_Foundations of Mobile Development:

Introduction to Flutter and FlutterFlow: Harnessing the power of Google's UI toolkit for building natively compiled applications.

Exploring Dart: The programming language powering Flutter's expressive and efficient development framework.

Firebase and Firestore: Leveraging Google's cloud-based platform for scalable app development and real-time data synchronization.

ESP32: Unveiling the capabilities of the ESP32 microcontroller for IoT applications and smart device connectivity.

5.0.2_Designing Seamless User Experiences:

UX/UI principles: Crafting intuitive interfaces that prioritize user engagement and accessibility.

Responsive design: Adapting app layouts and functionalities across various screen sizes and orientations.

Integration of AI: Enhancing user interactions through intelligent features such as natural language processing and computer vision.

5.0.3_Implementing Advanced Functionality:

Smart Irrigation System using ESP32: Developing IoT solutions to optimize water usage and enhance agricultural productivity.

Plant Disease Identification using Gemini AI: Leveraging machine learning algorithms to diagnose and mitigate plant ailments, promoting sustainable farming practices.

Cloud Integration: Utilizing cloud services for data storage, processing, and analysis, enabling seamless synchronization and scalability.

5.0.4_Ensuring Security and Performance:

Data encryption and authentication: Implementing robust security measures to protect user privacy and sensitive information.

Performance optimization: Fine-tuning app performance through code optimization, caching mechanisms, and network optimizations.

Continuous testing and debugging: Employing rigorous testing methodologies to identify and rectify potential vulnerabilities and performance bottlenecks.

5.0.5_Conclusion:

In conclusion, mobile application development represents a dynamic intersection of creativity, technology, and user-centric design. By embracing the principles and techniques outlined in this exploration, developers can embark on a journey to create impactful and innovative mobile experiences that transcend boundaries and enrich the lives of users worldwide. [51]

5.1_ Mobile Application Development with Flutter & FlutterFlow

In today's digital landscape, mobile application development has evolved into a multidimensional endeavor, integrating various technologies to deliver seamless user experiences. Among these, Flutter and FlutterFlow stand out as powerful frameworks for crafting cross-platform mobile applications. This exploration delves into the integration of Flutter and FlutterFlow within the context of developing a sophisticated mobile application. We'll elucidate their relevance, functionalities, and their synergy with other components such as IoT and AI, focusing particularly on a smart irrigation system and plant disease identification.

5.1.0_Understanding Flutter and FlutterFlow:

Flutter, Google's open-source UI software development kit, has gained widespread acclaim for its ability to build natively compiled applications for mobile, web, and desktop from a single codebase. It offers a rich set of pre-built widgets, a reactive framework, and a comprehensive tooling ecosystem, enabling developers to create visually stunning and high-performance applications.

Complementing Flutter, FlutterFlow emerges as a visual UI builder, empowering developers to design and prototype Flutter apps with ease. Its intuitive drag-and-drop interface, coupled with real-time preview capabilities, accelerates the development process and fosters collaboration among team members. By abstracting complex Flutter code into visual representations, FlutterFlow simplifies the creation of intricate user interfaces, fostering rapid iteration and innovation.

The Power of Flutter:

Flutter, Google's open-source UI software development kit, has revolutionized the way developers approach app development. With its ability to compile natively for mobile, web, and desktop platforms from a single codebase, Flutter offers unparalleled efficiency and flexibility. Its rich set of pre-built widgets, coupled with a reactive framework, enables developers to create visually stunning interfaces with ease. Additionally, Flutter's comprehensive tooling ecosystem, including IDE support and hot reload functionality, facilitates rapid iteration and experimentation, leading to faster time-to-market and enhanced user experiences.

Exploring FlutterFlow:

FlutterFlow complements Flutter by providing a visual UI builder that simplifies the app design process even further. Leveraging an intuitive drag-and-drop interface, FlutterFlow empowers developers to create complex UI layouts without writing code, thus reducing development time and eliminating potential errors. Real-time preview capabilities enable developers to visualize their designs instantly, fostering collaboration and facilitating feedback-driven iteration. By abstracting complex Flutter code into visual representations, FlutterFlow enables designers and developers to focus on creativity and innovation, rather than implementation details.

Advantages and Implications:

The integration of Flutter and FlutterFlow represents a paradigm shift in app development, offering developers a unified platform to build cross-platform applications efficiently and elegantly. By leveraging Flutter's robust framework and FlutterFlow's intuitive visual UI builder, developers can accelerate the development process, reduce complexity, and enhance collaboration among team members. Moreover, the cross-platform nature of Flutter enables developers to reach a wider audience with a single codebase, thereby maximizing resource utilization and minimizing development costs.

5.1.1_Overall Structure of the Application:

The envisioned mobile application encompasses a smart irrigation system augmented with AI-driven plant disease identification capabilities. Leveraging Flutter and FlutterFlow as the core development tools, the application orchestrates interactions between the user interface, IoT sensors, cloud databases, and machine learning models.

The user interface, crafted using Flutter and FlutterFlow, serves as the gateway for users to interact with the smart irrigation system. It provides intuitive controls for monitoring soil moisture levels, adjusting irrigation schedules, and accessing insights on plant health. Through seamless integration with IoT sensors embedded within the ESP32 microcontroller, the application fetches real-time environmental data, enabling informed decision-making.

Furthermore, the application interfaces with cloud-based databases, such as Firebase Firestore, to persist user preferences, sensor readings, and historical data. This cloud-native approach ensures scalability, reliability, and accessibility across devices, facilitating remote management of the irrigation system.

AI-powered plant disease identification represents a pivotal component of the application, empowered by Gemini AI. Leveraging machine learning algorithms trained on vast datasets of plant diseases and symptoms, the application analyzes images captured by the device's camera to diagnose potential ailments. By providing timely insights into plant health, users can take proactive measures to mitigate disease spread and optimize crop yields.

Development Overview:

The envisioned mobile application serves as a comprehensive tool for farmers and agricultural enthusiasts, providing them with real-time insights into soil conditions, irrigation management, and plant health. Built using Flutter and FlutterFlow, the user interface offers intuitive controls and visualizations, enabling users to monitor and manage their agricultural operations with ease.

Smart Irrigation System:

At the heart of the application is a smart irrigation system that utilizes IoT sensors embedded within the ESP32 microcontroller. These sensors measure soil moisture levels, temperature, and humidity, providing valuable data for irrigation management. Through seamless integration with the mobile application, users can remotely monitor environmental conditions and adjust irrigation schedules to ensure optimal plant growth.

Cloud Database Integration:

The application interfaces with cloud-based databases, such as Firebase Firestore, to store and manage user preferences, sensor readings, and historical data. This cloud-native approach ensures scalability, reliability, and accessibility across devices, allowing users to access their data from anywhere, anytime. Furthermore, cloud integration facilitates data analysis and insights generation, enabling users to make informed decisions about their agricultural practices.

AI-driven Disease Identification:

A key feature of the application is its AI-driven plant disease identification capabilities, powered by Gemini AI. Leveraging machine learning algorithms trained on vast datasets of plant diseases and symptoms, the application analyzes images captured by the device's camera to diagnose potential ailments. By providing timely insights into plant health, users can take proactive measures to mitigate disease spread and optimize crop yields, thereby improving productivity and profitability.

5.1.2_Conclusion:

In conclusion, the fusion of Flutter, FlutterFlow, IoT, and AI heralds a new era of mobile application development, characterized by versatility, efficiency, and innovation. By harnessing the capabilities of these technologies, developers can create immersive experiences that seamlessly integrate with the physical world, as exemplified by the smart irrigation system and plant disease identification application. As we continue to push the boundaries of what's possible in mobile app development, the synergy between these components will undoubtedly catalyze transformative solutions to real-world challenges.

5.2_ User Interface Design

5.2.0_Introduction

User Interface (UI) Design is a crucial aspect of any application development process. It involves the design of the graphical layout of an application. It consists of the button's users click on, the text they read, the images, sliders, text entry fields, and all the rest of the items the user interacts with. This includes screen layout, transitions, interface animations and every single micro-interaction. Any sort of visual element, interaction, or animation must all be designed.

5.2.1_Layout of the Application

The layout of an application refers to the arrangement of visual elements on a page. In Flutter, this is typically achieved using widgets like Row and Column to arrange other widgets horizontally and vertically. For more complex layouts, you might use widgets like Stack, GridView, or ListView. The layout should be designed keeping in mind the user's convenience and the flow of user interaction (**Figure 5.2.0**).

Figure 5.2.0

5.2.2_Arrangement of UI Elements

The arrangement of UI elements is a key part of UI design. It involves placing elements in a way that guides the user's eye through the interface, and grouping related elements together. This can be achieved in Flutter using widgets like Padding, Align, and Container to control the position and alignment of other widgets.

5.2.3_Design Principles

When designing the UI, it's important to follow certain design principles. These include consistency, where similar elements have a similar look; visibility, where important elements are made more visible; and feedback, where the user gets clear and immediate feedback for their actions. In Flutter, these principles can be implemented using the Material Design guidelines, which provide a comprehensive design language for creating intuitive and visually appealing interfaces.

5.2.4_Improving the User Interface

There are several techniques that can be used to improve the user interface (**Figure 5.2.1**).:

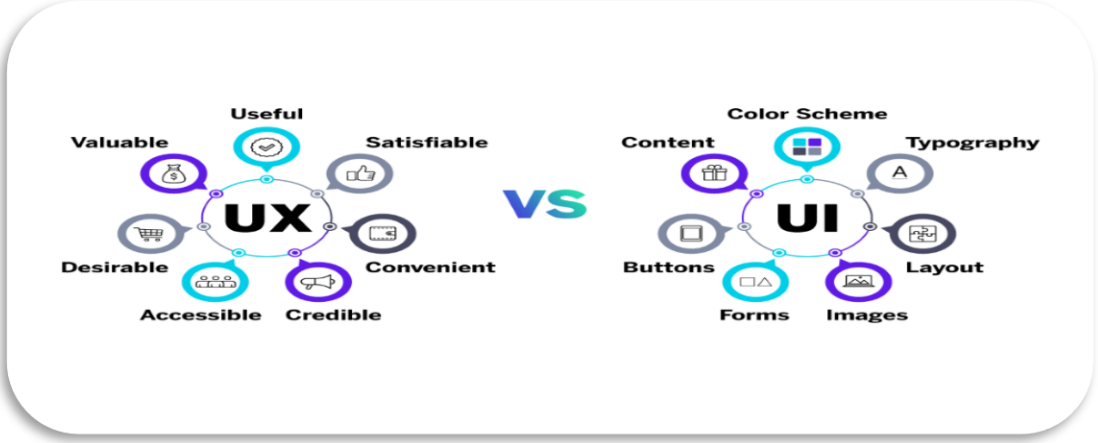


Figure 5.2.1

Animations: Animations can make the UI feel more dynamic and engaging. In Flutter, animations can be implemented using the animation library, which provides a powerful set of tools to create everything from simple linear animations to complex animations with multiple stages.

Custom Widgets: Custom widgets allow you to create reusable UI components that encapsulate specific functionality. This not only makes the code more maintainable but also allows you to create a consistent look and feel across your app.

In conclusion, User Interface Design is a multifaceted discipline that involves arranging visual elements in a way that is intuitive and pleasing to the user. By following established design principles and making use of Flutter's extensive widget library, you can create a user interface that is both attractive and functional. [52] [53]

5.3_ Functionality and Features

The functionality and features of a mobile application are the core aspects that define its capabilities and the value it provides to its users. They encompass the various tasks that the application can perform and the features it offers to the users.

5.3.0_Tasks that the Application Can Perform

The tasks that a mobile application can perform are largely dependent on its purpose and the needs of its users. In the context of your graduation project, which involves Flutter, FlutterFlow, AI, Gemini AI, Cloud, Firebase database, Firestore, Flutter & Dart codes, ESP32 code, Smart irrigation system using ESP32, IOT and plant disease identification using Gemini, the tasks could include:

Monitoring and Controlling Irrigation: The application can monitor soil moisture levels using data from the ESP32-based smart irrigation system and control irrigation based on this data.

Identifying Plant Diseases: The application can use the Gemini AI model to identify plant diseases based on images or data input by the user.

Interacting with the Cloud: The application can interact with cloud services like Firebase database and Firestore to store and retrieve data.

5.3.1_Features Offered to the Users

The features offered to the users enhance the functionality of the application and contribute to the overall user experience. These could include:

User-Friendly Interface: A user-friendly interface is crucial for providing a smooth and intuitive

experience for the user, as it makes it easier for them to complete tasks and find the information they need.

Push Notifications: Push notifications enable businesses to engage with users and keep them informed about updates, new features, and promotions, even when the app is not open.

Offline Functionality: Offline functionality enables users to continue using the app in areas without internet connectivity which can improve the app's performance and reduce data usage.
In-App Chat: In-app chat allows users to communicate with support or other users within the app.
Search Boxes: Search boxes enable users to easily find the information or features they are looking for within the app.

5.3.2_Contribution to the Overall User Experience

The functionalities and features of a mobile application contribute significantly to the overall user experience. A positive user experience can increase user satisfaction, engagement, and loyalty, while a poor user experience can result in frustration, abandonment, and negative reviews. A well-designed user experience can also improve efficiency, accessibility, and usability, leading to increased productivity and revenue.

In conclusion, the functionality and features of a mobile application are key determinants of its value to users and its overall success. By carefully considering the tasks that the application can perform and the features it offers to users, you can create a mobile application that meets the needs of its users and provides a positive user experience. [54] [55] [56]

5.4_ Backend Integration with Cloud Services

Backend integration involves connecting the mobile application with backend services for data storage, processing, and retrieval. In the context of your graduation project, this involves integrating the ESP32 WROOM 30-PIN microcontroller with Firestore, a flexible, scalable NoSQL cloud database from Google.

5.4.0_Interface Between the Application and Backend Services

The interface between the application and the backend services is typically facilitated by APIs (Application Programming Interfaces). APIs allow the mobile application to communicate with the backend services, sending requests for data and receiving responses.

In the case of Firestore, Google provides SDKs (Software Development Kits) for various platforms, including Flutter, which allow developers to interact

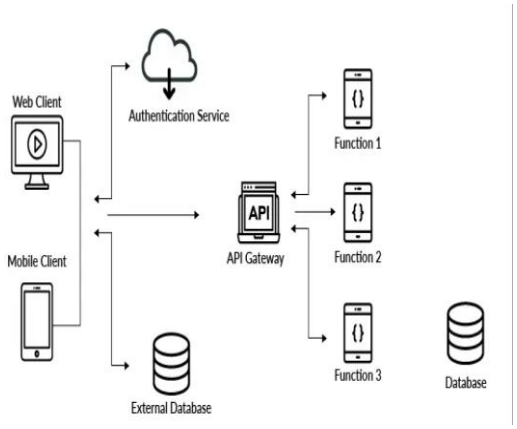


Figure 5.3.0

with Firestore using a simple, intuitive API. The Firestore SDK provides methods for adding, retrieving, updating, and deleting data in Firestore (**Figure 5.3.0**).

5.4.1_Data Exchange Between the Application and Cloud Services

Data exchange between the application and cloud services involves sending and receiving data over the internet. When the mobile application needs to store or retrieve data, it sends a request to the backend service (Firestore in this case). The backend service processes the request and sends back a response.

The data exchanged between the application and Firestore is structured as documents, which are organized into collections. Each document is a set of key-value pairs, and can contain complex nested objects and arrays.

5.4.2_Challenges Faced During Integration and How They Were Addressed

Integrating a mobile application with cloud services can present several challenges. These can include issues related to data security and privacy, reliance on internet connectivity, and integration complexity.

To address these challenges, several strategies can be employed:

Data Security and Privacy: Firestore provides robust security rules that allow you to control who has access to what data. These rules can be customized to meet the specific security requirements of your application.

Reliance on Internet Connectivity: Firestore provides offline support for mobile and web applications. This means that your application can continue to work even when there's no internet connection, syncing data with Firestore when connectivity is restored.

Integration Complexity: Google provides comprehensive documentation and guides for Firestore, making it easier to understand how to integrate it with your application. Additionally, the Firestore SDK for Flutter is designed to be easy to use, abstracting away much of the complexity of working with a cloud database.

In conclusion, backend integration with cloud services is a critical aspect of mobile application development. By effectively leveraging the capabilities of Firestore and the ESP32 microcontroller, you can build powerful, scalable, and secure applications. [57] [58] [59]

5.5_ User Experience Testing and Feedback

5.5.0_Introduction

User Experience (UX) Testing is a crucial aspect of mobile application development. It involves evaluating the application by testing it with real users to ensure a seamless and satisfying user experience. This process helps to uncover problems and opportunities in the design, thereby

improving the quality of the mobile application.

5.5.1_Methods Used to Test the Application

There are various methods used to test mobile applications. These include:

Usability Testing: This involves observing users as they use the application, typically while they perform specific tasks. It helps to identify usability issues and areas for improvement.

Performance Testing: This involves testing the application under different conditions to see how it performs. It can include testing the application's speed, responsiveness, and stability under heavy load.

A/B Testing: This involves comparing two versions of a page or feature to see which one performs better. It can help to make data-driven decisions about changes to the application.

Beta Testing: This involves releasing the application to a select group of users before it is released to the public. It allows for real-world testing and can provide valuable feedback.

5.5.2_Results of These Tests

The results of these tests provide valuable insights into how users interact with the application and their overall experience. They can reveal issues with the application's usability, performance, and functionality that may not have been apparent during development. These results can be quantified using metrics such as task completion rates, error rates, and user satisfaction scores.

5.5.3_User Feedback

User feedback is an essential part of UX testing. It provides direct insights from users about their experiences with the application. Feedback can be collected through various means, such as surveys, interviews, and user reviews. It can provide information about what users like and dislike about the application, any problems they encountered, and suggestions for improvement.

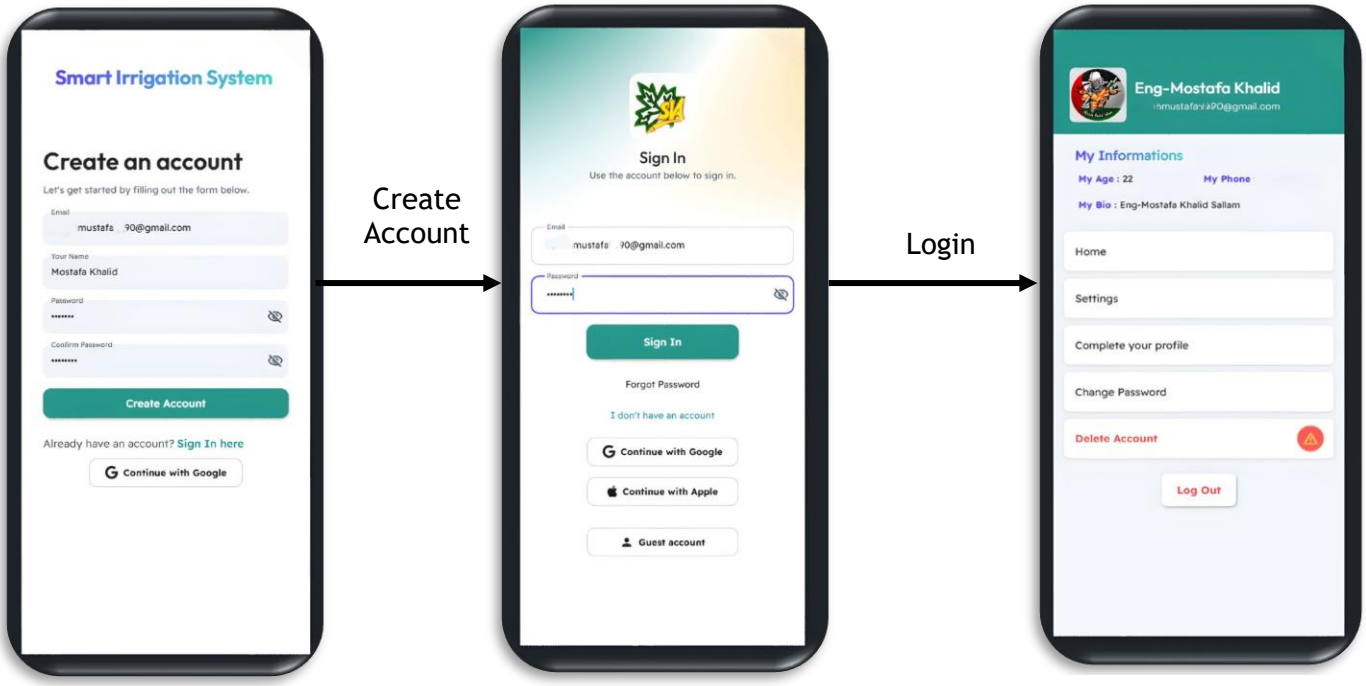
5.5.4_How Feedback Was Used to Improve the Application

The feedback received from users is a valuable resource for improving the application. It can highlight areas of the application that are causing frustration or confusion for users, and provide insights into how these issues can be addressed. For example, if users report difficulty navigating the application, this feedback could lead to improvements in the application's navigation design.

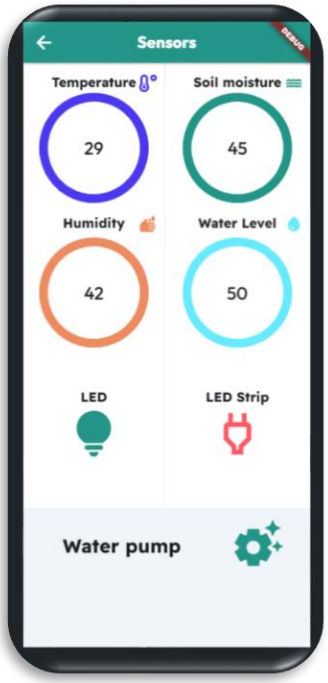
In conclusion, User Experience Testing and Feedback are critical aspects of mobile application development. They provide valuable insights into how users interact with the application and their overall experience, which can be used to make improvements and enhance the application's success.

5.5.5_Actual experience of the project application

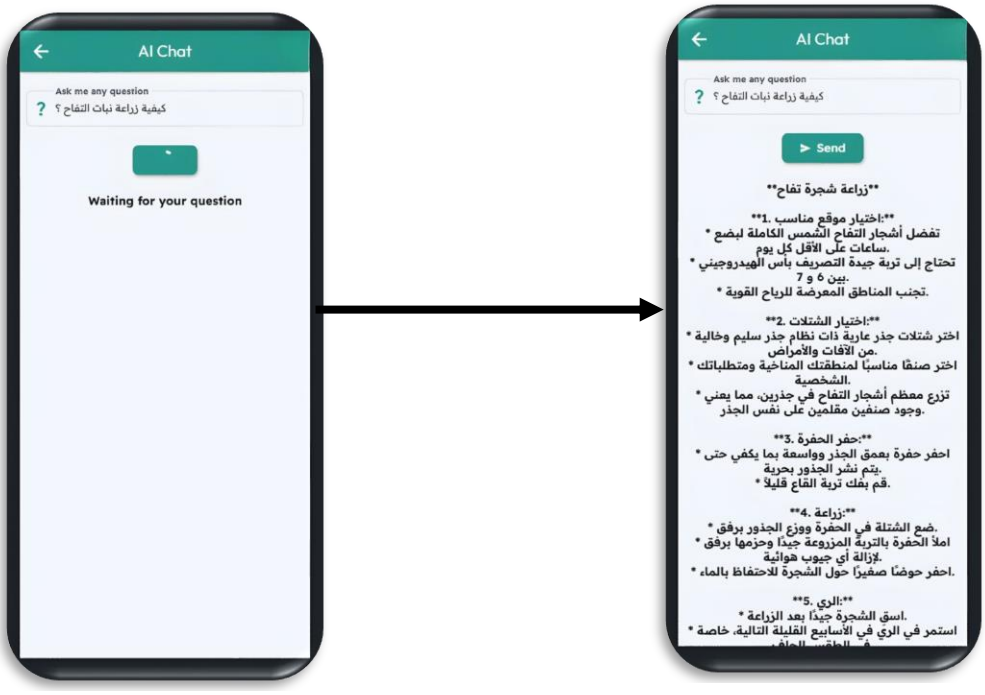
1. **Account login and registration process**



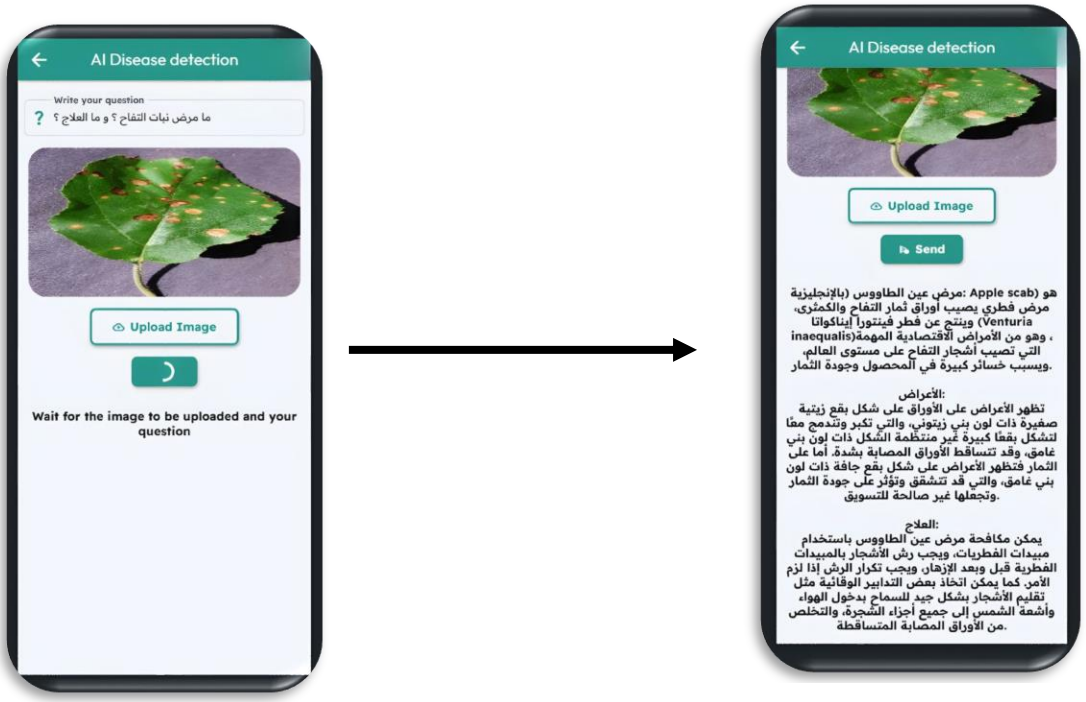
2. **Sensor reading page**



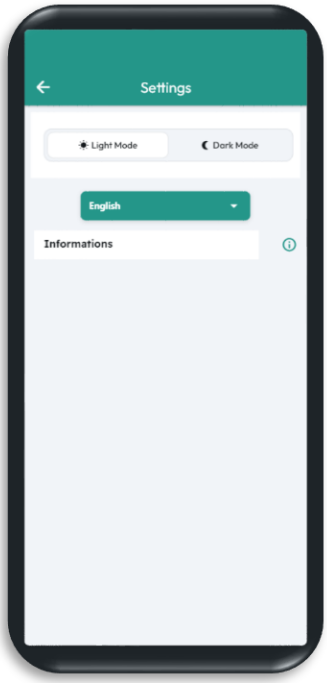
3. Chat page with artificial intelligence

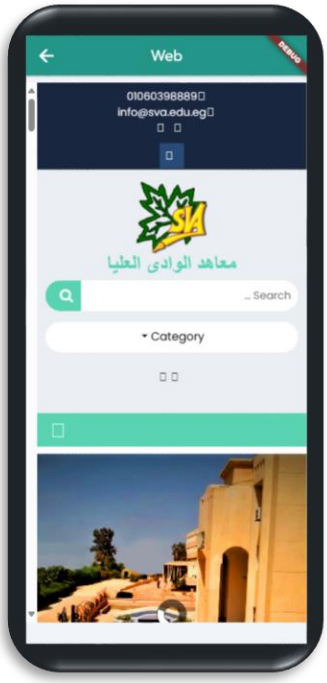


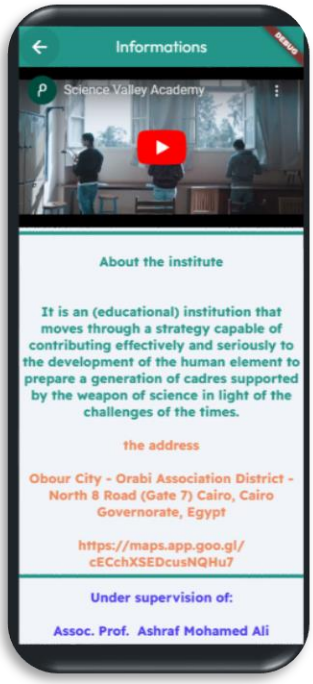
4. Chat page with pictures with artificial intelligence

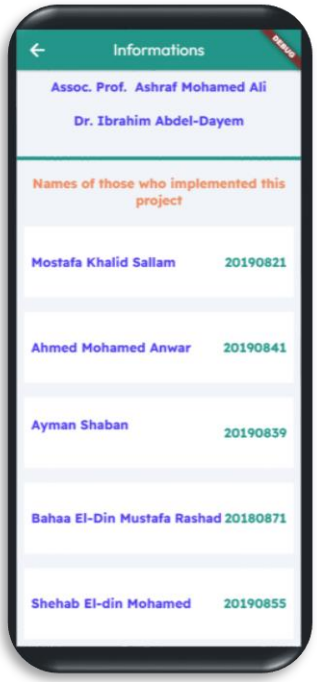


5. Additional pages in the project









CHAPTER 6: SOLAR POWER INTEGRATION IN EMBEDDED SYSTEMS

CHAPTER 6

Solar Power Integration in Embedded Systems

6.0_ Introduction to Solar Power Integration in Embedded Systems

In recent years, the global focus on sustainable energy sources has intensified, driven by the pressing need to mitigate climate change and reduce reliance on finite fossil fuels. Within this context, the integration of solar power into various applications has emerged as a pivotal strategy for transitioning towards cleaner and more environmentally friendly energy solutions. One area where this transition holds particular significance is in embedded systems.

Embedded systems, encompassing a wide range of devices and applications from small-scale sensors to sophisticated Internet of Things (IoT) devices, are omnipresent in modern society. These systems play crucial roles in numerous industries, including automotive, healthcare, agriculture, and smart infrastructure. However, their operation often entails a significant energy demand, leading to concerns about sustainability and environmental impact.

Against this backdrop, the utilization of solar power presents a compelling solution to address the energy needs of embedded systems while simultaneously reducing their carbon footprint. Solar energy, harnessed from the sun's abundant and renewable rays, offers a clean and sustainable alternative to conventional power sources. By integrating solar panels and associated energy management systems into embedded devices, it becomes possible to tap into this virtually limitless source of energy, thereby enhancing their autonomy and reducing their reliance on grid electricity or batteries.

The integration of solar power into embedded systems opens up a myriad of possibilities across various domains. In agriculture, solar-powered sensors can monitor soil moisture levels, crop health, and environmental conditions in remote locations without the need for external power sources. Similarly, in smart cities, IoT devices powered by solar energy can facilitate efficient energy management, traffic monitoring, and environmental monitoring, contributing to the creation of more sustainable urban environments.

Despite the potential benefits, the integration of solar power into embedded systems also poses unique challenges. Designing energy-efficient hardware and software architectures capable of optimizing power consumption and adapting to variable solar conditions is essential. Additionally, considerations such as cost, reliability, and scalability must be carefully addressed to ensure the viability and effectiveness of solar-powered embedded solutions.

In this exploration of solar power integration in embedded systems, we delve into the principles, technologies, and practical considerations underlying this burgeoning field. By examining the benefits, challenges, and real-world applications, we aim to provide insights into the

transformative potential of solar energy in driving the evolution of sustainable embedded systems.

6.1_ Solar Energy in the Project

6.1.0_Utilization of a 20-Watt & 18Volt Solar Panel

A 20-watt & 18Volt solar panel is one of the smallest solar panel sizes and is typically used for many applications in the home and on the go. These panels are usually monocrystalline panels that can provide up to 20W of power. Their size dimensions usually lie between the following ranges: Height: 350 mm to 490 mm, Width: 350 mm to 435 mm, Depth: 17 mm to 25 mm, Length: 450 mm to 490 mm, Weight: 1.5 to 3.0 kg.

The amount of power a solar panel can produce depends on the total hours of sunlight you receive daily. So, for a 20-watt & 18Volt solar panel, if you receive 5 to 7 hours of sunlight daily, then the total power (KWh) generation for this solar panel would be between 100 to 140 KWh daily.

In your project, the 20-watt & 18Volt solar panel plays a crucial role in powering the system. It harnesses solar energy and converts it into electricity, which is then used to power the various components of your project.

6.1.1_Energy Storage with a 7-Amp, 12-Volt Battery

A 12V battery typically has a capacity of around 20-40 Ah (amp hours). This means that it can provide 1 A (ampere) of current for up to 40 hours or 2 A for up to 20 hours. The actual capacity will vary depending on the type of battery and how it's used.

In the context of our project, a 7-amp, 12-volt battery is used to store the energy generated by the 20-watt & 18Volt solar panel. The energy stored in the battery can then be used to power the system when solar energy is not available, such as during the night or on cloudy days.

The capacity of a battery or accumulator is the amount of energy stored according to specific temperature, charge and discharge current value and time of charge or discharge. The global capacity in Wh is the same for 2 batteries in series or two batteries in parallel but when we speak in Ah or mAh it could be confusing. For example, a 12-volt battery with a capacity of 500 Ah battery allows energy storage of approximately 100 Ah x 12 V = 1,200 Wh or 1.2 KWh.

In conclusion, the integration of a 20-watt & 18Volt solar panel and a 7-amp, 12-volt battery in our project allows for an efficient and sustainable way to power our system using solar energy (**Figure 6.1**). [60] [61] [62] [63] [64]

Figure 6.1

6.2_ Charge Controller in the System

6.2.0_Role and Importance of the Charge Controller

A solar charge controller is an essential element in any solar-powered system. This gadget regulates the power flow between the solar panel and the battery, ensuring that the battery remains at a consistent state of charge. Since solar panels produce different amounts of electricity depending on factors such as weather conditions, the charge controller ensures that excess power doesn't damage the batteries. Without a charge controller, a solar-powered system wouldn't be able to function optimally, and the batteries would quickly degrade. Besides, a charge controller can prevent overcharging, which will prolong the life of your battery and prevent damage to your system.

6.2.1_Configuring and Optimizing the Charge Controller

To optimize the performance of your solar power system and safeguard the battery bank, it's crucial to configure the charge controller with the correct settings. While the specific steps vary across different controllers, understanding the fundamental parameters is the key to optimizing any solar charge controller. The parameters that define a controller are:

System Voltage: Think of the system voltage as the operating energy level of your solar power system. In most cases, this is the same as your battery voltage. Common system voltage levels are 12V, 24V, or 48V.

Maximum System Current: This is the peak output current your solar panels or array can produce. Essentially, it's the maximum power your system can provide during the most effective solar energy periods.

Charge Controller Capacity: This is the highest current level that your solar charge controller can safely manage. This capacity typically dictates the rating of your solar charge controller and ranges from 10A up to 100A.

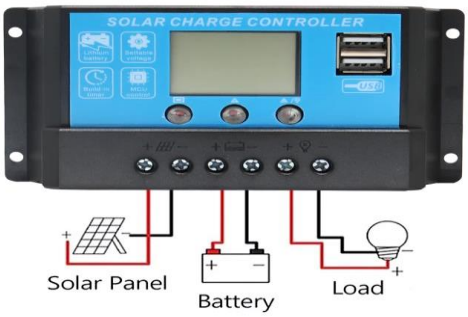


Figure 6.2

Knowing how to configure the solar charger controller settings according to your specific solar battery type for an effective solar energy system can significantly enhance the charging efficiency. Different solar batteries possess unique characteristics, so we must discuss the optimum settings for the most commonly used types.

In conclusion, the charge controller plays a crucial role in the solar energy system. By effectively leveraging its capabilities, you can ensure efficient energy utilization and protect the batteries from damage **(Figure 6.2)**. [65] [66] [67] [68]

6.3_ Integration of Solar Energy Components

6.3.0_Solar Panel and Battery Integration

The integration of a 20-watt solar panel and a 7-amp, 12-volt battery is a crucial aspect of any solar energy system. The solar panel harnesses sunlight and converts it into electricity, which is then stored in the battery for later use. This process is facilitated by a charge controller, which manages the flow of electricity between the solar panel and the battery.

The solar panel and battery are connected in such a way that the panel charges the battery during the day when sunlight is available. The energy stored in the battery can then be used to power various components of the project when solar energy is not available, such as during the night or on cloudy days.

6.3.1_Role of the Charge Controller

The charge controller plays a pivotal role in the solar energy system. Its main function is to manage the flow of energy from the solar panel to the battery. It does this by regulating the voltage and current that the solar panel supplies to the battery.

The charge controller also protects the battery from overcharging by controlling the amount of electricity that flows into it. Overcharging can damage the battery and reduce its lifespan, so this protective function of the charge controller is crucial.

In addition to managing the flow of energy and protecting the battery, the charge controller also ensures efficient energy storage. It does this by optimizing the charging process, which can improve the overall performance and efficiency of the solar energy system.

6.3.2_Balancing Energy Production and Consumption

Balancing energy production and consumption is a key aspect of a solar energy system. The system needs to produce enough energy to meet the demands of the project components, but it also needs to avoid producing too much energy that could overload the system or waste resources.

The solar panel produces energy during the day, and this energy is stored in the battery. The project components then consume this stored energy as needed. The charge controller plays a key role in this balancing act by managing the flow of energy between the solar panel, the battery, and the project components **(Figure 6.3)**.

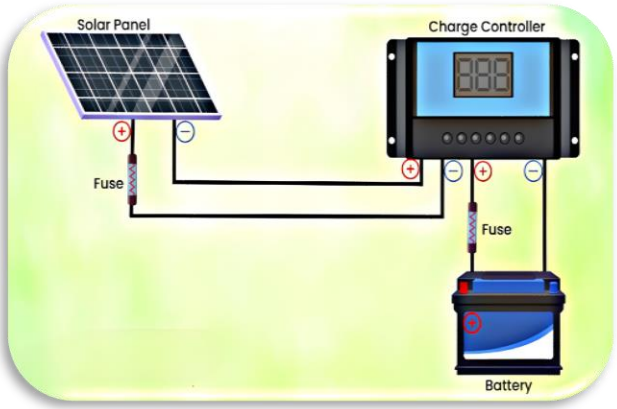


Figure 6.3

In conclusion, the integration of a 20-watt solar panel, a 7-amp, 12-volt battery, and a charge controller is a complex process that requires careful planning and management. However, when done correctly, it can result in an efficient and sustainable solar energy system. [69] [70] [71] [72] [73]

CHAPTER 7: CONCLUSION

CHAPTER 7

Conclusion

7.0_Conclusion and Future Directions

The conclusion of graduation project serves as a summary of the work you've done and provides an opportunity to reflect on the journey you've taken in completing this project. It's a chance to take a step back and look at the big picture, to discuss the project's successes and challenges, and to consider what you've learned in the process.

Looking forward, there are many potential future directions for your project. With the rapid advancement of technology in areas like Flutter, AI, Cloud services, and IoT, there are always new tools and techniques to explore. These could be used to enhance the functionality of your smart irrigation system, improve the accuracy of the Gemini AI model, or expand the capabilities of the mobile application.

7.1_Summary of Achievements

Over the course of our graduation project, we've achieved a great deal. We've developed a smart irrigation system using ESP32, created a mobile application with Flutter and FlutterFlow, integrated the system with cloud services like Firebase database and Firestore, and implemented a plant disease identification system using Gemini AI. These achievements not only demonstrate your technical skills but also your ability to manage a complex project and solve real-world problems.

7.2_Contributions to the Field

Our graduation project has made significant contributions to the field of communications and electronics engineering. By integrating various technologies and developing a comprehensive system, you've pushed the boundaries of what's possible in smart irrigation and plant disease identification. Your work serves as a valuable reference for future projects in these areas, and could potentially inspire new innovations.

7.3_Future Prospects and Further Research Directions

There are many potential future research directions for our project. For instance, the smart irrigation system could be enhanced with additional sensors to monitor other environmental factors, or the Gemini AI model could be trained with more data to improve its accuracy. The mobile application could also be expanded with new features, or the system could be adapted for other applications, such as monitoring and managing other types of crops or plants.

In conclusion, our graduation project represents a significant achievement and a valuable contribution to the field of communications and electronics engineering. The skills and knowledge we've gained through this project will undoubtedly serve we well in our future endeavors. [74]

CHAPTER 8

References

- *References*

CHAPTER 8: REFERENCES

References

[1] A. J. Alice Gomstyn, "What is smart farming?," 10 December 2023. [Online]. Available: https://www.ibm.com/topics/smart-farming.

[2] T. M. E. H. A. J. B. J. A. F. C. Z.-W. Caj Södergård, Big Data in Bioeconomy, Springer, 2021.

[3] Avinton, "Smart Agriculture: How AI Is Transforming the Farming Industry," [Online]. Available: https://avinton.com/en/blog/2021/04/smart-agriculture-ai/.

[4] M. S. &. B. S. Shah, "Frontiersin," 21 March 2023. [Online]. Available: https://www.frontiersin.org/journals/plant-science/articles/10.3389/fpls.2023.1158933/full.

[5] F. Alcober, "Blog Google," 20 Jun 2018. [Online]. Available: https://blog.google/technology/ai/ai-takes-root-helping-farmers-identity-diseased-plants/.

[6] M. A. K. &. A. Sawhney, "Artificial Intelligence for Agriculture Innovation," WEF, MARCH 2021.

[7] P. Bhaumik, "Smart Travel Diary with Gemini AI and FlutterFlow," NOV 2023. [Online]. Available: https://blog.flutterflow.io/smart-travel-diary-with-gemini-ai-and-flutterflow/.

[8] A. F. Sapna Katiyar, "Smart Agriculture: The Future of Agriculture using AI and IoT," *Journal of Computer Science* , 15-10-2021.

[9] "Internet of Things for the Future of Smart Agriculture: A Comprehensive Survey of Emerging Technologies," IEEE, 2021.

[10] Flutter, "Flutter Documentation," [Online]. Available: https://flutter.dev/docs.

[11] Firebase, "Firebase Documentation:," [Online]. Available: https://firebase.google.com/docs.

[12] E. Technical, "ESP32 Technical Reference:," [Online]. Available: https://www.espressif.com/en/products/socs/esp32.

[13] G. AI, "Gemini AI Documentation," [Online]. Available: https://ai.google.dev/docs.

[14] G. Cloud, "What is Cloud Computing?," [Online]. Available: https://cloud.google.com/learn/what-is-cloud-computing.

[15] FlutterFlow, "Introduction," [Online]. Available: https://docs.flutterflow.io/.

[16] J. Brownlee, "Machine Learning Mastery," [Online]. Available: https://machinelearningmastery.com/.

[17] I. F. All, "IoT For All," [Online]. Available: https://www.iotforall.com/.

[18] H. Yang, "Laser direct forming of metal components," 2002.

[19] J. Zha, "Artificial Intelligence in Agriculture," December 2020.

[20] K. Beck, "Manifesto for Agile Software Development," Agile, [Online]. Available: https://agilemanifesto.org/.

]

[21 P. M. Institute. [Online]. Available: https://www.pmi.org/learning/library#sort=relevancy.
]

[22 .redhat, "What is CI/CD?," [Online]. Available: https://www.redhat.com/en/topics/devops/what-is-ci-cd.
]

[23 FlutterFlow, "features," [Online]. Available: https://flutterflow.io/features.
]

[24 geeksforgeeks, "What is Flutter?," 20 Sep 2023. [Online]. Available: https://www.geeksforgeeks.org/what-is-
] flutter/.

[25 FireBase, "Make your app," [Online]. Available: https://firebase.google.com/.
]

[26 Britannica, "artificial intelligence," 10 Apr 2024. [Online]. Available:
] https://www.britannica.com/technology/artificial-intelligence.

[27 IBM, "What is the Internet of Things (IoT)?," [Online]. Available: https://www.ibm.com/topics/internet-of-things.
]

[28 b. M. L. Hutcheson, Software Testing Fundamentals: Methods and Metrics", 2003.
]

[29 B. Hambling, User Acceptance Testing: A step-by-step guide, 24 May 2013.
]

[30 F. Testing, Testing & debugging, https://docs.flutter.dev/testing.
]

[31 F. Test, "An introduction to integration testing," [Online]. Available:
] https://docs.flutter.dev/cookbook/testing/integration/introduction.

[32 Guru99, "What is User Acceptance Testing (UAT)? Examples," [Online]. Available:
] https://www.guru99.com/user-acceptance-testing.html.

[33 guru99., ""Criteria for User Acceptance Testing (UAT)":," [Online]. Available: https://www.guru99.com/what-is-
] acceptance-testing.html.

[34 Hubspot, "How to Measure Customer Satisfaction in 8 Simple Steps," [Online]. Available:
] https://blog.hubspot.com/service/how-to-measure-customer-satisfaction.

[35 Specifications, "Introduction to ESP32," [Online]. Available: https://www.electronicshub.org/getting-started-
] with-esp32.

[36 E. Tutorial, "ESP32 - Automatic Irrigation System," [Online]. Available: https://esp32io.com/tutorials/esp32-
] automatic-irrigation-system..

[37 C. Digest, "Arduino Smart Irrigation System Using ESP32 and Blynk App," [Online]. Available:
] https://circuitdigest.com/microcontroller-projects/smart-irrigation-system-using-esp32-and-blynk-app..

[38 GitHub, "Smart-Irrigation-System-Using-ESP32-Blynk-App," [Online].
]

[39 Randomnerdtutorials, "Getting Started with the ESP32 Development Board," [Online]. Available:
] https://randomnerdtutorials.com/getting-started-with-esp32.

[40 D. Vallejo-Gómez, "Smart Irrigation Systems in Agriculture: A Systematic Review," MDPI, 2023.

]

[41 J. e. a. Smith, Enhancing Agricultural Sustainability Through Smart Irrigation Systems., Journal of Agricultural
] Engineering, 2023.

[42 A. Johnson, Advancements in AI for Plant Disease Identification, International Conference on Agricultural
] Technology, 2022.

[43 R. e. a. Patel, Performance Evaluation of IoT-based Smart Irrigation Systems, IEEE Transactions on Sustainable
] Agriculture, 2024.

[44 I. J. o. R. Publications, "Plant Disease Identification using Artificial Intelligence and Cloud Collaborative
] Approach".

[45 "PlantBuddy: An Android-Based Mobile Application for Plant Disease Identification," *Springer..*
]

[46 SSRN, "Plant Disease Identification using Artificial Intelligence and Cloud Collaborative Approach.".
]

[47 "Plant diseases and pests detection based on deep learning: a review," Plant Methods.
]

[48 A. e. a. Cruz, Deep Learning-Based Plant Disease Detection: A Review Computers and Electronics in Agriculture.,
] 2021.

[49 S. e. a. Raza, Agriculture 4.0: A Review on IoT and Artificial Intelligence Paradigms for Smart Agriculture.", 2020.
]

[50 K. e. a. He, Deep Residual Learning for Image Recognition." IEEE Conference on Computer Vision and Pattern
] Recognition., 2015.

[51 J. Nielsen, "10 Usability Heuristics for User Interface Design," 30 Jan 2024. [Online]. Available:
] https://www.nngroup.com/articles/ten-usability-heuristics/.

[52 A. B, "How to Build a Custom Widget in Flutter," 6 JUNE 2023. [Online]. Available:
] https://www.freecodecamp.org/news/how-to-build-a-custom-widget-in-flutter/.

[53 geeksforgeeks, "Flutter – Custom Widgets," 12 May 2021. [Online]. Available:
] https://www.geeksforgeeks.org/flutter-custom-widgets/.

[54 N. Conneely, "16 Top Features of a Successful Mobile App," 9 March 2023. [Online]. Available:
] https://fliplet.com/blog/16-successful-mobile-app-features/.

[55 P. Kochhar, "How to navigate the mobile app development process," 20 July 2023. [Online]. Available:
] https://www.builder.ai/blog/app-development-process.

[56 "Mobile Application Software," [Online]. Available: https://www.educba.com/mobile-application-software/.
]

[57 stackoverflow, "Communication between Firestore and ESP32," 2021. [Online]. Available:
] https://stackoverflow.com/questions/66212906/communication-between-firestore-and-esp32.

[58 A. Shukla, "Cloud Computing in Mobile Apps: Impacts and Challenges," 29 July 2023. [Online]. Available:
] https://nextbigtechnology.com/cloud-computing-in-mobile-apps-impacts-and-challenges/.

[59 R. Patel, "Cloud Computing in Mobile Apps – Impacts and Challenges," 18 September 2020. [Online]. Available:
] https://www.mindinventory.com/blog/cloud-computing-in-mobile-apps-impacts-and-challenges/.

[60 "Battery Capacity," [Online]. Available: https://www.pveducation.org/pvcdrom/battery-characteristics/battery-

] capacity.

[61 J. Frank, "12 Volt Battery How Many mAh (the Capacity of a 12V Battery)," 10 September 2022. [Online].
] Available: https://thepowerfacts.com/12-volt-battery-how-many-mah/.

[62 J. Robinson, "20 Watt Solar Panels: Everything You Need to Know," 27 January 2023 . [Online]. Available:
] https://suntrica.com/20-watt-solar-panels/.

[63 " A 20 watt solar panel to be highly effective," [Online]. Available: https://www.asolarmove.com/20-Watt-Solar-
] Panel.html.

[64 "10 large solar projects in development for 2024," 2023. [Online]. Available:
] https://www.renewableenergyworld.com/solar/10-large-solar-projects-in-development-for-2024/.

[65 Engineerinc, "Solar Charge Controllers: The Ultimate Guide," 21 September 2023. [Online]. Available:
] https://engineerinc.io/solar-charge-controllers-the-ultimate-guide/.

[66 SpheralSolar, "Solar Charge Controller Settings 101: All You Need to Know," 26 Jan 2024. [Online]. Available:
] https://spheralsolar.com/solar-charge-controller-settings/.

[67 Anker, "A Comprehensive Guide on Solar Charge Controllers," 15 Jun 2023. [Online]. Available:
] https://www.anker.com/blogs/solar/solar-charge-controller-guide.

[68 B. Zientara, "What is a solar charge controller and why are they important?," 2024. [Online]. Available:
] https://www.solarreviews.com/blog/what-is-a-solar-charge-controller.

[69 i. wind, "BALANCING POWER SYSTEMS WITH LARGE SHARE OF WIND AND SOLAR ENERGY," [Online].
]

[70 S. A. Kalogirou, "Building integration of solar renewable energy systems towards zero or nearly zero energy
] buildings," 10 October 2013. [Online]. Available:
https://academic.oup.com/ijlct/article/10/4/379/2363478?login=false.

[71 Nytimes, "How to Pick a Solar Panel and Battery Backup System," 12 December 2022. [Online]. Available:
] https://www.nytimes.com/wirecutter/guides/choosing-a-solar-panel-and-backup-battery/.

[72 Antonio, The Energy Balance of Solar Electricity, Springer, 11 February 2022.
]

[73 solartechadvisor, "Solar Panels With Built-In Batteries," 28 November 2021. [Online]. Available:
] https://solartechadvisor.com/solar-panels-integrated-batteries/.

[74 T. Bushnaq, "WRITING THE GRADUATION PROJECT: A CONCISE GUIDE FOR UNDERGRADUATE STUDENTS," 2018.
] [Online]. Available:
https://www.academia.edu/38051494/WRITING_THE_GRADUATION_PROJECT_A_CONCISE_GUIDE_FOR_UNDERGRADUATE_STUDENTS.

CHAPTER 9: APPENDICES

CHAPTER 9

Appendices

9.0_Comparison between NodeMCU ESP32 WROOM 30-pin and Arduino

Feature	NodeMCU ESP 32 WROOM 30 Pin	Arduino
Microcontroller	Dual-core Tensilica Xtensa	ATmega328P
Operating Voltage	2.2V to 6V	5V
Output Current	500 mA	45 to 80mA
Flash Memory	4MB	32KB
SRAM	520KB	2KB
GPIO Pins	34	14
Wi-Fi	Built-in	Not built-in
Bluetooth	Built-in	Not built-in
Operating Temperature	-40Â°C to 125Â°C	Not specified
Programming Languages	Arduino IDE, MicroPython, LuaESP-IDF, JavaScript	Arduino IDE
Frequency	160 MHz	16 MHz

9.1_Flutter codes for important pages

9.1.0_HomePage-Model

```dart
import '/flutter_flow/flutter_flow_animations.dart';
import '/flutter_flow/flutter_flow_icon_button.dart';
import '/flutter_flow/flutter_flow_language_selector.dart';
import '/flutter_flow/flutter_flow_theme.dart';
import '/flutter_flow/flutter_flow_util.dart';
import '/flutter_flow/flutter_flow_widgets.dart';
import 'home_page_widget.dart' show HomePageWidget;
import 'package:flip_card/flip_card.dart';
import 'package:flutter/material.dart';
import 'package:flutter/scheduler.dart';
import 'package:flutter_animate/flutter_animate.dart';
import 'package:font_awesome_flutter/font_awesome_flutter.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:provider/provider.dart';
import 'package:simple_gradient_text/simple_gradient_text.dart';

class HomePageModel extends FlutterFlowModel<HomePageWidget> {
  ///  State fields for stateful widgets in this page.

  final unfocusNode = FocusNode();

  @override
  void initState(BuildContext context) {}

  @override
```

```
26   void dispose() {
27     unfocusNode.dispose();
28   }
29 }
```

9.1.1_HomePage-Widget

```
1  import '/flutter_flow/flutter_flow_animations.dart';
2  import '/flutter_flow/flutter_flow_icon_button.dart';
3  import '/flutter_flow/flutter_flow_language_selector.dart';
4  import '/flutter_flow/flutter_flow_theme.dart';
5  import '/flutter_flow/flutter_flow_util.dart';
6  import '/flutter_flow/flutter_flow_widgets.dart';
7  import 'package:flip_card/flip_card.dart';
8  import 'package:flutter/material.dart';
9  import 'package:flutter/scheduler.dart';
10 import 'package:flutter_animate/flutter_animate.dart';
11 import 'package:font_awesome_flutter/font_awesome_flutter.dart';
12 import 'package:google_fonts/google_fonts.dart';
13 import 'package:provider/provider.dart';
14 import 'package:simple_gradient_text/simple_gradient_text.dart';
15
16 import 'home_page_model.dart';
17 export 'home_page_model.dart';
18
19 class HomePageWidget extends StatefulWidget {
20   const HomePageWidget({super.key});
21
22   @override
23   State<HomePageWidget> createState() => _HomePageWidgetState();
24 }
25
26 class _HomePageWidgetState extends State<HomePageWidget>
27     with TickerProviderStateMixin {
28   late HomePageModel _model;
29
30   final scaffoldKey = GlobalKey<ScaffoldState>();
31
32   final animationsMap = <String, AnimationInfo>{};
33
34   @override
35   void initState() {
36     super.initState();
37     _model = createModel(context, () => HomePageModel());
38
39     logFirebaseEvent('screen_view', parameters: {'screen_name': 'HomePage'});
40     animationsMap.addAll({
41       'imageOnActionTriggerAnimation': AnimationInfo(
42         trigger: AnimationTrigger.onActionTrigger,
43         applyInitialState: true,
44         effectsBuilder: () => [
45           RotateEffect(
46             curve: Curves.linear,
47             delay: 0.0.ms,
48             duration: 500.0.ms,
49             begin: 0.0,
50             end: 1.0,
51           ),
52         ],
53       ),
54       'imageOnPageLoadAnimation': AnimationInfo(
55         trigger: AnimationTrigger.onPageLoad,
```

```
56          applyInitialState: true,
57          effectsBuilder: () => [
58            ScaleEffect(
59              curve: Curves.easeInOut,
60              delay: 0.0.ms,
61              duration: 600.0.ms,
62              begin: Offset(1.0, 1.0),
63              end: Offset(1.0, 1.0),
64            ),
65            FadeEffect(
66              curve: Curves.easeInOut,
67              delay: 600.0.ms,
68              duration: 600.0.ms,
69              begin: 0.0,
70              end: 1.0,
71            ),
72          ],
73        ),
74        'containerOnActionTriggerAnimation': AnimationInfo(
75          trigger: AnimationTrigger.onActionTrigger,
76          applyInitialState: true,
77          effectsBuilder: () => [
78            MoveEffect(
79              curve: Curves.easeInOut,
80              delay: 0.0.ms,
81              duration: 600.0.ms,
82              begin: Offset(0.0, 0.0),
83              end: Offset(115.0, 0.0),
84            ),
85          ],
86        ),
87      });
88      setupAnimations(
89        animationsMap.values.where((anim) =>
90            anim.trigger == AnimationTrigger.onActionTrigger ||
91            !anim.applyInitialState),
92        this,
93      );
94
95      WidgetsBinding.instance.addPostFrameCallback((_) => setState(() {}));
96    }
97
98    @override
99    void dispose() {
100     _model.dispose();
101
102     super.dispose();
103   }
104
105   @override
106   Widget build(BuildContext context) {
107     return GestureDetector(
108       onTap: () => _model.unfocusNode.canRequestFocus
109           ? FocusScope.of(context).requestFocus(_model.unfocusNode)
110           : FocusScope.of(context).unfocus(),
111       child: Scaffold(
112         key: scaffoldKey,
113         backgroundColor: FlutterFlowTheme.of(context).primaryBackground,
114         body: SafeArea(
115           top: true,
116           child: Padding(
117             padding: EdgeInsetsDirectional.fromSTEB(0.0, 0.0, 0.0, 50.0),
118             child: Column(
119               mainAxisSize: MainAxisSize.max,
120               mainAxisAlignment: MainAxisAlignment.center,
```

```
121                children: [
122                  Align(
123                    alignment: AlignmentDirectional(0.0, 0.0),
124                    child: ClipRRect(
125                      borderRadius: BorderRadius.only(
126                        bottomLeft: Radius.circular(0.0),
127                        bottomRight: Radius.circular(0.0),
128                        topLeft: Radius.circular(0.0),
129                        topRight: Radius.circular(0.0),
130                      ),
131                      child: Image.asset(
132                        Theme.of(context).brightness == Brightness.dark
133                            ? 'assets/images/SVA.png'
134                            : 'assets/images/SVA.png',
135                        width: 209.0,
136                        height: 187.0,
137                        fit: BoxFit.contain,
138                        alignment: Alignment(0.0, 0.0),
139                      ),
140                    )
141                        .animateOnPageLoad(
142                            animationsMap['imageOnPageLoadAnimation']!)
143                        .animateOnActionTrigger(
144                          animationsMap['imageOnActionTriggerAnimation']!,
145                        ),
146                  ),
147                  Align(
148                    alignment: AlignmentDirectional(0.0, 0.0),
149                    child: SelectionArea(
150                        child: GradientText(
151                      FFLocalizations.of(context).getText(
152                        'hxlfjusb' /* Smart Irrigation System  */,
153                      ),
154                      textAlign: TextAlign.center,
155                      style: FlutterFlowTheme.of(context).titleLarge.override(
156                            fontFamily: 'Noto Serif',
157                            fontSize: 24.0,
158                            letterSpacing: 0.0,
159                            fontWeight: FontWeight.w600,
160                          ),
161                      colors: [
162                        FlutterFlowTheme.of(context).warning,
163                        FlutterFlowTheme.of(context).success
164                      ],
165                      gradientDirection: GradientDirection.btt,
166                      gradientType: GradientType.linear,
167                    )),
168                  ),
169                  FlipCard(
170                    fill: Fill.fillBack,
171                    direction: FlipDirection.HORIZONTAL,
172                    speed: 400,
173                    front: Padding(
174                      padding:
175                          EdgeInsetsDirectional.fromSTEB(0.0, 0.0, 0.0, 16.0),
176                      child: Container(
177                        width: 250.0,
178                        height: 50.0,
179                        decoration: BoxDecoration(
180                          color: Color(0xFFF1F4F8),
181                          borderRadius: BorderRadius.circular(12.0),
182                          border: Border.all(
183                            color: Color(0xFFE0E3E7),
184                            width: 1.0,
185                          ),
```

```
186                       ),
187                       child: Padding(
188                         padding: EdgeInsets.all(4.0),
189                         child: Row(
190                           mainAxisSize: MainAxisSize.max,
191                           mainAxisAlignment: MainAxisAlignment.start,
192                           children: [
193                             Expanded(
194                               child: InkWell(
195                                 splashColor: Colors.transparent,
196                                 focusColor: Colors.transparent,
197                                 hoverColor: Colors.transparent,
198                                 highlightColor: Colors.transparent,
199                                 onTap: () async {
200                                   logFirebaseEvent(
201                                       'HOME_PAGE_PAGE_Container_flm5vljq_ON_TAP');
202                                   logFirebaseEvent(
203                                       'Container_set_dark_mode_settings');
204                                   setDarkModeSetting(context, ThemeMode.light);
205                                 },
206                                 child: Container(
207                                   width: 115.0,
208                                   height: 100.0,
209                                   decoration: BoxDecoration(
210                                     color: Theme.of(context).brightness ==
211                                             Brightness.light
212                                         ? Colors.white
213                                         : Color(0xFFF1F4F8),
214                                     borderRadius: BorderRadius.circular(10.0),
215                                     border: Border.all(
216                                       color: valueOrDefault<Color>(
217                                         Theme.of(context).brightness ==
218                                                 Brightness.light
219                                             ? Color(0xFFE0E3E7)
220                                             : Color(0xFFF1F4F8),
221                                         Color(0xFFE0E3E7),
222                                       ),
223                                       width: 1.0,
224                                     ),
225                                   ),
226                                   child: Row(
227                                     mainAxisSize: MainAxisSize.max,
228                                     mainAxisAlignment: MainAxisAlignment.center,
229                                     children: [
230                                       Icon(
231                                         Icons.wb_sunny_rounded,
232                                         color: Theme.of(context).brightness ==
233                                                 Brightness.light
234                                             ? Color(0xFF14181B)
235                                             : Color(0xFF57636C),
236                                         size: 16.0,
237                                       ),
238                                       Padding(
239                                         padding: EdgeInsetsDirectional.fromSTEB(
240                                             4.0, 0.0, 0.0, 0.0),
241                                         child: Text(
242                                           FFLocalizations.of(context).getText(
243                                             'red20v28' /* Light Mode */,
244                                           ),
245                                           style: FlutterFlowTheme.of(context)
246                                               .bodyMedium
247                                               .override(
248                                                 fontFamily: 'Outfit',
249                                                 color: Theme.of(context)
250                                                         .brightness ==
```

```
                                                  Brightness.light
                                              ? Color(0xFF14181B)
                                              : Color(0xFF57636C),
                                          fontSize: 14.0,
                                          letterSpacing: 0.0,
                                          fontWeight: FontWeight.w500,
                                        ),
                                    ),
                                  ),
                                ],
                              ),
                            ),
                          ),
                        ),
                      ),
                      Expanded(
                        child: InkWell(
                          splashColor: Colors.transparent,
                          focusColor: Colors.transparent,
                          hoverColor: Colors.transparent,
                          highlightColor: Colors.transparent,
                          onTap: () async {
                            logFirebaseEvent(
                                'HOME_PAGE_PAGE_Container_rhznpnad_ON_TAP');
                            logFirebaseEvent(
                                'Container_set_dark_mode_settings');
                            setDarkModeSetting(context, ThemeMode.dark);
                          },
                          child: Container(
                            width: 115.0,
                            height: 100.0,
                            decoration: BoxDecoration(
                              color: Theme.of(context).brightness ==
                                      Brightness.dark
                                  ? Colors.white
                                  : Color(0xFFF1F4F8),
                              borderRadius: BorderRadius.circular(10.0),
                              border: Border.all(
                                color: valueOrDefault<Color>(
                                  Theme.of(context).brightness ==
                                          Brightness.dark
                                      ? Color(0xFFE0E3E7)
                                      : Color(0xFFF1F4F8),
                                  Color(0xFFF1F4F8),
                                ),
                                width: 1.0,
                              ),
                            ),
                            child: Row(
                              mainAxisSize: MainAxisSize.max,
                              mainAxisAlignment: MainAxisAlignment.center,
                              children: [
                                Icon(
                                  Icons.nightlight_round,
                                  color: Theme.of(context).brightness ==
                                          Brightness.dark
                                      ? Color(0xFF14181B)
                                      : Color(0xFF57636C),
                                  size: 16.0,
                                ),
                                Padding(
                                  padding: EdgeInsetsDirectional.fromSTEB(
                                      4.0, 0.0, 0.0, 0.0),
                                  child: Text(
                                    FFLocalizations.of(context).getText(
                                      'igdljdio' /* Dark Mode */,
```

```
                                        ),
                                        style: FlutterFlowTheme.of(context)
                                            .bodyMedium
                                            .override(
                                              fontFamily: 'Outfit',
                                              color: Theme.of(context)
                                                          .brightness ==
                                                      Brightness.dark
                                                  ? Color(0xFF14181B)
                                                  : Color(0xFF57636C),
                                              fontSize: 14.0,
                                              letterSpacing: 0.0,
                                              fontWeight: FontWeight.w500,
                                            ),
                                      ),
                                    ),
                                  ],
                                ),
                              ),
                            ).animateOnActionTrigger(
                              animationsMap[
                                  'containerOnActionTriggerAnimation']!,
                            ),
                          ),
                        ],
                      ),
                    ),
                  ),
                ),
                back: Container(),
              ),
              FlutterFlowLanguageSelector(
                width: 258.0,
                backgroundColor: FlutterFlowTheme.of(context).success,
                borderColor: Colors.transparent,
                dropdownIconColor: Colors.white,
                borderRadius: 8.0,
                textStyle: TextStyle(
                  color: Colors.white,
                  fontWeight: FontWeight.bold,
                  fontSize: 14.0,
                ),
                hideFlags: true,
                flagSize: 24.0,
                flagTextGap: 8.0,
                currentLanguage: FFLocalizations.of(context).languageCode,
                languages: FFLocalizations.languages(),
                onChanged: (lang) => setAppLanguage(context, lang),
              ),
              Align(
                alignment: AlignmentDirectional(0.0, -1.0),
                child: Padding(
                  padding:
                      EdgeInsetsDirectional.fromSTEB(0.0, 25.0, 0.0, 0.0),
                  child: FlutterFlowIconButton(
                    borderColor: FlutterFlowTheme.of(context).success,
                    borderRadius: 20.0,
                    borderWidth: 1.0,
                    buttonSize: 65.0,
                    fillColor: FlutterFlowTheme.of(context).success,
                    icon: Icon(
                      Icons.logout,
                      color: FlutterFlowTheme.of(context).primaryText,
                      size: 24.0,
                    ),
```

```
                        onPressed: () async {
                          logFirebaseEvent('HOME_PAGE_PAGE_logout_ICN_ON_TAP');
                          logFirebaseEvent('IconButton_navigate_to');

                          context.pushNamed('Login');
                        },
                      ),
                    ),
                  ),
                ],
              ),
            ),
          ),
        ),
      );
  }
}
```

9.1.2_AI Disease Detection-Model

```dart
import '/backend/gemini/gemini.dart';
import '/flutter_flow/flutter_flow_icon_button.dart';
import '/flutter_flow/flutter_flow_theme.dart';
import '/flutter_flow/flutter_flow_util.dart';
import '/flutter_flow/flutter_flow_widgets.dart';
import '/flutter_flow/upload_data.dart';
import 'a_i_chat_image_widget.dart' show AIChatImageWidget;
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:provider/provider.dart';

class AIChatImageModel extends FlutterFlowModel<AIChatImageWidget> {
///  State fields for stateful widgets in this page.

final unfocusNode = FocusNode();
// State field(s) for TextField widget.
FocusNode? textFieldFocusNode;
TextEditingController? textController;
String? Function(BuildContext, String?)? textControllerValidator;
bool isDataUploading = false;
FFUploadedFile uploadedLocalFile =
FFUploadedFile(bytes: Uint8List.fromList([]));

// Stores action output result for [Gemini - Text From Image] action in Button
widget.
String? outPut;

@override
void initState(BuildContext context) {}

@override
void dispose() {
unfocusNode.dispose();
textFieldFocusNode?.dispose();
textController?.dispose();
}
}
```

9.1.3_AI Disease Detection-Widget

```dart
import '/backend/gemini/gemini.dart';
import '/flutter_flow/flutter_flow_icon_button.dart';
import '/flutter_flow/flutter_flow_theme.dart';
import '/flutter_flow/flutter_flow_util.dart';
import '/flutter_flow/flutter_flow_widgets.dart';
import '/flutter_flow/upload_data.dart';
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:provider/provider.dart';

import 'a_i_chat_image_model.dart';
export 'a_i_chat_image_model.dart';

class AIChatImageWidget extends StatefulWidget {
  const AIChatImageWidget({super.key});

  @override
  State<AIChatImageWidget> createState() => _AIChatImageWidgetState();
}

class _AIChatImageWidgetState extends State<AIChatImageWidget> {
  late AIChatImageModel _model;

  final scaffoldKey = GlobalKey<ScaffoldState>();

  @override
  void initState() {
    super.initState();
    _model = createModel(context, () => AIChatImageModel());

    logFirebaseEvent('screen_view',
        parameters: {'screen_name': 'AI_Chat_Image'});
    _model.textController ??= TextEditingController();
    _model.textFieldFocusNode ??= FocusNode();

    WidgetsBinding.instance.addPostFrameCallback((_) => setState(() {}));
  }

  @override
  void dispose() {
    _model.dispose();

    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: () => _model.unfocusNode.canRequestFocus
          ? FocusScope.of(context).requestFocus(_model.unfocusNode)
          : FocusScope.of(context).unfocus(),
      child: Scaffold(
        key: scaffoldKey,
        backgroundColor: FlutterFlowTheme.of(context).primaryBackground,
        appBar: AppBar(
          backgroundColor: FlutterFlowTheme.of(context).success,
          automaticallyImplyLeading: false,
          leading: FlutterFlowIconButton(
            borderColor: Colors.transparent,
            borderRadius: 30.0,
            borderWidth: 1.0,
            buttonSize: 60.0,
```

```
63              icon: Icon(
64                Icons.arrow_back_rounded,
65                color: Colors.white,
66                size: 30.0,
67              ),
68              onPressed: () async {
69                logFirebaseEvent('A_I_CHAT_IMAGE_arrow_back_rounded_ICN_ON');
70                logFirebaseEvent('IconButton_navigate_back');
71                context.pop();
72              },
73            ),
74            title: Text(
75              FFLocalizations.of(context).getText(
76                't9ioq7ns' /* AI Disease detection */,
77              ),
78              style: FlutterFlowTheme.of(context).headlineMedium.override(
79                    fontFamily: 'Outfit',
80                    color: Colors.white,
81                    fontSize: 22.0,
82                    letterSpacing: 0.0,
83                  ),
84            ),
85            actions: [],
86            centerTitle: true,
87            elevation: 2.0,
88          ),
89          body: SafeArea(
90            top: true,
91            child: SingleChildScrollView(
92              child: Column(
93                mainAxisSize: MainAxisSize.max,
94                children: [
95                  Padding(
96                    padding: EdgeInsetsDirectional.fromSTEB(8.0, 25.0, 8.0, 0.0),
97                    child: TextFormField(
98                      controller: _model.textController,
99                      focusNode: _model.textFieldFocusNode,
100                     autofocus: true,
101                     obscureText: false,
102                     decoration: InputDecoration(
103                       labelText: FFLocalizations.of(context).getText(
104                         'zo8mwzzr' /* Write your question */,
105                       ),
106                       labelStyle:
107                           FlutterFlowTheme.of(context).labelMedium.override(
108                                 fontFamily: 'Readex Pro',
109                                 fontSize: 16.0,
110                                 letterSpacing: 0.0,
111                                 fontWeight: FontWeight.bold,
112                               ),
113                       hintStyle:
114                           FlutterFlowTheme.of(context).labelMedium.override(
115                                 fontFamily: 'Readex Pro',
116                                 fontSize: 16.0,
117                                 letterSpacing: 0.0,
118                                 fontWeight: FontWeight.w600,
119                               ),
120                       enabledBorder: OutlineInputBorder(
121                         borderSide: BorderSide(
122                           color: FlutterFlowTheme.of(context).alternate,
123                           width: 2.0,
124                         ),
125                         borderRadius: BorderRadius.circular(8.0),
126                       ),
127                       focusedBorder: OutlineInputBorder(
```

```
                        borderSide: BorderSide(
                          color: FlutterFlowTheme.of(context).primary,
                          width: 2.0,
                        ),
                        borderRadius: BorderRadius.circular(8.0),
                      ),
                      errorBorder: OutlineInputBorder(
                        borderSide: BorderSide(
                          color: FlutterFlowTheme.of(context).error,
                          width: 2.0,
                        ),
                        borderRadius: BorderRadius.circular(8.0),
                      ),
                      focusedErrorBorder: OutlineInputBorder(
                        borderSide: BorderSide(
                          color: FlutterFlowTheme.of(context).error,
                          width: 2.0,
                        ),
                        borderRadius: BorderRadius.circular(8.0),
                      ),
                      prefixIcon: Icon(
                        Icons.question_mark_outlined,
                        color: FlutterFlowTheme.of(context).success,
                      ),
                    ),
                    style: FlutterFlowTheme.of(context).bodyMedium.override(
                          fontFamily: 'Readex Pro',
                          letterSpacing: 0.0,
                        ),
                    validator:
                        _model.textControllerValidator.asValidator(context),
                  ),
                ),
                Padding(
                  padding:
                      EdgeInsetsDirectional.fromSTEB(10.0, 15.0, 10.0, 15.0),
                  child: ClipRRect(
                    borderRadius: BorderRadius.only(
                      bottomLeft: Radius.circular(25.0),
                      bottomRight: Radius.circular(25.0),
                      topLeft: Radius.circular(25.0),
                      topRight: Radius.circular(25.0),
                    ),
                    child: Image.memory(
                      _model.uploadedLocalFile.bytes ?? Uint8List.fromList([]),
                      width: 365.0,
                      height: 200.0,
                      fit: BoxFit.fill,
                    ),
                  ),
                ),
                FFButtonWidget(
                  onPressed: () async {
                    logFirebaseEvent('A_I_CHAT_IMAGE_UPLOAD_IMAGE_BTN_ON_TAP');
                    logFirebaseEvent('Button_store_media_for_upload');
                    final selectedMedia =
                        await selectMediaWithSourceBottomSheet(
                      context: context,
                      allowPhoto: true,
                    );
                    if (selectedMedia != null &&
                        selectedMedia.every((m) =>
                            validateFileFormat(m.storagePath, context))) {
                      setState(() => _model.isDataUploading = true);
                      var selectedUploadedFiles = <FFUploadedFile>[];
```

```
193
194                     try {
195                       selectedUploadedFiles = selectedMedia
196                           .map((m) => FFUploadedFile(
197                                 name: m.storagePath.split('/').last,
198                                 bytes: m.bytes,
199                                 height: m.dimensions?.height,
200                                 width: m.dimensions?.width,
201                                 blurHash: m.blurHash,
202                               ))
203                           .toList();
204                     } finally {
205                       _model.isDataUploading = false;
206                     }
207                     if (selectedUploadedFiles.length ==
208                         selectedMedia.length) {
209                       setState(() {
210                         _model.uploadedLocalFile =
211                             selectedUploadedFiles.first;
212                       });
213                     } else {
214                       setState(() {});
215                       return;
216                     }
217                   }
218                 },
219                 text: FFLocalizations.of(context).getText(
220                   'pmqybh08' /* Upload Image */,
221                 ),
222                 icon: Icon(
223                   Icons.cloud_upload_outlined,
224                   size: 15.0,
225                 ),
226                 options: FFButtonOptions(
227                   width: 200.0,
228                   height: 50.0,
229                   padding:
230                       EdgeInsetsDirectional.fromSTEB(24.0, 15.0, 24.0, 15.0),
231                   iconPadding:
232                       EdgeInsetsDirectional.fromSTEB(0.0, 0.0, 0.0, 0.0),
233                   color: FlutterFlowTheme.of(context).primaryBackground,
234                   textStyle: FlutterFlowTheme.of(context).titleSmall.override(
235                         fontFamily: 'Readex Pro',
236                         color: FlutterFlowTheme.of(context).success,
237                         letterSpacing: 0.0,
238                         fontWeight: FontWeight.bold,
239                       ),
240                   elevation: 3.0,
241                   borderSide: BorderSide(
242                     color: FlutterFlowTheme.of(context).success,
243                     width: 3.0,
244                   ),
245                   borderRadius: BorderRadius.circular(8.0),
246                 ),
247               ),
248               Padding(
249                 padding:
250                     EdgeInsetsDirectional.fromSTEB(10.0, 15.0, 10.0, 15.0),
251                 child: FFButtonWidget(
252                   onPressed: () async {
253                     logFirebaseEvent('A_I_CHAT_IMAGE_PAGE_SEND_BTN_ON_TAP');
254                     logFirebaseEvent('Button_gemini');
255                     await geminiTextFromImage(
256                       context,
257                       _model.textController.text,
```

```
258                         uploadImageBytes: _model.uploadedLocalFile,
259                       ).then((generatedText) {
260                         safeSetState(() => _model.outPut = generatedText);
261                       });
262 
263                       setState(() {});
264                     },
265                     text: FFLocalizations.of(context).getText(
266                       'jmmivw4g' /* Send */,
267                     ),
268                     icon: Icon(
269                       Icons.schedule_send_outlined,
270                       size: 15.0,
271                     ),
272                     options: FFButtonOptions(
273                       height: 40.0,
274                       padding:
275                           EdgeInsetsDirectional.fromSTEB(24.0, 0.0, 24.0, 0.0),
276                       iconPadding:
277                           EdgeInsetsDirectional.fromSTEB(0.0, 0.0, 0.0, 0.0),
278                       color: FlutterFlowTheme.of(context).success,
279                       textStyle:
280                           FlutterFlowTheme.of(context).titleSmall.override(
281                                 fontFamily: 'Readex Pro',
282                                 color: Colors.white,
283                                 letterSpacing: 0.0,
284                                 fontWeight: FontWeight.w800,
285                               ),
286                       elevation: 3.0,
287                       borderSide: BorderSide(
288                         color: Colors.transparent,
289                         width: 1.0,
290                       ),
291                       borderRadius: BorderRadius.circular(8.0),
292                     ),
293                   ),
294                 ),
295                 Padding(
296                   padding:
297                       EdgeInsetsDirectional.fromSTEB(10.0, 15.0, 10.0, 15.0),
298                   child: Text(
299                     valueOrDefault<String>(
300                       _model.outPut,
301                       'Wait for the image to be uploaded and your question',
302                     ),
303                     textAlign: TextAlign.center,
304                     style: FlutterFlowTheme.of(context).bodyMedium.override(
305                           fontFamily: 'Readex Pro',
306                           fontSize: 16.0,
307                           letterSpacing: 0.0,
308                           fontWeight: FontWeight.bold,
309                         ),
310                   ),
311                 ),
312               ],
313             ),
314           ),
315         ),
316       ),
317     );
318   }
319 }
```

9.1.4_Sensors-Model

```dart
import '/backend/backend.dart';
import '/flutter_flow/flutter_flow_icon_button.dart';
import '/flutter_flow/flutter_flow_theme.dart';
import '/flutter_flow/flutter_flow_util.dart';
import '/flutter_flow/flutter_flow_widgets.dart';
import 'sensors_widget.dart' show SensorsWidget;
import 'package:flutter/material.dart';
import 'package:font_awesome_flutter/font_awesome_flutter.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:percent_indicator/percent_indicator.dart';
import 'package:provider/provider.dart';

class SensorsModel extends FlutterFlowModel<SensorsWidget> {
  ///  State fields for stateful widgets in this page.

  final unfocusNode = FocusNode();

  @override
  void initState(BuildContext context) {}

  @override
  void dispose() {
    unfocusNode.dispose();
  }
```

9.1.5_Sensors-Widget

```dart
import '/backend/backend.dart';
import '/flutter_flow/flutter_flow_icon_button.dart';
import '/flutter_flow/flutter_flow_theme.dart';
import '/flutter_flow/flutter_flow_util.dart';
import '/flutter_flow/flutter_flow_widgets.dart';
import 'package:flutter/material.dart';
import 'package:font_awesome_flutter/font_awesome_flutter.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:percent_indicator/percent_indicator.dart';
import 'package:provider/provider.dart';

import 'sensors_model.dart';
export 'sensors_model.dart';

class SensorsWidget extends StatefulWidget {
  const SensorsWidget({super.key});

  @override
  State<SensorsWidget> createState() => _SensorsWidgetState();
}

class _SensorsWidgetState extends State<SensorsWidget> {
  late SensorsModel _model;

  final scaffoldKey = GlobalKey<ScaffoldState>();

  @override
  void initState() {
    super.initState();
    _model = createModel(context, () => SensorsModel());
```

```
31
32    logFirebaseEvent('screen_view', parameters: {'screen_name': 'Sensors'});
33    WidgetsBinding.instance.addPostFrameCallback((_) => setState(() {}));
34  }
35
36  @override
37  void dispose() {
38    _model.dispose();
39
40    super.dispose();
41  }
42
43  @override
44  Widget build(BuildContext context) {
45    return GestureDetector(
46      onTap: () => _model.unfocusNode.canRequestFocus
47          ? FocusScope.of(context).requestFocus(_model.unfocusNode)
48          : FocusScope.of(context).unfocus(),
49      child: Scaffold(
50        key: scaffoldKey,
51        backgroundColor: FlutterFlowTheme.of(context).primaryBackground,
52        appBar: AppBar(
53          backgroundColor: FlutterFlowTheme.of(context).success,
54          automaticallyImplyLeading: false,
55          leading: FlutterFlowIconButton(
56            borderColor: Colors.transparent,
57            borderRadius: 30.0,
58            borderWidth: 1.0,
59            buttonSize: 60.0,
60            icon: Icon(
61              Icons.arrow_back_rounded,
62              color: Colors.white,
63              size: 30.0,
64            ),
65            onPressed: () async {
66              logFirebaseEvent('SENSORS_arrow_back_rounded_ICN_ON_TAP');
67              logFirebaseEvent('IconButton_navigate_back');
68              context.pop();
69            },
70          ),
71          title: Text(
72            FFLocalizations.of(context).getText(
73              'r5bsyeoz' /* Sensors */,
74            ),
75            style: FlutterFlowTheme.of(context).headlineMedium.override(
76                  fontFamily: 'Outfit',
77                  color: Colors.white,
78                  fontSize: 22.0,
79                  letterSpacing: 0.0,
80                  fontWeight: FontWeight.w800,
81                ),
82          ),
83          actions: [],
84          centerTitle: true,
85          elevation: 2.0,
86        ),
87        body: SafeArea(
88          top: true,
89          child: Column(
90            mainAxisSize: MainAxisSize.max,
91            children: [
92              Row(
93                mainAxisSize: MainAxisSize.max,
94                mainAxisAlignment: MainAxisAlignment.spaceAround,
95                children: [
```

```
 96                   StreamBuilder<List<SensorsRecord>>(
 97                     stream: querySensorsRecord(
 98                       singleRecord: true,
 99                     ),
100                     builder: (context, snapshot) {
101                       // Customize what your widget looks like when it's loading.
102                       if (!snapshot.hasData) {
103                         return Center(
104                           child: SizedBox(
105                             width: 50.0,
106                             height: 50.0,
107                             child: CircularProgressIndicator(
108                               valueColor: AlwaysStoppedAnimation<Color>(
109                                 FlutterFlowTheme.of(context).primary,
110                               ),
111                             ),
112                           ),
113                         );
114                       }
115                       List<SensorsRecord> containerSensorsRecordList =
116                           snapshot.data!;
117                       // Return an empty Container when the item does not exist.
118                       if (snapshot.data!.isEmpty) {
119                         return Container();
120                       }
121                       final containerSensorsRecord =
122                           containerSensorsRecordList.isNotEmpty
123                               ? containerSensorsRecordList.first
124                               : null;
125                       return Container(
126                         width: 192.0,
127                         height: 202.0,
128                         decoration: BoxDecoration(
129                           color:
130                               FlutterFlowTheme.of(context).secondaryBackground,
131                         ),
132                         child: Padding(
133                           padding: EdgeInsetsDirectional.fromSTEB(
134                               0.0, 10.0, 0.0, 0.0),
135                           child: Stack(
136                             children: [
137                               Align(
138                                 alignment: AlignmentDirectional(0.0, 0.0),
139                                 child: Padding(
140                                   padding: EdgeInsetsDirectional.fromSTEB(
141                                       0.0, 20.0, 0.0, 0.0),
142                                   child: CircularPercentIndicator(
143                                     percent: 1.0,
144                                     radius: 75.0,
145                                     lineWidth: 12.0,
146                                     animation: true,
147                                     animateFromLastPercent: true,
148                                     progressColor:
149                                         FlutterFlowTheme.of(context).primary,
150                                     backgroundColor:
151                                         FlutterFlowTheme.of(context).accent4,
152                                     center: Text(
153                                       containerSensorsRecord!.temperature
154                                           .toString(),
155                                       style: FlutterFlowTheme.of(context)
156                                           .headlineSmall
157                                           .override(
158                                             fontFamily: 'Outfit',
159                                             letterSpacing: 0.0,
160                                           ),
```

```
                                    ),
                                  ),
                                ),
                              ),
                              Align(
                                alignment: AlignmentDirectional(0.0, -1.0),
                                child: Padding(
                                  padding: EdgeInsetsDirectional.fromSTEB(
                                      0.0, 5.0, 0.0, 0.0),
                                  child: Text(
                                    FFLocalizations.of(context).getText(
                                      'dn2ethry' /* Temperature */,
                                    ),
                                    textAlign: TextAlign.center,
                                    style: FlutterFlowTheme.of(context)
                                        .bodyMedium
                                        .override(
                                          fontFamily: 'Readex Pro',
                                          fontSize: 18.0,
                                          letterSpacing: 0.0,
                                          fontWeight: FontWeight.bold,
                                        ),
                                  ),
                                ),
                              ),
                              Align(
                                alignment: AlignmentDirectional(1.0, -1.0),
                                child: Padding(
                                  padding: EdgeInsetsDirectional.fromSTEB(
                                      0.0, 5.0, 10.0, 0.0),
                                  child: FaIcon(
                                    FontAwesomeIcons.temperatureLow,
                                    color: FlutterFlowTheme.of(context).primary,
                                    size: 24.0,
                                  ),
                                ),
                              ),
                            ],
                          ),
                        ),
                      );
                    },
                  ),
                  StreamBuilder<List<SensorsRecord>>(
                    stream: querySensorsRecord(
                      singleRecord: true,
                    ),
                    builder: (context, snapshot) {
                      // Customize what your widget looks like when it's loading.
                      if (!snapshot.hasData) {
                        return Center(
                          child: SizedBox(
                            width: 50.0,
                            height: 50.0,
                            child: CircularProgressIndicator(
                              valueColor: AlwaysStoppedAnimation<Color>(
                                FlutterFlowTheme.of(context).primary,
                              ),
                            ),
                          ),
                        );
                      }
                      List<SensorsRecord> containerSensorsRecordList =
                          snapshot.data!;
                      // Return an empty Container when the item does not exist.
```

```
                    if (snapshot.data!.isEmpty) {
                      return Container();
                    }
                    final containerSensorsRecord =
                        containerSensorsRecordList.isNotEmpty
                            ? containerSensorsRecordList.first
                            : null;
                    return Container(
                      width: 198.0,
                      height: 202.0,
                      decoration: BoxDecoration(
                        color:
                            FlutterFlowTheme.of(context).secondaryBackground,
                      ),
                      child: Padding(
                        padding: EdgeInsetsDirectional.fromSTEB(
                            0.0, 10.0, 0.0, 0.0),
                        child: Stack(
                          children: [
                            Align(
                              alignment: AlignmentDirectional(0.0, 0.0),
                              child: Padding(
                                padding: EdgeInsetsDirectional.fromSTEB(
                                    0.0, 20.0, 0.0, 0.0),
                                child: CircularPercentIndicator(
                                  percent: 1.0,
                                  radius: 75.0,
                                  lineWidth: 12.0,
                                  animation: true,
                                  animateFromLastPercent: true,
                                  progressColor:
                                      FlutterFlowTheme.of(context).success,
                                  backgroundColor:
                                      FlutterFlowTheme.of(context).accent4,
                                  center: Text(
                                    containerSensorsRecord!.soilMoisture
                                        .toString(),
                                    style: FlutterFlowTheme.of(context)
                                        .headlineSmall
                                        .override(
                                          fontFamily: 'Outfit',
                                          letterSpacing: 0.0,
                                        ),
                                  ),
                                ),
                              ),
                            ),
                            Align(
                              alignment: AlignmentDirectional(0.0, -1.0),
                              child: Padding(
                                padding: EdgeInsetsDirectional.fromSTEB(
                                    0.0, 5.0, 0.0, 0.0),
                                child: Text(
                                  FFLocalizations.of(context).getText(
                                    '76is3u2p' /* Soil moisture */,
                                  ),
                                  textAlign: TextAlign.center,
                                  style: FlutterFlowTheme.of(context)
                                      .bodyMedium
                                      .override(
                                        fontFamily: 'Readex Pro',
                                        fontSize: 18.0,
                                        letterSpacing: 0.0,
                                        fontWeight: FontWeight.bold,
                                      ),
```

```
                                    ),
                                  ),
                                ),
                                Align(
                                  alignment: AlignmentDirectional(1.0, -1.0),
                                  child: Padding(
                                    padding: EdgeInsetsDirectional.fromSTEB(
                                        0.0, 5.0, 10.0, 0.0),
                                    child: Icon(
                                      Icons.water,
                                      color: FlutterFlowTheme.of(context).success,
                                      size: 24.0,
                                    ),
                                  ),
                                ),
                              ],
                            ),
                          ),
                        );
                      },
                    ),
                  ],
                ),
                Row(
                  mainAxisSize: MainAxisSize.max,
                  mainAxisAlignment: MainAxisAlignment.spaceAround,
                  children: [
                    StreamBuilder<List<SensorsRecord>>(
                      stream: querySensorsRecord(
                        singleRecord: true,
                      ),
                      builder: (context, snapshot) {
                        // Customize what your widget looks like when it's loading.
                        if (!snapshot.hasData) {
                          return Center(
                            child: SizedBox(
                              width: 50.0,
                              height: 50.0,
                              child: CircularProgressIndicator(
                                valueColor: AlwaysStoppedAnimation<Color>(
                                  FlutterFlowTheme.of(context).primary,
                                ),
                              ),
                            ),
                          );
                        }
                        List<SensorsRecord> containerSensorsRecordList =
                            snapshot.data!;
                        // Return an empty Container when the item does not exist.
                        if (snapshot.data!.isEmpty) {
                          return Container();
                        }
                        final containerSensorsRecord =
                            containerSensorsRecordList.isNotEmpty
                                ? containerSensorsRecordList.first
                                : null;
                        return Container(
                          width: 192.0,
                          height: 202.0,
                          decoration: BoxDecoration(
                            color:
                                FlutterFlowTheme.of(context).secondaryBackground,
                          ),
                          child: Stack(
                            children: [
```

```
                            Align(
                              alignment: AlignmentDirectional(0.0, 0.0),
                              child: Padding(
                                padding: EdgeInsetsDirectional.fromSTEB(
                                    0.0, 20.0, 0.0, 0.0),
                                child: CircularPercentIndicator(
                                  percent: 1.0,
                                  radius: 75.0,
                                  lineWidth: 12.0,
                                  animation: true,
                                  animateFromLastPercent: true,
                                  progressColor:
                                      FlutterFlowTheme.of(context).tertiary,
                                  backgroundColor:
                                      FlutterFlowTheme.of(context).accent4,
                                  center: Text(
                                    containerSensorsRecord!.humidity.toString(),
                                    style: FlutterFlowTheme.of(context)
                                        .headlineSmall
                                        .override(
                                          fontFamily: 'Outfit',
                                          letterSpacing: 0.0,
                                        ),
                                  ),
                                ),
                              ),
                            ),
                            Align(
                              alignment: AlignmentDirectional(0.0, -1.0),
                              child: Padding(
                                padding: EdgeInsetsDirectional.fromSTEB(
                                    0.0, 5.0, 0.0, 0.0),
                                child: Text(
                                  FFLocalizations.of(context).getText(
                                    '8n1rgfug' /* Humidity */,
                                  ),
                                  textAlign: TextAlign.center,
                                  style: FlutterFlowTheme.of(context)
                                      .bodyMedium
                                      .override(
                                        fontFamily: 'Readex Pro',
                                        fontSize: 18.0,
                                        letterSpacing: 0.0,
                                        fontWeight: FontWeight.bold,
                                      ),
                                ),
                              ),
                            ),
                            Align(
                              alignment: AlignmentDirectional(1.0, -1.0),
                              child: Padding(
                                padding: EdgeInsetsDirectional.fromSTEB(
                                    0.0, 5.0, 10.0, 0.0),
                                child: Icon(
                                  Icons.wash,
                                  color: FlutterFlowTheme.of(context).tertiary,
                                  size: 24.0,
                                ),
                              ),
                            ),
                          ],
                        ),
                      );
                    },
                  ),
```

```
421                   StreamBuilder<List<SensorsRecord>>(
422                     stream: querySensorsRecord(
423                       singleRecord: true,
424                     ),
425                     builder: (context, snapshot) {
426                       // Customize what your widget looks like when it's loading.
427                       if (!snapshot.hasData) {
428                         return Center(
429                           child: SizedBox(
430                             width: 50.0,
431                             height: 50.0,
432                             child: CircularProgressIndicator(
433                               valueColor: AlwaysStoppedAnimation<Color>(
434                                 FlutterFlowTheme.of(context).primary,
435                               ),
436                             ),
437                           ),
438                         );
439                       }
440                       List<SensorsRecord> containerSensorsRecordList =
441                           snapshot.data!;
442                       // Return an empty Container when the item does not exist.
443                       if (snapshot.data!.isEmpty) {
444                         return Container();
445                       }
446                       final containerSensorsRecord =
447                           containerSensorsRecordList.isNotEmpty
448                               ? containerSensorsRecordList.first
449                               : null;
450                       return Container(
451                         width: 198.0,
452                         height: 202.0,
453                         decoration: BoxDecoration(
454                           color:
455                               FlutterFlowTheme.of(context).secondaryBackground,
456                         ),
457                         child: Stack(
458                           children: [
459                             Align(
460                               alignment: AlignmentDirectional(0.0, 0.0),
461                               child: Padding(
462                                 padding: EdgeInsetsDirectional.fromSTEB(
463                                     0.0, 20.0, 0.0, 0.0),
464                                 child: CircularPercentIndicator(
465                                   percent: 1.0,
466                                   radius: 75.0,
467                                   lineWidth: 12.0,
468                                   animation: true,
469                                   animateFromLastPercent: true,
470                                   progressColor: Color(0xE759E9FF),
471                                   backgroundColor:
472                                       FlutterFlowTheme.of(context).accent4,
473                                   center: Text(
474                                     containerSensorsRecord!.waterLevel
475                                         .toString(),
476                                     style: FlutterFlowTheme.of(context)
477                                         .headlineSmall
478                                         .override(
479                                           fontFamily: 'Outfit',
480                                           letterSpacing: 0.0,
481                                         ),
482                                   ),
483                                 ),
484                               ),
485                             ),
```

```
                              Align(
                                alignment: AlignmentDirectional(0.0, -1.0),
                                child: Padding(
                                  padding: EdgeInsetsDirectional.fromSTEB(
                                      0.0, 5.0, 0.0, 0.0),
                                  child: Text(
                                    FFLocalizations.of(context).getText(
                                      'v9sxqaz1' /* Water Level */,
                                    ),
                                    textAlign: TextAlign.center,
                                    style: FlutterFlowTheme.of(context)
                                        .bodyMedium
                                        .override(
                                          fontFamily: 'Readex Pro',
                                          fontSize: 18.0,
                                          letterSpacing: 0.0,
                                          fontWeight: FontWeight.bold,
                                        ),
                                  ),
                                ),
                              ),
                              Align(
                                alignment: AlignmentDirectional(1.0, -1.0),
                                child: Padding(
                                  padding: EdgeInsetsDirectional.fromSTEB(
                                      0.0, 5.0, 10.0, 0.0),
                                  child: Icon(
                                    Icons.water_drop_sharp,
                                    color: Color(0xE759E9FF),
                                    size: 24.0,
                                  ),
                                ),
                              ),
                            ],
                          ),
                        );
                      },
                    ),
                  ],
                ),
                Row(
                  mainAxisSize: MainAxisSize.max,
                  mainAxisAlignment: MainAxisAlignment.spaceAround,
                  children: [
                    StreamBuilder<List<SensorsRecord>>(
                      stream: querySensorsRecord(
                        singleRecord: true,
                      ),
                      builder: (context, snapshot) {
                        // Customize what your widget looks like when it's loading.
                        if (!snapshot.hasData) {
                          return Center(
                            child: SizedBox(
                              width: 50.0,
                              height: 50.0,
                              child: CircularProgressIndicator(
                                valueColor: AlwaysStoppedAnimation<Color>(
                                  FlutterFlowTheme.of(context).primary,
                                ),
                              ),
                            ),
                          );
                        }
                        List<SensorsRecord> containerSensorsRecordList =
                            snapshot.data!;
```

```
551                     // Return an empty Container when the item does not exist.
552                     if (snapshot.data!.isEmpty) {
553                       return Container();
554                     }
555                     final containerSensorsRecord =
556                         containerSensorsRecordList.isNotEmpty
557                             ? containerSensorsRecordList.first
558                             : null;
559                     return Container(
560                       width: 192.0,
561                       height: 202.0,
562                       decoration: BoxDecoration(
563                         color:
564                             FlutterFlowTheme.of(context).secondaryBackground,
565                       ),
566                       child: Stack(
567                         children: [
568                           Align(
569                             alignment: AlignmentDirectional(0.0, -1.0),
570                             child: Padding(
571                               padding: EdgeInsetsDirectional.fromSTEB(
572                                   0.0, 40.0, 0.0, 0.0),
573                               child: Text(
574                                 FFLocalizations.of(context).getText(
575                                   '6ur606ef' /* LED */,
576                                 ),
577                                 textAlign: TextAlign.center,
578                                 style: FlutterFlowTheme.of(context)
579                                     .bodyMedium
580                                     .override(
581                                       fontFamily: 'Readex Pro',
582                                       fontSize: 18.0,
583                                       letterSpacing: 0.0,
584                                       fontWeight: FontWeight.bold,
585                                     ),
586                               ),
587                             ),
588                           ),
589                           Builder(
590                             builder: (context) {
591                               if (containerSensorsRecord?.ledPin ?? false) {
592                                 return Align(
593                                   alignment: AlignmentDirectional(0.0, 0.0),
594                                   child: Icon(
595                                     Icons.lightbulb_rounded,
596                                     color:
597                                         FlutterFlowTheme.of(context).success,
598                                     size: 75.0,
599                                   ),
600                                 );
601                               } else {
602                                 return Align(
603                                   alignment: AlignmentDirectional(0.0, 0.0),
604                                   child: Icon(
605                                     Icons.lightbulb_outline_rounded,
606                                     color: FlutterFlowTheme.of(context).error,
607                                     size: 75.0,
608                                   ),
609                                 );
610                               }
611                             },
612                           ),
613                         ],
614                       ),
615                     );
```

```
616                     },
617                   ),
618                   StreamBuilder<List<SensorsRecord>>(
619                     stream: querySensorsRecord(
620                       singleRecord: true,
621                     ),
622                     builder: (context, snapshot) {
623                       // Customize what your widget looks like when it's loading.
624                       if (!snapshot.hasData) {
625                         return Center(
626                           child: SizedBox(
627                             width: 50.0,
628                             height: 50.0,
629                             child: CircularProgressIndicator(
630                               valueColor: AlwaysStoppedAnimation<Color>(
631                                 FlutterFlowTheme.of(context).primary,
632                               ),
633                             ),
634                           ),
635                         );
636                       }
637                       List<SensorsRecord> containerSensorsRecordList =
638                           snapshot.data!;
639                       // Return an empty Container when the item does not exist.
640                       if (snapshot.data!.isEmpty) {
641                         return Container();
642                       }
643                       final containerSensorsRecord =
644                           containerSensorsRecordList.isNotEmpty
645                               ? containerSensorsRecordList.first
646                               : null;
647                       return Container(
648                         width: 198.0,
649                         height: 202.0,
650                         decoration: BoxDecoration(
651                           color:
652                               FlutterFlowTheme.of(context).secondaryBackground,
653                         ),
654                         child: Stack(
655                           children: [
656                             Align(
657                               alignment: AlignmentDirectional(0.0, -1.0),
658                               child: Padding(
659                                 padding: EdgeInsetsDirectional.fromSTEB(
660                                     0.0, 40.0, 0.0, 0.0),
661                                 child: Text(
662                                   FFLocalizations.of(context).getText(
663                                     's4nqfbcj' /* LED Strip */,
664                                   ),
665                                   textAlign: TextAlign.center,
666                                   style: FlutterFlowTheme.of(context)
667                                       .bodyMedium
668                                       .override(
669                                         fontFamily: 'Readex Pro',
670                                         fontSize: 18.0,
671                                         letterSpacing: 0.0,
672                                         fontWeight: FontWeight.bold,
673                                       ),
674                                 ),
675                               ),
676                             ),
677                             Builder(
678                               builder: (context) {
679                                 if (containerSensorsRecord?.ledStripState ??
680                                     false) {
```

```
                                    return Align(
                                      alignment: AlignmentDirectional(0.0, 0.0),
                                      child: Icon(
                                        Icons.power_rounded,
                                        color:
                                            FlutterFlowTheme.of(context).success,
                                        size: 75.0,
                                      ),
                                    );
                                  } else {
                                    return Align(
                                      alignment: AlignmentDirectional(0.0, 0.0),
                                      child: Icon(
                                        Icons.power_outlined,
                                        color: FlutterFlowTheme.of(context).error,
                                        size: 75.0,
                                      ),
                                    );
                                  }
                                },
                              ),
                            ],
                          ),
                        );
                      },
                    ),
                  ],
                ),
                Stack(
                  children: [
                    Padding(
                      padding:
                          EdgeInsetsDirectional.fromSTEB(0.0, 10.0, 0.0, 10.0),
                      child: StreamBuilder<List<SensorsRecord>>(
                        stream: querySensorsRecord(
                          singleRecord: true,
                        ),
                        builder: (context, snapshot) {
                          // Customize what your widget looks like when it's
loading.
                          if (!snapshot.hasData) {
                            return Center(
                              child: SizedBox(
                                width: 50.0,
                                height: 50.0,
                                child: CircularProgressIndicator(
                                  valueColor: AlwaysStoppedAnimation<Color>(
                                    FlutterFlowTheme.of(context).primary,
                                  ),
                                ),
                              ),
                            );
                          }
                          List<SensorsRecord> rowSensorsRecordList =
                              snapshot.data!;
                          // Return an empty Container when the item does not exist.
                          if (snapshot.data!.isEmpty) {
                            return Container();
                          }
                          final rowSensorsRecord = rowSensorsRecordList.isNotEmpty
                              ? rowSensorsRecordList.first
                              : null;
                          return Row(
                            mainAxisSize: MainAxisSize.max,
                            mainAxisAlignment: MainAxisAlignment.spaceAround,
```

```
                          children: [
                            Padding(
                              padding: EdgeInsetsDirectional.fromSTEB(
                                  20.0, 10.0, 0.0, 0.0),
                              child: Text(
                                FFLocalizations.of(context).getText(
                                  'p0ulqlbz' /* Water pump */,
                                ),
                                style: FlutterFlowTheme.of(context)
                                    .bodyMedium
                                    .override(
                                      fontFamily: 'Readex Pro',
                                      fontSize: 26.0,
                                      letterSpacing: 0.0,
                                      fontWeight: FontWeight.bold,
                                    ),
                              ),
                            ),
                            Builder(
                              builder: (context) {
                                if (rowSensorsRecord?.ledPin ?? false) {
                                  return Align(
                                    alignment: AlignmentDirectional(0.0, 0.0),
                                    child: Icon(
                                      Icons.settings_suggest,
                                      color:
                                          FlutterFlowTheme.of(context).success,
                                      size: 75.0,
                                    ),
                                  );
                                } else {
                                  return Align(
                                    alignment: AlignmentDirectional(0.0, 0.0),
                                    child: Icon(
                                      Icons.settings_rounded,
                                      color: FlutterFlowTheme.of(context).error,
                                      size: 75.0,
                                    ),
                                  );
                                }
                              },
                            ),
                          ],
                        );
                      },
                    ),
                  ),
                ],
              ),
            ],
          ),
        ),
      );
  }
}
```

9.2_ NodeMCU ESP 32 WROOM 30 Pin code

```
#define FIREBASE_PROJECT_ID "smart-irrigation-v77nzz"
#define API_KEY "q7HrtzhZSlHH8vvuOhw9qrcQ10QJwD9FXA5yGevD"

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

```
#include <LiquidCrystal.h>  // Include the library for LCD display
#include <FastLED.h>
#include <Arduino.h>
#include "DHT.h"
FirebaseAuth firebaseAuth;
FirebaseConfig firebaseConfig;
FirebaseData firebaseData;


const char *ssid = "Honor 7X"; // Wifi network name.
const char *pass = "Mo244466666";

#define LED_PIN 21
#define BRIGHTNESS 255
#define LED_TYPE WS2811
#define COLOR_ORDER GRB

const uint8_t kMatrixWidth = 16;
const uint8_t kMatrixHeight = 16;
const bool kMatrixSerpentineLayout = true;

#define NUM_LEDS (kMatrixWidth * kMatrixHeight)
#define MAX_DIMENSION ((kMatrixWidth > kMatrixHeight) ? kMatrixWidth :
kMatrixHeight)

// The leds
CRGB leds[kMatrixWidth * kMatrixHeight];
bool ledStripOn = false;
// The 16 bit version of our coordinates
static uint16_t x;
static uint16_t y;
static uint16_t z;

uint16_t speed = 20;  // speed is set dynamically once we've started up
uint16_t scale = 30;  // scale is set dynamically once we've started up
uint8_t noise[MAX_DIMENSION][MAX_DIMENSION];

CRGBPalette16 currentPalette(PartyColors_p);
uint8_t colorLoop = 1;

#define DHTPIN 4
#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);
const int soilMoisturePin = 35;  // Analog pin for the soil moisture sensor
const int waterLevelPin = 34;    // Analog pin for the water level sensor
const int ledPin = 2;
bool Led_Pin = false;
// Constants for soil moisture thresholds
const int SOIL_MOISTURE_LOW = 50;   // Adjusted to match the 0-100 range
const int SOIL_MOISTURE_HIGH = 50;  // Adjusted to match the 0-100 range

const int pumpPin = 5;  // Change this to the actual pin connected to the DC pump
on ESP32
int pumpState = LOW;    // Initialize pumpState to LOW (off)
int pumpMode = 0;       // 0 for automatic, 1 for manual
bool pump_on_off=true;
int waterLevel = 0;  // Water level indicator value
unsigned long previousMillis = 0;
const unsigned long interval = 1000;  // Interval in milliseconds

float mappedSoilMoisture = 0;  // Declare mappedSoilMoisture as a global variable

// Initialize the LCD object with the appropriate pins
const int rs = 12, en = 14, d4 = 27, d5 = 26, d6 = 25, d7 = 33 ; // rw=GND Define
```

```
71 RW pin
72 LiquidCrystal lcd(rs, en, d4, d5, d6, d7); // Initialize LCD with RW pin
73
74 void setup() {
75   Serial.begin(115200);
76   Serial.print("Connecting With");
77   Serial.println(ssid);
78   WiFi.begin(ssid, pass);
79   while (WiFi.status() != WL_CONNECTED) {
80     delay(500);
81     Serial.print(".");
82   }
83   Serial.println("");
84   Serial.print("WiFi connected. IP:");
85   Serial.println(WiFi.localIP());
86
87   firebaseConfig.signer.test_mode = true;
88
89
90   Firebase.reconnectWiFi(true);
91
92   Firebase.begin(&firebaseConfig, &firebaseAuth);
93
94
95   digitalWrite(pumpPin, LOW);
96   pinMode(ledPin, OUTPUT);
97   pinMode(pumpPin, OUTPUT);
98   dht.begin();
99   delay(3000);
100  LEDS.addLeds<LED_TYPE, LED_PIN, COLOR_ORDER>(leds, 80);
101  LEDS.setBrightness(BRIGHTNESS);
102
103  // Initialize our coordinates to some random values
104  x = random16();
105  y = random16();
106  z = random16();
107
108  // Initialize the LCD
109  lcd.begin(16, 2);
110 }
111
112 void loop() {
113   updateLEDStrip();
114   unsigned long currentMillis = millis();
115
116   if (currentMillis - previousMillis >= interval) {
117     previousMillis = currentMillis;
118
119     // Read soil moisture or water level based on the flag
120     if (pumpMode == 0) {
121       int soilSensorValue = analogRead(soilMoisturePin);
122
123        Serial.print("soilSensor: ");
124
125       Serial.println(soilSensorValue);
126       mappedSoilMoisture = map(soilSensorValue, 0, 4095, 100, 0);
127
128       autoControlPump(mappedSoilMoisture);
129       Serial.print("mapped_SoilMoisture: ");
130
131       Serial.println(mappedSoilMoisture);                          //
132 Automatically control the pump based on soil moisture if in automatic mode
133     }
134
135     // Read water level
```

```
136     waterLevel = analogRead(waterLevelPin);
137     Serial.print("waterLevel: ");
138     Serial.println(waterLevel);
139     if (waterLevel < 450) {
140       waterLevel = 0;  // Set water level to 0 if below a threshold
141     } else {
142       waterLevel = map(waterLevel, 0, 4095, 0, 100);  // Map water level to 0-100
143       Serial.print("map_waterLevel: ");
144     Serial.println(waterLevel);
145     }
146 
147     // Control LED
148     controlLED(mappedSoilMoisture);
149 
150     // Read and send sensor data
151     readAndSendSensorData();
152 
153     // Display sensor readings on LCD
154 
155   }
156 }
157 
158 void displaySensorReadingsOnLCD(float temperature, float humidity) {
159   lcd.clear();
160 
161   // Display temperature
162   lcd.setCursor(0, 0);
163   lcd.print("Temp: ");
164   lcd.print(temperature);
165   lcd.print(" C");
166 
167   // Display humidity
168   lcd.setCursor(0, 1);
169   lcd.print("Humidity: ");
170   lcd.print(humidity);
171   lcd.print(" %");
172   delay(2500);
173 
174   // Display soil moisture
175   lcd.clear(); // Clear the screen to display new data
176   lcd.setCursor(0, 0);
177   lcd.print("Soil: ");
178   lcd.print(mappedSoilMoisture);
179   lcd.print(" %");
180 
181   // Display water level
182   lcd.setCursor(0, 1);
183   lcd.print("Wat.Lev: ");
184   lcd.print(waterLevel);
185   lcd.print(" %");
186   delay(2500);
187 }
188 
189 void autoControlPump(float mappedSoilMoisture) {
190   if (waterLevel < 30 || waterLevel < 20) {
191     // Water level is too low, do not turn on the pump
192     pumpState = LOW;
193     digitalWrite(ledPin, LOW);
194       pump_on_off=false;
195   } else if (mappedSoilMoisture <= SOIL_MOISTURE_LOW) {
196     // If soil moisture is below the threshold and water level is okay, turn the
197 pump ON
198     pumpState = HIGH;
199      pump_on_off=true;
200   } else if (mappedSoilMoisture >= SOIL_MOISTURE_HIGH) {
```

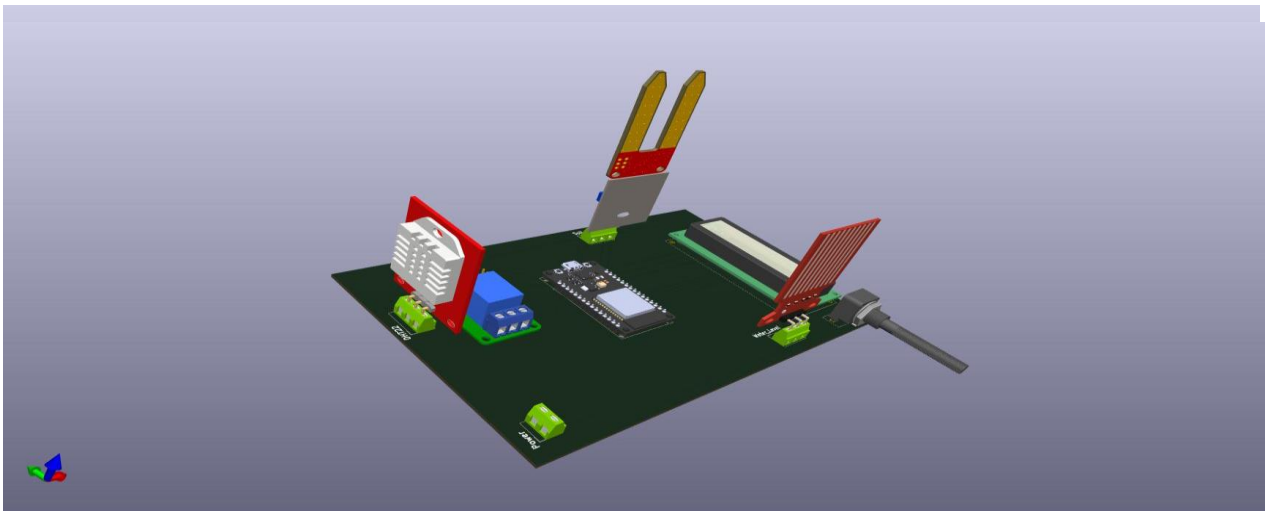
```
    // If soil moisture is above the threshold, turn the pump OFF
    pumpState = LOW;
     pump_on_off=false;

  }

  digitalWrite(pumpPin, pumpState);  // Update the pump state
}

void controlLED(float mappedSoilMoisture) {
  if (mappedSoilMoisture <= SOIL_MOISTURE_LOW) {
    digitalWrite(ledPin, HIGH);
    Serial.print("LED IS HIGH (TRUE)");
       Led_Pin = true;
  } else if (mappedSoilMoisture >= SOIL_MOISTURE_HIGH) {
    digitalWrite(ledPin, LOW);
    Serial.print("LED IS LOW (FALSE)");
       Led_Pin = false;
  }
}

void readAndSendSensorData() {
  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();

  if (!isnan(humidity) && !isnan(temperature)) {
    Serial.print("temperature: ");
    Serial.println(temperature);
    Serial.print("humidity: ");
    Serial.println(humidity);
  } else {
    Serial.println(F("Failed to read from DHT sensor!"));
  }

  updateFirebase(temperature, humidity);
      displaySensorReadingsOnLCD( temperature, humidity);
}

void updateFirebase(float temperature, float humidity) {
  // booleanValue, doubleValue, integerValue, stringValue
  FirebaseJson content;
  String documentPath = "sensors/esp32";  // This collection/document must exist
in firestore in order for `patchDocument` to work.
  content.clear();
  content.set("fields/temperature/doubleValue", temperature);
  content.set("fields/humidity/doubleValue", humidity);
  content.set("fields/soil_moisture/doubleValue", mappedSoilMoisture);
  content.set("fields/water_level/integerValue", waterLevel);
  content.set("fields/ledPin/booleanValue", Led_Pin);
  content.set("fields/pumpState/booleanValue", pump_on_off);
  content.set("fields/ledStripState/booleanValue", ledStripOn);

  if (Firebase.Firestore.patchDocument(&firebaseData, FIREBASE_PROJECT_ID,
"(default)", documentPath.c_str(), content.raw(), "", "", "true")) {
    Serial.printf("ok\n%s\n\n", firebaseData.payload().c_str());
  } else {
    Serial.println(firebaseData.errorReason());
  }
}

void updateLEDStrip() {
  if (ledStripOn) {
    ChangePaletteAndSettingsPeriodically();
    fillnoise8();                  // Call the noise generation function
    mapNoiseToLEDsUsingPalette();  // Map the noise data to LEDs
```

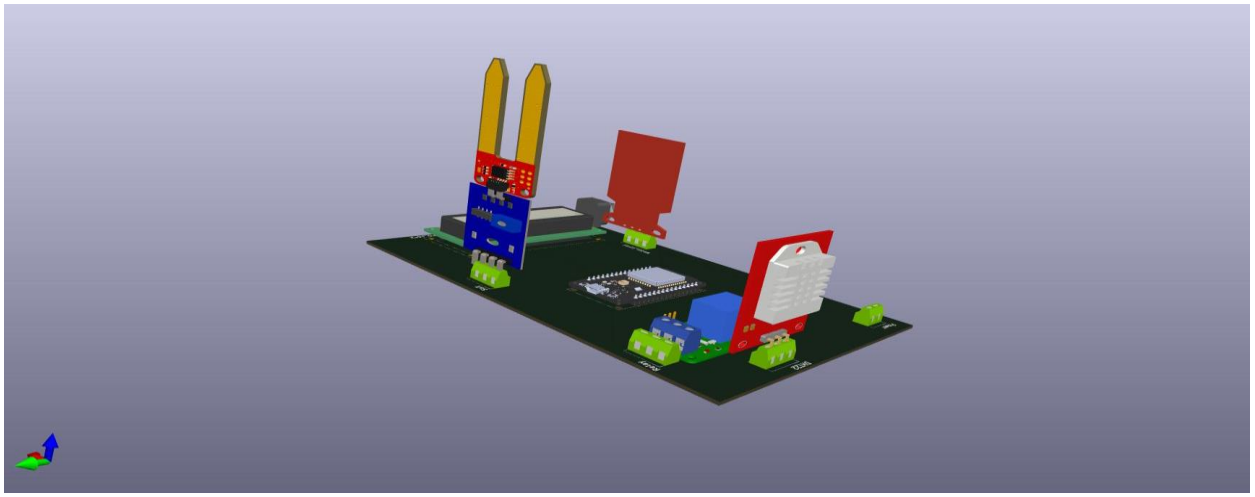
```
266     LEDS.show();                    // Show the LEDs
267     // You may need to adjust the animation speed based on your preference
268     delay(10);  // Delay between frames (adjust as needed)
269   } else {
270     // Turn off the LED strip
271     fill_solid(leds, NUM_LEDS, CRGB::Black);
272     LEDS.show();
273   }
274 }
275
276 void fillnoise8() {
277   uint8_t dataSmoothing = 0;
278   if (speed < 50) {
279     dataSmoothing = 200 - (speed * 4);
280   }
281
282   for (int i = 0; i < MAX_DIMENSION; i++) {
283     int ioffset = scale * i;
284     for (int j = 0; j < MAX_DIMENSION; j++) {
285       int joffset = scale * j;
286
287       uint8_t data = inoise8(x + ioffset, y + joffset, z);
288
289       data = qsub8(data, 16);
290       data = qadd8(data, scale8(data, 39));
291
292       if (dataSmoothing) {
293         uint8_t olddata = noise[i][j];
294         uint8_t newdata = scale8(olddata, dataSmoothing) + scale8(data, 256 -
295 dataSmoothing);
296         data = newdata;
297       }
298
299       noise[i][j] = data;
300     }
301   }
302
303   z += speed;
304
305   x += speed / 8;
306   y -= speed / 16;
307 }
308
309 void mapNoiseToLEDsUsingPalette() {
310   static uint8_t ihue = 0;
311
312   for (int i = 0; i < kMatrixWidth; i++) {
313     for (int j = 0; j < kMatrixHeight; j++) {
314       uint8_t index = noise[j][i];
315       uint8_t bri = noise[i][j];
316
317       if (colorLoop) {
318         index += ihue;
319       }
320
321       if (bri > 127) {
322         bri = 255;
323       } else {
324         bri = dim8_raw(bri * 2);
325       }
326
327       CRGB color = ColorFromPalette(currentPalette, index, bri);
328       leds[XY(i, j)] = color;
329     }
330   }
```

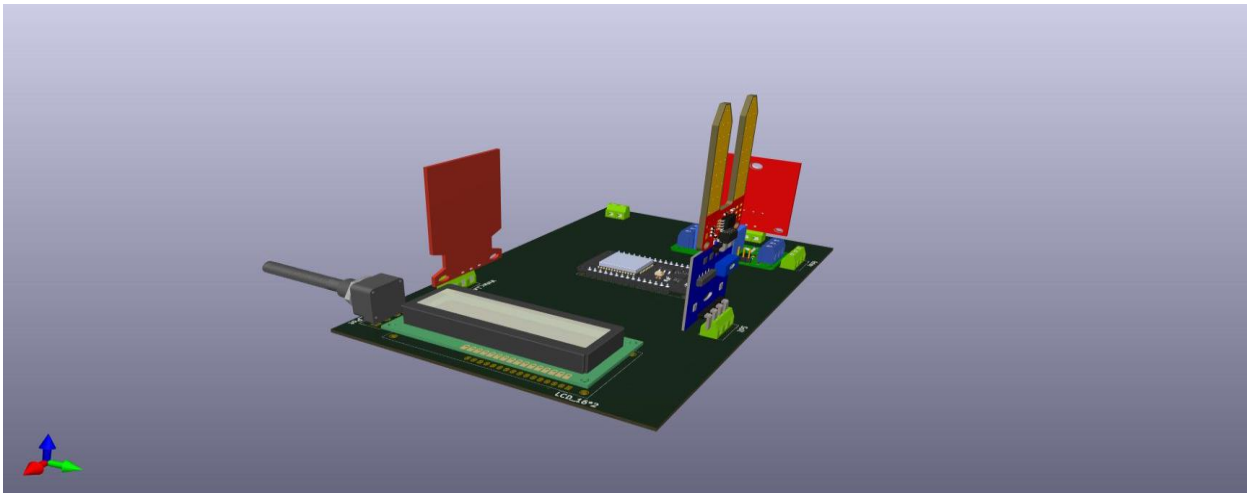
```
331
332   ihue += 1;
333 }
334
335 uint16_t XY(uint8_t x, uint8_t y) {
336   uint16_t i;
337   if (kMatrixSerpentineLayout == false) {
338     i = (y * kMatrixWidth) + x;
339   }
340   if (kMatrixSerpentineLayout == true) {
341     if (y & 0x01) {
342       uint8_t reverseX = (kMatrixWidth - 1) - x;
343       i = (y * kMatrixWidth) + reverseX;
344     } else {
345       i = (y * kMatrixWidth) + x;
346     }
347   }
348   return i;
349 }
350
351 #define HOLD_PALETTES_X_TIMES_AS_LONG 1
352
353 void ChangePaletteAndSettingsPeriodically() {
354   uint8_t secondHand = ((millis() / 1000) / HOLD_PALETTES_X_TIMES_AS_LONG) % 60;
355   static uint8_t lastSecond = 99;
356
357   if (lastSecond != secondHand) {
358     lastSecond = secondHand;
359     if (secondHand == 0) {
360       currentPalette = RainbowColors_p;
361       speed = 20;
362       scale = 30;
363       colorLoop = 1;
364     }
365     if (secondHand == 5) {
366       SetupPurpleAndGreenPalette();
367       speed = 10;
368       scale = 50;
369       colorLoop = 1;
370     }
371     if (secondHand == 10) {
372       SetupBlackAndWhiteStripedPalette();
373       speed = 20;
374       scale = 30;
375       colorLoop = 1;
376     }
377     if (secondHand == 15) {
378       currentPalette = ForestColors_p;
379       speed = 8;
380       scale = 120;
381       colorLoop = 0;
382     }
383     if (secondHand == 20) {
384       currentPalette = CloudColors_p;
385       speed = 4;
386       scale = 30;
387       colorLoop = 0;
388     }
389     if (secondHand == 25) {
390       currentPalette = LavaColors_p;
391       speed = 8;
392       scale = 50;
393       colorLoop = 0;
394     }
395     if (secondHand == 30) {
```

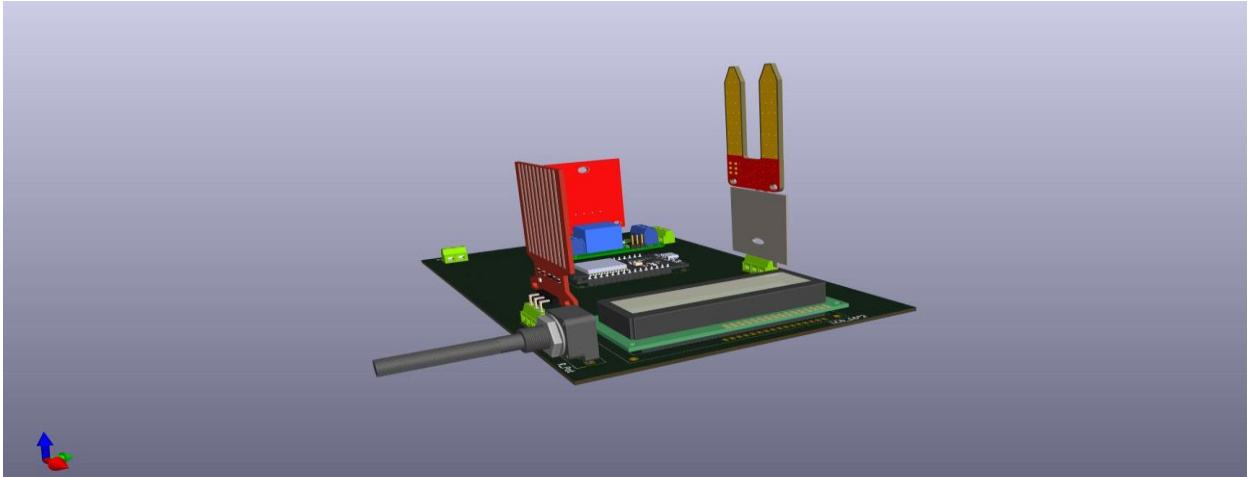
```
396       currentPalette = OceanColors_p;
397       speed = 20;
398       scale = 90;
399       colorLoop = 0;
400     }
401     if (secondHand == 35) {
402       currentPalette = PartyColors_p;
403       speed = 20;
404       scale = 30;
405       colorLoop = 1;
406     }
407     if (secondHand == 40) {
408       SetupRandomPalette();
409       speed = 20;
410       scale = 20;
411       colorLoop = 1;
412     }
413     if (secondHand == 45) {
414       SetupRandomPalette();
415       speed = 50;
416       scale = 50;
417       colorLoop = 1;
418     }
419     if (secondHand == 50) {
420       SetupRandomPalette();
421       speed = 90;
422       scale = 90;
423       colorLoop = 1;
424     }
425     if (secondHand == 55) {
426       currentPalette = RainbowStripeColors_p;
427       speed = 30;
428       scale = 20;
429       colorLoop = 1;
430     }
431   }
432 }
433
434 void SetupRandomPalette() {
435   currentPalette = CRGBPalette16(
436     CHSV(random8(), 255, 32),
437     CHSV(random8(), 255, 255),
438     CHSV(random8(), 128, 255),
439     CHSV(random8(), 255, 255));
440 }
441
442 void SetupBlackAndWhiteStripedPalette() {
443   fill_solid(currentPalette, 16, CRGB::Black);
444   currentPalette[0] = CRGB::White;
445   currentPalette[4] = CRGB::White;
446   currentPalette[8] = CRGB::White;
447   currentPalette[12] = CRGB::White;
448 }
449
450 void SetupPurpleAndGreenPalette() {
451   CRGB purple = CHSV(HUE_PURPLE, 255, 255);
452   CRGB green = CHSV(HUE_GREEN, 255, 255);
      CRGB black = CRGB::Black;

      currentPalette = CRGBPalette16(
        green, green, black, black,
        purple, purple, black, black,
        green, green, black, black,
        purple, purple, black, black);
    }
```

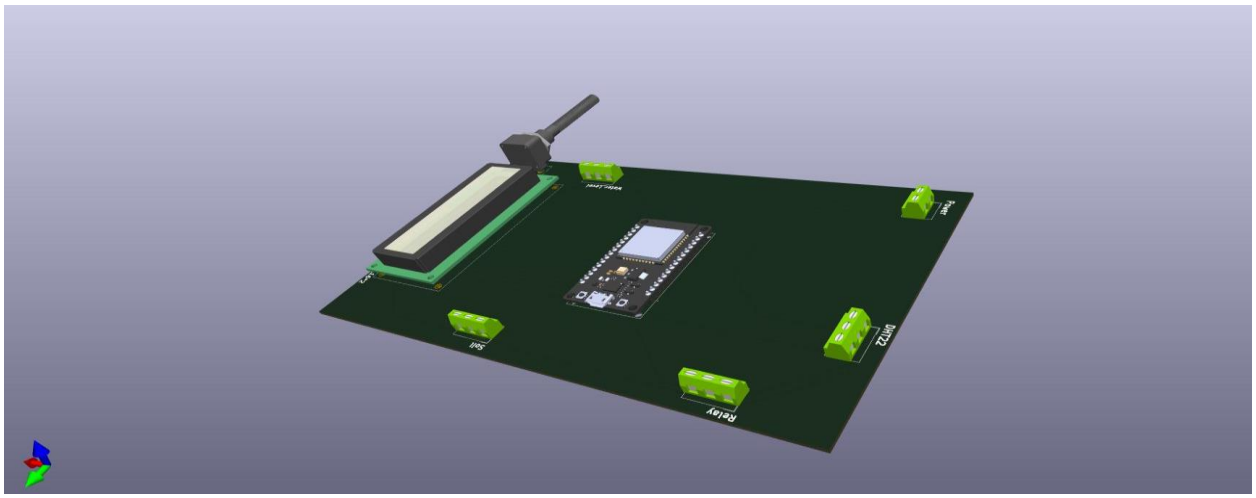
9.3_PCB circuit design using KiCAD

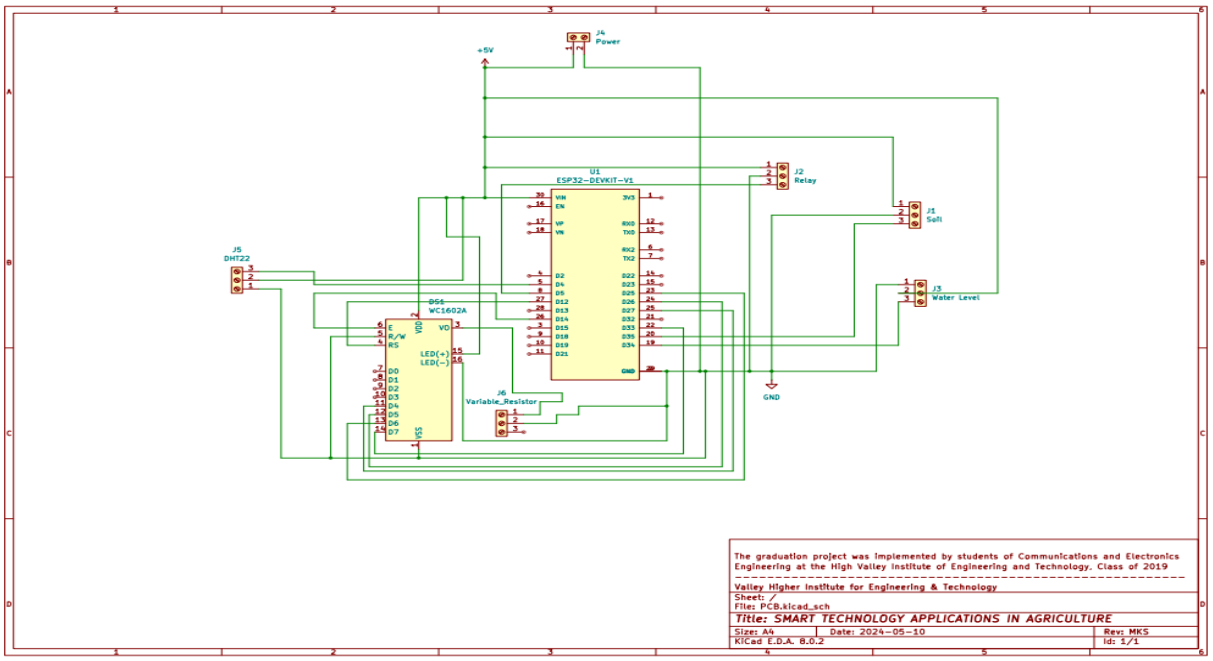


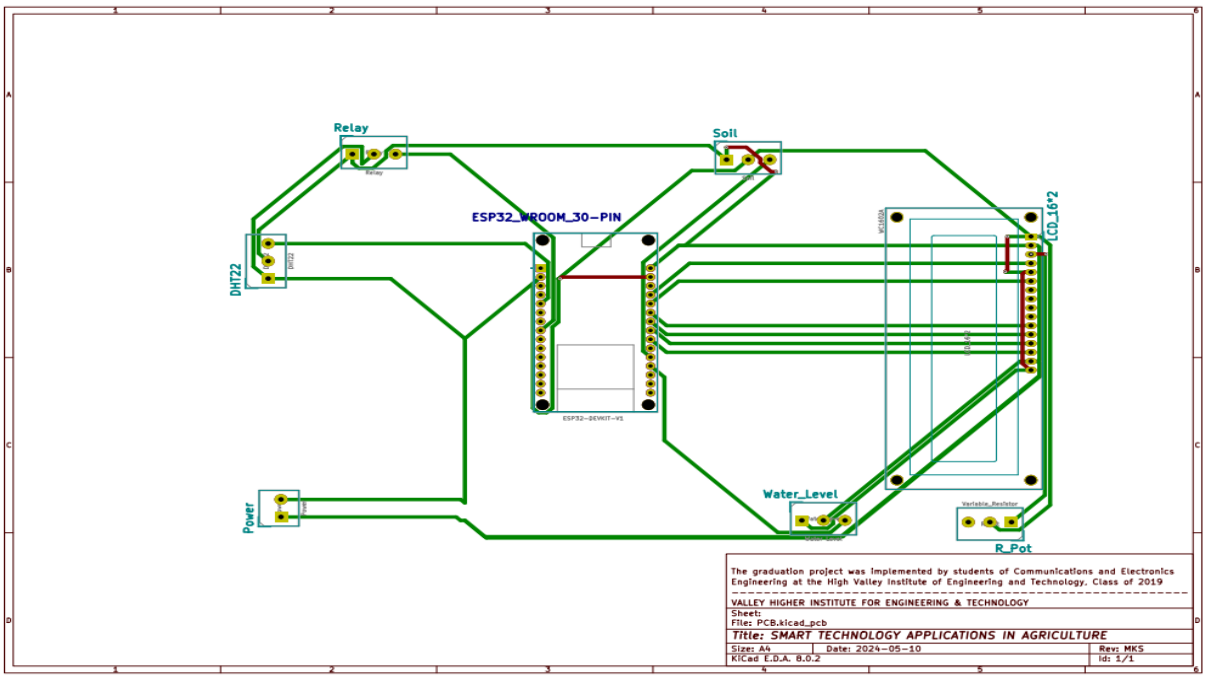


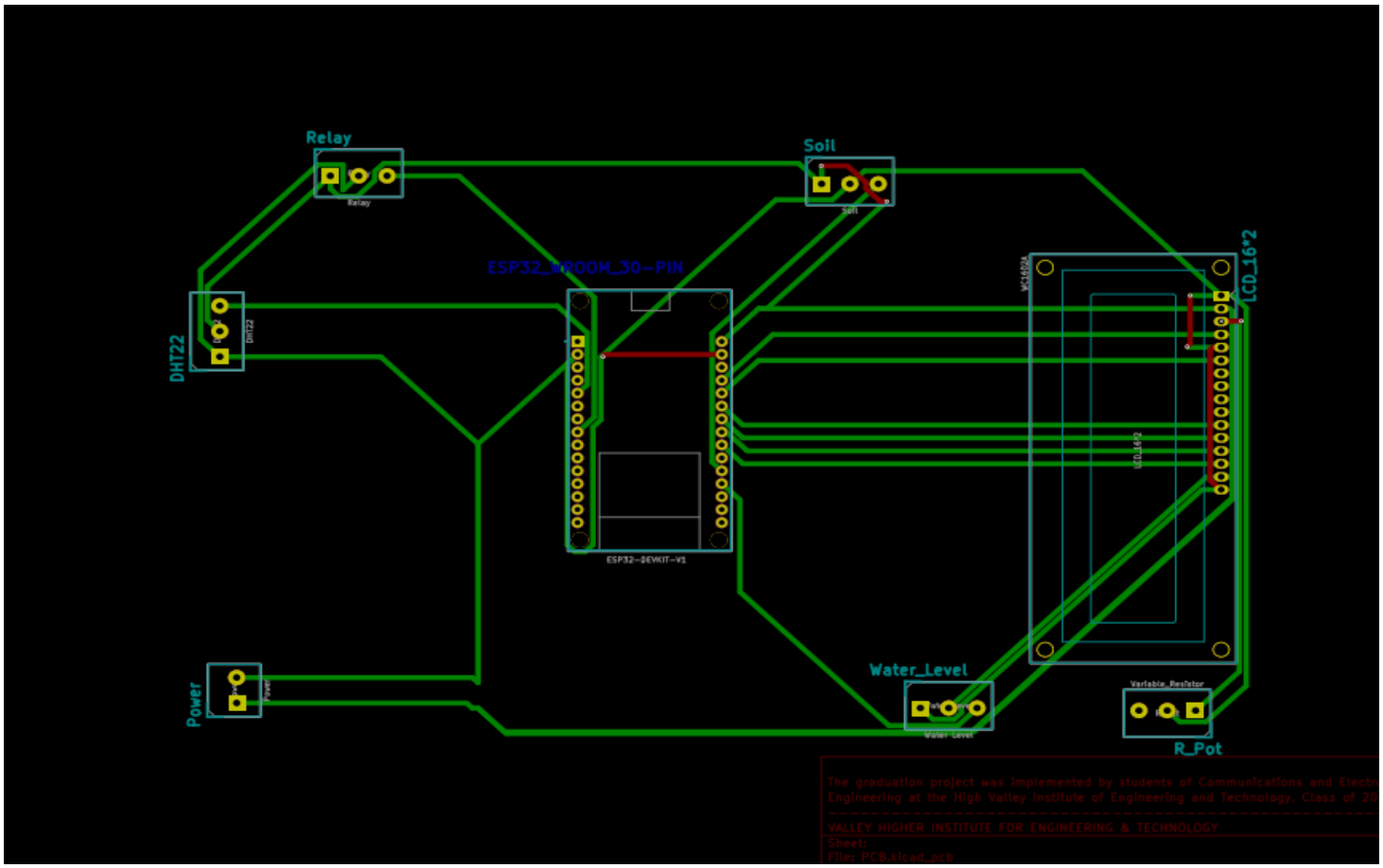


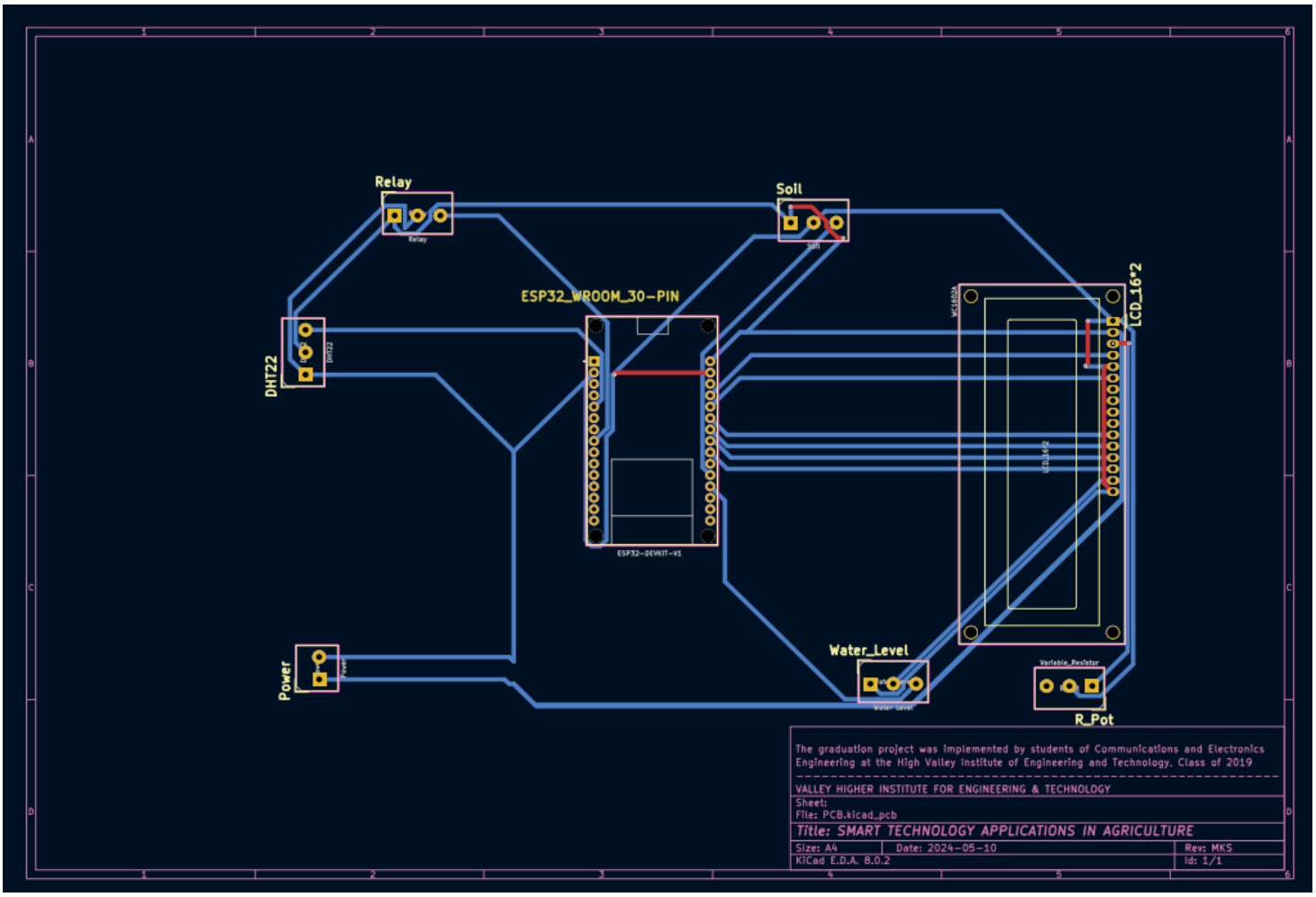



Relay
Soil
LCD


J4
Power
+5V
U1
ESP32-DEVKIT-V1
J2
Relay
J1
Soil
J3
Water Level
J5
DHT22
DS6
WC1602A
J6
Variable_Resistor
GND
The graduation project was implemented by students of Communications and Electronics Engineering at the High Valley Institute of Engineering and Technology, Class of 2019
Valley Higher Institute for Engineering & Technology
Sheet: /
File: PCB.kicad_sch
Title: SMART TECHNOLOGY APPLICATIONS IN AGRICULTURE
Size: A4
Date: 2024-05-10
Rev: MKS
KiCad E.D.A. 8.0.2
Id: 1/1


Relay
Soil
ESP32_WROOM_30-PIN
DHT22
LCD_16*2
Power
Water_Level
R_Pot
The graduation project was implemented by students of Communications and Electronics Engineering at the High Valley Institute of Engineering and Technology, Class of 2019
VALLEY HIGHER INSTITUTE FOR ENGINEERING & TECHNOLOGY
Sheet:
File: PCB.kicad_pcb
Title: SMART TECHNOLOGY APPLICATIONS IN AGRICULTURE
Size: A4
Date: 2024-05-10
Rev: MKS
KiCad E.D.A. 8.0.2
Id: 1/1


Relay
Soil
ESP32_WROOM_30-PIN
DHT22
LCD_16*2
Power
Water_Level
R_Pot
The graduation project was implemented by students of Communications and Electr
Engineering at the High Valley Institute of Engineering and Technology, Class of 20
VALLEY HIGHER INSTITUTE FOR ENGINEERING & TECHNOLOGY
Sheet:
File: PCB.kicad_pcb


Relay
Soil
ESP32_WROOM_30-PIN
DHT22
LCD_16*2
Power
Water_Level
R_Pot
The graduation project was implemented by students of Communications and Electronics
Engineering at the High Valley Institute of Engineering and Technology, Class of 2019
VALLEY HIGHER INSTITUTE FOR ENGINEERING & TECHNOLOGY
Sheet:
File: PCB.kicad_pcb
Title: SMART TECHNOLOGY APPLICATIONS IN AGRICULTURE
Size: A4
Date: 2024-05-10
Rev: MKS
KiCad E.D.A. 8.0.2
Id: 1/1

CHAPTER 10

Acknowledgments

Thanks and appreciation to everyone who has given us credit

THANKS, AND APPRECIATION

To the one who lit the first candle for me, to the fragrance of my childhood, to the warmth of my life, and the fragrance of my youth, to my refuge and refuge, to the one who endured every moment of pain in my life and turned it into moments of joy, to the one who protected me from the summer heat with spring flowers, to my beloved and the soul of my heart. To my father, to the one who supported me on my weak day, to my beloved who shared my worries and sadness, to the one who shed tears for me, to the one who gave me love when I was young until the veins of my body quenched from it, to the one to whom my soul traveled to embrace her sweet soul and shrink before her serenity, to my mother.

You have given and are still giving your best, and giving and generosity are still your address. Accept from me all expressions of thanks, love and gratitude. To you, gentlemen, I offer all expressions of appreciation and praise.

Under supervision of:

Assoc. Prof. Ashraf Mohamed Ali

Dr. Ibrahim Abdel-Dayem

CHAPTER 11

About the Authors

In honor of all those who made effort and time to achieve this project

The graduation project was implemented by students of Communications and Electronics Engineering at the High Valley Institute of Engineering and Technology, Class of 2019

Name	*ID*
Mostafa Khalid Sallam	***20190821***
Ahmed Mohamed Anwar	***20190841***
Bahaa El-din Mostafa	***20180871***
Ayman Shaban	***20190839***
Shihab El-Din Mohamed	***20190855***

About the Authors

Thus, for every beginning there is an end, and the best deed is that which has a good end, and the best speech is that which is little and meaningful. After this humble effort, I hope that I will be successful in narrating the previous elements in a narration without boredom or negligence, explaining the positive and negative effects of this interesting and enjoyable topic. May God grant me success and you in what is in the interest of all of us.