BIRZEIT UNIVERSITY
FACULTY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

SMART AGRICULTURE IRRIGATION SYSTEM

Prepared by:

– Oday Ziq	1201168
– Janna Hamad	1200853
– Sawsan Omair	1200822

Supervised by:
Dr. Alhareth Zyoud

Section: #9

Graduation Project submitted to the Department of Electrical and Computer engineering in partial fulfillment of the requirements for the degree of B.Sc. in Electrical Engineering

BIRZEIT
Feb 2025

المستخلص

يهدف المشروع المقترح إلى بناء نظام ري زراعي ذكي يعتمد على تقنيات إنترنت الأشياء (IoT) والذكاء الاصطناعي (AI)والتعلم الآلي (ML) لتحسين إدارة المياه وزيادة إنتاجية المحاصيل. ويهدف النظام المقترح إلى تخصيص جداول الري وأوقات التشغيل تلقائيًا لتلبية احتياجات المناطق الزراعية بناءً على مستوى المياه في الخزان ورطوبة التربة. وتعتمد هذه الأنظمة الذكية على جمع البيانات البيئية ومعالجتها بدقة لتحديد الاحتياجات المائية في الوقت الفعلي، مما يساهم في تحسين كفاءة استخدام المياه الخارجية.

تعمل وحدات التحكم الذكية في الري على مراقبة الرطوبة وظروف التربة والتبخر واستخدام المياه من قبل النباتات، وتقوم تلقائيًا بضبط جداول الري بناءً على ظروف الموقع الفعلية. على سبيل المثال، عندما ترتفع درجات الحرارة الخارجية أو ينخفض هطول الأمطار، تأخذ وحدات التحكم الذكية في الاعتبار المتغيرات الخاصة بالموقع، مثل نوع التربة ومعدل استخدام الرش، لضبط أوقات التشغيل أو جداول الري.

يتكون SAIS من عدة مكونات رئيسية، بما في ذلك أجهزة استشعار مثل مستشعر درجة الحرارة DS18B20 ، الذي يقيس درجات الحرارة من -55 درجة مئوية إلى 125 درجة مئوية بدقة تبلغ ± 0.5 درجة مئوية؛ ومستشعر درجة الحرارة والرطوبةDHT11 ، الذي يقيس درجات الحرارة من 0 درجة مئوية إلى 50 درجة مئوية والرطوبة من 20% إلى 90% بدقة ±1 درجة مئوية و±1%؛ ومستشعر رطوبة التربةYL-69 ، الذي يقيس محتوى رطوبة التربة باستخدام مستشعر التوصيل الكهربائي. تم دمج هذه المستشعرات مع متحكم Arduino UNO لجمع ومعالجة البيانات والتحكم في عمليات الري. تُستخدم تقنية LoRa أيضًا في الاتصالات اللاسلكية طويلة المدى ومنخفضة الطاقة لنقل البيانات من أجهزة الاستشعار إلى مركز معالجة البيانات المركزي.

سيتم استخدام الشبكات العصبية الاصطناعية (ANN) لتحليل احتياجات الري والتنبؤ بها. بحيث تقوم الشبكات العصبية الاصطناعية بتحليل البيانات التاريخية والحالية لضمان الإدارة المثلى للمياه.

Abstract

In this project, we are proposing a Smart Agricultural Irrigation System (SAIS), where it will be able to control its watering schedules according to the farm's requirements automatically. It depends on the farm water tank's status and soil moisture. Consequently, such actions contribute to better outdoor water usage. SAIS: For data acquisition and processing, the system will rely on the Internet of Things (IoT), Artificial Intelligence (AI), and Machine Learning (ML). This guarantees that the irrigation will be managed with high precision. Moisture, soil conditions, evaporation and the amount of water used by plants will all be verified by a smart irrigation controller. Then it will retune the watering schedule on its own to fit the actual site conditions. On the contrary to regular irrigation controllers that run on fixed schedules, intelligent controllers are not rigid and could alter their functioning basing on factors such as temperature differences and precipitation patterns. For instance, when there is an increase in temperature or reduced rainfall, a smart controller will change watering times or program to match the soil type and sprinkler rate. The sensors are very important elements in SAIS. The DS18B20 temperature sensor will detect temperatures between -55°C and 125°C with an accuracy of ±0.5°C. Similarly, the DHT11 temperature and humidity sensor measures temperatures within a range of 0 to 50oC with an error of ±1oC and humidity from 20% to 90% accuracy ±1%. The YL-69 soil moisture sensor will use electrical conductivity sensing to measure soil moisture content. These sensors will communicate with Arduino UNO microcontroller that will be responsible for acquiring the data as well as processing it to ensure irrigation control happens. Moreover, long-range low-power wireless communication via Long Range (LoRa) technology will provide sensors' connectivity to the central data processing center. Artificial Neural Networks (ANN) uses previous and current information for the most effective water management systems.

Abbreviations

AAA	Artificial Algae Algorithm
ADI	Alternating Drip Irrigation
AI	Artificial Intelligence
ANN	Artificial Neural Network
LS-SVM	Least Squares Support Vector Machine
CoAP	Constrained Application Protocol
CNN	Convolutional Neural Networks
CSS	Chirp Spread Spectrum
DI	Drip Irrigation
DSS	Decision Support Systems
ET	Evapotranspiration
EH	Energy-Harvesting
GN	Gateway Node
GIS	Geographic Information System
GM	Genetically Modified
GSM	Global System for Mobile Communication
IoT	Internet of Things
LoRa	Long Range
MC	Moisture Content
MAE	Mean Absolute Error
MSE	Mean Square Error
MQTT	Message Queuing Telemetry Transport
NN	Neural Network
PIC	Peripheral Interface Controller
RFID	Radio Frequency Identification
RF	Radio Frequency
RS	Remote Sensing
SAIS	Smart Agricultural Irrigation System
SCE	Shuffled Complex Evolution

SDI	Subsurface Drip Irrigation
SSIM	Structural Similarity Index Methods
WSN	Wireless Sensor Networks

Acknowledgments

We would like to express our gratitude and appreciation to all the supporters of our Smart Agricultural Irrigation System (SAIS) project, and to everyone who contributed to the successful completion of this work. Our greatest gratitude goes to our supervisor Dr. Al-Harith Zyoud for his cooperation with the group, and his comments at every step of the project. We also extend our thanks to Dr. Ibrahim Nimer for his valuable guidance and Eng. Abdullah Dahlah for providing and reviewing the data, which greatly enhanced the accuracy and efficiency of our system.

We also express our appreciation to our families and friends for their continuous support and patience throughout our project, from the beginning until now. Their encouragement and understanding have given us the strength and motivation to continue from the first step of this project to this level.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

As the global population grows, so does the demand for food, making it essential to modernize agriculture to boost productivity, quality, and reduce human labor. This shift to "Agriculture 4.0" incorporates technologies like the Internet of Things (IoT), Artificial Intelligence (AI), robotics, and nanotechnology to advance intelligent farming [1]. The IoT is defined on the one hand as an addition to the existing Internet of all things directly or indirectly connected to electronic gadgets that connect to the Internet. On the other hand, it is defined as a new approach that advances modern wireless communications so that components that support this technology can communicate to achieve common goals, such as actuators, mobile phones, sensors, and Radio Frequency Identification (RFID) tags.

Communication technologies such as wireless fidelity (Wi-Fi), Global System for Mobile communication (GSM) and Long Range (LoRa) play an important role in the successful and efficient operation of IoT devices in agriculture, enabling wireless connectivity for data transmission and remote monitoring. Widely, Wi-Fi is used because it is easy to access and supported by low-cost devices, and all that makes it suitable for small farms, while GSM provides long-range communications while the Wi-Fi is absent. ZigBee provides low power consumption and supports multi-node networking, and LoRa is ideal for remote areas without service coverage. These technologies contribute to real-time data collection, analysis, and control of irrigation systems [2]. Despite the many benefits and improvements that IoT brings, it faces problems such as high initial investment costs, lack of technical expertise among farmers, and concerns about privacy and data security.

Palestine depends mainly on water extracted from groundwater and surface water sources. The main reason for the misuse of surface water is due to the Israeli occupation's control over the waters of the Jordan River and the Dead Sea. The average daily Palestinian per person consumption of water was 86.3 liters. This rate reached 89.0 liters per day in the West Bank, compared to 82.7 liters in Gaza Strip. The quantities of polluted water classified as unfit for human use were much greater compared to the quantities of usable water, so that the percentage of usable water in Gaza decreased to 21.3 liters [3]. To solve this water deficit, our project was proposed to help give each plant only the amount of water it needs, reducing the amount of water wasted. Palestine also suffers from another problem in rural areas where there is no reliable internet connection. To solve these problems, the government can take the initial steps by providing grants to remove the financial burden from farmers, improve Internet infrastructure in rural areas, provide training programs on IoT systems, and improve farmers' technical skills. Information security and privacy concerns can be reduced through data encryption protocols and strong security measures. Integrating blockchain and AI into IoT technology for agriculture can enhance security and improve system management, leading to a better understanding of crops, improved descriptions of Genetically Modified (GM) crops, and enhanced water use efficiency [4] [5].

Research indicates that AI applications in agriculture can leverage various techniques, including (ANN), and other advanced models that address complex problems related to water and nutrient distribution within soil and the estimation of soil moisture. This approach is useful in developing Decision Support Systems (DSS) for automated irrigation, utilizing Machine Learning (ML), ANN, and other algorithms to estimate crop water requirements and improve irrigation management practices [6] [7].

1.2 Motivation

Improper irrigation practices in agriculture can lead to significant resource inefficiencies, either through excessive or insufficient water application. This imbalance not only contributes to the depletion of water resources but also undermines crop yield and quality. Furthermore, the increasing variability associated with climate change introduces additional challenges to sustainable water management, thereby posing a threat to both food security and the long-term viability of agricultural systems. Addressing these issues requires advanced irrigation strategies that adapt to changing climatic conditions and optimize water use efficiency [8][9].

Water resources management: Poor irrigation practices, such as traditional methods, lead to wasting of water, as it becomes unavailable at the time when plants need it most. Incorrect and inefficient use of water during heavy rains or dry seasons also leads to depletion of water resources.

Limited real-time monitoring: Conventional methods delay crop monitoring at the required time and therefore do not allow adjustments to be made at that time depending on changing environmental and soil conditions.

High initial investment costs: Farmers, especially those who farm on a small scale, have great difficulty paying the investment costs needed by developed irrigation systems and technologies.

So, it was mandatory to develop a system that introduced a smart irrigation system that uses IoT, ML and unified technologies. This system will provide real-time environmental monitoring, effective water management, and give safe and cost-effective solutions, leading to sustainability, enhanced productivity, and economic health in agriculture [10].

Our initiative correlates with the second and the sixth objectives of the Sustainable Development Goals by the UN because it aims at increasing agricultural productivity under minimum water consumption. The second goal, concerning health, is ‘Zero Hunger,’ which is to enhance people’s nutritional status, ensure availability of food, eliminate hunger, and promote agriculture that is sustainable. Our findings also assist in providing crops with the right amount of water at the right time that in return enhances food production making food secure through efficient use of water through smart irrigation.

In addition, ‘Clean Water and Sanitation for All’ is another goal number 6 that aims to make water and sanitation affordable to everyone and to be sustainable. This objective is in harmonious with the focus of our project that entails the efficient use of water and monitoring the surrounding environment. We ensure that water is properly used in agriculture and that we do not use too much water or too little of the water both of which are detrimental to water resources.

Consequently, the mentioned important areas contribute not only to the implementation of the project objectives and launch of Clean Water and Hunger Relief but also to further large-scale projects of sustainable development and proper agricultural resource management [11].

1.3 Problem Statement

Currently, we have adopted agricultural operations that are carried out automatically or nearly completely, aiming to increase crop productivity while saving time and effort.

Some problems must be taken into account, such as excessive or low humidity, because they cause problems for plants. In addition, at the same time, excessive irrigation should not be done, because this harms the plants and reduces the yield. It is also possible that long distances between farmers' residence and farmland cause lack of follow-up. Therefore, attention leads to wasting money, effort and water.

To solve this problem, SAIS is suggested, as this device contains sensors, a microcontroller, and LoRa as the use of sensors and monitoring devices to monitor data in real time helps farmers make informed and accurate decisions about environmental factors such as humidity and temperature and also helps farmers make accurate decisions about the amount of water they use. Crops need these things to reduce operating costs and increase production.

To solve the problem of communication in remote places, we relied on integrating LoRa, and other communication technologies that provide complete monitoring, control, and operation of agricultural systems in areas that do not have sufficient communication.

However, using IoT technology in agriculture has certain drawbacks, particularly due to farmers' limited technical knowledge. Despite these challenges, smart irrigation offers substantial benefits. To address its limitations, governments are supporting infrastructure improvements, and the integration of blockchain and artificial intelligence can enhance data security, build trust, and provide advanced analytics for predictive crop management.

Stakeholders play a pivotal role in the development of smart agricultural irrigation systems, as they recognize the pressing need for data-driven and efficient agricultural practices. These systems not only enhance productivity and sustainability for farmers but also contribute to broader objectives, such as ensuring global food security.

1.4 Report Objectives

1. Improving the performance of the smart irrigation system using IoT, AI and LoRa technology.
2. Promoting sustainability, efficiency and economic viability in agriculture.
3. Testing and validating the performance of our system to ensure its performance in different conditions.

1.5 Report Methodology

The flowchart shows how SAIS operates by executing environmental monitoring followed by data processing on Arduino for assessment prior to irrigation decision making and consequent water management according to live conditions.

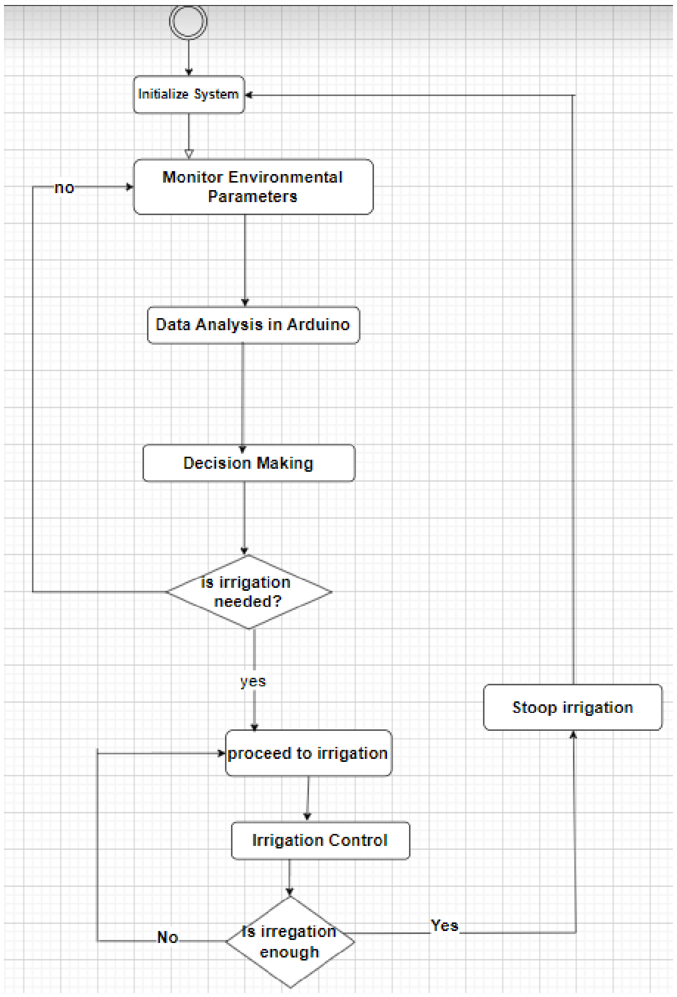


Figure 1.1: Methodology flow chart.

1.6 Report Scope

- Environmental Monitoring: data related to ground moisture content, temperature, and other important environmental factors are collected via IoT sensors.

- Water Management: the application of AI technology facilitates optimized irrigation strategies by determining precise water requirements, adjusting irrigation frequency, and scheduling based on real-time data analysis. This approach enhances resource efficiency by aligning watering intervals and volumes with environmental conditions and crop needs.

- Technical Integration: advanced technologies are used and integrated such as machine learning models, ANN to predict irrigation requirements and optimize water use.

- System Validation: the system must undergo various tests to ease the determination of the ability of the system to perform optimally under all such conditions.

- Economic and Sustainability Objectives: working collectively for improving water use efficiency, reducing, and wasting, increasing crop productivity along with issues of economic profitability for farmers.

1.7 Organization of the Report

The report is organised in four chapters and the rest of this report is organised as follows:

- Chapter 1: this chapter discusses the motivation for the project and additional background information related to the research.
- Chapter 2: this chapter presents literature and previous works related to smart irrigation systems, and the technologies associated with it.
- Chapter 3: this chapter aims to explain the research methods and procedures employed in this study.
- Chapter 4: this chapter presents the results obtained from the implementation of SAIS and provides a discussion on the findings.
- Chapter 5: provides the conclusion, future work and limitations, summarizing findings and outlining next steps for system development and validation.

Chapter 2

Related Works

2.1 Overview

The development of smart irrigation systems which combine ML, IoT, AI, and UAVs is evidence of a key evolution in agricultural technology. These technologies enhance the complex process and intelligent methods of agriculture development. The current research presents a literature review that discusses irrigation control as well as smart agriculture which identifies the core significance of real-time data, intelligent sensing and energy management in boosting irrigation performance and handling soil. Next section summarizes the recent studies related to smart irrigation systems.

2.2 Related Works

The article [1] consolidates the findings and approaches from research works that center on irrigation control and intelligent farming emphasizing the crucial importance of real time data, intelligent sensing, and energy generation, in improving irrigation effectiveness and soil supervision as explained in Figure 2.1.

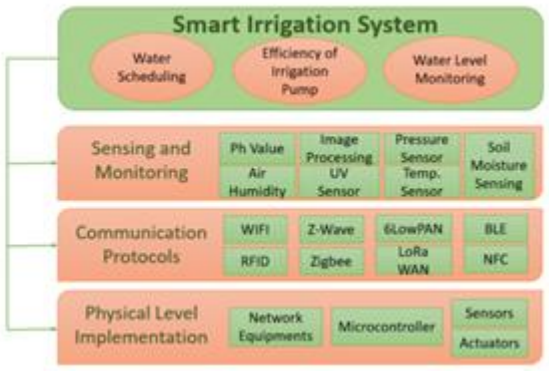


Figure 2.1: Smart irrigation systems building layers.

The integration of intelligent sensing technologies in farming allows for the monitoring of soil conditions empowering farmers to make informed decisions about when to irrigate. This method differs from approaches that rely on factors like soil moisture levels, Evapotranspiration (ET) thresholds and crop growth stages. One significant progress in this area is the based Neural Network (NN) irrigation management strategy, which has shown effectiveness to the Root Zone Water Quality Model 2 with Water Stress (RZWQM2 WS) method. It outperforms ET based techniques. Enhances water efficiency by up to 20%.

Artificial intelligence plays a role in comprehending the complexities of soil moisture within the soil and crop environment. A research project employed a network model trained on data from RZWQM2 to predict changes in soil moisture with accuracy. In the realm of estimating evapotranspiration (ETo) cutting edge learning techniques, like one dimensional Convolutional Neural Networks (1D CNN) and Long Short-Term Memory (LSTM) Convolutional LSTM (ConvLSTM) and were examined, highlighting the potential of AI in enhancing precision agriculture.

Geographic Information System (GIS) and Employing Remote Sensing (RS) methods has proven effective, in addressing challenges such as percolation and uneven distribution of irrigation water both of which have an impact, on the efficiency of irrigation practices. The importance of integrating modeling and optimization techniques into irrigation management is underscored by the development and validation of the Optimization of Irrigational Model (EDOSIM), Evaluation and Design well as the successful application of the Shuffled Complex Evolution (SCE) approach to enhance field performance.

A study [12] comparing surface drip irrigation (DI), alternating drip irrigation (ADI) and subsurface drip irrigation (SDI) on tomato production and soil microbial activity showed that SDI was more effective, in improving water distribution, root growth and yield. These results emphasize the benefits of drip irrigation techniques in conserving water and enhancing crop yields.

The combination of IoT, AI cloud computing and edge computing is revolutionizing farming methods by allowing monitoring of crops and soil analyzing data and making decisions. Wireless Sensor Networks (WSN), as a component of this blend of technologies present an answer for optimizing water usage and tackling the water scarcity issue, with smart irrigation management.

Diving, into methods of measuring soil moisture highlights the significance of monitoring at specific depths for agricultural, hydrological, and meteorological purposes. Although volumetric and tensiometric approaches offer information, newer methods such, as techniques and microwave sensing provide more sophisticated detailed measurements though they come with challenges and constraints related to complexity and scale.

The authors in [13] proposed a smart agricultural system called Swamp, which is applying a technology related to the IoT that has many benefits in irrigation systems shown in Figure 2.2. They piloted this project in these countries: Brazil, Spain, and Italy, and during that period they focused on improving water distribution and consumption. As for the main stages that the SWAMP system went through, the first stage was monitoring the water reserve by assessing the water level in the tank and other water sources. Next, they plan efficient use of this water source. The next stage is water distribution, where water is effectively distributed to agricultural areas, and the IoT here ensures accurate distribution and precise irrigation and reduces water waste. As for the final stage, which is water consumption, here water use and plant needs are monitored, as the system adjusts irrigation times to ensure optimal growth and better productivity, as well as preserving water resources at the same time. All these steps serve as the backbone of SWAMP's smart irrigation system. As for how Swamp uses IoT and Future Internet Ware (FIWARE (technology, it is done through a platform. This platform manages water using IoT and leverages FIWARE components to enable monitoring and control of water distribution. However, a problem has been discovered for this system that the system may crash due to its heavy Central Processing Unit. CPU consumption. There are non-precision components in FIWARE that need to be reconfigured to enhance scalability and performance.

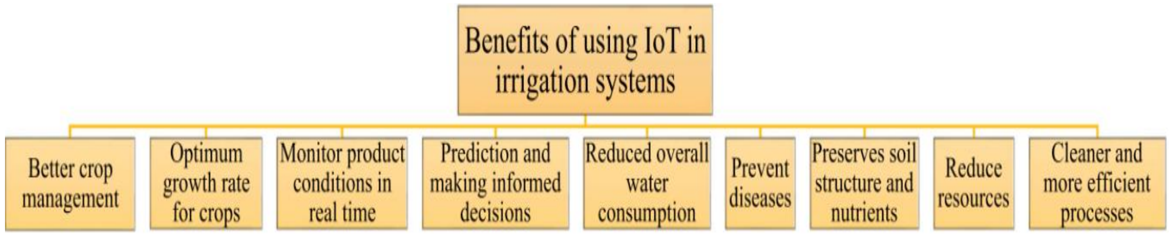

Benefits of using IoT in irrigation systems
Better crop management
Optimum growth rate for crops
Monitor product conditions in real time
Prediction and making informed decisions
Reduced overall water consumption
Prevent diseases
Preserves soil structure and nutrients
Reduce resources
Cleaner and more efficient processes

Figure 2.2: Benefits of IoT.

The weather conditions in the countries we mentioned above also affected the experiment, as well as the difference in the nature of the soil. All this led to a difference in results.

In Brazil, for example, the experiment faced a problem due to the instability of communications and the distance between the agricultural office and the central irrigation system. This led to a defect in the operating mechanism of the smart irrigation system, and the soil also differed, which led to modifications in the composition of systems that rely on the IoT for precision irrigation.

To improve this program, some authors suggested re-engineering some FIWARE components to improve performance and provide better scalability by using fewer computational resources.

The study in [2] discusses many forecasting models and smart technologies in agriculture, in particular smart irrigation systems based on the IoT. It also addresses the importance of technology in water management. The smart irrigation system uses sensors, data analysis, automation, and some important platforms such as: Node-RED and FavorIoT in order to improve water use and increase producing crops, enhancing their health, and enabling remote monitoring and control of sustainable agricultural practices.

Smart agricultural forecasting models contribute to precision agriculture by enabling the precise design of irrigation schedules based on plant water requirements in real time. These models resort to the use of sensors, data analytics and automation to conserve water, reduce its waste, and reduce impacts environmental.

As for the components of the smart irrigation system supported by the IoT, they are sensors to sense humidity, temperature, and soil moisture, and a central control unit that uses Raspberry Pi, and Node-RED to manage data flow and integrate with the preferred platform for data flow and cloud storage. Node-RED facilitates the process of making decision and control in data management, while the Favor IoT platform is based on effective communication and data management and secure storage of data in real time. This leads to improved water management and irrigation schedules. Figure 2.3 shows system connection.

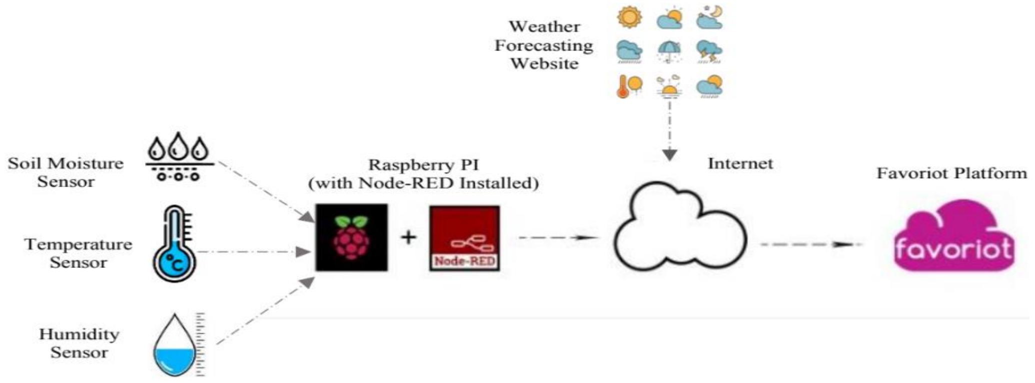


Figure 2.3: System connection.

The smart irrigation methodology includes certain steps such as defining system requirements, integrating sensors, equipping, and preparing Node-RED, visually designing the system flow, obtaining data from the real world, processing this data, implementing control mechanisms, testing and monitoring the system performance. This approach is efficiently used by the visual interface of Node-RED. To read sensor data in real time and automate irrigation operations to optimize water use and increase yields.

Researchers faced a problem during the development of the smart system, which is the need for effective water management. This includes improving water use, enhancing crop health, and increasing their productivity, but in order to ensure sustainability, researchers resort to solving this problem and improving through the use of IoT technology, data analysis, and automation in order to make decisions about irrigation based on data in Real time from sensors and other external sources, and by integrating Node-RED and the FavorIoT platform to process and store data, the system aims to overcome challenges related to water management and promote sustainable agricultural practices

The idea of this paper [14] is to stimulate the economy in Algeria in the field of agriculture. Algeria enjoys suitable conditions for agriculture in terms of water resources, good sunlight, and large areas, but due to poor irrigation water management and its shortage, agricultural production is affected, so it was necessary to use the IoT to improve water use and continuous monitoring, controlling the system remotely.

The system focuses on several goals to ensure that it reaches the largest possible number of farmers:

- Smooth and easy to deploy and use, it simplifies planning for irrigation tasks.
- Easy to operate and maintain because it is a modular and adaptable system.
- Design the system to be robust, ensuring reliability in operation.

For the system design, the team primarily required a 6LoWPAN smart gateway—which bridges ZigBee networks to the Internet via the 4G LTE mobile network—and a wireless sensor network system. This smart gateway links both the wireless sensor networks and controllers to the Internet and smartphones, effectively integrating the ZigBee network with online connectivity through the 4G LTE mobile network.

To continuously measure soil moisture values and levels of water wells and water tanks, sensors were used in the agricultural field. The collected data were sent to the smart portal via the ZigBee network, and then this information was sent to the web through 4G LTE network communications. The results were then analyzed, and according to it, the application selectively activates the controllers as needed. So, the user can access the results through their smartphone using Constrained Application Protocol (CoAP) or Hypertext Transfer Protocol Secure HTTPs interfaces.

Based on these values, the farmer decides whether to irrigate the dry areas or fill the tank. Then, communication will be made with the smart portal, which acts as an intermediary for the communication process via the web application. The smart gateway then sends this information to the coordination node to control the turning on or off of the electric pump or the opening and closing of valves. In cases where the soil moisture values or the water level in the tank are critical, and this indicates either that the areas are dry or that the water level in the tank is minimal, the system switches to autonomous mode so that it operates without the need for manual intervention from the farmer.

Based on the initial results obtained using the Cooja simulator, the project proved to be acceptable and emphasized the importance of adopting the IoT and WSN technologies in agriculture. This encourages efforts to create a special web for smartphones to complete the proposed system.

This article [15] discusses the Smart and Green framework, which is created to improve irrigation management in agriculture. This framework utilizes IoT technologies to enhance water efficiency by incorporating services, for preprocessing, data monitoring, synchronization, storage, fusion, and soil moisture forecasting. By combining the Van Genutchen model with predicted matric potential, the goal of the framework is to optimize irrigation scheduling and reduce water consumption significantly by 56.4% to 90% through the utilization of removal methods such, as Zscore, MZscore and Chauvenet.

Smart and Greens effectiveness lies in its capacity to gather and analyze information from origins such, as weather stations, soil sensors (such as tensiometers) and IoT devices. This enables an examination of soil moisture levels and weather conditions. This process streamlines data preprocessing, which's essential, for irrigation control. Users have the flexibility to personalize the system by specifying data origins and parameters for identifying and eliminating anomalies. As a result, the credibility of soil and weather data utilized in irrigation control is bolstered.

The process involves steps; collecting data, preparing it and managing irrigation incorporating machine learning methods to forecast soil moisture. Approaches, like Gradient Boosting, Random Forest, Decision Tree and Linear Regression and are assessed within the system showing reductions in water usage, for irrigation through analysis and handling of data.

Smart and Greens design features a four-tiered framework consisting of Application, Services, Communication and Physical layers. This setup enables blending of data gathering, analysis and organization for smart irrigation. The technology stack involves the use of Python 3, for building the framework MySQL, for storing data and diverse IoT communication protocols.

Tests show a new plan saves water by finding better ways to water plants. Removing unusual data helps make information more accurate and saves even more water. Using computers to guess how much water is in the ground shows one way that works well to decide when to water.

The primary goal of this study [6] is to create an accurate irrigation monitoring and control system to achieve the following goals:

- Create a cheap, highly accurate, IoT-enabled distributed wireless sensor network environment for environmental and soil monitoring, which is used to predict real-time water needs.
- Design an improved optimization technique for rapid prediction of Moisture Content (MC) levels.
- Creating a system based on interpolation technology to map the distribution of soil moisture content. By using an IoT-enabled wireless sensor network (WSN) environment, soil moisture content is measured at nine different locations in the farmland to create a distribution map as shown in Figure 2.4.

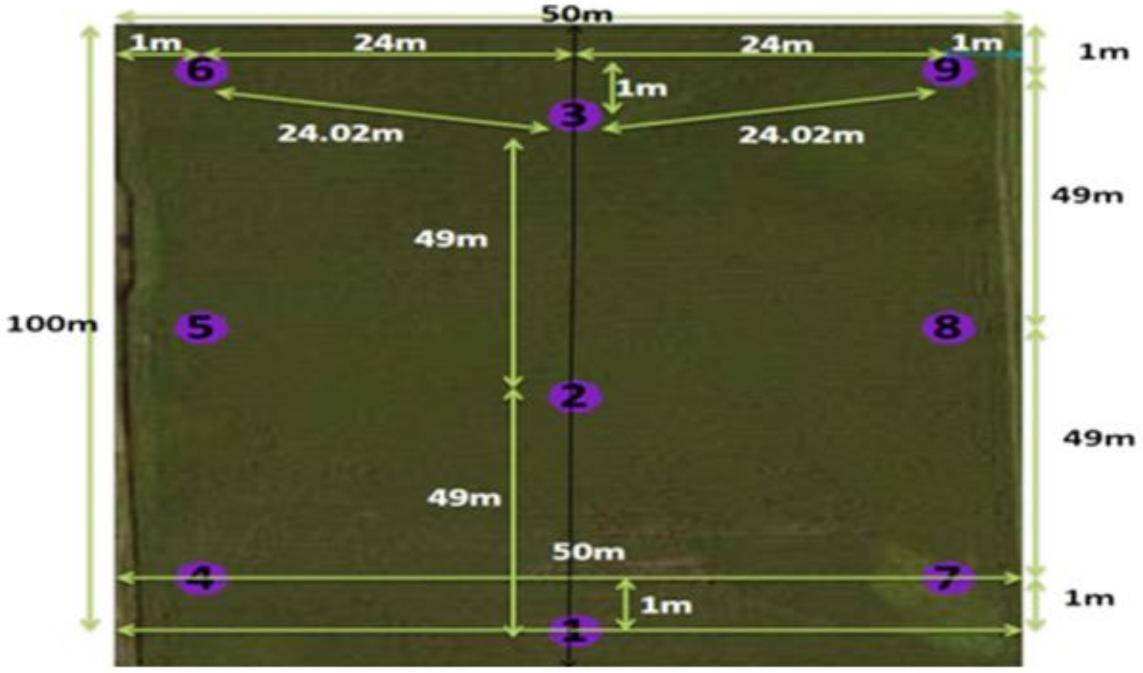


Figure 2.4: Example of sensor deployment as in [6].

- Updated data is linked to produce guidance on irrigation control and regulation of irrigation valves using Structural Similarity Index Methods (SSIM).
- Based on fuzzy logic-based weather model, an irrigation management system is implemented according to weather conditions.

This work [6] relies on an autonomous IoT-enabled WSN framework, to collect farm information in real-time via multi-point measurement. This framework consists of a soil moisture probe, a daylight intensity device (light-dependent resistor), and sensors for soil temperature, environmental humidity, environmental temperature, and carbon dioxide.

To determine the areas of farms that need water, Structural Similarity Index Method SSIM-based water valve management is used in the proposed irrigation control scheme. In addition, optimization methods are examined, including pattern classification based on feedforward neural networks and gradient descent with variable learning rates. Hence, the best method for estimating hourly soil MC is applied and combined with Interpolation techniques to provide a map of the distribution of soil moisture content. Then, the MC deficiency in the soil is calculated based on the SSIM index to set dedicated valves, which will ensure consistent water distribution throughout the farm. Valve control commands are also processed by a weather modelling system based on fuzzy logic, to consider different weather conditions.

The network consists of nine wireless sensor nodes and one Gateway Node (GN), also known as a coordinator node. The coordination node, powered by a Raspberry Pi3 and equipped with a Zigbee transceiver (S2 series) and Wi-Fi circuits, collects sensor data, and transmits it to the cloud platform to predict soil water requirements. In addition, the gateway node connects to Internet applications through Wi-Fi to analyse soil water requirements.

To estimate soil MC, Wi-Fi is connected to the Internet server unit to obtain basic data. It is also used to obtain a map of soil MC distribution in the ground. This results in the calculation of soil MC deficiency based on SSIM to process irrigation valves to maintain the needed level of water content in the farm area.

Wireless nodes were used to collect sensor data and transmit it to a designated storage node, where the data is recorded in a dedicated database. The system is powered by a 6V/4.5A lead-acid battery and supplemented with solar panels for daytime charging. The proposed soil moisture (MC) prediction approach utilizes a feedforward neural network that takes real-time sensor data as input. Once the soil MC is accurately predicted at each wireless sensor network (WSN) point across the field, linear interpolation is employed to create a full distribution map of soil moisture over the farmland. This predicted moisture content is then compared to the desired levels using SSIM, a process that plays a key role in controlling the agricultural irrigation valve by assessing actual water needs through a weather-based irrigation mechanism combined with fuzzy logic.

To enable the nodes to collect sensor data in real time, the system is fed by a combination of lead-acid batteries and solar panels. MATLAB is then used to process data to predict soil MC and control the irrigation valve. Using feedforward neural networks and binary interpolation, the system provides accurate predictions of soil MC as well as generating distribution maps across entire farmlands.

Since the performance of a neural network is related to the learning rate, which affects the convergence of error rates, the study relied on calculating statistical errors such as Root Mean Square Error (RMSE), R-squared error, and Mean Squared Error (MSE), through two distinct optimization techniques, such as Variable Learning Rate Gradient Descent VLRGD and GD-based NN data prediction strategies. A higher learning rate can lead to faster convergence error

rates while training the neural network. However, very high learning rates may make the neural network unstable. It was observed that VLRGD significantly reduced RMSE and MSE, and this indicates its success, so this algorithm relied on a greater learning rate. While the GD-based NN prediction system leads to much higher MSE and RMSE values.
In this paper [7] the use of IoT technology in agriculture is addressed, focusing specifically on smart irrigation systems. The study explained the required equipment's used, including various sensors and motors used in irrigation systems based on the IoT, communications technologies, and cloud platforms used in storing data. The importance of water management in agriculture is presented. In addition, the study highlights the challenges facing systems using the IoT in agriculture and future developments in this field.

The most common sensors used here that contributed to developing the irrigation process and increasing crop production are devices for sensing soil moisture and temperature sensors, for example: DHT11, DHT22, LM35, and humidity sensors in general, such as BME280, EE160, HTU210, devices for sensing light intensity, and others for sensing radiation. All these devices have contributed to improving the irrigation process by providing and obtaining data in real time. This data is about the level of humidity, the temperature, humidity, intensity of light and radiation. All of these lead to scheduling irrigation accurately and with high efficiency. By monitoring the criteria mentioned above, farmers can avoid excessive irrigation, thus avoiding some diseases that result from excessive irrigation of crops, conserving water, and increasing the production of healthier and more efficient crops.

The international conference in India witnessed various presentations on the use of the IoT in agriculture, including field monitoring. These presentations highlighted the benefits of the IoT in improving efficiency and productivity, as the cloud is used to store and also process the data collected through sensors and the cloud platform, which was more the most widely used is thing speak followed by more platforms like FIWARE and Dynamo DB. Data can also be stored in databases (MySQL and SQL). Managing and analyzing big data is very important to improve the irrigation system with the use of techniques, for example fuzzy logic and machine learning in countries where farmers cannot afford commercial solutions. Low-cost independent sensors are being used and sustainable irrigation systems are also being developed. In the end, and in general, what this time wants to convey is a comprehensive view of irrigation systems based on the IoT, focusing on modern matters in sensors and IoT technology for smart agriculture. The importance of water management, the challenges facing farmers, and developments in low-cost sensors were discussed. The study also compares methods. Remote sensing and irrigation systems based on the IoT. The role of technology in agriculture to manage water and improve crop production was highlighted.

This paper [16] outlines an advanced irrigation system for precision agriculture called AREThOU5A, which uses big data, IoT technologies including WSN and edge computing to optimize the system. It underlines the use of Energy-Harvesting (EH) and Radio Frequency (RF) as an ingenious technique of energy accumulation and delivery in systems that require minimal power.

Key Technologies and Methodologies:

IoT and Machine Learning: AREThOU5A integrate IoT and machine learning at the fingertip of breath to extend the capabilities of 5G smart irrigation system. The system will prove the alternative energy delivery method using RF EH technology.

Wireless Sensor Networks: GSM and General Packet Radio Service GPRS based WSNs along with data transmission and communication protocols are shown in various findings.

Energy Harvesting and Management: This paper discusses the issue of using of RF EH for powering IoT devices and exact models to match the power from renewable energy sources like solar photovoltaics.

Data Analysis and Machine Learning: Reliance on thermal imaging methods in conjunction with Partial Least Squares Regression PLSR and Adaptive Neuro-Fuzzy Inference System ANFIS techs for the taking of agriculture decisions.

Emerging Technologies: Message Queuing Telemetry Transport (MQTT) and LoRa contains both integration for effective data transmission.

AREThOU5A System Architecture: The platform consists of four primary subsystems: Catalogue design is divided into four main subsystems: measurement, routing, user interface, and server with several secondary data collection and processing subsystems to supplement the result as shown in the Figure 2.5. The scheme follows layered architecture pattern which serves as a water management tool in smart irrigation.

Measurement Subsystem: Uses of LoPy4 microcontroller and IoT sensors for generating data related to temperature and soil moisture.

Routing Subsystem: Investigates and enrolls new applicants to the platform by examining and starting the onboarding process.

User-interface Subsystem: Alternative UI implementation which is based on LoRaWAN network server stack.

Server Subsystem: Establish connectivity with email and database services respectively.

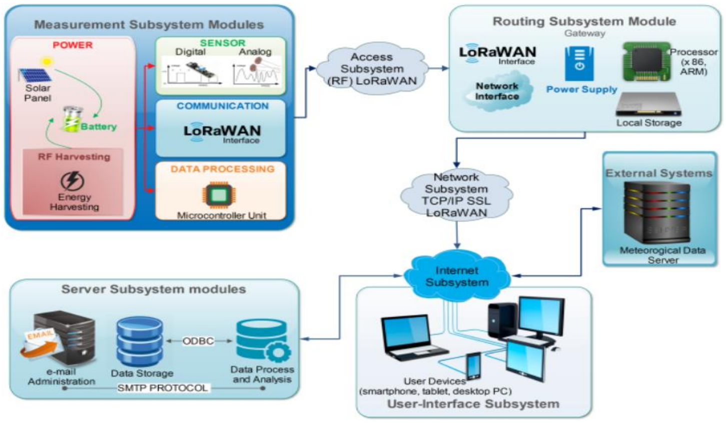


Figure 2.5: AREThOU5A smart irrigation IoT platform and its subsystems.

Layered Architecture:

L1: Physical Layer

L2: Data Link Layer

L3: Network Layer

L4: Authentication Layer

L5: Application Layer

RF Energy Harvesting Module: However, rectenna for the RF-to-DC power is a key part of the system, which is one of a few unique features. The experimental trial demonstrates how successful this approach is through the two designs, antenna, and rectifier, that are proven to be effective in harvesting and generating energy.

Experimental Evaluation: Evaluation of the RF EH module revealed a very high level of RF signal to DC voltage conversion efficiency with the antenna E-shaped UWB type with minimal return loss and optimal power transfer efficiency. The findings substantiates the capacity of RF EH to feed power sources of smart sensors in agriculture practice.

AREThOU5A is a breakthrough approach in precision farming, showing this phenomenal fact that combining RF power adsorption with IoT technologies is possible and may be applied to the sustainable and smart irrigation systems. The work is therefore the plan for modern agriculture technology integrating each other for purposeful management of production so as encourage innovation on addressing energy constraints in IoT networks.

The work in [17] uses a drip system to provide the required amount of water by determining the level of moisture.

The project determines the level of moisture sensing for crops to provide the required water when needed by focusing on the drip system.

The equipment used in this study are ATmega 328 as a microcontroller, GSM module for the communication system, and two types of sensors for humidity and soil moisture.

The initial idea for the project is a closed-loop real time irrigation control system to use soil moisture level and temperature.

By using the right quantity of water at the right time and reducing human labor to turn on/off valves, the project has proven a good experience for automated irrigation.

The system works by setting the irrigation time according to the humidity level reading from the sensors, then irrigating the field automatically. Soil moisture level is measured through a capacitive sensor placed in different places in crop fields. The counters are automatically valued with a built-in microcontroller in different areas to be used to control data at different periods. As for the consumer, a Peripheral Interface Controller (PIC) microcontroller and an Advanced RISC Machine ARM microcontroller are used in the polar station. Periodic data is continuously sent by the PIC microcontroller and then the data is processed by the ARM microcontroller. Depending on the transmitted data, if the value differs from the previously taken data, this indicates that the humidity level has changed in some way, and the user receives a message to the mobile phone through the GSM module. However, if the data exceeds the threshold values, the motor will start, and water will be delivered to the specified field. If the values obtained are less than the threshold values, the motor stops working, and the crops are saved from damage from excessive irrigation.

The system works through messages sent by the user to the GSM module, to start processing instructions. Data from soil moisture and humidity sensors are analyzed and sent back to the user's mobile device for monitoring. Users can command the GSM unit to activate “Zone 1” or “Zone 2” while deactivating the other unit, specify operation run times and give a duration of time for the engine to run. They receive confirmation when the engine is running and can issue the command to shut it down.

Based on the results, the project proved to work well and was simulated through Proteus software. The previous steps for both temperature and humidity have also been verified and relied upon to turn on/off the motor through the GSM module of the users' phones. In the case of multiple agricultural fields, when water is supplied to the first field, once enough is provided, the motor activates. The motor state is then automatically transferred to the second field, and the process is reversed when needed. It was also noted that the duration of operation of the engine is set, and that when the engine stops, a message is sent to the farmers' phone.

In [18], implementing a smart agricultural system using several sensors was proposed. Internet of Things devices as shown in Figure 2.6 Arduino, and a wireless sensor network were used to monitor environmental conditions in agricultural fields and also to improve water use and reduce water waste and thus increase agricultural crop production. The system is an Android application for farmers to control irrigation. The system provides notifications in the event of an animal attack, for example, and also receives suggestions for combating agricultural pests.

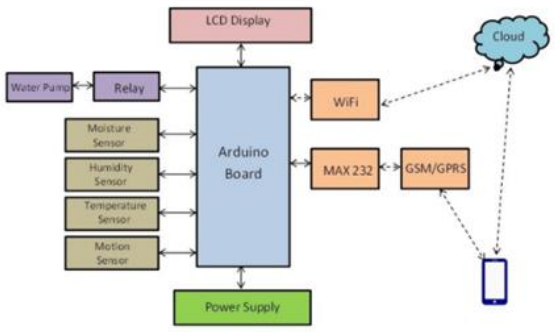


Figure 2.6: IoT implementation

Sensors will be used to sense humidity in the air, soil moisture, temperature, and movement connected to the Arduino Uno R3 microcontroller board. The mechanism of these sensors is as follows: These sensors will continuously monitor environmental conditions and send data to the Arduino board for processing because it is considered a gateway to the IoT. The system also contains on a distributed wireless network of soil humidity and temperature sensors deployed in places of plant roots for real-time monitoring and transmission of this data. This application includes the following main features:

1. Choose whether the user wants to start or stop the water pump.
2. Choose irrigation files to schedule irrigation times.
3. There are also suggestions for using pesticides suitable for crops.
4. Alert if there is an animal invasion.

All these four features help farmers with irrigation control, appropriate and effective scheduling, crop care recommendations, and alerts in the event of threats to the fields.
The proposed system will benefit farmers in terms of controlling water use and increase crop productivity. This is done by monitoring environmental conditions such as air humidity, soil moisture, and temperature, and through sensors and the Internet of Things, farmers can obtain very accurate data about the requirements of crops and their need for irrigation. This allows the use of water more accurately, which reduces water waste and works to improve crops and increase their productivity.

The suggested system is based on the following steps: constantly obtaining sensor data from the temperature, humidity, soil moisture and motion sensor attached to the Arduino Uno R3 microcontroller board and transferring the data to the cloud through the Internet of Things gateway (using the ESP8266 Wi-Fi module). If the data is higher than the specified threshold values for temperature, humidity, or other parameters, trigger an action in the decision logic in the cloud. It then sends notifications to the Smart Farming Android application based on the output. Finally, the farmer is contacted via a text message to his mobile phone and is allowed to take control. With the irrigation system by turning, it on or off.

There are two main limitations of the proposed solution, the first one is its reliance mainly on Internet connection for monitoring and time control. This is considered a restriction in agricultural places where communication is weak. To address the problem, an improvement can be made by integrating a GSM unit to send signals directly to the farmer's phone via a short message. This is an alternative method. To connect to the Internet
The second limitations is with the animal movement sensor. There are limitations on accurately distinguishing between different types of animals. Improvement can be achieved by adding advanced image recognition technology or advanced sensors to provide more accurate detection of certain types of animals.

A paper [19] considers building a state-of-the-art irrigation system which improves efficiency in the farming practices with IoT, cloud-computing and optimization methods as explained in Figure 2.7. It employs low-cost sensors to keep watch on the different environmental and the soil conditions for instance moisture levels, pH, and weather data recording in the Thing Speak cloud service. Next, the water demand for each block from different models is input into an optimization model that is solved using an ARM controller, which subsequently triggers the on/off switching of solenoid valves with the appropriate irrigation rate, and mobile devices are also available for continuous monitoring.

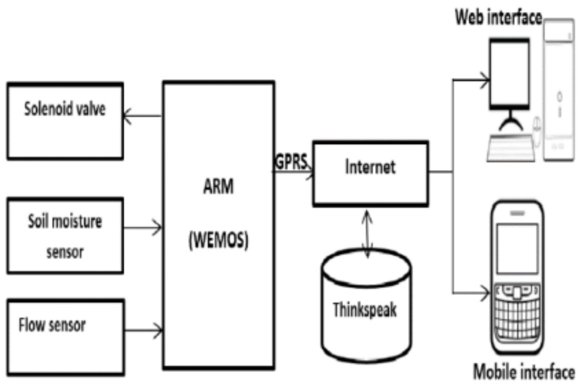


Figure 2.7: IoT architecture of the smart irrigation system.

The system design comprises sensors network and WEMOS D1 controller, which works by Company name performing the functions of data communication surveying (Wi-Fi and GSM GPRS) uniting irrigation management in different sectors. The optimization model developed for the system considers water usage, thus taking soil moisture, rainfall, and water flow rates into consideration and ensuring that the constraint limitations are properly accounted for. This model is solved by means of applications like MATLAB's linprog routine, giving the decision support framework that the water for irrigation will be used wisely.

Experimental results from a three-week study comparing traditional manual irrigation methods with the automated system revealed significant water savings: drip irrigation reduced water by 24%, sprinkler systems saved 20%, and closal use of solenoid valves reduced water usage by 16%. Makes it to the greatest extend possible the predicted savings significantly more substantial. Firstly, this research revealed that soil pH values were lowered with the increase of moisture, hence the integration of IoT into the Cloud Computing platforms and optimization models would enable substantial productive efficiencies from water management.

This study [20] discusses the role of technology in a smart irrigation system based on the IoT and sensors to enhance crop production as well as reduce water waste. The role and importance of solar energy, monitoring temperature, humidity, and soil moisture, in this system are also discussed.

It alternates in detail about the role of these technologies in achieving sustainable development goals, the most important goal of which is preserving the environment. It also explains how water can be used without wasting it in this system.

Here some layers play an important role in transmitting and collecting data and implementing the smart irrigation process. These layers are the perception layer, the network layer, and the application layer. These layers also play a role in efficient water management.
In this system, machine learning and software were used to address challenges such as climate change, water scarcity, and the decline in the agricultural workforce. Artificial intelligence has also been integrated that collects crop information and enhances irrigation scheduling.

The system consists of sensors, actuators, and cloud services (databases). Arduino and Raspberry microcontrollers, photoelectric panels, and control devices are used to control irrigation operations. Irrigation is adjusted based on several factors such as the amount of humidity, temperature, and real-time weather.
Also, a weather modeling system based on logic was used to control the valves in different weather conditions.

The system often contains applications that are easy for farmers to use, allowing them to monitor irrigation operations remotely. The application provides real-time information and data on weather conditions, humidity, and irrigation schedules. It also allows users to adjust irrigation settings, receive notifications about possible problems, and track the water use process through application.

After collecting the data (from the sensors), it is transferred to the cloud and accessed through an application on smartphones.

The Water Bit portal collects information related to soil moisture and weather conditions and sends it through wireless technology to the cloud in a reliable and safer way, and this data is updated after every few minutes.

As for solar energy and its role here, it reduces environmental pollution by reducing the release of greenhouse gases and reducing the use of fossil fuels, as solar energy directly supplies the system with energy, increases crop production, and works to improve its quality.

To summarize, these steps were adopted in building the application: Installing sensors in the field to monitor humidity levels and weather conditions, then the sensors were connected to a gateway device to transfer data to the cloud, then access to the data and control features is through the application, and real-time information about soil moisture is monitored. And temperature. Irrigation settings are adjusted remotely based on the information coming from the sensors, and alerts are received about potential problems.

As for the result, it was noted that this application helped to significantly conserve water and save costs. For example, the automated system for monitoring soil moisture in Ipswich, Australia, led to a significant reduction in water use, and the efficiency was better compared to traditional methods. Also, the use of the IoT in a mushroom farm in Thailand led to Energy savings range between 20% and 29% through network division and automatic irrigation control.

Reducing water waste and improving use efficiency led to a noticeable increase in crop production.

The main problem that the system faces is the effective management of huge data. This is the excessive load of data, its analysis and its use. To solve such a problem, blockchain technology was proposed to get rid of unnecessary data, use aggregation technology, and implement an effective algorithm. Also, as we mentioned previously, the use of solar energy to operate devices and ensure appropriate transmission. Data is very important to improve the irrigation process and the use of artificial intelligence also improves data analysis.

This study [21] presents the development of a new intelligent irrigation system (IoTML-SIS) supported by the IoT and ML. The system integrates IoT-based sensors to monitor light conditions, humidity, temperature, and soil moisture light conditions in agricultural land. The collected data is then transferred to the cloud server for analysing information and making decision. The main feature of the system is the classification algorithm using Artificial Algae Algorithm (AAA) combined with LS-SVM model to predict the irrigation requirements Specifically, AAA was used to optimize the parameters LS-SVM model, significantly improving efficiency of the classification, where the highest accuracy reached 0.975.

ICT and IoT technologies such as WSN and CC make the IoTML-SIS system part of the information communication technologies used for remote sensing and data management. These technologies enable large-scale water allocation by monitoring weather conditions and terrain contours. A variety of sensors are used in the system to estimate crop water needs, including variable soil sensors with dielectric properties, IR radiometers, satellites, multispectral and thermal camera. Moreover, this paper uses mobile applications of intelligent irrigation systems based on sensors that have been implemented.

The materials and methods of IoTML-SIS system is generalized in this paper which comprises of data acquisition using specialized sensors including soil moisture sensor (HL-69) and air moisture sensor (AM2302 DHT11) coupled with the (light) sensor (BH1750 FVI) for light intensity measurements. Sensors collect the data and texture features for the training of the LS-SVM model within the AAA which updates the model every time for better performance and response. The reliability of the suggested IoTML-SIS model is tested using real sensor data and passing through several class labels to verify that the irrigation needs are determined adequately.

The coming of the IoTML-SIS model brings with it a significant step forward in precision agriculture, thereby providing a more complex smart irrigation solution via the combination of IoT and ML technologies. The system in addition to being used for doing the tasks of managing the water efficiently also allows remote monitoring as well as controlling through mobile applications that turns it into a useful tool for modern agriculture methods.

The system described in [22] applies traditional smart irrigation techniques which control water delivery through automatic sensors of soil moisture and temperature and humidity levels. The system operates with a fixed ON/OFF action for its pump whenever the moisture threshold is reached. The method provides water to plants yet it fails to achieve precise measurement which could result in excessive watering or insufficient watering.

The SAIS implements sophisticated water control capabilities to both detect water requirements and exactly determine the needed water distribution time. The system operates beyond basic ON/OFF control by executing precise pump duration periods. An LED indicator functions for a duration equivalent to the necessary water quantity to guarantee precise watering. The system differs from cloud-based approaches by carrying out data processing at the Arduino microcontroller level together with DS18B20, DHT11 and YL-69 sensors to deliver instant irrigation decisions. Through LoRa communication technology the system provides distant and energy-efficient data exchanges which make it suitable for regions without internet connection. The localized real-time functionality decreases water waste and maximizes irrigation performance and helps farmers maintain sustainable farming practices beyond standard binary irrigation options.

2.3 Chapter Summary

Thus, the studied works on smart irrigation systems show IoT, AI, and some other innovations in technology in agriculture lead to the yield rise and water management efficiency improvement. They are designed with enhanced sensors and data-driven algorithms which make them to deliver high level of precise control over irrigation processes or irrigation methods thus saving the farmers water wasted and improving the health of crop yield. Through integrating different scientific studies, a picture of the modern role and impact from technology of improving agricultural sustainability and resource management under the challenge of environment takes shape. The previous studies show that recent technology could improve and enhance the irrigation systems.

Chapter 3

Research Methodology

3.1 Overview

The purpose of this chapter is to describe the necessary methodological processes in designing, implementing and assessing the SAIS. The methodologies are designed in a way that take advantage of IoT, AI, and ML to improve traditional approaches of irrigation with a goal of reducing water wastage, increasing crop yields efficiently. There is more focus on the need to implement technology that is flexible and applicable to numerous agricultural contexts, in addition to the components that must comprise the enabling technology, which should meet the current agricultural requirements.

3.2 Methodology

The methodology used in this project includes qualitative and quantitative approaches to analyze the SAIS. The methodology will depend on the following steps:

1. System design and development:

- Hardware components:

Sensors: Appropriate sensors will be selected to monitor soil moisture and temperature in addition to other weather conditions.

Microcontroller: A microcontroller such as Arduino will be used to process the data received from the sensors.

Communications module: LoRa will be used to transmit data wirelessly.

2. System implementation:

- Field deployment: sensors and microcontrollers are used in some agricultural fields.

- Data Collection: Data will be collected continuously from the environment of a particular site, such as soil moisture, temperature and other elements over a specified period.

3. Data analysis:

- Data processing: This stage refers to cleaning and pre-processing the collected data to ensure that it is suitable for training machine learning models.
- Machine Learning Models: ANN will be used to predict required irrigation levels and maximize the use of available water resources. These models analyze historical and actual data to make irrigation scheduling decisions.
- Artificial intelligence algorithms: Artificial intelligence can establish crop irrigation patterns with efficient methods and times and thus optimize irrigation schedules to ensure efficient water use and improve crop production.

5. Notification and Control

- User Notification via Telegram: The predicted amount of water required for irrigation will be sent to the farmer through a Telegram notification.

- LED Indicator: If irrigation is required, an LED will turn on to indicate that water needs to be supplied.

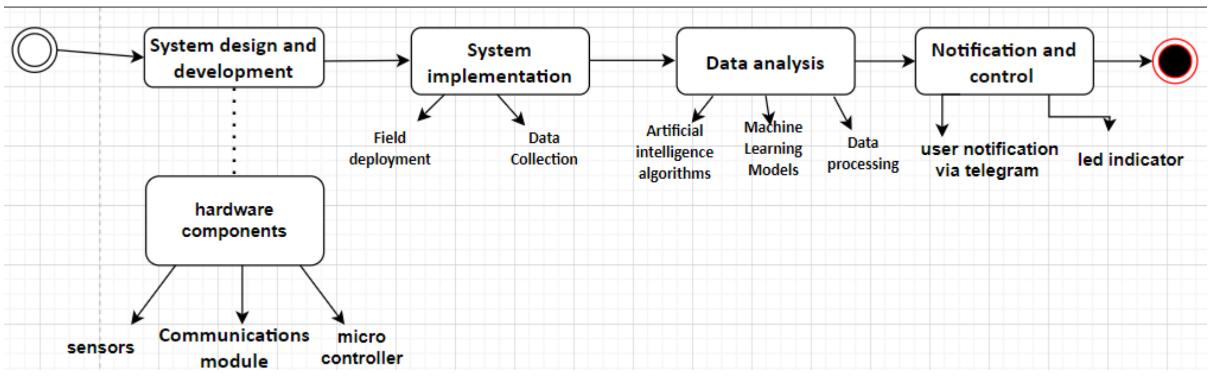


Figure 3.1: Methodology flow chart

3.3 Components

3.3.1 DS18B20 Temperature Sensor

The DS18B20 is a temperature sensor that has been developed by Maxim Integrated and fat its core forms a 1-wire bus. It is expected they are very resistant, which makes them suitable to work in circumstances such as chemical solutions, mines or soil. The sensor is very well built, and this one can even be purchased with a shell that makes it waterproof, hence easy to install.

It can take a temperature as low as -55°C and as high as 125°C with a stand error of ±0. The temperature is around +5°C. Each sensor is given a unique ID and data are transmitted through a single pin of the microcontroller. This makes it good for use in designs where temperature at several points could be read but are limited on the number of digital pins in the microcontroller.

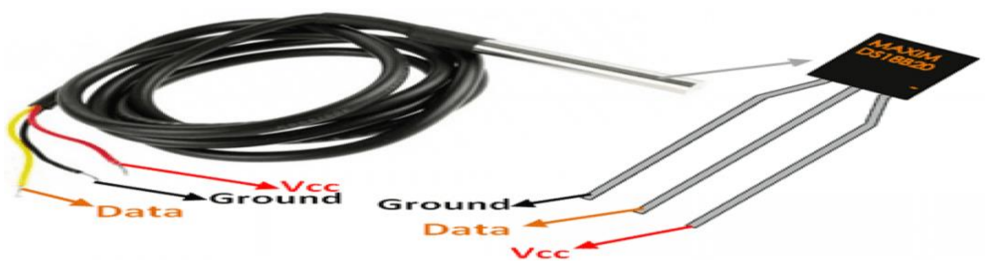


Figure 3.2: Sensor DS18B20

Applications:

- Measuring temperature in challenging environments.
- Measuring liquid temperatures.
- Temperature monitoring in two or more points in the process or at various stages of the process.

Pin Configuration:

Table 3-1: Configuration of DS18B20 sensor

Pin Name	Description
Ground	This must be connected to circuit ground.
Vcc	Powers the sensor; it can be powered with 3.3V or 5V.

Data	Displays the value of the temperature which can be only read via one-wire communication.

3.3.2 DHT11 Sensor

The DHT11 is a typical temperature and humidity sensor used to measure temperature as well as humidity. For temperature sensing, it is equipped with an NTC, and for humidity sensing, an 8-bit microcontroller is used to output the data of temperature and humidity in a serial form. Hypothetically, the sensor was factory calibrated; therefore, it does not require a lot of effort to interface with other microcontrollers.

Temperature sensor sensing range is 0°C – 50 and humidity sensing range is 20% – 90% with accuracy of ± 1 °C and ± 1% respectively.

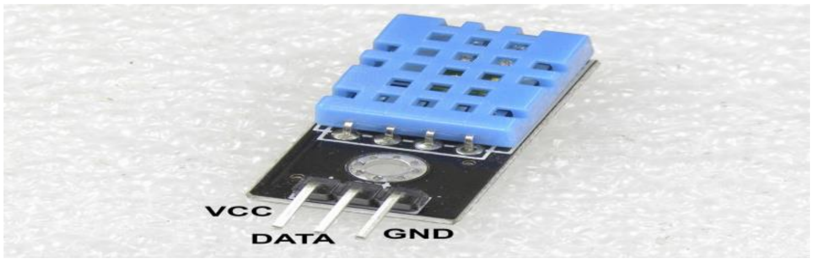


Figure 3.3: Sensor DHT11

Applications:

- Measure temperature and humidity
- Local weather station
- Automatic climate control
- Environment monitoring

DHT11 Specifications:

- Operating Voltage: 3.5V to 5.5V
- Operating current: 0. 3mA (measuring) 60uA (standby).
- Output: Serial data.
- Temperature range: It works at temperatures ranging from 0 degree centigrade to 50 degrees centigrade.
- Humidity range: It may range from as low as 20 percent to as high as 90 percent on some topics.

- Resolution: The parameters temperature and humidity both should be 16-bit
- Accuracy: ±1°C and ±1%.

Pin Configuration:

Table 3-2: Configuration of DHT11 sensor

Pin Name	Description
Vcc	Power supply 3. 5V to 5. 5V
Data	It can output both Temperature and Humidity readings to serial data.
Ground	It is situated near the ground of the circuit

3.3.3 YL-69 Sensor

The ability of YL-69 soil moisture sensor is that it is a kind of electrical conductivity sensor which is used to measure the moisture in the soil area. This component has two probes that are inserted into the ground. When the condition of the soil changes, the electrical current between the two probes will change. Another part of the platform is the sensor module, the LM393 core comparator that integrates the sensor and converts the analog signal received from it into a digital signal. This is then amplified and passed through a digital signal received by the Arduino Uno where the amount of moisture in the soil can be detected in real time.

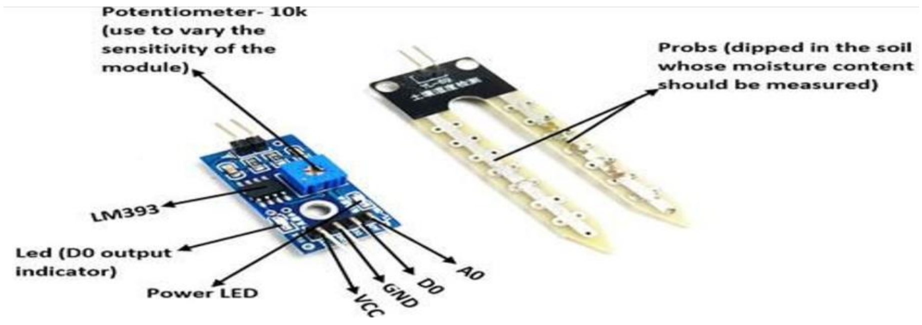

Potentiometer- 10k (use to vary the sensitivity of the module)
Probs (dipped in the soil whose moisture content should be measured)
LM393
Led (D0 output indicator)
Power LED
VCC
GND
D0
A0

Figure 3.4: Sensor YL-69

Specifications:

1. Moisture Detection Range: The YL-69 Soil Moisture Sensor Module helps measuring the soil moisture within the range then giving accurate results.
2. Digital Output: The sensor module comes with digital output thus can be connected to Arduino UNO digital input pin 3.

3. Operating Voltage: It works in a voltage range suitable for Arduino UNO thus integrating well with projects.
4. Compact Size: The compact design of the YL-69 Sensor Module makes it possible to insert it directly into the soil.

Pin Configuration:

Table 3-3: Configuration of YL-69 sensor

Pin Name	Description
VCC	Vcc pin provides the power supply to the module, which may be +5V.
GND	Power Supply Ground.
DO	Digital Out Pin for Digital Output.
AO	Analog Out Pin for Analog output.

3.3.4 Arduino

Arduino UNO is a microcontroller board which has the capacity of the ATMega328P microcontroller. This one is used in this project because of its ease of use and simplicity. The board offers an excellent opportunity for development and prototyping and as such should be preferred for sensors and other peripherals integration.

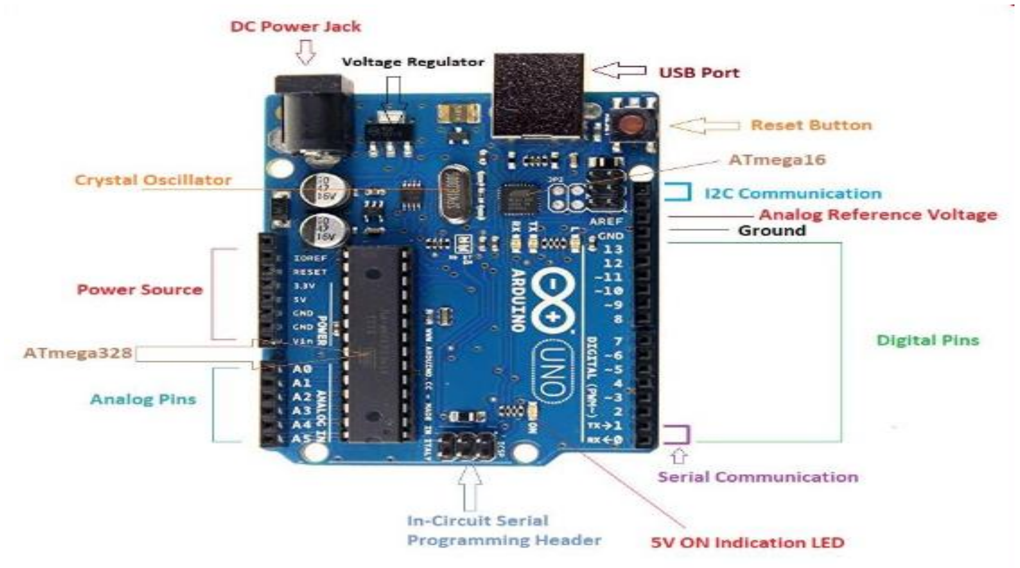


Figure 3.5: Arduino UNO

Key Features:

- Digital I/O Ports: It has 14 digital I/O pins of which 6 pins are programmable as pulse width modulation (PWM). These pins can be used to read inputs from many sensors and switch the outputs of many actuators as well as to communicate with other digital devices.

- Analog Inputs: 6 Analog Inputs can measure different voltage levels which are ideal for Sensors that put out Analog signals.
- USB: usually employed in transferring sketches to the board and in serial communication between the board and the computer; can power up the board.
- Power Socket: allows the panel to be powered from an external source to meet the needs of the panel members.
- Reset Button enables the user to clear all information displayed on the panel and continue with the new command.

3.3.5 LoRa

LoRa (short for "Long Range") is a proprietary radio communication technology originally created by Cycleo and later acquired by Semtech. It utilizes spread spectrum modulation techniques that stem from Chirp Spread Spectrum (CSS) technology, making it especially suitable for long-distance, low-power wireless communication. The use of LoRa in a project has many benefits, especially in projects that involve use of sensors and monitoring of environment. LoRa is particularly well-suited for IoT applications, allowing devices to communicate over several kilometers while consuming minimal power. LoRa provides coverage of more than a few kilometers, so it is ideal for large scale applications such as agriculture, monitoring the environment, infrastructure in smart city and more; such applications require little to no infrastructure. It is specifically useful in battery-operated machines due to its low power consumption to help maintain battery life in sensor devices in remote areas with limited access to charging facilities for long periods. LoRa guarantees dependable information transfer in different environments that can be crucial to the operation of sensors, such as the YL-69 Soil Moisture Sensor, DHT11 Temperature and Humidity Sensor, and the DS18B20 Temperature Sensor in providing precise readings routinely. Due to LoRa modules, interfacing low-power microcontrollers is relatively easy, such as the Arduino Uno, to collect data from multiple sensors and send the information through LoRa to a main data processing and analysis center, making the overall system and project design relatively convenient and less complicated.

Figure 3.6: LoRa

Connections

Sensors to Arduino Uno

Table 3-4: Connections for YL-69 soil moisture sensor to Arduino Uno

YL-69 Soil Moisture Sensor pins	**Arduino Uno**
Vcc	5V
GND	GND
AO (Analog Out)	A0

Table 3-5: Connections for DHT11 temperature and humidity sensor to Arduino Uno

DHT11 Temperature and Humidity Sensor pins	**Arduino Uno**
VCC	5V
GND	GND
Data	digital input pin 3

Table 3-6: Connections DS18B20 temperature sensor to Arduino Uno

DS18B20 Temperature Sensor pins	**Arduino Uno**
VCC	5V
GND	GND
Data	Digital input pin 4 on the Arduino, with a 10kΩ pull-up resistor between the Data pin and VCC.

3.4 AI Algorithms

Artificial Neural Networks (ANN):

It is a kind of deep learning method that relies on the neural structures and neural processing of the human brain. It is useful in analyzing and predicting patterns. It was included among the various machine learning models used in the project, as it can be applied to predict the potential level of irrigation from past and current data.

3.5 Chapter Summary

This chapter explained all the methods followed for designing and testing the proposed SAIS. These comprised of the system design and implementation, data gathering, interpretation, verification, and evaluation, assessable influence, the scheme viability test, and security concerns. These steps make the SAIS very reliable in terms of its potential applications to modern agriculture, especially in the improvement of water management for crop production.

Chapter 4

Results and Discussion

4.1 Dataset

The dataset for this project comprises approximately 10,000 samples, carefully curated from multiple sources to ensure accuracy and reliability. Approximately 4,000 samples were filled by an agricultural engineer, whose expertise was instrumental in defining precise irrigation requirements. These samples include detailed insights into crop-specific growth stages, water requirements, and environmental factors. This expert also reviewed and approved the dataset after additional data integration, further enhancing its credibility.

The remaining 6,000 samples were sourced from three publicly available datasets and five tables provided by the expert[23][24][25].

In addition to these publicly available datasets, the expert provided five detailed tables. These tables offer valuable insights into crop-specific growth stages, water requirements, and environmental factors affecting irrigation. The tables outline critical parameters such as:

- Growth stages, including germination, seedling, vegetative growth, flowering, and harvesting.
- Weekly and stage-wise water requirements for different crops, providing a granular understanding of irrigation needs.
- Variations in water needs based on environmental conditions such as temperature, humidity, and soil moisture.

The dataset incorporates a diverse range of features essential for predicting irrigation requirements in a (SAIS). These features include:

- Crop type: Categorical data representing five types of crops (tomato, cucumber, potato, onion, and lettuce), encoded using one-hot encoding.
- Age: The age of the crop in weeks, correlating with its growth stage.
- Moisture and humidity levels: Indicators of soil and air conditions.

- Air temperature and soil temperature: Environmental factors affecting water evaporation and absorption.

The dependent variable, ‘Exact_Water_Needed,’ specifies the precise quantity of water required for each sample in milliliters.

Our dataset development and methodology align with the research presented in[22], which proposes an IoT-based Smart Irrigation Management System. This study highlights the significance of using sensors such as DHT11 for air temperature and humidity, DS18B20 for soil temperature, and Capacitive Soil Moisture v2.0 for soil moisture monitoring. The research underscores the importance of real-time monitoring, data-driven irrigation decisions, and cloud-based automation, all of which have been integrated into our dataset for enhanced precision irrigation.

Crop-Specific Water Requirements:

Table 4-1: Tomato growth stages and water requirements.

Growth stage	Age(weeks)	Approximate water requirement per stage	Water per week
Seedling	1-4 week	20-30 liters	5-7.5 liters/week
Early vegetative	5-8 week	30-40 liters	7.5-10 liters/week
vegetative	9-12 week	40-50 liters	10-12.5 liters/week
Pre-flowering	13-16 week	50-60 liters	12.5-15 liters/week
Flowering	17-20 week	60-70 liters	15-17.5 liters/week
Fruit setting	21-24 week	60-70 liters	15-17.5 liters/week
Fruit development	25-32 week	80-100 liters	10-12.5 liters/week
Mature harvesting	33-36 week	35-50 liters	8.75-12.5 liters/week

This table presents the weekly and stage-wise water requirements for tomato plants under standard loamy soil conditions. Water demand increases significantly during flowering and fruit-setting stages, emphasizing the need for careful irrigation management. Adjustments may be required for different soil types: sandy soil requires 10–20% more water, while clay soil requires 10–15% less. Drip irrigation is recommended to optimize efficiency and minimize water wastage.

Table 4-2:Cucumber growth stages and water requirements.

Growth stage	Age(weeks)	Approximate water requirement per stage	Water per week
Germination	1-2 week	10-15 liters	5-7.5 liters/week
seedling	3-5 week	20-30 liters	6.7-10 liters/week
Vegetative growth	6-10 week	50-70 liters	10-14 liters/week
Flowering	11-13 week	40-60 liters	13.3-20 liters/week
Fruiting	14-18 week	30-75 liters	6-15 liters/week

The cucumber crop follows a structured irrigation requirement, with the highest demand during flowering and fruit development. The table provides a weekly breakdown of water requirements across different growth stages. Proper soil drainage is crucial, especially in clay soils, to prevent waterlogging. Loamy soil is considered optimal, while sandy soil may require additional irrigation adjustments (10–20% increase).

Table 4-3: Potato growth stages and water requirements.

Growth stage	Age(weeks)	Approximate water requirement per stage	Water per week
Seedling	0-2 week	2-4 liters	1-2 liters/week
Vegetative growth	3-5 week	6-12 liters	2-4 liters/week
Tuber initiation	6-7 week	6-10 liters	3-5 liters/week
Tuber bulking	8-12 week	12-20 liters	2.4-4 liters/week
Maturation	13-15 week	6-12 liters	2-4 liters/week

The water requirements for potatoes vary across different growth stages, with peak demand occurring during the tuber initiation and bulking phases. The table illustrates the irrigation needs at each stage, ensuring adequate water supply for tuber development. Sandy soil requires more frequent watering due to rapid drainage, while clay soil should be avoided unless well-drained to prevent tuber damage.

Table 4-4: Onion growth stages and water requirements.

Growth stage	Age(weeks)	Approximate water requirement per stage	Water per week
Germination	1-2 week	1-2 liters	0.5-1 liters/week
Seedling	3-5 week	1.5-2 liters	0.5-0.7 liters/week
Vegetative growth	6-12 week	3-4 liters	0.5-0.7 liters/week
Bulb formation	13-17 week	1-2 liters	0.2-0.5 liters/week
Bulb maturation	18-21 week	0.5-1 liters	0.2-0.3 liters/week

Onion plants require consistent moisture levels, particularly during germination and bulb formation. The table provides a detailed distribution of water needs, showing how irrigation demand gradually decreases as the plant matures. Proper drainage is essential in clay soils to prevent waterlogging, while sandy soil may necessitate more frequent irrigation to retain optimal moisture levels.

Table 4-5: Lettuce growth stages and water requirements.

Growth stage	Age(weeks)	Approximate water requirement per stage	Water per week
Germination	1 week	0.5-1 liters	0.5-1 liters/week
Seedling	2-3 week	1-2 liters	0.5-0.7 liters/week
Vegetative growth	3-5 week	2-3 liters	0.6-1 liters/week
Heading	6-8 week	2-3 liters	0.7-1 liters/week
Maturity/Harvest	9-10 week	1-2 liters	0.5-1 liters/week

The irrigation requirements for lettuce are relatively moderate but must be carefully managed across different growth stages. The table provides a structured breakdown of water distribution from germination to harvest, ensuring optimal plant development. Loamy soil is the best option, as it retains moisture while ensuring proper drainage. Adjustments in irrigation should be made for sandy or clay soils to maintain appropriate water levels.

Soil Considerations for All Crops:

- Loamy Soil: Best for most crops, offering a balance between moisture retention and drainage.
- Sandy Soil: Drains quickly; requires 10–20% more irrigation.
- Clay Soil: Retains water but risks waterlogging; irrigation must be adjusted to avoid excess moisture.

Preprocessing and Final Considerations:

Preprocessing steps were applied to enhance the dataset's usability for machine learning models. These steps include:

- Scaling of numerical features to standardize the data.
- One-hot encoding for categorical variables such as crop type.
- Logarithmic transformation of the target variable to address its skewed distribution and ensure better model performance.

This meticulously curated dataset represents a balanced integration of expert knowledge and publicly available resources. The combination of structured preprocessing and validation by an agricultural engineer ensures its robustness and reliability for predicting total water requirements in a (SAIS). By leveraging this dataset, the project aims to enhance precision irrigation techniques, optimize water usage, and contribute to sustainable agricultural practices.

4.2 Preparing Dataset For ANN

The following preprocessing is applied to ensure that the dataset is ready for ANN. First, the target variable ‘Exact_Water_Needed’ is log transformed due to skewness and heterogeneous variance for applying regression. The categorical feature `Crop_Type’ is encoded using one-hot encoding to be compatible with the requirements of ANN; it is represented by binary columns from five crop types (Tomato, Potato, Cucumber, Onion, and Lettuce). The features are then normalized by using the RobustScaler feature scaling algorithm that uses the median and quartiles values to scale our data, thereby reducing the effect of outliers. The dataset is divided into three subsets: the training dataset, which constitutes 70% of the data, validation dataset, which constitutes 15% of the data, and the testing set, which constitutes 15% of the data. By splitting the data in such a manner, it is possible to train the model, adjust it for performance using the validation data, and test it on new data to get the performance estimates.

4.3 Training ANN

The ANN model developed from the prepared dataset provides the exact water requirements of crops. The model contains many layers that are connected densely, as well as normalization and dropout layers for improving model stability and reducing overfitting. The ReLU activation function is implemented for the hidden layers, and the output layer is activated with linear activation for considering the regression task. In addition, a kernel regularization term [L 2] is also applied for further reduction of overfitting.

For effective convergence, the Adam optimizer, with a learning rate of 0.0005, was used for the training process. To reduce the prediction error, the Mean Squared Error (MSE) is used as the loss function. During the training process, the option of early stopping and the learning rate scheduling are used. Early stopping based on the validation loss is set to stop training if no improvement occurs for 7 epochs, whereas the learning rate is lowered by 0.3 if no improvement is found after 5 epochs, to guarantee the precision of weight adjustment.

The model is trained for 100 epochs, with a batch size of 32. Split is done; 70% of data is used for training, 15% for validation, and 15% for testing of the model. Therefore, the training and validation loss curves show convergence, as shown in Figure 4.1, confirming the efficiency of model's learning.

The trained model showed very good results as shown by the evaluation results. As shown in Figure 4.2, it is evident that the errors (MAE, MSE and RMSE) on both the training as well as the testing set are low, and that provides precise prediction. The R^2 score also remained high for both the training and the testing sets, ensuring that the model can explain a significant amount of variance in the dataset.

Furthermore, the threshold-based indices, like: Accuracy, Recall, F1-score, and Precision all provide perfect score for both sets, and that confirms reliability of the model. This performance emphasizes the suitability of the model in predicting the irrigation requirements in the (SAIS) with unseen data sets.

Upon the completion of training, the model is saved under the name model.h5 and can be used in determining the required irrigation for the crops in the (SAIS).

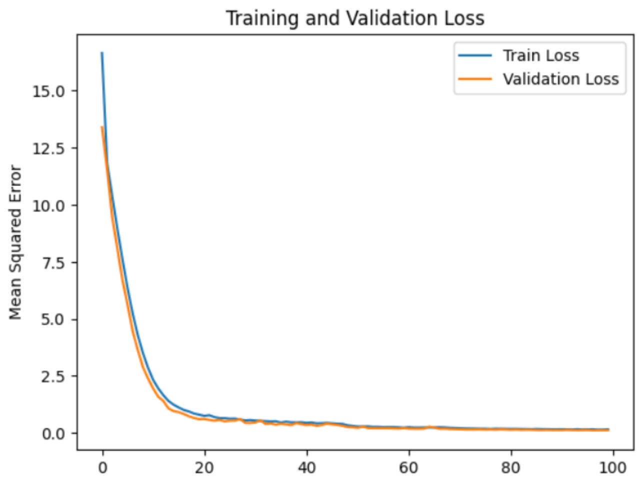
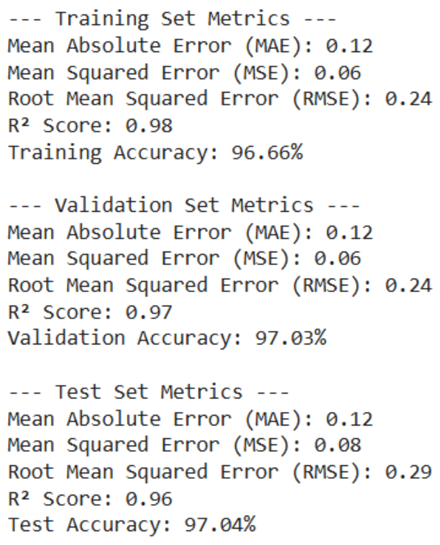


Figure 4.1: Training and validation loss.



```
--- Training Set Metrics ---
Mean Absolute Error (MAE): 0.12
Mean Squared Error (MSE): 0.06
Root Mean Squared Error (RMSE): 0.24
R² Score: 0.98
Training Accuracy: 96.66%

--- Validation Set Metrics ---
Mean Absolute Error (MAE): 0.12
Mean Squared Error (MSE): 0.06
Root Mean Squared Error (RMSE): 0.24
R² Score: 0.97
Validation Accuracy: 97.03%

--- Test Set Metrics ---
Mean Absolute Error (MAE): 0.12
Mean Squared Error (MSE): 0.08
Root Mean Squared Error (RMSE): 0.29
R² Score: 0.96
Test Accuracy: 97.04%
```

Figure 4.2: Evaluation results of model.

The scatter plot in Figure 4.3 evaluates model performance by comparing the actual data set values to the predicted values within the validation data set. The ideal prediction line is shown as the dashed red lines representing the ideal predictions. The actual value measures match the predicted values effectively at scores above 4 but show less accuracy at scores between 0 and 3. A limited number of outliers show cases of excessive and insufficient prediction errors.

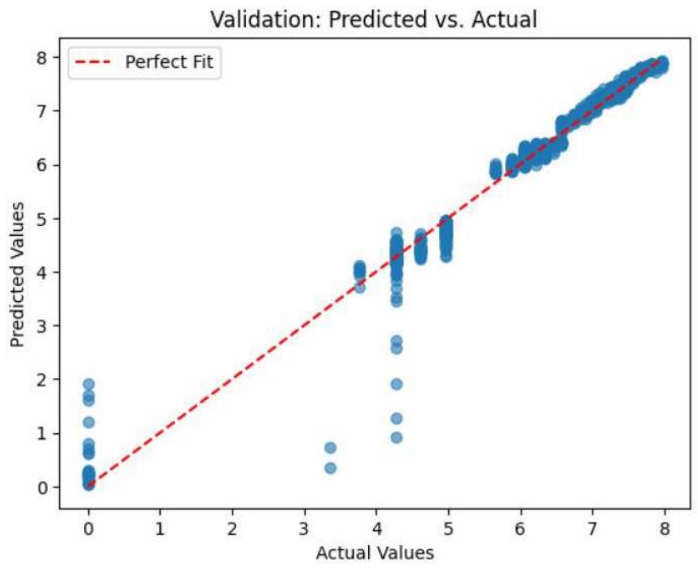


Figure 4.3: Predicted vs actual validation.

The validation set distribution of residuals appears through the histogram Figure 4.4 below which reveals the model prediction precision level. The histogram shows residual differences between actual and predicted values which are displayed along the horizontal axis and how often they occur on the vertical axis. The model predictions demonstrate minimum bias because most residuals cluster around the zero value. The prediction errors show minimal variability which can be seen through the narrow distribution with scarce outliers present. The model demonstrates robust performance because most prediction errors are both small in size and symmetrically arranged, which leads to accurate predictions.

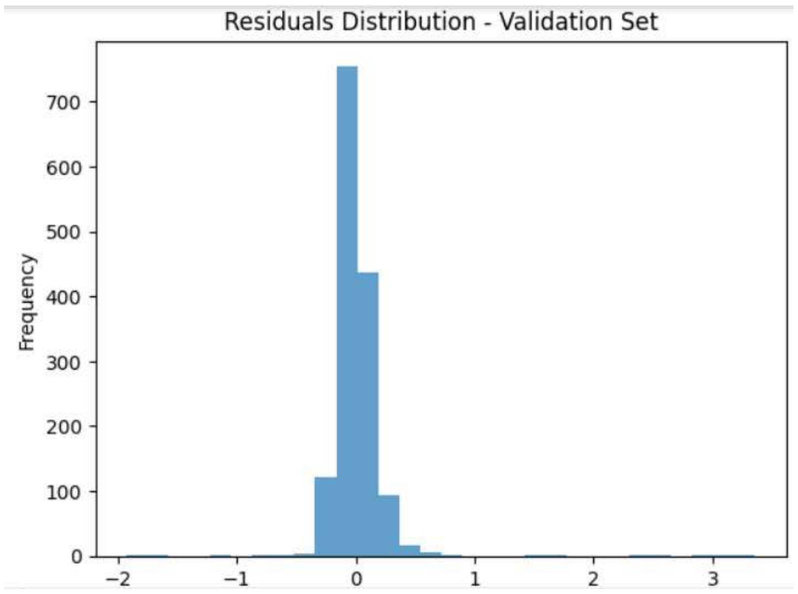


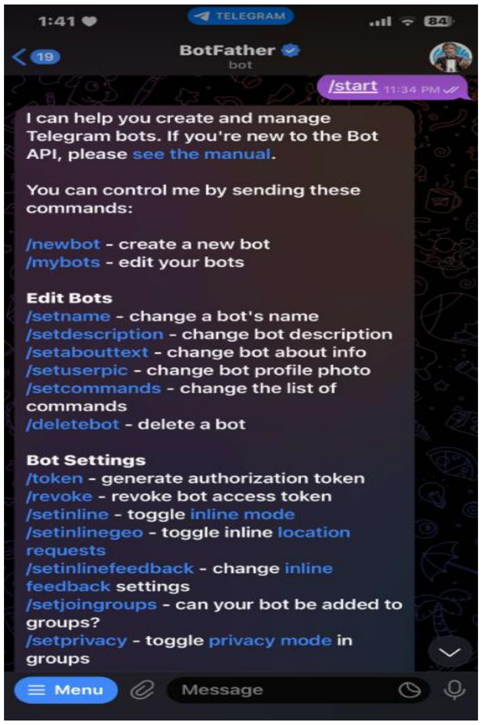
Figure 4.4: Residuals distribution-validation set.

4.4 User Interface (Telegram Bot)

The farmer receives updates through Telegram because people use this platform often. Updates are received directly without need for extra applications or website visits, so farmers can stay in touch and use the software easily.

4.4.1 Creating The Telegram Bot

User begins developing Telegram bots by starting conversations with “BotFather” through the Telegram app, as shown in Figure 4.5. When chatting with “BotFather”, user selects “New Bot” from the options, and then follows the instructions, where user is asked to choose a name and username of the bot. by this, API is ready for use, as shown in Figure 4.6. This API is needed to prove that the bot can connect to Telegram’s main servers.



/start 11:34 PM

I can help you create and manage Telegram bots. If you're new to the Bot API, please see the manual.

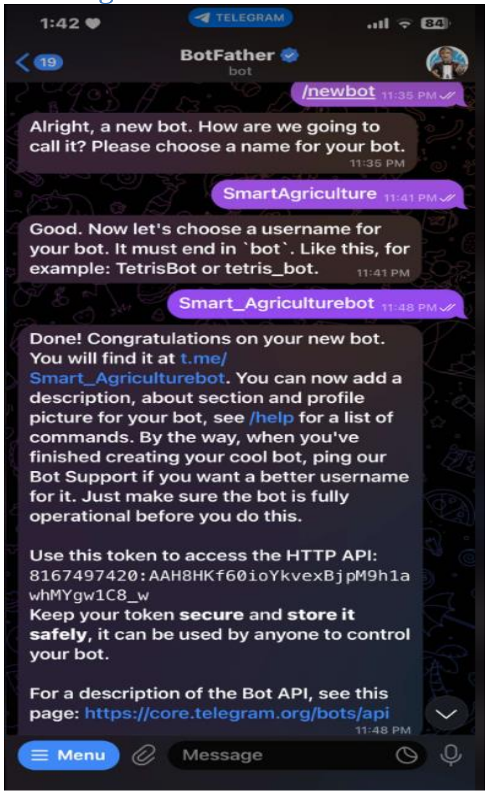
You can control me by sending these commands:

/newbot - create a new bot
/mybots - edit your bots

Edit Bots
/setname - change a bot's name
/setdescription - change bot description
/setabouttext - change bot about info
/setuserpic - change bot profile photo
/setcommands - change the list of commands
/deletebot - delete a bot

Bot Settings
/token - generate authorization token
/revoke - revoke bot access token
/setinline - toggle inline mode
/setinlinegeo - toggle inline location requests
/setinlinefeedback - change inline feedback settings
/setjoingroups - can your bot be added to groups?
/setprivacy - toggle privacy mode in groups

Figure 4.5: BotFather chat.



/newbot 11:35 PM

Alright, a new bot. How are we going to call it? Please choose a name for your bot. 11:35 PM

SmartAgriculture 11:41 PM

Good. Now let's choose a username for your bot. It must end in `bot`. Like this, for example: TetrisBot or tetris_bot. 11:41 PM

Smart_Agriculturebot 11:48 PM

Done! Congratulations on your new bot. You will find it at t.me/Smart_Agriculturebot. You can now add a description, about section and profile picture for your bot, see /help for a list of commands. By the way, when you've finished creating your cool bot, ping our Bot Support if you want a better username for it. Just make sure the bot is fully operational before you do this.

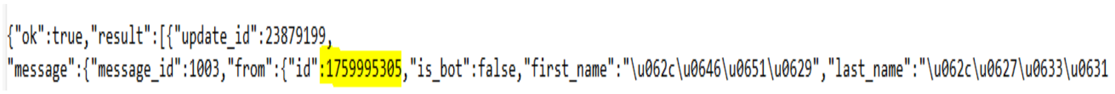
Use this token to access the HTTP API:
8167497420:AAH8HKf60ioYkvexBjpM9h1awhMYgw1C8_w
Keep your token **secure** and **store it safely**, it can be used by anyone to control your bot.

For a description of the Bot API, see this page: https://core.telegram.org/bots/api 11:48 PM

Figure 4.6: Chatting with the BotFather.

4.4.2 Extracting Chat ID

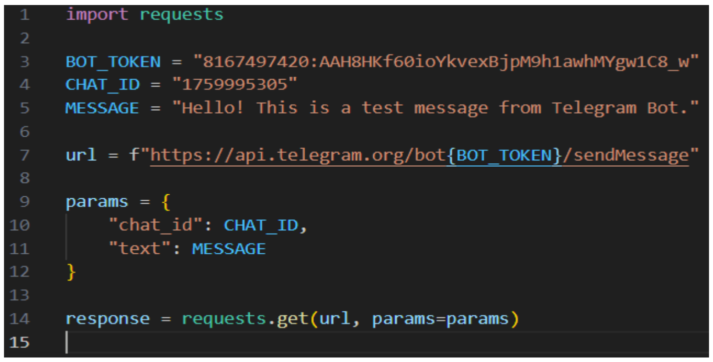
For the bot to communicate with the farmer on Telegram, the chat ID is required. Chat ID is obtained by using the API in the following URL: “https://api.telegram.org/bot<API>/getUpdates”. The API responds when its URL is accessed that provides the unique chat ID. In Figure 4.7, the response shows that chat ID is ‘1759995305’. Using both the Telegram chat ID and the API, the system creates a protected network path that allows the bot to deliver its messages directly to the specified user.



```
{"ok":true,"result":[{"update_id":23879199,
"message":{"message_id":1003,"from":{"id":1759995305,"is_bot":false,"first_name":"\u062c\u0646\u0651\u0629","last_name":"\u062c\u0627\u0633\u0631
```

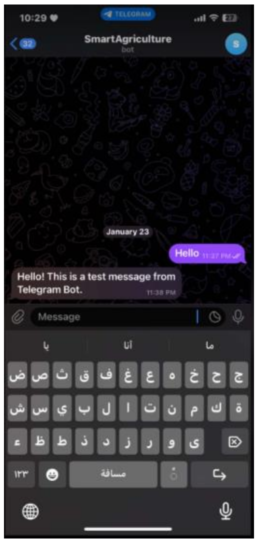
Figure 4.7: Getting chat ID.

This section shows that a message was successfully sent to the conversation ID extracted from section 4.4.2 shown in Figure 4.7 by writing a simple function shown in Figure 4.8



```python
import requests

BOT_TOKEN = "8167497420:AAH8HKf60ioYkvexBjpM9h1awhMYgw1C8_w"
CHAT_ID = "1759995305"
MESSAGE = "Hello! This is a test message from Telegram Bot."

url = f"https://api.telegram.org/bot{BOT_TOKEN}/sendMessage"

params = {
    "chat_id": CHAT_ID,
    "text": MESSAGE
}

response = requests.get(url, params=params)

```

Figure 4.8: Testing send message.



SmartAgriculture
bot
January 23
Hello 11:37 PM
Hello! This is a test message from Telegram Bot. 11:38 PM

Figure 4.9: Succeed sent message.

4.5 LoRa Setup

4.5.1 LoRa Transmitter

To ensure that the LoRa module is ready to act as a transmitter and send packets including sensor readings, the LoRa SX1276RFJAS module was connected to the Arduino UNO board as shown in the figure below in Figure 4.10 and Figure 4.11 shows the schematic diagram of the transmitter according to the connection mechanism followed from Table 4-6. It is ready to send packets. The LoRa transmitter was programmed using the code shown in Appendix A.

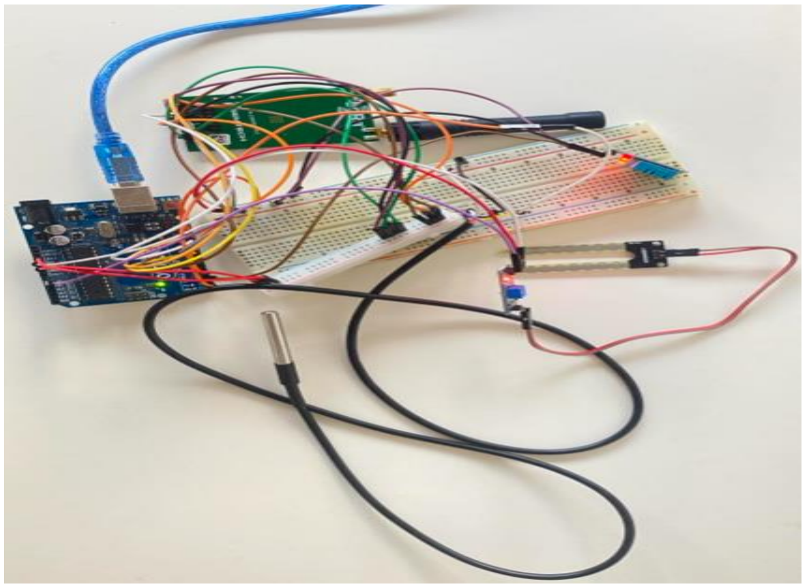


Figure 4.10: LoRa transmitter connection with Arduino and sensors.

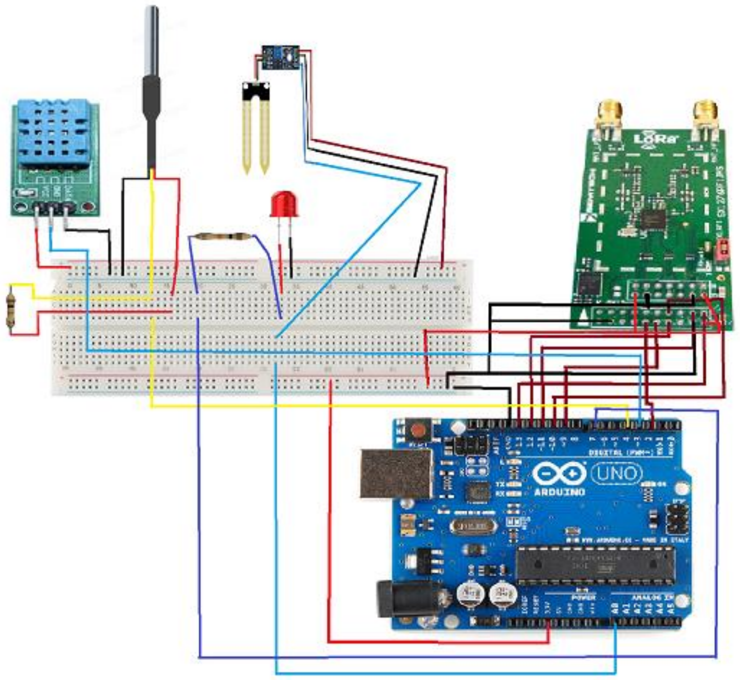


Figure 4.11: LoRa transmitter connection diagram with Arduino and sensors.

Table 4-6: LoRa transmitter and Arduino Uno pins connection.

SX1276RFJAS-Pins	Arduino UNO Pins
RST-10	9
MOSI-3	11
MISO-8	12
SCK-1	13
DIO0-12	2
NSS-7	10
GND-4,24,32	GND (0V)
VDD-2,22,34	VDD (3.3V)
Low Frequency (433MHz) →FEM_CPS-18	GND (0V)

4.5.2 LoRa Receiver

We have configured a LoRa receiver to receive transmitted data packets and sensor readings over a wide and long range with low power consumption. The LoRa SX1276RFJAS module is connected to the Arduino board as shown in the figure below in Figure 4.12 and Figure 4.13 shows the receiver schematic according to the wiring mechanism from Table 4-7 and configured to receive packets. The LoRa receiver uses the code in Appendix B to program the receiver.

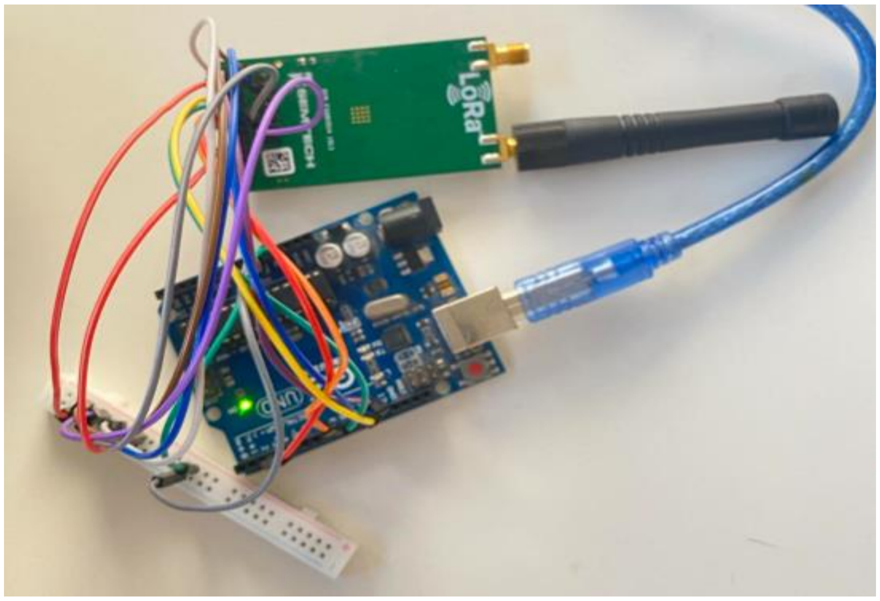


Figure 4.12: LoRa receiver connection with Arduino.

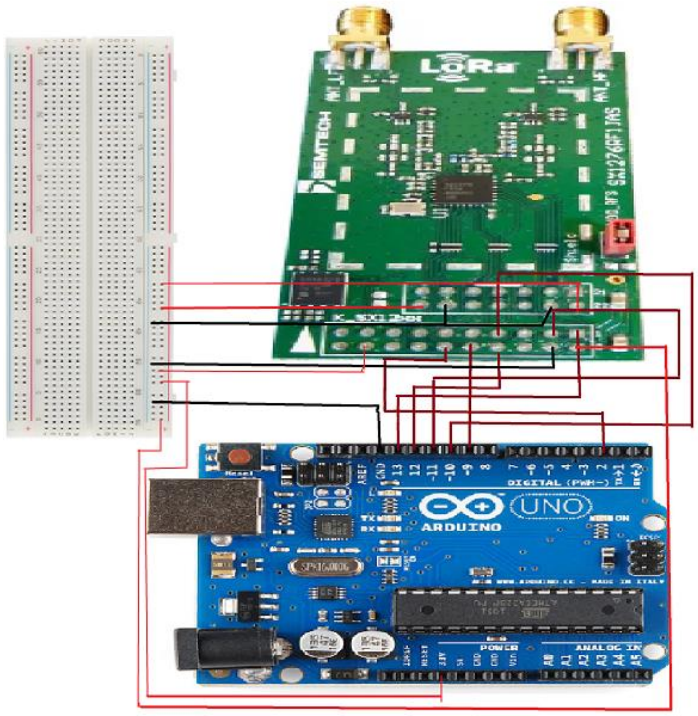


Figure 4.13: LoRa receiver connection diagram with Arduino and sensors.

Table 4-7: LoRa receiver and Arduino Uno pins connection.

SX1276RFJAS-Pins	Arduino UNO Pins
RST-10	9
MOSI-3	11
MISO-8	12
SCK-1	13
DIO0-12	2
NSS-7	10
GND-4,24,32	GND (0V)
VDD-2,22,34	VDD (3.3V)
Low Frequency (433MHz) →FEM_CPS-18	VDD (3.3V)

4.5.3 Sending And Receiving Between The Two LoRa

Last, both Arduino circuits were connected to the computer to bring the chips into the Arduino IDE as demonstrated in Figure 4.14 below and we note the success of the data sending and receiving process between the LoRa sensors.

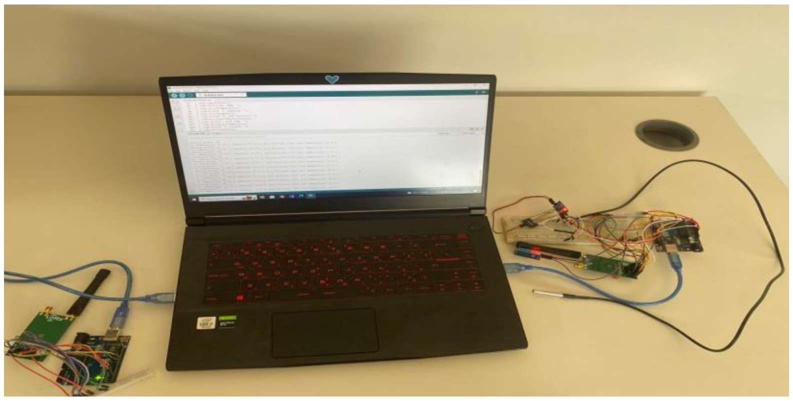
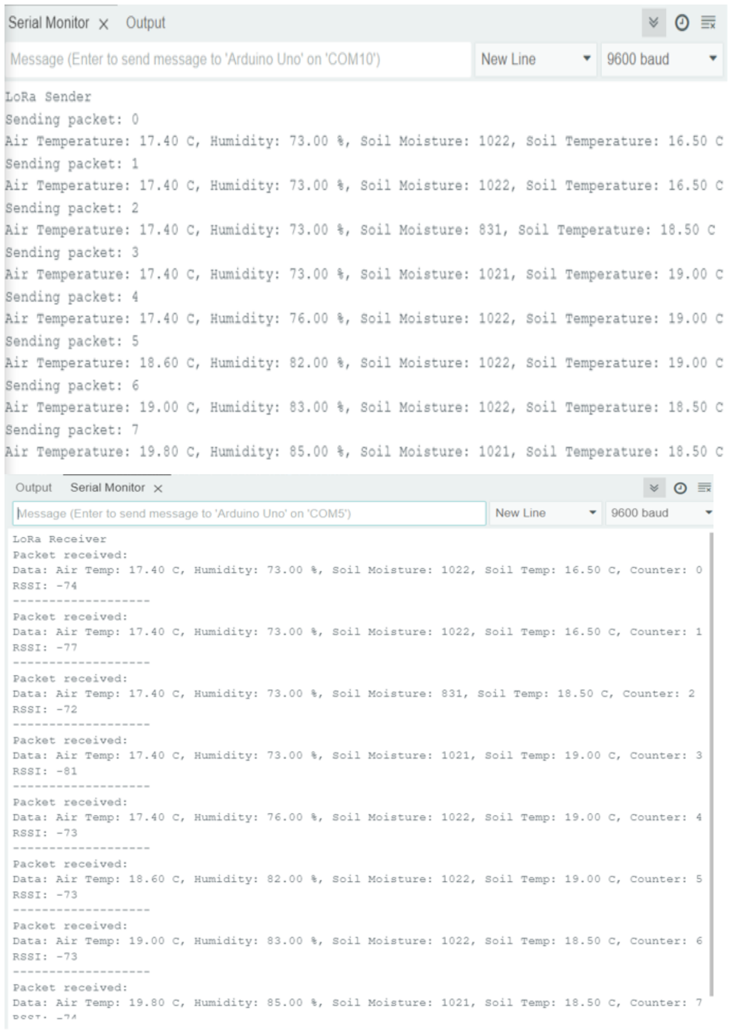


Figure 4.14: LoRa transmitter and receiver circuits.

The figure 4.15 below shows the successful transmission of readings from the sensors from the LoRa transmitter, as well as the successful reception of that data by the second LoRa receiver.



```
LoRa Sender
Sending packet: 0
Air Temperature: 17.40 C, Humidity: 73.00 %, Soil Moisture: 1022, Soil Temperature: 16.50 C
Sending packet: 1
Air Temperature: 17.40 C, Humidity: 73.00 %, Soil Moisture: 1022, Soil Temperature: 16.50 C
Sending packet: 2
Air Temperature: 17.40 C, Humidity: 73.00 %, Soil Moisture: 831, Soil Temperature: 18.50 C
Sending packet: 3
Air Temperature: 17.40 C, Humidity: 73.00 %, Soil Moisture: 1021, Soil Temperature: 19.00 C
Sending packet: 4
Air Temperature: 17.40 C, Humidity: 76.00 %, Soil Moisture: 1022, Soil Temperature: 19.00 C
Sending packet: 5
Air Temperature: 18.60 C, Humidity: 82.00 %, Soil Moisture: 1022, Soil Temperature: 19.00 C
Sending packet: 6
Air Temperature: 19.00 C, Humidity: 83.00 %, Soil Moisture: 1022, Soil Temperature: 18.50 C
Sending packet: 7
Air Temperature: 19.80 C, Humidity: 85.00 %, Soil Moisture: 1021, Soil Temperature: 18.50 C
```

```
LoRa Receiver
Packet received:
Data: Air Temp: 17.40 C, Humidity: 73.00 %, Soil Moisture: 1022, Soil Temp: 16.50 C, Counter: 0
RSSI: -74
-------------------
Packet received:
Data: Air Temp: 17.40 C, Humidity: 73.00 %, Soil Moisture: 1022, Soil Temp: 16.50 C, Counter: 1
RSSI: -77
-------------------
Packet received:
Data: Air Temp: 17.40 C, Humidity: 73.00 %, Soil Moisture: 831, Soil Temp: 18.50 C, Counter: 2
RSSI: -72
-------------------
Packet received:
Data: Air Temp: 17.40 C, Humidity: 73.00 %, Soil Moisture: 1021, Soil Temp: 19.00 C, Counter: 3
RSSI: -81
-------------------
Packet received:
Data: Air Temp: 17.40 C, Humidity: 76.00 %, Soil Moisture: 1022, Soil Temp: 19.00 C, Counter: 4
RSSI: -73
-------------------
Packet received:
Data: Air Temp: 18.60 C, Humidity: 82.00 %, Soil Moisture: 1022, Soil Temp: 19.00 C, Counter: 5
RSSI: -73
-------------------
Packet received:
Data: Air Temp: 19.00 C, Humidity: 83.00 %, Soil Moisture: 1022, Soil Temp: 18.50 C, Counter: 6
RSSI: -73
-------------------
Packet received:
Data: Air Temp: 19.80 C, Humidity: 85.00 %, Soil Moisture: 1021, Soil Temp: 18.50 C, Counter: 7
RSSI: -74
```

Figure 4.15: Sending and receiving data between the two LoRa.

The coming sections will discuss the practical implementation of the project. We will cover the entire process, from monitoring plants on the farm to sending real-time farm status updates and water requirements directly to the farmer’s phone. We will also share implementation results to show how well it works in practice.

4.6 Sensor Node Implementation

Our system starts by placing three sensors in the soil and near the plants on the farm connected to an Arduino board to measure the agricultural conditions Figure 4.16 illustrates this system. Our sensors detect the environmental conditions that affect plant growth by measuring temperature, humidity, soil temperature, and soil moisture; Once the data is collected, it is processed directly on the Arduino board to provide accurate, real-time information about the conditions of interest affecting the plants, which is then ready to be sent to a receiver via LoRa connected to the same Arduino board.

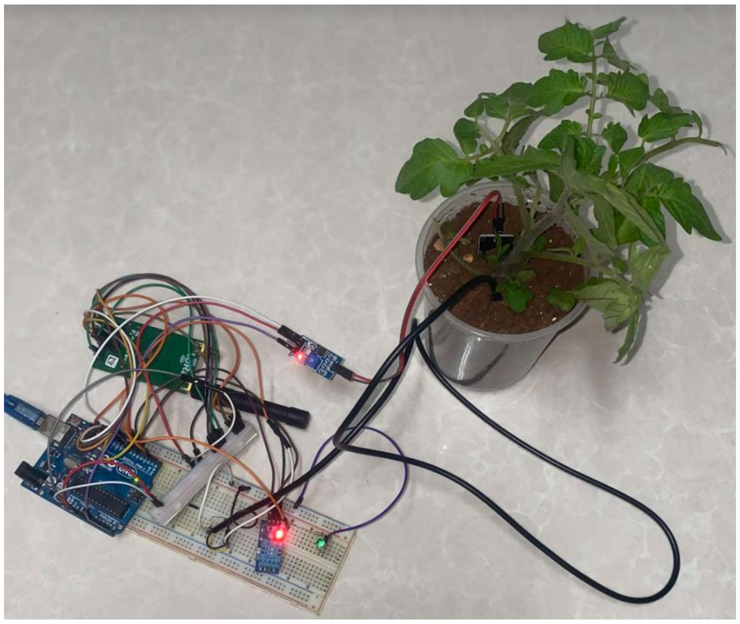
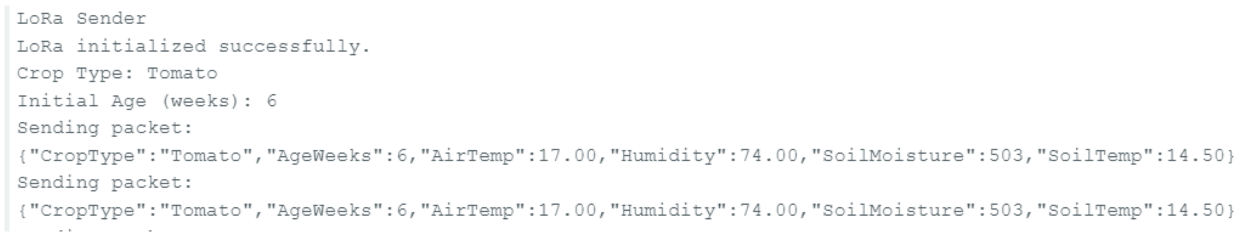


Figure 4.16: Sensors and LoRa connections.

Figure 4.17 shows how the data including sensor readings, plant type and age are sent to the receiver. However, the farmer must first specify the plant type and age in weeks via Arduino when placing the sensors on the plant and then the plant age is updated every week via the code in Appendix C.



```
LoRa Sender
LoRa initialized successfully.
Crop Type: Tomato
Initial Age (weeks): 6
Sending packet:
{"CropType":"Tomato","AgeWeeks":6,"AirTemp":17.00,"Humidity":74.00,"SoilMoisture":503,"SoilTemp":14.50}
Sending packet:
{"CropType":"Tomato","AgeWeeks":6,"AirTemp":17.00,"Humidity":74.00,"SoilMoisture":503,"SoilTemp":14.50}
```

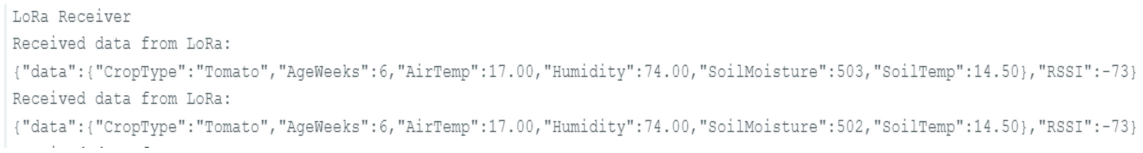
Figure 4.17: Reading data in serial monitor by LoRa transmitter.

The sensor node consists of the following components:

- Temperature sensor DS18B20: used to read the soil temperature with high accuracy.
- Sensor DHT11: responsible for measuring air temperature and humidity.
- Soil moisture sensor YL-69: used to monitor soil moisture levels.
- LoRa transmitter: responsible for transmitting the sensor readings wirelessly to the receiver.
- Arduino Uno: a microcontroller board that processes the sensor data and connects the sensors to the LoRa module, allowing these components to work together efficiently on the farm without requiring a lot of processing power.

4.7 Collector Node

The LoRa transmitter at the farm sends data that reaches the LoRa receiver located at the farmer's site. The illustrated data is presented in Figure 4.18 after receiving information.



```
LoRa Receiver
Received data from LoRa:
{"data":{"CropType":"Tomato","AgeWeeks":6,"AirTemp":17.00,"Humidity":74.00,"SoilMoisture":503,"SoilTemp":14.50},"RSSI":-73}
Received data from LoRa:
{"data":{"CropType":"Tomato","AgeWeeks":6,"AirTemp":17.00,"Humidity":74.00,"SoilMoisture":502,"SoilTemp":14.50},"RSSI":-73}
```

Figure 4.18: Reading data in the serial monitor by LoRa receiver.

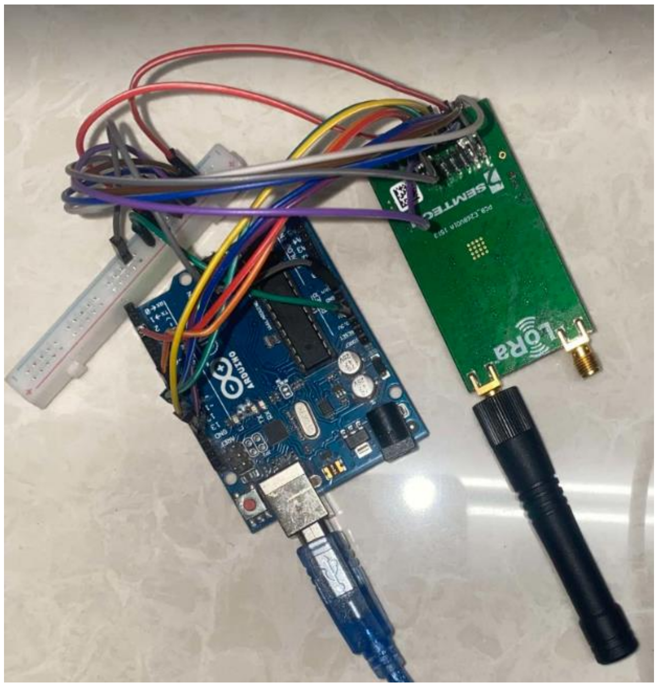
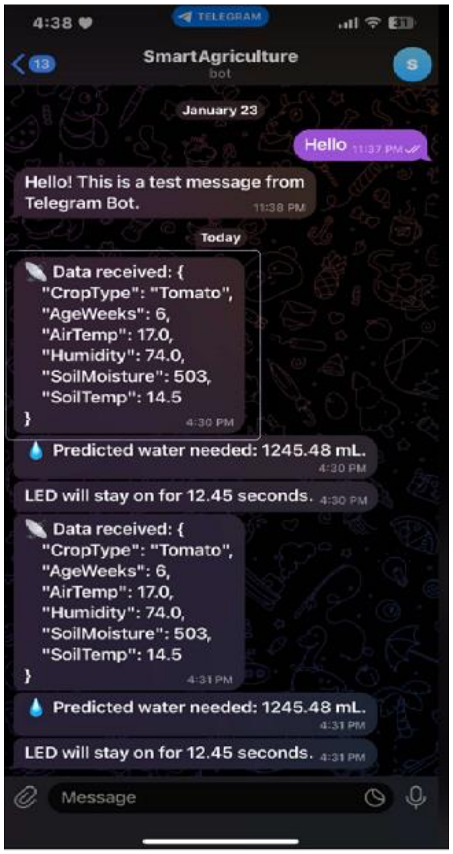


Figure 4.19: LoRa receiver.

The collector node consists of the following components:

- LoRa receiver: The device is responsible for receiving the data from the sensor node transmissions.
- Arduino UNO: It connects the LoRa receiver to the PC and is the processing point without the need for a large processing unit.
- PC: The PC collects data from the LoRa receiver by connecting the Arduino to the serial port on the PC. This data is then sent to the farmer’s phone through Telegram API and chat ID as shown in Figure 4.20. This data is then attached to the ANN for machine learning. This data includes: soil moisture and temperature measurements, temperature and humidity readings of the environment surrounding the plant, and crop species and ages. All this is done by writing Python code in Appendix E.


4:38

TELEGRAM

SmartAgriculture
bot

January 23

Hello 11:37 PM

Hello! This is a test message from Telegram Bot. 11:38 PM

Today

```
Data received: {
  "CropType": "Tomato",
  "AgeWeeks": 6,
  "AirTemp": 17.0,
  "Humidity": 74.0,
  "SoilMoisture": 503,
  "SoilTemp": 14.5
} 4:30 PM
```

💧 Predicted water needed: 1245.48 mL. 4:30 PM

LED will stay on for 12.45 seconds. 4:30 PM

```
Data received: {
  "CropType": "Tomato",
  "AgeWeeks": 6,
  "AirTemp": 17.0,
  "Humidity": 74.0,
  "SoilMoisture": 503,
  "SoilTemp": 14.5
} 4:31 PM
```

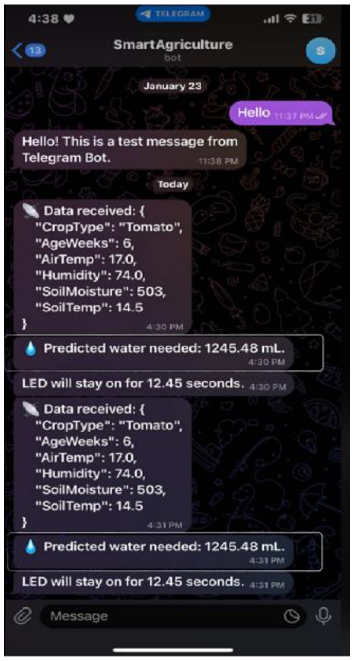
💧 Predicted water needed: 1245.48 mL. 4:31 PM

LED will stay on for 12.45 seconds. 4:31 PM

Figure 4.20: Data readings from telegram.

4.8 Predict The Water Amount

After feeding the data into the model trained to predict the amount of water needed for the plant, it predicts the amount in milliliters and sends it to the farmer's phone through Telegram as shown of the figure 4.21.



Hello 11:37 PM

Hello! This is a test message from Telegram Bot. 11:38 PM

```
Data received: {
  "CropType": "Tomato",
  "AgeWeeks": 6,
  "AirTemp": 17.0,
  "Humidity": 74.0,
  "SoilMoisture": 503,
  "SoilTemp": 14.5
} 4:30 PM
```

💧 Predicted water needed: 1245.48 mL. 4:30 PM

LED will stay on for 12.45 seconds. 4:30 PM

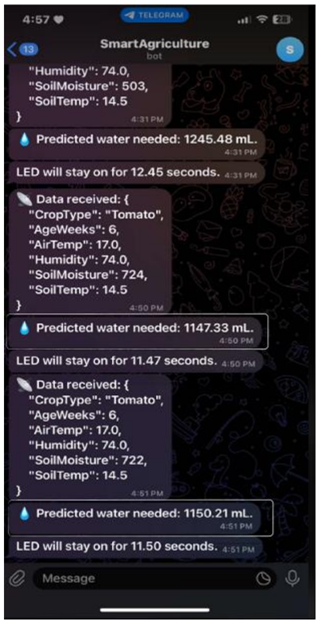
```
Data received: {
  "CropType": "Tomato",
  "AgeWeeks": 6,
  "AirTemp": 17.0,
  "Humidity": 74.0,
  "SoilMoisture": 503,
  "SoilTemp": 14.5
} 4:31 PM
```

💧 Predicted water needed: 1245.48 mL. 4:31 PM

LED will stay on for 12.45 seconds. 4:31 PM

Figure 4.21:Water quantity prediction on telegram.

After some water was added, the new readings obtained showed that the needed water for the plant decreased, as shown in the figure 4.22 below.



```
  "Humidity": 74.0,
  "SoilMoisture": 503,
  "SoilTemp": 14.5
} 4:31 PM
```

💧 Predicted water needed: 1245.48 mL. 4:31 PM

LED will stay on for 12.45 seconds. 4:31 PM

```
Data received: {
  "CropType": "Tomato",
  "AgeWeeks": 6,
  "AirTemp": 17.0,
  "Humidity": 74.0,
  "SoilMoisture": 724,
  "SoilTemp": 14.5
} 4:50 PM
```

💧 Predicted water needed: 1147.33 mL. 4:50 PM

LED will stay on for 11.47 seconds. 4:50 PM

```
Data received: {
  "CropType": "Tomato",
  "AgeWeeks": 6,
  "AirTemp": 17.0,
  "Humidity": 74.0,
  "SoilMoisture": 722,
  "SoilTemp": 14.5
} 4:51 PM
```

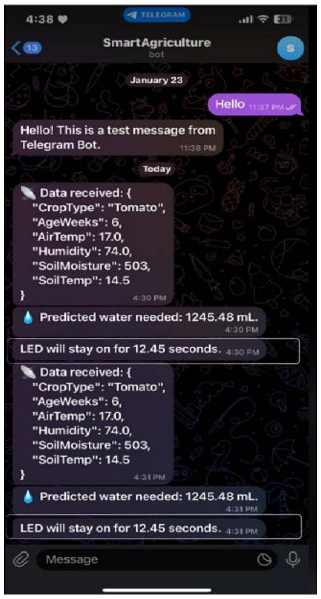
💧 Predicted water needed: 1150.21 mL. 4:51 PM

LED will stay on for 11.50 seconds. 4:51 PM

Figure 4.22: New readings of and predicted water volume after adding a small amount of water.

4.9 Control The Irrigation LED

Then the LED connected to the Arduino in the farm is controlled which will act as an output expressing the period during which the water pump will remain open depending on the amount of water where it was assumed that for every 100 ml of water the LED will remain lit for one second. This is done by sending the duration to keep the LED lit to the Arduino in the farm in addition to sending the duration during which the LED will remain lit to the farmer through telegram as shown in figure 4.23.



```
Hello 11:37 PM

Hello! This is a test message from
Telegram Bot. 11:38 PM

Today

Data received: {
  "CropType": "Tomato",
  "AgeWeeks": 6,
  "AirTemp": 17.0,
  "Humidity": 74.0,
  "SoilMoisture": 503,
  "SoilTemp": 14.5
} 4:30 PM

Predicted water needed: 1245.48 mL. 4:30 PM

LED will stay on for 12.45 seconds. 4:30 PM

Data received: {
  "CropType": "Tomato",
  "AgeWeeks": 6,
  "AirTemp": 17.0,
  "Humidity": 74.0,
  "SoilMoisture": 503,
  "SoilTemp": 14.5
} 4:31 PM

Predicted water needed: 1245.48 mL. 4:31 PM

LED will stay on for 12.45 seconds. 4:31 PM
```

Figure 4.23: The time that the LED will stay on.

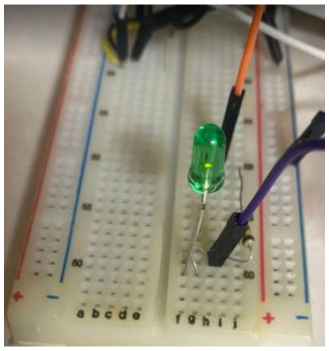


Figure 4.24: LED is ON.

Chapter 5

Conclusion and Future Works

5.1 Conclusion

In conclusion, this project will make farmers' work easier and improve the irrigation process. The SAIS will use real-time data from sensors to automatically supply water to farms, ensuring optimal irrigation efficiency while reducing manual labor and water wastage.

Through this project, a dataset of 10,000 samples was considered, with 70% of the data used for training the system and algorithms. Additionally, 15% was utilized for validation, while another 15% was reserved for testing, ensuring a comprehensive evaluation of the model's performance. The study covered four to five types of plants, where the irrigation requirements were analyzed and optimized based on real-time environmental data.

The system's effectiveness was evaluated using key error metrics, demonstrating high accuracy in irrigation predictions. The results showed a Mean Absolute Error (MAE) of 0.12, Mean Squared Error (MSE) of 0.06, Root Mean Squared Error (RMSE) of 0.24, an R^2 Score of 0.98, and a Training Accuracy of 96.66%. These values highlight the system's precision in estimating water needs and its ability to enhance resource efficiency.

Furthermore, Telegram and LoRa played a significant role in facilitating remote communication, allowing farmers to monitor and control irrigation settings efficiently. The integration of LoRa transceivers enabled seamless data exchange between the sensors and the central control unit, enhancing the system's reliability in areas with limited internet connectivity.

An important aspect of this project is its scalability and adaptability. The system was designed to be modular, allowing for easy expansion to different crops and farming environments. Additionally, the integration of AI models with real-time environmental monitoring ensures continuous optimization of irrigation schedules, adapting to varying climate conditions.

This work demonstrates the potential of AI-driven smart irrigation in optimizing water usage, reducing waste, and improving agricultural efficiency. By continuously refining and scaling the system, it could become an essential tool for sustainable and data-driven farming.

5.2 Future Works

To enhance the performance, reliability, and overall effectiveness of the (SAIS), several crucial steps are planned for future work. Firstly, we will visit farms to assess how well SAIS operates in real-world conditions and gather feedback from farmers on its performance under various weather and soil conditions. The next step will focus on improving the programming code to enhance its efficiency and effectiveness. A key objective is to incorporate and expand the number of wireless sensors in the system. This will enable users to collect more granular environmental data over larger agricultural areas, improving real-time monitoring and decision-making capabilities. The dataset size will also be significantly increased to enhance the accuracy of predictive analytics and irrigation strategies.

To further refine SAIS, an electronic valve will be integrated to enable precise water flow control and automation, optimizing irrigation efficiency. Furthermore, a new AI model incorporating computer vision will be developed to detect plant diseases and adjust irrigation accordingly. To ensure seamless communication, the system's connectivity range will be extended by either adopting an improved version of LoRa or exploring alternative long-range communication technologies. This will involve optimizing antenna design and integrating high-gain directional antennas to enhance signal transmission and reception over longer distances.

Additionally, a mobile app will be developed to allow users to remotely control and monitor the irrigation system, receive live updates, and make real-time adjustments as needed. This feature will provide farmers with greater control over their irrigation processes, enabling them to focus more on crop management. By implementing these advancements, SAIS will evolve into a more flexible, user-friendly, and efficient solution that significantly contributes to sustainable and productive farming.

5.3 Limitations

There was no component that would allow the same LoRa module to handle both sending and receiving, so each LoRa module was specified to either sending or receiving data. Each of the two Arduino boards was connected to the laptop. Since our project achieved success by this method, then it could also work by connecting each Arduino to its own Raspberry Pi, but this would be expensive, so we did not use such a method.

Another limitation was identifying the type and age of the plant. At first, we attempted to use a switch, where each number represented a specific type of plant and its age. However, this way did not work, and we could not find a compatible component to achieve this point. So, we finally decided to set the type and age of the plant directly in the Arduino code prior to placing the sensors next to the plant.

References

[1] Gamal, Yomna, Ahmed Soltan, Lobna A. Said, Ahmed H. Madian, and Ahmed G. Radwan. "Smart irrigation systems: Overview." IEEE Access (2023).

[2] Darshna, S., Sangavi, T., Mohan, S., Soundharya, A., & Desikan, S. (2015). "Smart irrigation system." IOSR Journal of Electronics and Communication Engineering (IOSR-JECE), 10(3), 32-36.

[3] "The Occupation of Water," Amnesty International, accessed June 2024, https://www.amnesty.org/en/latest/campaigns/2017/11/the-occupation-of-water/.

[4] "Water Resources Management Crisis in Palestine," SpringerLink, https://link.springer.com/chapter/10.1007/698_2023_975" , accessed June 2024,

[5] "Israel-Palestine: Water Sharing Conflict," Climate-Diplomacy, accessed June 2024, https://climate-diplomacy.org/case-studies/israel-palestine-water-sharing-conflict.

[6] Keswani, B., Mohapatra, A. G., Mohanty, A., Khanna, A., Rodrigues, J. J., Gupta, D., & De Albuquerque, V. H. C. (2019). "Adapting weather conditions based IOT enabled smart irrigation technique in precision agriculture mechanisms." Neural Computing and Applications, 31, 277-292.

[7] García, L., Parra, L., Jimenez, J. M., Lloret, J., & Lorenz, P. (2020). "IOT-based smart irrigation systems: An overview on the recent trends on sensors and IOT systems for irrigation in precision agriculture." Sensors, 20(4), 1042.

[8] Expansion, environmental impacts of irrigation by 2050 greatly underestimated. Princeton University. Accessed June 2024. https://www.princeton.edu/news/2021/05/04/expansion-environmental-impacts-irrigation-2050-greatly-underestimated.

[9] The future of food in a changing climate. Climate-Diplomacy. Accessed June 2024. https://climate-diplomacy.org/magazine/environment/future-food-changing-climate.

[10] Smart & green: An internet-of-things framework for smart irrigation. Sensors. 20(1), 190. Accessed June 2024. https://link.springer.com/article/10.1007/s00542-018-4079-z.

[11] World Top 20 Global Movement. Accessed June 2024. World Top 20 Global Movement.

[12] J. Wang, Y. Du, W. Niu, J. Han, Y. Li, and P. Yang, "Drip irrigation mode affects tomato yield by regulating root–soil–microbe interactions," Agricultural Water Management, vol. 260, p. 107188, 2022.

[13] Obaideen, K., Yousef, B. A., AlMallahi, M. N., Tan, Y. C., Mahmoud, M., Jaber, H., & Ramadan, M. (2022). An overview of smart irrigation systems using IOT. Energy Nexus, 7, 100124.

[14] Khelifa, B., Amel, D., Amel, B., Mohamed, C., & Tarek, B. (2015, July). Smart irrigation using internet of things. In 2015 Fourth International Conference on future generation communication technology (FGCT) (pp. 1-6). IEEE.

[15] GS Campos, N., Rocha, A. R., Gondim, R., Coelho da Silva, T. L., & Gomes, D. G. (2019). Smart & green: An internet-of-things framework for smart irrigation. Sensors, 20(1), 190.

[16] Boursianis, A. D., Papadopoulou, M. S., Gotsis, A., Wan, S., Sarigiannidis, P., Nikolaidis, S., & Goudos, S. K. (2020). Smart irrigation system for precision agriculture—The AREThOU5A IOT platform. IEEE Sensors Journal, 21(16), 17539-17547.

[17] Vimal, S. P., Kumar, N. S., Kasiselvanathan, M., & Gurumoorthy, K. B. (2021, June). Smart irrigation system in agriculture. In Journal of Physics: Conference Series (Vol. 1917, No. 1, p. 012028). IOP Publishing.

[18] Sushanth, G., & Sujatha, S. (2018, March). IOT based smart agriculture system. In 2018 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET) (pp. 1-4). IEEE.

[19] Ramachandran, V., Ramalakshmi, R., & Srinivasan, S. (2018, November). An automated irrigation system for smart agriculture using the Internet of Things. In 2018 15th International conference on control, automation, robotics and vision (ICARCV) (pp. 210-215). IEEE.

[20] Kamienski, C., Soininen, J. P., Taumberger, M., Dantas, R., Toscano, A., Salmon Cinotti, T., ... & Torre Neto, A. (2019). Smart water management platform: IOT-based precision irrigation for agriculture. Sensors, 19(2), 276.

[21] Abuzanouneh, K. I. M., Al-Wesabi, F. N., Albraikan, A. A., Al Duhayyim, M., Al-Shabi, M., Hilal, A. M., ... & Muthulakshmi, K. (2022). Design of machine learning based smart irrigation System for precision Agriculture. Computers, Materials & Continua, 72(1), 109-124.

[22] Hamdoon, Waseem, and Ahmet Zengin. "A NEW IOT-BASED SMART IRRIGATION MANAGEMENT SYSTEM." Middle East Journal of Science 9, no. 1 (2023): 42-56.

[23] Dataset for Predicting watering the plants. Accessed December 2025. https://www.kaggle.com/datasets/nelakurthisudheer/dataset-for-predicting-watering-the-plants.

[24] Soil Moisture, Air temperature, humidity, and Motor on/off Monitoring data. Accessed December 2025. https://www.kaggle.com/datasets/harshilpatel355/autoirrigationdata.

[25] INTELLIGENT IRRIGATION SYSTEM. Accessed December 2025. https://www.kaggle.com/datasets/harshilpatel355/autoirrigationdata.

[26] Mosa, A., Smith, J., & Brown, R. (2016). Advancements in Smart Irrigation Systems. Journal of Agricultural Technology, 45(3), 123-130.

Appendices

Appendix A LoRa Transmitter Code

```
#include <LoRa.h>
#include <SPI.h>
#include <DHT.h>
#include <OneWire.h>
#include <DallasTemperature.h>

#define DHTPIN 3
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

#define SOIL_MOISTURE_PIN A0

#define ONE_WIRE_BUS 4
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature ds18b20(&oneWire);

int counter = 0;
int Senderled = 5;
int Errorled = 6;

void setup() {
  Serial.begin(9600);
  while (!Serial);

  Serial.println("LoRa Sender");

  pinMode(Senderled, OUTPUT);
  pinMode(Errorled, OUTPUT);
  digitalWrite(Senderled, LOW);
  digitalWrite(Errorled, LOW);

  dht.begin();
  ds18b20.begin();

  if (!LoRa.begin(433E6)) {
    Serial.println("Starting LoRa failed!");
    digitalWrite(Errorled, HIGH);
    delay(2000);
    while (1);
  }
}

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

  int soilMoistureValue = analogRead(SOIL_MOISTURE_PIN);

  ds18b20.requestTemperatures();
  float soilTemperature = ds18b20.getTempCByIndex(0);

  if (isnan(temperature) || isnan(humidity)) {
    Serial.println("Failed to read from DHT sensor!");
    digitalWrite(Errorled, HIGH);
    delay(2000);
    return;
  }

  if (soilTemperature == DEVICE_DISCONNECTED_C) {
    Serial.println("Failed to read from DS18B20 sensor!");
    digitalWrite(Errorled, HIGH);
    delay(2000);
    return;
  }

  digitalWrite(Senderled, LOW);
  digitalWrite(Errorled, LOW);

  Serial.print("Sending packet: ");
  Serial.println(counter);
  Serial.print("Air Temperature: ");
  Serial.print(temperature);
  Serial.print(" C, Humidity: ");
  Serial.print(humidity);
  Serial.print(" %, Soil Moisture: ");
  Serial.print(soilMoistureValue);
  Serial.print(", Soil Temperature: ");
  Serial.print(soilTemperature);
  Serial.println(" C");

  LoRa.beginPacket();
  LoRa.print("Air Temp: ");
  LoRa.print(temperature);
  LoRa.print(" C, Humidity: ");
  LoRa.print(humidity);
  LoRa.print(" %, Soil Moisture: ");
  LoRa.print(soilMoistureValue);
  LoRa.print(", Soil Temp: ");
  LoRa.print(soilTemperature);
  LoRa.print(" C, Counter: ");
  LoRa.print(counter);
  LoRa.endPacket();

  digitalWrite(Senderled, HIGH);
  delay(500);
  digitalWrite(Senderled, LOW);

  counter++;
  delay(2000);
}
```

Appendix B LoRa Receiver Code

```
#include <LoRa.h>
#include <SPI.h>

void setup() {
  Serial.begin(9600);
  while (!Serial);

  Serial.println("LoRa Receiver");

  if (!LoRa.begin(433E6)) {
    Serial.println("Starting LoRa failed!");
    while (1);
  }
}

void loop() {
  int packetSize = LoRa.parsePacket();
  if (packetSize) {
    Serial.println("Packet received:");

    String receivedData = "";
    while (LoRa.available()) {
      receivedData += (char)LoRa.read();
    }

    Serial.print("Data: ");
    Serial.println(receivedData);

    Serial.print("RSSI: ");
    Serial.println(LoRa.packetRssi());
    Serial.println("-------------------");

    delay(2000);
  }
}
```

Appendix C Final LoRa Sender Code

```
#include <LoRa.h>
#include <SPI.h>
#include <DHT.h>
#include <OneWire.h>
#include <DallasTemperature.h>

// DHT11 Sensor
#define DHTPIN 3
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

// YL-69 Sensor
#define SOIL_MOISTURE_PIN A0

// DS18B20 Sensor
#define ONE_WIRE_BUS 4
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature ds18b20(&oneWire);

// LoRa Pins and Settings
int counter = 0;
int Senderled = 7;

// Variables for crop details
String cropType = "Tomato";
int ageWeeks = 6;
unsigned long lastUpdate = 0;
unsigned long lastMessageTime = 0;
const unsigned long updateInterval = 7 * 24 * 60 * 60 * 1000;
const unsigned long messageInterval = 60000;
const unsigned long sensorReadInterval = 60000;
unsigned long lastSensorReadTime = 0;

float temperature = 0;
float humidity = 0;
int soilMoistureValue = 0;
float soilTemperature = 0;

void setup() {
  Serial.begin(9600);
  while (!Serial);
  Serial.println("LoRa Sender");

  pinMode(Senderled, OUTPUT);
  pinMode(Errorled, OUTPUT);
  digitalWrite(Senderled, LOW);
  digitalWrite(Errorled, LOW);

  // Initialize sensors
  dht.begin();
  ds18b20.begin();

  // Initialize LoRa
  if (!LoRa.begin(433E6)) {
    Serial.println("Starting LoRa failed!");
    digitalWrite(Errorled, HIGH);
    delay(2000);
    while (1);
  }

  Serial.println("LoRa initialized successfully.");
  Serial.print("Crop Type: ");
  Serial.println(cropType);
  Serial.print("Initial Age (weeks): ");
  Serial.println(ageWeeks);
}

void loop() {
  unsigned long currentMillis = millis();

  // Update age every 7 days
  if (currentMillis - lastUpdate >= updateInterval) {
    ageWeeks++;
    lastUpdate = currentMillis;
    Serial.println("Age updated automatically by 1 week.");
  }

  // Take sensor readings and send data every 1 minute
  if (currentMillis - lastSensorReadTime >= sensorReadInterval) {
    lastSensorReadTime = currentMillis;

    temperature = dht.readTemperature();
    humidity = dht.readHumidity();

    soilMoistureValue = 1023 - analogRead(SOIL_MOISTURE_PIN);
    ds18b20.requestTemperatures();
    soilTemperature = ds18b20.getTempCByIndex(0);

    if (isnan(temperature) || isnan(humidity)) {
      Serial.println("Failed to read from DHT sensor!");
      digitalWrite(Errorled, HIGH);
      delay(2000);
      return;
    }

    if (soilTemperature == DEVICE_DISCONNECTED_C) {
      Serial.println("Failed to read from DS18B20 sensor!");
      digitalWrite(Errorled, HIGH);
      delay(2000);
      return;
    }

    String payload = "{";
    payload += "\"CropType\":\"" + cropType + "\",";
    payload += "\"AgeWeeks\":" + String(ageWeeks) + ",";
    payload += "\"AirTemp\":" + String(temperature, 2) + ",";
    payload += "\"Humidity\":" + String(humidity, 2) + ",";
    payload += "\"SoilMoisture\":" + String(soilMoistureValue) + ",";
    payload += "\"SoilTemp\":" + String(soilTemperature, 2) + "}";

    Serial.println("Sending packet:");
    Serial.println(payload);

    LoRa.beginPacket();
    LoRa.print(payload);
    LoRa.endPacket();

    digitalWrite(Senderled, HIGH);
    delay(500);
    digitalWrite(Senderled, LOW);

    counter++;
  }

  if (Serial.available() > 0) {
    String receivedData = Serial.readString();

    int irrigationDuration = receivedData.toInt();

    if (irrigationDuration > 0) {
      digitalWrite(Senderled, HIGH);
      Serial.println("Irrigation triggered: LED on");
      delay(irrigationDuration);
      digitalWrite(Senderled, LOW);
      Serial.println("Irrigation completed: LED off");
    } else {
      digitalWrite(Senderled, LOW);
      Serial.println("No irrigation needed: LED off");
    }
  }
}
```

Appendix D Final LoRa Receiver Code

```
#include <LoRa.h>
#include <SPI.h>

void setup() {
  Serial.begin(9600);
  while (!Serial);

  Serial.println("LoRa Receiver");

  if (!LoRa.begin(433E6)) {
    Serial.println("Starting LoRa failed!");
    while (1);
  }
}

void loop() {
  int packetSize = LoRa.parsePacket();
  if (packetSize) {
    String receivedData = "";

    while (LoRa.available()) {
      receivedData += (char)LoRa.read();
    }

    String message = "{";
    message += "\"data\":" + receivedData + ",";
    message += "\"RSSI\":" + String(LoRa.packetRssi());
    message += "}";

    Serial.println("Received data from LoRa:");
    Serial.println(message);

    delay(2000);
  }
}
```

Appendix E ANN

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import RobustScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score, precision_score, recall_score, f1_score
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization
from tensorflow.keras.regularizers import l2
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
import math
import matplotlib.pyplot as plt

file_path = '/content/Dataa.csv'
data = pd.read_csv(file_path)

data['Exact_Water_Needed'] = np.log(data['Exact_Water_Needed'] + 1)

# One-hot encode Crop_Type
data = pd.get_dummies(data, columns=['Crop_Type'], drop_first=True)

X = data.drop(columns=['Exact_Water_Needed'])
y = data['Exact_Water_Needed']

# Split the data into training, validation, and testing sets
X_train, X_temp, y_train, y_temp = train_test_split(X, y,
test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp,
test_size=0.5, random_state=42)

scaler = RobustScaler()
X_train = scaler.fit_transform(X_train)
X_val = scaler.transform(X_val)
X_test = scaler.transform(X_test)

model = Sequential([
    Dense(1024, activation='relu', kernel_regularizer=l2(0.01),
input_shape=(X_train.shape[1],)),
    BatchNormalization(),
    Dropout(0.4),
    Dense(512, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    Dropout(0.4),
    Dense(256, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    Dropout(0.3),
    Dense(128, activation='relu', kernel_regularizer=l2(0.01)),
    Dense(1)  # Output layer
])

model.compile(optimizer=Adam(learning_rate=0.0005), loss='mse',
metrics=['mae'])

early_stopping = EarlyStopping(monitor='val_loss', patience=7,
restore_best_weights=True, verbose=1)
lr_scheduler = ReduceLROnPlateau(monitor='val_loss', factor=0.3,
patience=5, min_lr=1e-6, verbose=1)

history = model.fit(
    X_train, y_train,
    epochs=100,
    batch_size=32,
    validation_data=(X_val, y_val),
    verbose=1,
    callbacks=[early_stopping, lr_scheduler]
)

y_train_pred = model.predict(X_train).flatten()

mse_train = mean_squared_error(y_train, y_train_pred)
rmse_train = math.sqrt(mse_train)
mae_train = mean_absolute_error(y_train, y_train_pred)
r2_train = r2_score(y_train, y_train_pred)

y_val_pred = model.predict(X_val).flatten()

mse_val = mean_squared_error(y_val, y_val_pred)
rmse_val = math.sqrt(mse_val)
mae_val = mean_absolute_error(y_val, y_val_pred)
r2_val = r2_score(y_val, y_val_pred)

def calculate_accuracy(y_true, y_pred, tolerance=0.10):
    y_true = np.array(y_true)
    y_pred = np.array(y_pred)
    within_tolerance = np.abs((y_true - y_pred) / y_true) <= tolerance
    accuracy = np.mean(within_tolerance)
    return accuracy

training_accuracy = calculate_accuracy(y_train, y_train_pred)
validation_accuracy = calculate_accuracy(y_val, y_val_pred)

y_pred = model.predict(X_test).flatten()

mse_test = mean_squared_error(y_test, y_pred)
rmse_test = math.sqrt(mse_test)
mae_test = mean_absolute_error(y_test, y_pred)
r2_test = r2_score(y_test, y_pred)

test_accuracy = calculate_accuracy(y_test, y_pred)

print(f"\n--- Training Set Metrics ---")
print(f"Mean Absolute Error (MAE): {mae_train:.2f}")
print(f"Mean Squared Error (MSE): {mse_train:.2f}")
print(f"Root Mean Squared Error (RMSE): {rmse_train:.2f}")
print(f"R² Score: {r2_train:.2f}")
print(f"Training Accuracy: {training_accuracy * 100:.2f}%")

print(f"\n--- Validation Set Metrics ---")
print(f"Mean Absolute Error (MAE): {mae_val:.2f}")
print(f"Mean Squared Error (MSE): {mse_val:.2f}")
print(f"Root Mean Squared Error (RMSE): {rmse_val:.2f}")
print(f"R² Score: {r2_val:.2f}")
print(f"Validation Accuracy: {validation_accuracy * 100:.2f}%")

print(f"\n--- Test Set Metrics ---")
print(f"Mean Absolute Error (MAE): {mae_test:.2f}")
print(f"Mean Squared Error (MSE): {mse_test:.2f}")
print(f"Root Mean Squared Error (RMSE): {rmse_test:.2f}")
print(f"R² Score: {r2_test:.2f}")
print(f"Test Accuracy: {test_accuracy * 100:.2f}%")

for epoch, (train_loss, val_loss) in
enumerate(zip(history.history['loss'], history.history['val_loss']),
start=1):
    print(f"Epoch {epoch}: Train Loss: {train_loss:.4f}, Validation
Loss: {val_loss:.4f}")

plt.figure()
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Mean Squared Error')
plt.legend()
plt.show()

training_accuracies = [calculate_accuracy(y_train,
model.predict(X_train).flatten()) for epoch in
range(len(history.history['loss']))]
validation_accuracies = [calculate_accuracy(y_val,
model.predict(X_val).flatten()) for epoch in
range(len(history.history['val_loss']))]

plt.figure()
plt.plot(training_accuracies, label='Train Accuracy')
plt.plot(validation_accuracies, label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

plt.figure()
plt.scatter(y_val, y_val_pred, alpha=0.6)
plt.title('Validation: Predicted vs. Actual')
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.plot([min(y_val), max(y_val)], [min(y_val), max(y_val)],
color='red', linestyle='--', label='Perfect Fit')
plt.legend()
plt.show()

plt.figure()
residuals = y_val - y_val_pred
plt.hist(residuals, bins=30, alpha=0.7)
plt.title('Residuals Distribution - Validation Set')
plt.xlabel('Residuals')
plt.ylabel('Frequency')
plt.show()
```

Appendix F Final Python Code

```python
import serial
import time
import requests
import json
import numpy as np
import pandas as pd
from sklearn.preprocessing import RobustScaler
import tensorflow as tf
from tensorflow.keras.metrics import MeanSquaredError
import re

custom_objects = {"mse": MeanSquaredError()}

model = tf.keras.models.load_model('D:/Download/New/PRO/model.h5',
custom_objects=custom_objects)
scaler = RobustScaler()
scaler_data = pd.read_csv('D:/Download/New/PRO/Dataa.csv')

if 'Crop_Type' in scaler_data.columns:
    scaler_data = pd.get_dummies(scaler_data, columns=['Crop_Type'],
drop_first=True)
if 'Crop_Type_cucumber' not in scaler_data.columns:
    scaler_data['Crop_Type_cucumber'] = 0

X = scaler_data[['Age_Weeks', 'Soil_Moisture_YL69', 'Air_Temperature_DHT11',
'Humidity_DHT11', 'Soil_Temperature_DS18B20'] +
                [col for col in scaler_data.columns if 'Crop_Type' in col]]
scaler.fit(X)
model_input_columns = X.columns.tolist()

BOT_TOKEN = "8167497420:AAH8HKf60ioYkvexBjpM9h1awhMYgw1C8_w"
CHAT_ID = "1759995305"
TELEGRAM_URL = f"https://api.telegram.org/bot{BOT_TOKEN}/sendMessage"
TELEGRAM_GET_URL = f"https://api.telegram.org/bot{BOT_TOKEN}/getUpdates"

SERIAL_PORT_RECEIVER = "COM5"
ser_receiver = serial.Serial(SERIAL_PORT_RECEIVER, 9600, timeout=1)
ser_arduino = serial.Serial("COM10", 9600, timeout=1)

def send_to_telegram(message):
    """Send a message to the Telegram bot."""
    payload = {"chat_id": CHAT_ID, "text": message, "parse_mode": "Markdown"}
    response = requests.post(TELEGRAM_URL, data=payload)
    if response.status_code == 200:
        print("Message sent to Telegram.")
    else:
        print(f"Failed to send message. Error: {response.text}")

def is_valid_json(data):
    """Quick validation to check if a string is a JSON-like structure."""
    return data.startswith("{") and data.endswith("}")

def prepare_features(data_dict):
    """Prepare model features from data dictionary."""
    age_weeks = float(data_dict.get("AgeWeeks", 0))
    soil_moisture = float(data_dict.get("SoilMoisture", 0))
    air_temp = float(data_dict.get("AirTemp", 0))
    humidity = float(data_dict.get("Humidity", 0))
    soil_temp = float(data_dict.get("SoilTemp", 0))
    crop_type = data_dict.get("CropType", "").replace(" ", "_").lower()

    crop_one_hot = [0] * (len(model_input_columns) - 5)
    if f"Crop_Type_{crop_type}" in model_input_columns:
        crop_one_hot_index = model_input_columns.index(f"Crop_Type_{crop_type}")
        crop_one_hot[crop_one_hot_index - 5] = 1

    return [age_weeks, soil_moisture, air_temp, humidity, soil_temp] +
crop_one_hot

def read_from_receiver():
    """Read data from the LoRa receiver and send it to Telegram and Arduino."""
    if ser_receiver.in_waiting > 0:
        received_data = ser_receiver.readline().decode('utf-8').strip()
        print(f"Received data from LoRa receiver: {received_data}")

        if not is_valid_json(received_data):
            print("Data is not JSON-like. Skipping this message.")
            return

        try:
            outer_data = json.loads(received_data)
            nested_data = outer_data.get("data", outer_data)
            print(f"Nested data: {nested_data}")

            if isinstance(nested_data, dict):
                message = f"📡 Data received: {json.dumps(nested_data,
indent=4)}"
                send_to_telegram(message)

                crop_type = nested_data.get("CropType", "").replace(" ",
"_").lower()

                user_data = prepare_features(nested_data)

                user_data_df = pd.DataFrame([user_data],
columns=model_input_columns)

                user_data_scaled = scaler.transform(user_data_df)
                user_data_scaled = user_data_scaled[:, :model.input_shape[-1]]
                print(f"Scaled data: {user_data_scaled}")

                water_needed_log = model.predict(user_data_scaled).flatten()[0]
                water_needed_ml = np.exp(water_needed_log) - 1
                print(f"Predicted water needed: {water_needed_ml:.2f} mL.")

                send_to_telegram(f"💧 Predicted water needed:
{water_needed_ml:.2f} mL.")

                duration_ms = int(water_needed_ml * 10)
                ser_arduino.write(f"{duration_ms}\n".encode())
                send_to_telegram(f"LED will stay on for {duration_ms / 1000:.2f}
seconds.")
        except json.JSONDecodeError as e:
            print(f"Error decoding the message: {e}")
        except Exception as e:
            print(f"Unexpected error: {e}")

def main():
    """Main script execution for both transmitting and receiving."""
    try:
        while True:
            read_from_receiver()
            time.sleep(1)
    except KeyboardInterrupt:
        print("Program stopped manually.")
    finally:
        ser_receiver.close()
        ser_arduino.close()
        print("Serial connections closed.")
```

```python
if __name__ == "__main__":
    main()
```