DEVELOPMENT OF AN IoT-BASED SMART IRRIGATION SYSTEM FOR EFFICIENT WATER MANAGEMENT IN UASIN GISHU COUNTY

Winny Tuitoek Bundotich

A Project Report Submitted in Partial Fulfilment of the Requirements for the Degree of Master of Science in Embedded and Mobile Systems of the Nelson Mandela African Institution of Science and Technology

Arusha, Tanzania

August, 2024

ABSTRACT

Agriculture is the backbone of Kenya's economy, with Uasin Gishu county being the country's breadbasket. Food scarcity has recently increased due to climate change, population growth, and decreased land available for farming. Several measures have been implemented to mitigate food scarcity, including encouraging irrigation farming to ensure whole-year food production. However, irrigation practice faces a water scarcity challenge. Efforts have been put in place to improve water use efficiency. These measures include scheduled irrigation system technology. Most of these scheduled systems target greenhouse irrigation and leave out open-field irrigation farmers where rainfall is a factor in reducing water wastage whenever there is rainfall. This technology does not provide a precision of plant water needs and poses a risk of under or over-irrigation. Therefore, to overcome these challenges, this project developed an IoT-based system with sensors to monitor critical soil parameter measurements continuously. The main objective of this project was to develop an IoT-based smart irrigation system for efficient water management. An ESP32 microcontroller board is used to process information collected from the sensors. Openweather API is implemented on the Thingsboard cloud platform to fetch rainfall prediction information. While providing remote valve control, the system uses soil moisture level and rainfall prediction parameters to automatically control the irrigation valves. The system also offers rich farm information visualization through the Thingsboard cloud platform dashboard, which can be accessed remotely through the Thingsboard live mobile application, where a user can control the irrigation valves remotely. Mixed methods which involved questionnaires and focus group discussions was used to collect data from sixteen respondents. Purposve sampling was also used to identify the respondents during data collection. To develop the system,agile software development methodology, specifically extreme programming (XP) was implemented. To validate the system's functionalities, the system was demonstrated to twenty seven people and thereafter were allowed to interact with the system. A questionnaire was implemented to get feedback from the respondents. Generally, respondents agreed that the system satisfactorily met their needs. The developed system contributes to the value chain by providing precise water input. The system can be advanced to include other essential features needed to monitor and evaluate irrigated farms within Uasin Gishu county and any other region.

DECLARATION

I, Winny Tuitoek Bundotich, declare to the Senate of Nelson Mandela African Institution of Science and Technology that this project report is my original work and that it has never been submitted nor is being concurrently submitted for degree award in any other institution.

Winny Tuitoek Bundotich		21.08.2022
Name of Candidate	**Signature**	**Date**

The above declaration is confirmed by:

Dr. Ramadhani Sinde		21-08-2024
Name of Supervisor 1	**Signature**	**Date**

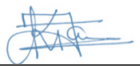
Dr. John K. Tarus		21.08.2024
Name of Supervisor 2	**Signature**	**Date**

COPYRIGHT

CERTIFICATION

The undersigned certify that they have read and hereby recommend for acceptance by the Nelson Mandela African Institution of Science and Technology, a project report titled **"An IoT-Based Smart Irrigation System for Efficient Water Management in Uasin Gishu County"** in partial fulfillment of the requirements for the degree of Master of Science in Embedded and Mobile Systems of the Nelson Mandela African Institution of Science and Technology.

Dr. Ramadhani Sinde		21-08-2024
Name of Supervisor 1	**Signature**	**Date**

Dr. John K. Tarus		21.08.2024
Name of Supervisor 2	**Signature**	**Date**

ACKNOWLEDGEMENTS

First and foremost, I thank God for everything it took to complete this project successfully. I appreciate the Centre of Excellence for ICT in East Africa (CENIT@EA) and DAAD GIZ's scholarship program for funding the project throughout its implementation.

My most sincere gratitude to my supervisors Dr. Ramadhani Sinde and Dr. John Tarus, for their generous support and patience during the entire Project implementation and the preparation of this dissertation.

I wish to pass my sincere appreciation to my industrial supervisor Mr. Titus Kimaiyo at Uasin Gishu county government for his support throughout the implementation of the project.

Special appreciation to my parents, spouse, and the entire family for their understanding, moral support, and continuous prayers.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

LIST OF ABBREVIATIONS AND SYMBOLS

ADC	Analog-to-Digital Converter
AES	Advanced Encryption Standard
AES-CCM	Advanced Encryption Standard- Cipher block chaining - Message authentication code
API	Application Programming Interface
ARM	Advanced RISC Machine
CAN	Controller Area Network
CIDP	County Integrated Development Plan
CMD	Command Prompt
CoAP	Constrained Application Protocol
COM	Component Object Model
CSS	Chirp Spread Spectrum
DAC	Digital Analog Converter
DIO	Digital Input/Output
DSI	Display Serial Interface
ECC	Excise Control Code
EEWMP	Energy-Efficient Water Management Platform
FAO	Food and Agriculture Organization
G-Code	Geometric Code
GDP	Gross Domestic Product
GND	Ground
GoK	Government of Kenya
GPIO	General Purpose Input/output
GSM	Global System for Mobile Communication
HCI	Human-Computer Interaction
HMAC	Hash-based Message Authentication Code
HTTP	Hypertext Transfer Protocol
I2C	Inter-Integrated Circuit

I2S	Inter-IC Sound
IC	Integrated Circuit
IDE	Integrated Development Environment
IoT	Internet of Things
IR	Infrared Radiation
JSON	JavaScript Object Notation
KCSAP	Kenya Climate-Smart Agriculture Project
LDO	Low-Dropout
LDR	Light Dependent Resistor
LoRa	Long Range
LoRaWAN	Long Range Wide Area Network
MALF	Ministry of Agriculture, Livestock, and Fisheries
MCU	Microcontroller Unit
MIME	Multipurpose Internet Mail Extensions
MISO	Slave In, Master Out
MOSI	Master Out, Slave In
MQTT	Message Queueing Telemetry Transport
NSS	Negative Slave Select
OLED	Organic Light-Emitting Diodes
PCB	Printed Circuit Board
PLA	Polylactic Acid
PWM	Pulse-Width Modulation
RAM	Random Access Memory
REST API	Representational State Transfer Application Programming Interface
RH	Relative Humidity
ROM	Read Only Memory
RPC	Remote Procedural Calls
RSA	Algorithm Rivest-Shamir-Adleman Algorithm
RSSI	Received Signal Strength Indication

RST	Reset
RTOS	Real-Time Operating System
SCK	Serial Clock Line
SD-Card	Secure Digital- Card
SDHC	Secure Digital High Capacity
SDIO	Secure Digital Input Output
SDIOSI	Secure Digital Input Output
SHA	Secure Hash Algorithm
SMD	Surface Mount Devices
SNR	Signal Noise Ratio
SPI	Serial Peripheral Interface
SVG	Scalable Vector Graphics
SWAMP	Smart Water Management Platform
TAM	Technology Acceptance Model
TCP/IP	Transmission Control Protocol/Internet Protocol
TSL	Thingsboard Scripting Language
TTL	Time To Live
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus
VCC	Voltage Common Collector
VDD	Voltage (at) Drain
WIFI	Wireless Fidelity
WPA	Wi-Fi Protected Access
XP	Extreme Programming

CHAPTER ONE

INTRODUCTION

1.1 Background of the Problem

In the recent past, the use of technology in agriculture has been on the rise. Through technology, the agricultural sector activities have involved superior farm machines, significantly reducing time and labor costs. Besides special farm machinery, advancement in technology has also seen the implementation of precision agriculture, where farmers can manage the variability of natural factors and time. Internet of Things technology (IoT), which is a network of devices communicating with each other and the cloud, has also been adopted to improve communication and minimize manual operations (Aggarwal & Singh, 2020). Through precision farming and the Internet of Things, farmers have witnessed improved yields.

The world population is projected to increase by 33% between 2017 and 2030 (World Population Prospects, 2017). This population increase translates to an expected increase in food production to feed the world's population. Several measures have been put in place to ensure the availability of enough food according to the population increase. These measures include adopting computer technologies, such as artificial intelligence, Internet of Things (IoT), and machine learning. Artificial Intelligence (AI) is the comprehensive science that aims to create intelligent systems, while Machine Learning (ML) is a specific approach within AI that uses data to enable machines to learn and adapt. These technologies have seen improvement in fighting pests and varying climatic conditions.

Kenya is a developing country whose economy mainly relies on agricultural activities. According to the (Food and Agriculture Organization of the United Nations, 2018), agriculture is critical to Kenya's economy. In its report, Food and Agriculture Organization (FAO) reports that while Agriculture directly contributes 26% of the Gross Domestic Product (GDP), it also indirectly contributes another 27% of the GDP through associations with other sectors. However, the productivity level has remained low due to unpredictable variability in climate change (World Bank Group, 2018). Following this stagnation, small-scale farmers and small agricultural enterprises have faced more challenges in growing their businesses and improving the quality of agricultural goods.

According to Mwaniki (2010), Kenya is considered a water-scarce country, with 80% covered by arid and semi-arid land. The scarcity is expected to rise with time due to several contributing factors to climate change. This scarcity, therefore, calls for efficiency in water use and management. The need for food security and nutrition is inscribed in Kenya's government blueprint, famously known as the big four agenda, which seeks to adopt irrigation to bring more land into farming use (World Bank Group, 2018).

Currently, the Government of Kenya (GoK), through the Ministry of Agriculture, Livestock, and Fisheries (MALF), is implementing the Kenya Climate-Smart Agriculture Project (KCSAP) (Ministry of Agriculture, Livestock and Fisheries, 2017). The KCSAP aims to increase agricultural productivity and build resilience to climate change risks in targeted smallholder farming and pastoral communities in Kenya. The other objective is to provide an immediate and effective response in a crisis.

According to GoK's climate risk profile report, Uasin Gishu county has a (0.419) vulnerability index to climate change (GoK, 2013). The county continues to experience severe weather occurrences such as drought. Drought has led to crop failures and reduced crop production areas. A report on Uasin Gishu's climate risk profile captures the decline of land under crop production from 92 500 hectares (ha) in 2014 to 77 225 ha in 2017 (*Climate Risk Profile Uasingishu County*, 2017). Uasin Gishu County is considered Kenya's breadbasket; hence, the climate change repercussions and agricultural variation call for an effort to mitigate the effect of climate change on food production.

Uasin Gishu county government has implemented measures to curb food insecurity by encouraging irrigation. The government has drilled boreholes and silted dams to provide alternative water sources to farmers during dry seasons. Besides the silted dams and drilled boreholes, the residents also harvest rainwater to supplement the water quantity (Climate Risk Profile Uasin Gishu County, 2017). Uasin Gishu county residents also practice traditional irrigation along river basins such as Moiben and Cheborорwa (Akenga *et al.*, 2020). The crops under irrigation in Uasin Gishu county include passion fruit, tomatoes, and other vegetables.

1.2 Statement of the Problem

Kenya is considered a water-scarce country, and agriculture uses an enormous volume of this scarce natural resource (Mwaniki, 2010). The increasing demand in agricultural production for food calls for efficient management of the limited resource. Research done by Lang'at (2019)

has shown that maximizing the utilization of every unit of water saves water and leads to an expansion of irrigable land by 15%. Water use management in irrigation farms in Uasin Gishu county is done manually, where farmers turn on irrigation taps for an estimated time in the morning and evening. Besides waterlogging, which causes soil degradation in the irrigated farms, this practice leads to inefficient water usage and, eventually, lower yields from the irrigated land. In the past, research has been done in open field surface irrigation to ensure efficient water management in open irrigated land. However, this studies have technologies with high implementation cost, and the infrastructure resources required are strenuous for developing countries. Therefore, to address this problem, using low cost resources, this project developed an IoT-based smart irrigation system for efficient water management aiming to increase water usage efficiency, provide remote visualization and eradicate repetitive tasks such as opening and closing irrigation valves. Massive production of the system could further reduce the overall cost of production.

1.3 Rationale of the Study

Several researches have been done utilization of IoT for smart water management systems. These studies include those that automatically control irrigation valves. There is still room to improve efficiency of these smart systems by including rainfall prediction for open field irrigated farms.Uasin Gishu county doesn't have any system for efficient water management and still relies on manual control which leads to inefficient water management.

1.4 Objectives of the Study

1.4.1 General Objective

To develop an IoT-based smart irrigation system for efficient water management.

1.4.2 Specific Objectives

The study aimed to achieve the following specific objectives:

(i) To identify requirements for a smart irrigation system for efficient water management

(ii) To develop an IoT based smart irrigation system for efficient water management.

(iii) To validate the developed smart irrigation system for efficient water management.

1.5 Research Questions

The study intended to answer the following questions:

(i) How to identify system requirements for a smart irrigation system for efficient water management?

(ii) How to design and develop a smart irrigation system for efficient water management?

(iii) How to validate the developed system?

1.6 Significance of the Study

The successful accomplishment of this project contributes to the body of knowledge on how the Internet of Things can be applied in irrigation agriculture to achieve efficiency in water management while improving crop yield for farmers. The project's successful accomplishment also contributes to adoption of technological solutions to irrigation, agriculture and food security.

1.7 Delineation of the Study

The project's main aim is to address the inefficiency in water management in open field irrigation farms to achieve precision in water input. The precision inwater management leads to expansion in irrigable land which eventually improves food pruction in the face of climate change.

The system met its specific objectives within the scope of the study. However, there are limitations to the implementation of the developed system. The major limitation is that to use the system; one must have an internet-enabled device and be online. Despite most of the population having smartphones, most farmers are in rural areas where internet connection is challenging. However, the government of Kenya is working towards 100% internet connectivity through National Fiber Optic Project (NOFBI) (Kelly & Dunand, 2021).

The other limitation comes with the IoT devices used to develop the system. The low-cost sensors used in this system have a short lifespan and would require replacement from time to time. This means maintenance from time to time is needed; however, the maintenance cost is affordable since the devices used are low-cost.

CHAPTER TWO

LITERATURE REVIEW

2.1 Review of Related Theories

Davis founded the Technology Acceptance Model (TAM) in 1986 (Silva, 2015), which models perceived usefulness, perceived ease of use, attitude, and the existing system. The model emphasizes these factors as crucial factors that determine technology adoption. The technology acceptance model is critical in assessing a system's viability in terms of adoption. This model is directly applicable to the Internet of Things projects and technologies since it can provide a quick guide to measure the adaptability of individual technologies.

The modified version of TAM followed Davis' Technology Acceptance Model theory. This new version also proposes perceived usefulness, ease of use, and attitude as important constructs to technology acceptance. Further to the two constructs, the latest version also adds behavior intention and actual usage to the technology acceptance concepts (Davis, 1989). The modified version of TAM recognizes gender, effort, social influence, facilitating conditions, and hedonic motivation as additional factors in adopting new technology.

A theoretical review by Mavrommati *et al.* (2013) reviewed different theories forming the basis of the Internet of Things design to support end-user development. From the activity theory postulated by Bannon, the authors derive three key factors in the design of IoT systems. These critical factors to be considered while designing any IoT system include End-user design, end-user programming, and semantic web/ IoT System design, which is equivalent to the system development methodologies and the development of the total system.

In his activity theory, Bannon describes tolls as "tools that reflect the experiences of other people who have tried to solve similar problems earlier and invented or adapted and modified the tool in order to make it more efficient. This accumulative social experience is accumulated in the properties of tools (structure, shape, material, etc.) and in the knowledge of how the tool is used" (Bannon, 1997). Bannon's definition of tools relates to a "thing" as used in the IoT. The Internet of Things field incorporates several tools, which include sensors and actuators that are required to communicate seamlessly and perform activities that humans can perform manually. Activity theory can be used to form the basis of IoT's conceptual framework

technological tools. It addresses how technology IoT and people's Cognitive internal and external processes can change as an ecosystem as time goes by.

Distributed cognition theory (Hollan *et al.,* 2000) by Hutchins is a cognition thinking framework that seeks to understand how the aggregation of cognitive properties emerges from component parts interactions. The theory attempts to explain how comprehension is achieved by accumulating facts from different environmental components. The distributed cognition theory has been used to advance human-computer interactions. It closely simulates the Internet of Things concept, where several elements in the environment interact together to achieve a particular task (often automation of manual activity.) This can also be applied in smart farming where sensors are used to collect physical parameters and actuators automatically perform otherwise manual human tasks.

Several theories are attempting to explain Human-computer interaction (HCI). In their study, Mavrommati *et al.* (2013) describe the four critical factors for human-computer exchange. The end-user design assumes the end-user as the system designer and deals with customization, envisioning, planning, designing, and adapting to the end user's own ambient experience. The authors emphasize customization settings to maximize user experience by providing them with the power to customize the systems to fit their desires. End-user design is the desired practice that ensures usability maximization and encourages easy-to-use multi-modal interface design. A self-explanatory interface is also a preferred practice in IoT. The authors propose including end-user programming and design as inseparable elements of ubiquitous systems. This inclusion ensures compliance with the TAM usability factor, as explained by Silva (2015).

The theories described in the literature above have evidently been considered in the IoT field to maximize IoT adoption. This project’s design was guided by Bannon’s activity theory where the focus was to develop a system that would reasonate with the end user’s expectations of a smart irrigation system. The distributed cognition theory was also adopted in getting different environmental components such as aoil moisture and rainfall prediction to accomplish one task i.e the automation of the irrigation system. Davi’s Advanced TAM guided our user acceptance testing exercise where the perceived ease of use, attitude and usefulness were among the feedback collected from respondents.

2.2 Empirical Literature Review

The Food and Agriculture Organization of the United Nations (FAO, 2009) estimates that within 20 years, there will be a 27% increase in land under irrigation. While this sheds some hope, the research also predicts only a 12% increase in water available for use in agricultural activities (Karina & Mwaniki, 2011)

Irrigation expansion, farmer information, and digital technologies are some proposed opportunities to improve agricultural productivity amid climate change challenges (Birch, 2018). Intertwining these opportunities can improve productivity by a more significant margin. The Internet of Things is one of the digital technologies that can be employed to improve productivity.

Machine learning systems have also been on trial in several parts of the world. The aim is to achieve water management efficiency in water reserves, among other irrigation parameters. An example of such a system is that developed by Togneri *et al.* (2019). The system named Smart water management platform (SWAMP) attempts to mitigate water wastage in water reserves, minimize irrigation problems such as under-irrigation and over-irrigation, and provide for planning and forecasting of water needs on the farm. The SWAMP is a huge project that combines various technologies such as sensor technology, semantic and cloud computing, drone technology, communication protocols, and IoT, among others. Nonetheless, the system described in this study requires full-time attention as the farmer must approve the plan or make changes to the irrigation plan before implementation. Besides, the SWAMP implementation cost is high, and the infrastructure resources required can be strenuous for developing countries.

Globally, automated water management platforms have been on trial. For instance, (Ullah *et al.* (2021) proposed and simulated an IoT-Based Energy-Efficient Water Management Platform for Smart Irrigation. Through a Matlab simulation, the authors found that using sink nodes in precision agriculture improves network performance. The proposed EEWMP (Energy-Efficient Water Management Platform) system is an improvement of an earlier model SWAMP aimed at improving network and energy efficiency consumption levels of SWAMP. The authors, however, did not implement the proposed system in an actual environment. The simulation was also set to allow water flow for a preset period. Still, it did not consider the sensor data collected from the field to determine the amount of water needed by the crop.

The lack of real-time parameters in automation leaves an unsealed loophole in maximizing water efficiency in irrigated land.

A mobile phone controllable smart irrigation system was also developed by Olatunji *et al.* (2020). The authors developed a prototype of a system that can be used for monitoring and controlling of greenhouse environment through an android mobile application. The researchers measured soil moisture and temperature to decide what action to take on the irrigation valves. The developed system automated valve control and is suitable for controlled environment such as greenhouse. The system is, however, applicable for a controlled environment that is small in space, which is a limiting factor for large open field irrigated farms as soil water can vary from one part of the farm to another which requires multiple sensors and irrigation valves in different parts of the farm.

The GIZ reported that, through the Kilimo bora insurance initiative based on IoT, farmers in Kenya, Rwanda, Tanzania, and Zimbabwe had improved their crop yields (Kreische *et al.*, 2015). This initiative provides farmers with accurate information on weather conditions so that farmers can take the correct action according to the report received. This initiative proves that IoT can be implemented even in small-scale open-field farms. However, the initiative doesn't automate the farming systems, leaving natural human weaknesses to chance. Integrating weather forecasts with real-time soil crop needs can improve irrigation precision and water use efficiency.

Smart irrigation systems in India have also been developed for data visualization (Rawal, 2017). The approach proposed by the researchers used various sensors to fetch parameters of the farm environment and transmit them to the Thing speak cloud using WIFI. The farmer could therefore view the fetched parameters by retrieving them from the Thingspeak cloud. Mbandi and Kisangari (2020) developed a wireless sensor network for collecting weather parameters for online visualization in Kitui - Kenya. The system employed various sensors to measure different environmental parameters and transmitted them to the cloud for online visualization. However, the two visualization systems did not achieve automation of optimal controls to aid in managing the measured parameters to the desired measurements for their respective applications.

Another study was conducted at Dedan Kimathi University in Kenya, irrigated farms and other farms in Ethiopia and South Africa and proposed the adoption of smart irrigation. The study

done by Nigussie *et al.* (2020) suggested architecture for a micro-climate monitoring system. The proposed architecture presents early farmer notification on what could happen shortly regarding weather fluctuations. The approach proposed by these authors gets data input from sensor nodes in the field and transmits it to the servers using LoRaWAN. The proposed system does not provide automated controls for desired farm actions based on weather prediction.

Another research by Haule and Michael (2014a) reviewed deploying a wireless sensor network to manage water in irrigation farms. The authors proposed the adoption of automated irrigation water management. The authors further implemented this project in a simulation of a real environment (Haule & Michael, 2014b). The author's research utilized only soil moisture as the parameter for the automation decision. The system can be improved by getting more parameters, such as rainfall prediction, which are critical in irrigation decisions and could further improve water efficiency in open-field irrigated land.

In his research, Njuu (2016) developed a sensor-based system for water monitoring in smallholder irrigation farms in Tanzania. The system measured the water level at the irrigation canal entry points. However, the system did not consider real-time soil needs in different parts of the farm to ensure efficient water use in the irrigated land which could improve efficiency as water would be used precisely when needed. The developed system uses soil sensors to gather real-time soil needs such as soil moisture levels which is used for irrigation valve control decisions.

The above work related to this study shows much has been done on IoT water management in irrigated lands. Much work has been done on weather visualization and monitoring of farm parameters. A few have been done on automating irrigated land water management based on soil moisture. However, none of the studies considers rainfall prediction as an input for water control in irrigated land. Rainfall prediction is a critical factor for farmers practicing open-field irrigation as they can save on water whenever there's a high possibility of rainfall.

2.3 Conceptual Framework

Figure 1 shows a conceptual model of the proposed system.

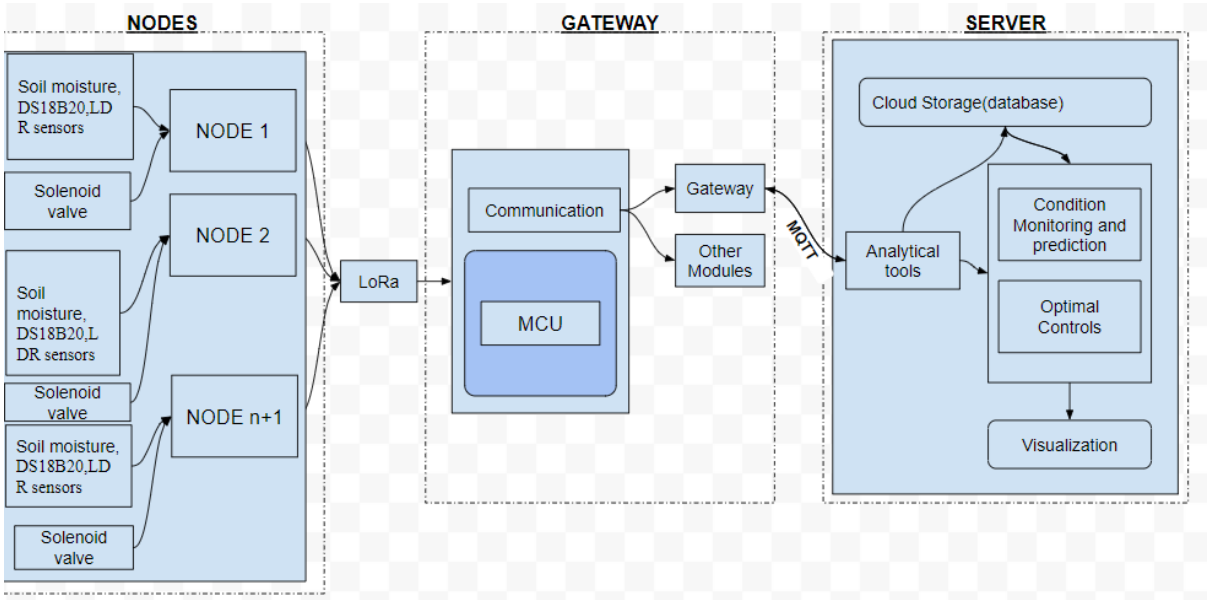


Figure 1: Conceptual model of the proposed system

Among the popular microcontrollers in IoT-based irrigation systems are ESP32, ESP8266, and Xbee (García *et al.*, 2020). The ESP32 was chosen for this project due to its low cost and better performance features compared to other microcontrollers in the field of IoT. Besides ESP32's low cost and high performance, Maier *et al.*, (2017) discussed the improved technical performance of ESP32 as compared to other microcontrollers. The comparison made in Table 1 shows that ESP32 has more technical advantages to leverage. These advantages include low power consumption and more connection interfaces.

Table 1: Performance comparison between ESP32 and other microcontrollers (Maier *et al.*, 2017)

Chip (Module)	ESP32 (ESP-WROOM-32)	ESP8266 (ESP8266-12E)	Xbee (XB2B-WFPS-001)
Voltage	2.2V to 3.6V	3.0V to 3.6V	3.14V to 3.46V
Operating Current	80 mA average	80 mA average	N/A
Programmable	Free (C, C++, Lua, etc.)	Free (C, C++, Lua, etc.)	AT and API commands
Open source	Yes	Yes	No
Wi-Fi	802.11 b/g/n	802.11 b/g/n	802.11 b/g/n
Bluetooth®	4.2 BR/EDR + BLE	-	-
UART	3	2	1
GPIO	32	17	10
SPI	4	2	1
I2C	2	1	-
PWM	8	-	-
ADC	18 (12-bit)	1 (10-bit)	4 (12-bit)
DAC	2 (8-bit)	-	-
Size	25.5 x 18.0 x 2.8 mm	24.0 x 16.0 x 3.0 mm	24.0 x 22.0 x 3.0 mm
Prize	£8	£5	£23

CHAPTER THREE

MATERIALS AND METHODS

3.1 Area of the Study and Scope of the Research

This study covered irrigated farms in Uasin Gishu County, Kenya. Specifically, the study focused on passion fruit irrigation. Besides its high value, Passion fruit is grown throughout the year. Therefore, an automated system is needed to maximize the efficient use of water and reduce intensive labor and costs associated with manual water management in irrigated farms.

Uasin Gishu county comprises six sub-counties: Turbo, Kesses, Soy, Ainabkoi, Kapseret, and Moiben.

3.2 Research Methods

To leverage the credibility and contextualization of results/findings, a mixed research method was used. Here, we incorporated both qualitative and quantitative research methods. Blending quantitative and qualitative data collection methods to collect data on the same topic enriches the results. We employed an exploratory sequential research design of mixed methods. With the exploratory sequential design, we first used qualitative methods (focused discussion groups and interviews) to explore initial questions and collect initial findings. The qualitative information collected was analyzed and used to develop questions for the quantitative data collection phase. Questionnaires were used in the quantitative phase mainly to confirm the findings from the qualitative method initially used. We also analyzed existing research publications on smart irrigation systems to identify challenges and opportunities in developing smart irrigation systems. Several scientific papers were reviewed to find out how other researchers have solved similar problems in the past. The county government of Uasin Gishu also provided documentation on past work and future plans regarding irrigation.

3.3 Target Population

The target population for this study was irrigated farm managers and county government irrigation system experts. An Irrigation farm manager is anyone tasked with overseeing the activities in the irrigated land. Sixteen respondents participated in the exrcise.

3.4 Sample Size and Sampling Techniques

Due to the limited number of participants that can provide the primary data required for the project requirement identification and validation, we used the purposive sampling technique to identify participants in the data collection phase. The purposive sampling technique is also cost-effective.

To ensure the project's effectiveness, five technical staff participated in focused discussion group meetings, while eleven farmers responded to our questionnaires.

3.5 System Requirements

To come up with system requirements, we used the tools and methods explained in this section

3.5.1 Data Collection

Data collection involves gathering and analyzing data about the research of interest to answer a hypothesis or a research question. The project methodology focused on getting accurate information regarding the IoT-based smart irrigation system for efficient water management. The data collection exercise aimed to understand the expectations and desires of different stakeholders concerning the developed system. Data collection also helped in making informed decisions and gaining technical insights into the critical aspects of passion fruit farming.

As explained below, the data collection methodology was based on secondary and primary data sources.

(i) Secondary Data Sources

The existing documents including published materials and scientific articles were reviewed. The documents from the Uasin Gishu county government provided regarding irrigation systems and passion fruit farming were used. The documents provided by the county government for this project include the county integrated development plan 2018-2022 (CIDP) and the climate risk profile report.

(ii) Primary Data Sources

To understand the challenges of the current manual irrigation systems, the Primary data were collected using questionnaires, and focus group discussions. In this stage the technical staff

were engaged through the focus group discussion. In addition, the questionnaires were administered to farmers practicing passion fruit farming within Uasin Gishu county.

3.5.2 Data Analysis

The questionnaires were administered online using google forms which reduced the cost of printing and also provided quantitative analysis of individual questions. Table 1 shows information collected from the focus discussion group. Detailed information on the questionnaire is contained in appendix 1, while that of the focus discussion group is covered in the discussion guide in Appendix 2.

3.6 System Development Approach

The Extreme programming (XP) agile software development life cycle model was used. XP is based on achievable iterative tasks and incremental processes. A review of XP by Al-Saqqa *et al.* (2020) explained the critical roles in the entire lifecycle. The positions described include programmer, customer, tester, tracker, coach, consultant, and manager.

The XP consists of seven phases in its life cycle. The exploration phase is the first phase, which involves collecting user stories from the customer and developing a sample prototype. The second phase is the planning phase, where the user stories are prioritized, while the iteration release phase is the third phase, where iterations are released, with the last iteration being the final product. Productionization, maintenance, and death phase are the fourth, fifth, and sixth phases, respectively. There are no additional user stories in the death phase, and the product is considered final.

A review of agile methodologies revealed that, unlike scrum methodology, XP does not address technical and managerial issues (Mirza & Datta, 2019). However, with XP, the work is continuously tested and released in short sprint cycles. The breaking down of huge tasks into smaller tasks eases error detection and correction and allows more focus on the product hence improving the quality of the end product. Furthermore, agile methodology is flexible to changes during implementation due to changes to a part of the process or requirements; this is because a change of functional requirements is implemented into the increment development process. The phases involved in the entire lifecycle are demonstrated in Fig. 2.

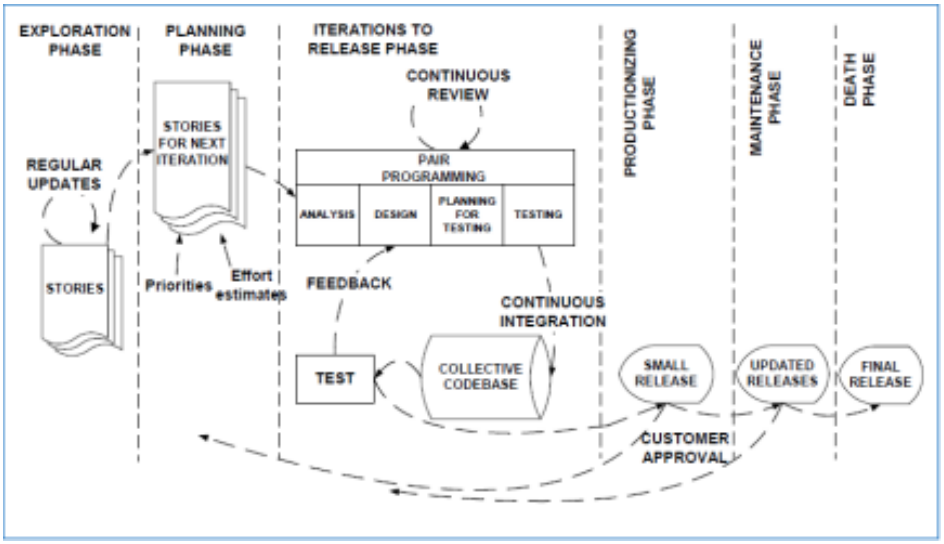


Figure 2: Extreme programming lifecycle (Al-Saqqa *et al.*, 2020)

3.7 System Design

3.7.1 Architecture Diagram

The proposed system collects soil parameter information using sensors. The irrigated land is divided into sections, each with a node. The most critical parameter required for precise water management is the level of soil moisture. Other parameters include temperature and light intensity, which we only collect for visualization. The parameters are sensed using sensors connected to Raspberry Pi Pico. In a review by García *et al.* (2020), LoRa is described as a technology with a longer transmission range compared to ZigBee and other transmission protocols. The review also describes LoRa as the most preferred long-range transmission technology for secluded areas. In this proposed system, the sensor parameters are transmitted through LoRa communication protocol to the Coordinator consisting of an ESP32 microcontroller. The ESP32 transmits using WIFI to the server (Thingsboard cloud). The server also receives information on rainfall prediction from a weather prediction API (OpenWeather). The system then analyzes the received parameters and implements optimal controls by sending a signal to turn on or off the irrigation valves (actuators) on the responsible nodes. Figure 3 shows the architecture diagram of the system.

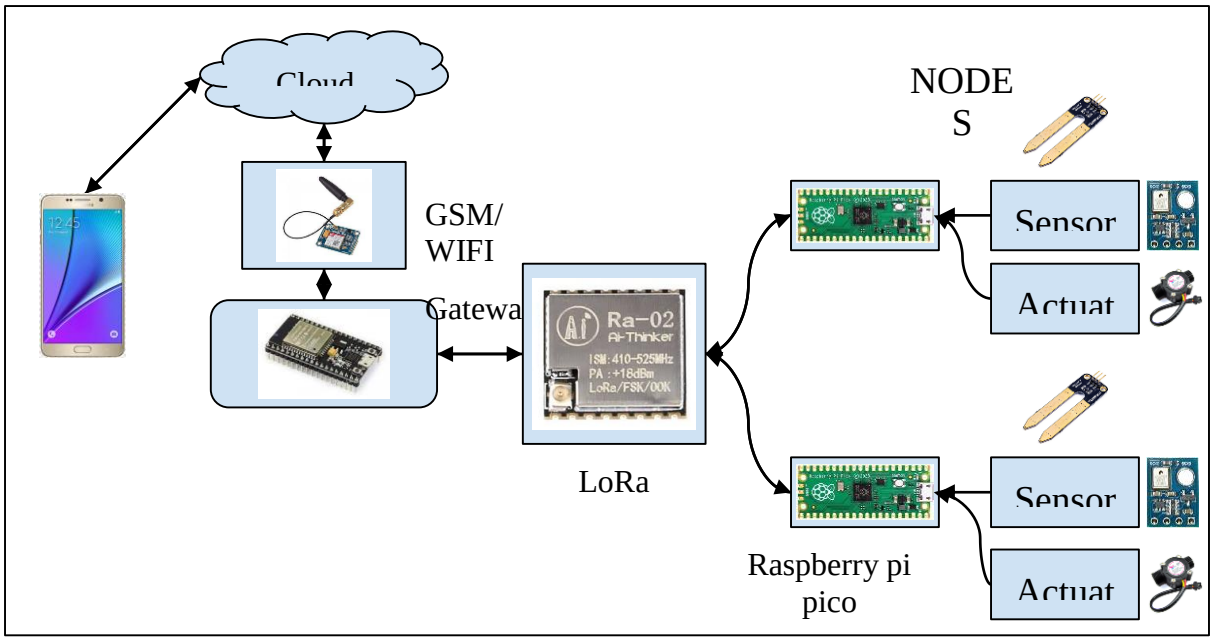


Figure 3: System architecture diagram

3.7.2 Printed Circuit Board Design

Two boards were designed using Kicad open-source software. KiCad, an electronic design automation software that enables individuals to design and simulate electronic hardware, was used to design the PCB. The design and simulation of electronic hardware enable the process of schematic capture, which is the first procedure of circuit design. The system is made up of the node and the Gateway, whose boards were designed. The system's node shown in Fig. 4 comprised of the sensors, actuators, and the RP2040, whereas the Gateway is shown in Fig. 5 composed of LoRa, OLED display module, and ESP32.

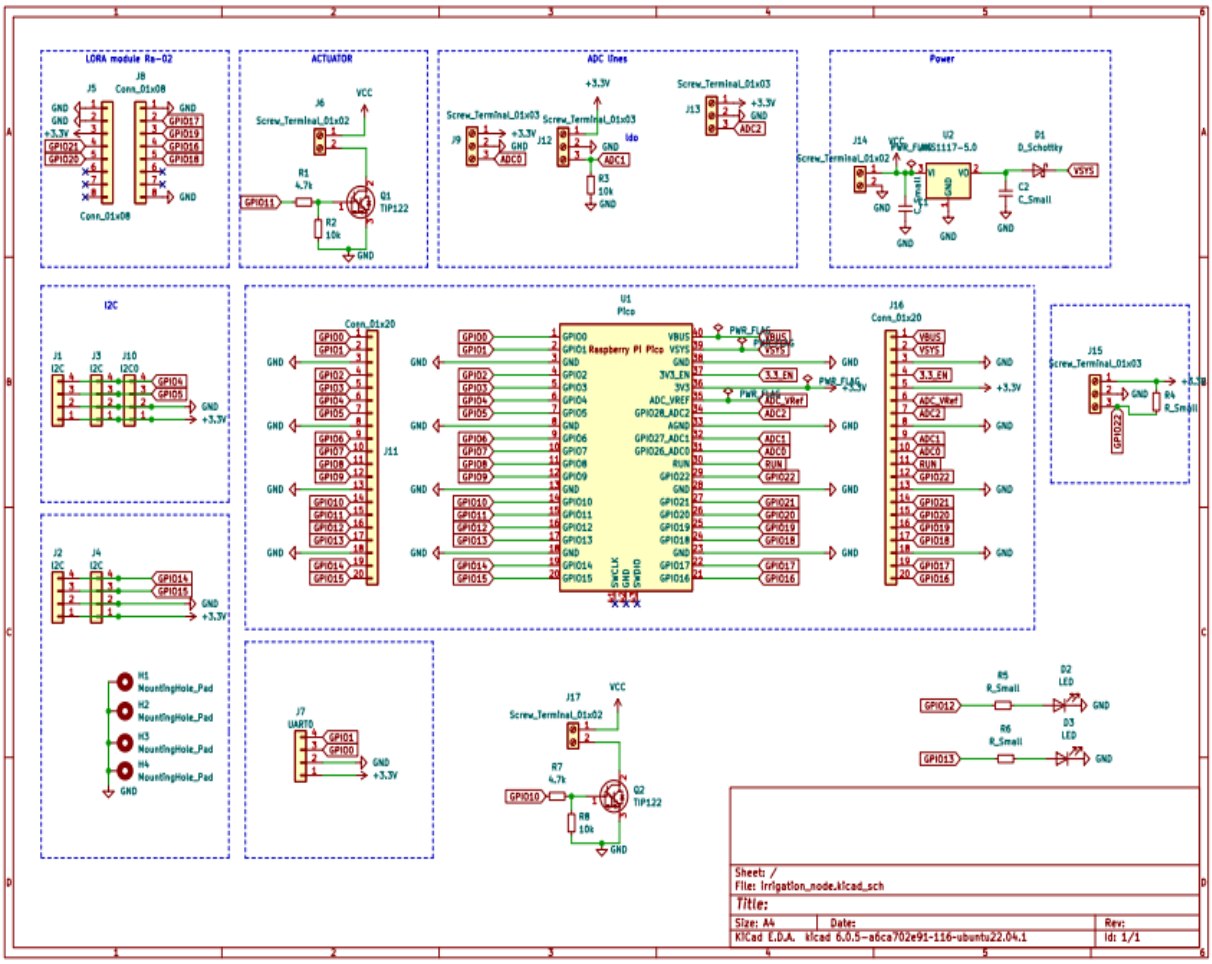


Figure 4: System node PCB circuit design diagram

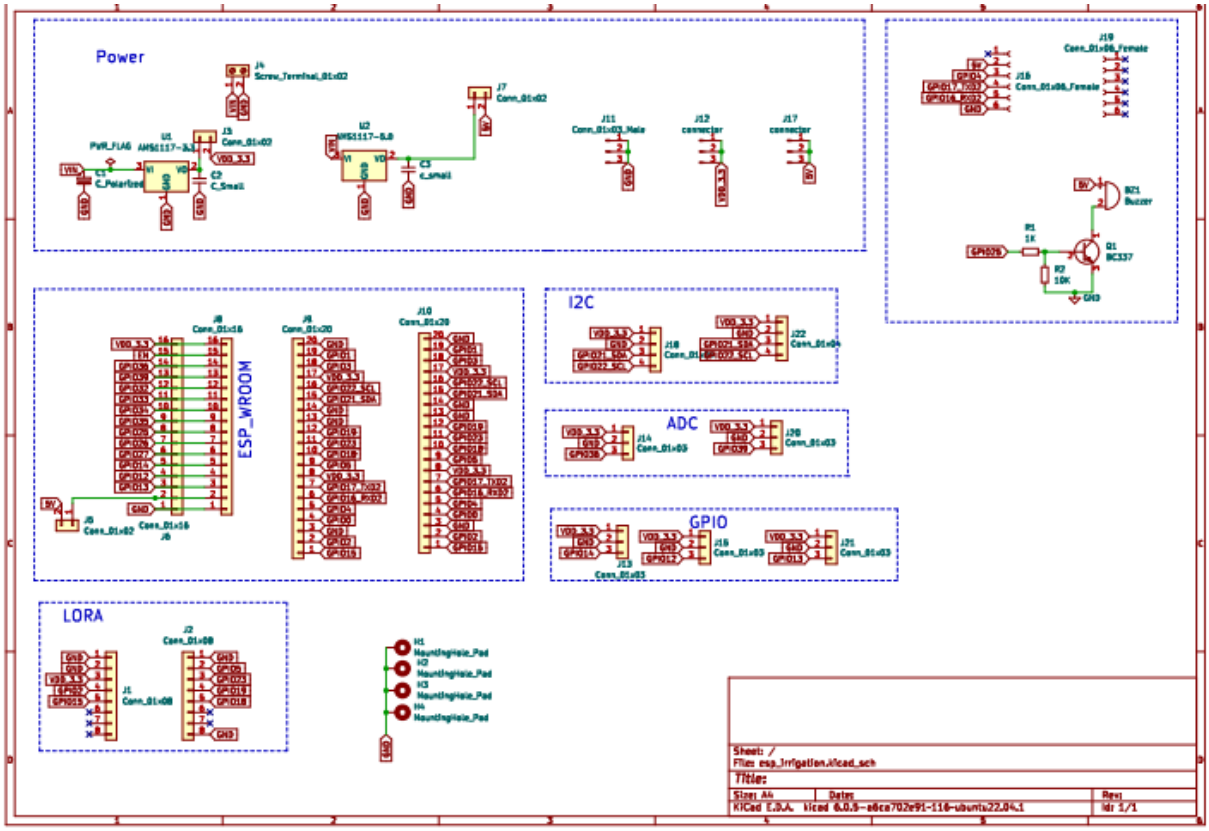


Figure 5: System gateway PCB circuit design diagram

3.7.3 The PCB Power Schematic Diagram

Figure 6 shows the PCB power schematic design captured from Kicad. The chip used for power regulation is the AMS1117 low dropout three-terminal regulator. The board is powered externally by a 12 V source and is regulated to 5 V and 3.3 V. The power distribution is elaborated on in the hardware development section.

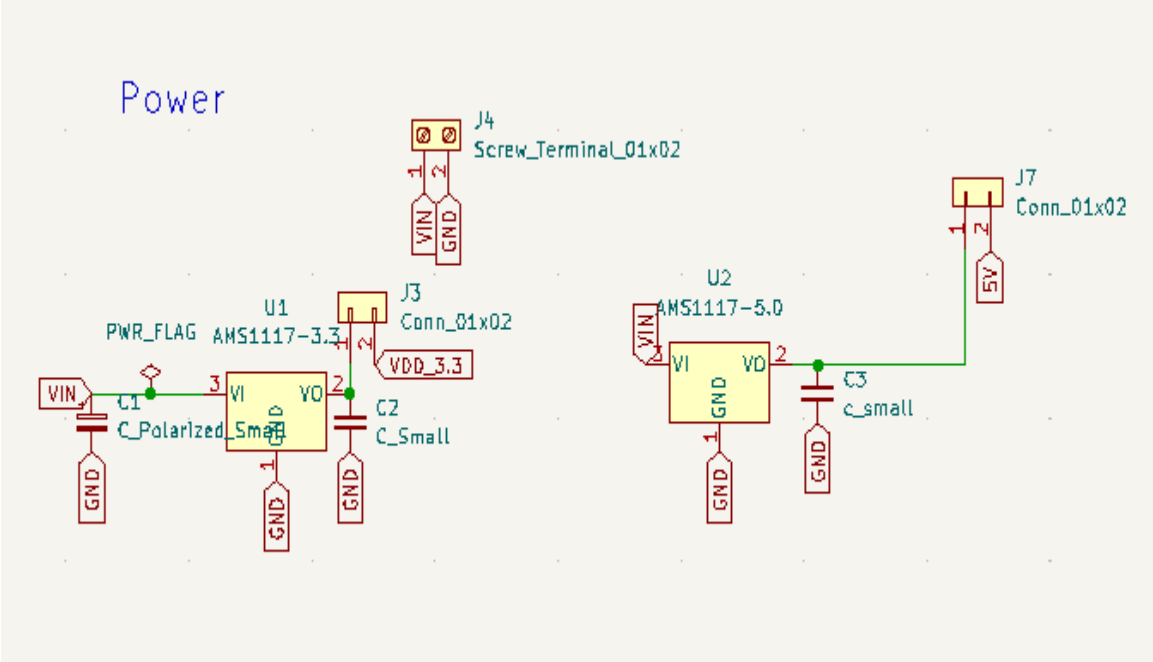


Figure 6: The PCB power schematic diagram

3.7.4 The 3D Case Design for the Gateway

The system's PCB comprises sensitive components susceptible to damage if left in the open air. For proper storage, a package case was designed to protect the electronic components from damage caused by environmental elements, e.g., water, dust, etc. The packaging case shown in Fig. 7 was designed using Freecad, and Cura software was used to slice and generate G-code, which was later transferred to a 3D printer to print out the case.

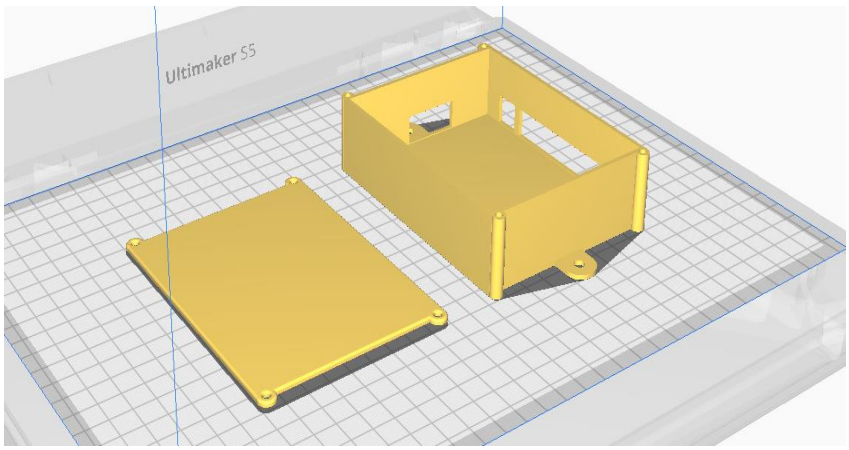


Figure 7: The 3D Gateway case design

3.7.5 Block Diagram

The gateway comprises an ESP32 microcontroller, OLED display, AHT20, and LoRa module. OLED display and AHT20 are connected to ESP32 using the I2C interface, while the LoRa module is connected to ESP32 using the SPI interface.

The gateway communicates with the node using LoRa (433 MHZ) protocol. The Raspberry Pi PICO is connected to LoRa through the SPI (serial peripheral interface) interface, whereas the soil moisture sensor and LDR are connected through ADC. Figure 8 shows an elaborate block diagram of the entire system, the components used, and their connections.

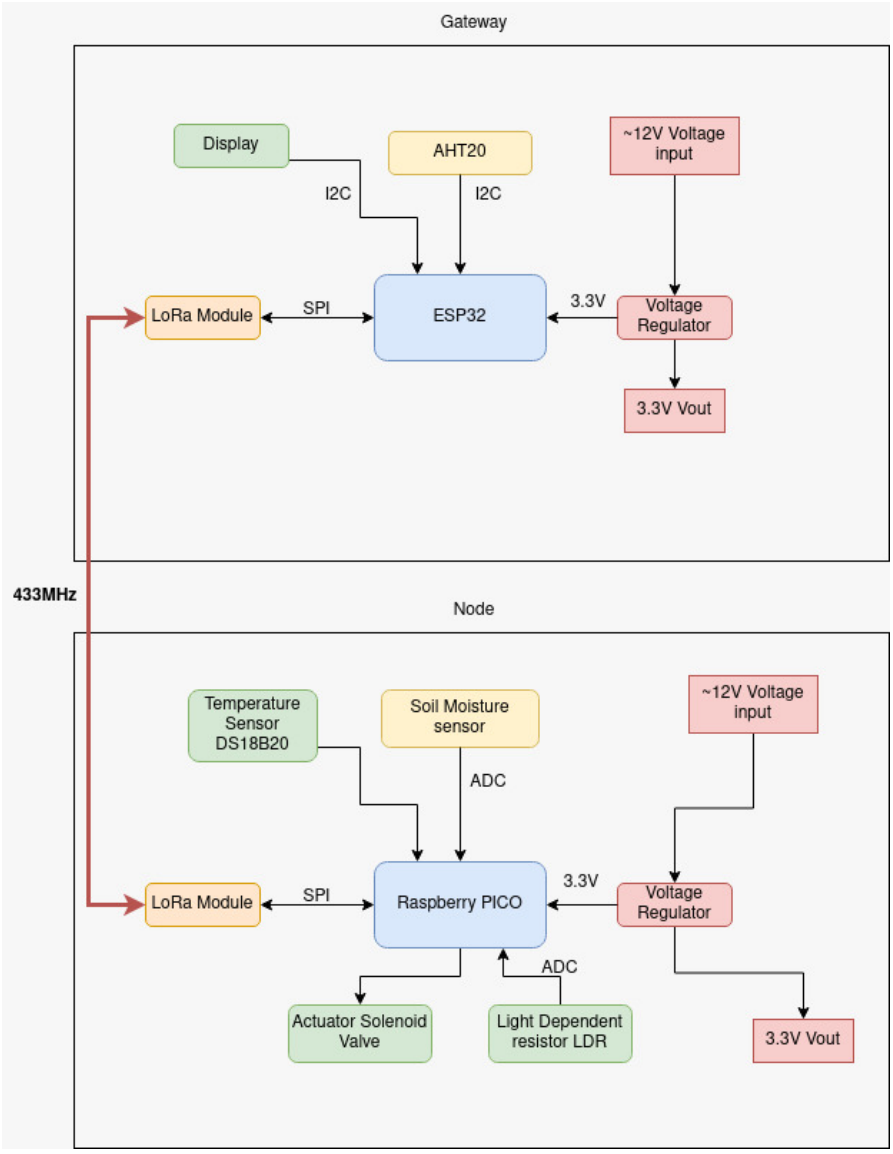


Figure 8: System block diagram

3.8 System Development

In this section, we delve deeper into the procedures involved in the development of both the hardware and software aspects of the entire system. This section also captures all the hardware components and software tools used in the system development phase, and Firmware development is also elaborated on in detail.

3.8.1 Hardware Requirements

A summary of hardware requirements is presented in Table 2. Each hardware component's description, specifications, and other information are provided in detail.

Table 2: A summary of the system's hardware requirements

Hardware component	Specification
Microcontroller boards	Pico RP2040, ESP 32
Sensors	AHT20, DS18B20, Analog capacitive soil moisture sensor, LDR (Light Dependent Resistor)
Transmission protocols	LoRa Ra-02 SX1278 Module 433MHZ
Irrigation valves	12 V Solenoid valve G1/2 DN15BSPP
Display	OLED 128 * 32
Others	PLA standard 1.75mm, Ferric Chloride acid, status LEDs

(i) The ESP32-WROOM-32

The ESP32 microcontroller board, integral to the gateway, processes data from nodes and forwards it to the cloud. According to Maier *et al.* (2017), the ESP32 is an excellent choice for IoT projects due to its affordability and impressive performance features, as highlighted in Tables 1 and 3. Figure 9 shows the ESP32 development board.

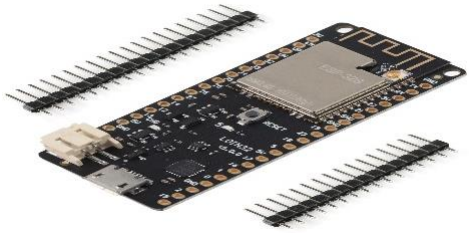


Figure 9: The ESP32 development board

Besides the features captured in Table 1, other features of ESP32 are listed in Table 3.

Table 3: The ESP32 features

Parameter	Specification
Microprocessors	Two low-power Xtensa® 32-bit LX6
ROM	448 KBytes
SRAM	520 Kbytes (on-chip), 8 Kbytes (in RTC SLOW), 8 Kbytes (in RTC FAST)
Wi-Fi	802.11 b/g/n/d/e/i/k/r (802.11n up to 150 Mbps)
Bluetooth	v4.2 BR/EDR and BLE
Security	WPA/WPA2/WPA2-Enterprise/WPS
Encryption	AES/RSA/ECC/SHA
Protocols	IPv4, IPv6, SSL, TCP/UDP/HTTP/FTP/MQTT
Interfaces	SD-card, UART, SPI, SDIO, I2C, LED PWM, Motor PWM, I2S, IR, GPIO, capacitive touch sensor, ADC, DAC, Hall sensor, and temperature sensor.
Operating temperature	-40 + 85℃
Operating voltage	2.2-3.6 V
Consumption	80 mA

(ii) Raspberry Pi Pico

The Raspberry Pi Pico shown in Fig. 10 is an efficient microcontroller board developed by the Raspberry Pi Foundation for embedded projects. The microcontroller is based on an RP2040 chip. Compared to Raspberry model zero, model three, and model four models A and B, the Pico is small, low cost, flexible, and consumes much less current. This project used Raspberry Pi Pico in the nodes with sensors and actuators to process data collected from the sensors and forward to the gateway. Raspberry Pi Pico operates at 3.3 V, the desired low power for the developed system. Table 4 shows the comparison between different microcontroller boards. The low cost of Raspberry Pi Pico also gives it an edge over the other microcontrollers.

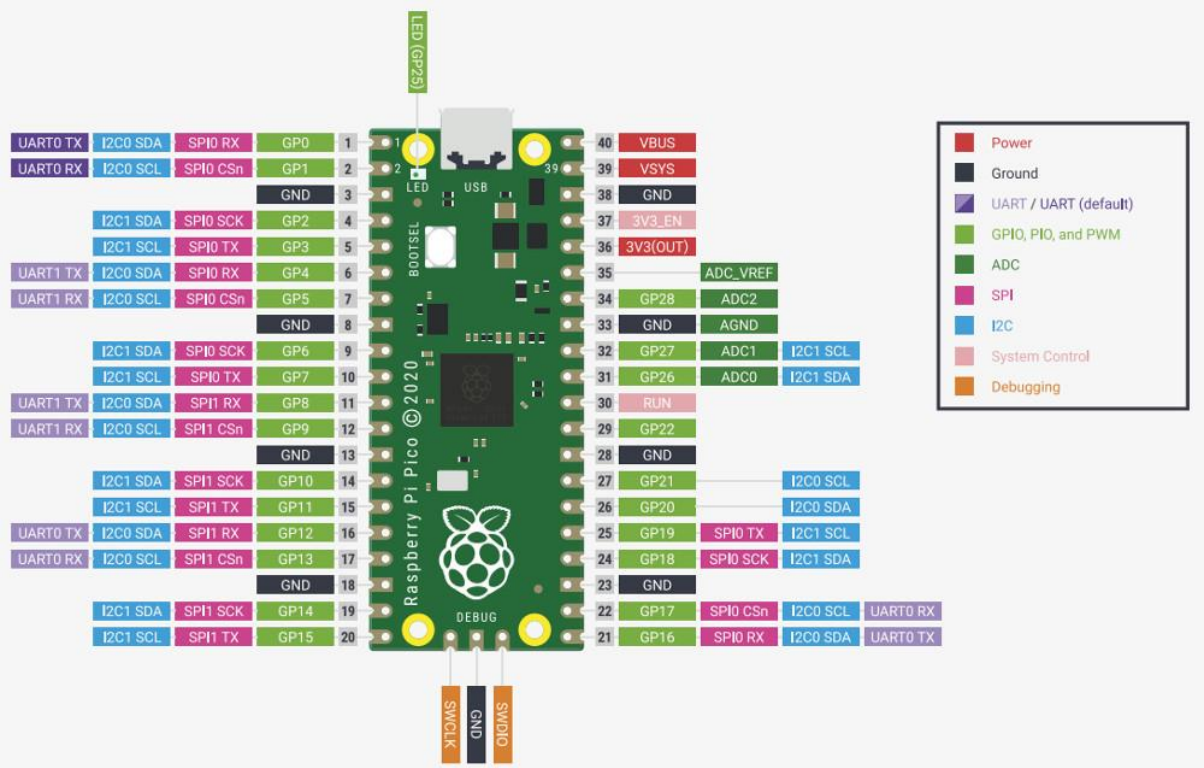


Figure 10: Raspberry pi pico (Brown, 2021)

Table 4: Microcontroller boards comparison

Item/Parameter	Raspberry Pi,3,4 series	Pico RP2040	ESP32	Arduino
Type	Single board microcontroller	Single board microcontroller	Single board microcontroller	Single board microcontroller
Operating Voltage	5 V	3.3 V	2.2 TO 3.6 V	5 V
CPU	ARM Cortex	ARM Cortex-MO+	Xtensa dual-core	Atmel, ARM, Intel
Developer	Raspberry Pi Foundation	Raspberry Pi Foundation	Open source community	Arduino
Peripheral interfaces	SPI, DSI, UART, SDIOCSI, GPIO	UART,SPI,I2C,USB,GPIO	UART, GPIO, SPI, I2C,Ethernet,CAN bus,	SPI, I2C, UART, GPIO
Operating system	Linux	RTOS(Real-time operating system)	Linux	None
Programming language	Python, c/c++, scratch	MicroPython, c/c++	C/C++, Python, Lua	Python, C, C#
Power	USB, Power Supply	USB-C, Power supply	USB, Power Supply	USB, Battery, Power supply
Clock Speed	1.2GHz	133 MHZ to 400 MHZ	160 or 240 MHz	16 MHz
Memory	1-4 GB	264 KB	320 KiB RAM, 448 KiB ROM	32 KB
Storage	Micro SDHC Slot	2MB	4 MB	1 KB
Price	From $35	$0.80	$25	$29

(iii) The AHT sensor

The AHT sensor shown in the Fig. 11 is a high-precision digital temperature and humidity sensor. However, according to García *et al.* (2020), DHT22 and DHT11 are the most utilized temperature sensors in irrigation systems. The DHT22 is a sophisticated version of DHT11; therefore, in Table 5, we compare AHT20 to DHT22. The AHT sensor is an improved version of DHT sensors and works the same as the DHT series. An AHT sensor is made up of two main parts, which are a thermistor and a capacitive humidity sensor. The AHT sensors have another chip that converts the received analog signals to digital signals. Its communication module is I2C. According to the manufacturer's declarations, AHT20 is three times better than its predecessors. Another research by Evstigneev *et al.* (2022) demonstrated that DHT22 and BME280 sensors have systemic errors that overestimate humidity, while AHT20 had close to zero discrepancies. The AHT20 sensor was used to collect the surrounding's humidity measurements and temperature.

Table 5: Comparison between AHT20 and DHT22 temperature and humidity sensors

Parameter	**AHT20**	**DHT22**
Humidity Accuracy	+- 2% RH	+-2% RH
Temperature Accuracy	+-0.3℃	+-0.5℃
Humidity Resolution	0.0024 RH	0.1 RH
Temperature Resolution	0.001℃	0.1℃.
Operational Voltage	DC: 2.0-5.5 V	DC: 3 to 6 V
Current supply	Sleep: 10 μA, Measuring: 950 μA	1 to 1.5 mA
Sampling period	2.0 seconds	2 seconds
Measuring temperature range	-40 to 85℃	-40 to 80℃
Measuring humidity range	0 to 100% RH	0 – 99.9% RH
Communication protocol	I²C, One-wire	One-wire
Price	$4.90	$4.99 to $9.90

Figure 11: The AHT sensor

(iv) The DS18B20 Temperature sensor

The DS18B20 shown in Fig. 12 was used to collect soil temperature measurements. The sensor is a one-wire interface with a 64-bit onboard ROM and operates in a voltage range between 3 V to 5 V.

Figure 12: The DS18B20

(v) Capacitive soil moisture sensor V1.2

This soil moisture sensor module detects soil moisture by measuring the volumetric content of water inside the soil and gives the moisture level as output. This sensor consists of a 555IC circuit that measures its capacity and produces a voltage proportional to the sensor's capacitance. The voltage produced ranges between 1.2 V and 3 V, while the sensor's operating voltage ranges between 3.2 V and 5 V. A built-in voltage regulator completes the sensor's functionality. The sensor is made of a waterproof probe with high corrosion resistance to

minimize corrosion speed, providing long-life assurance in the soil (Hobby, 2021). Figure 13 shows the image of this project's capacitive soil moisture sensor.

Figure 13: Capacitive soil moisture sensor (Hobby, 2021)

(vi) LoRa module

Due to popularity and ease of use, Wi-Fi, Zigbee, and bluetoth4.0 have increasingly been used in wireless transmission networks. However, these technologies communicate in a short range and are unsuitable for long-range applications such as smart agriculture and global positioning (Choi *et al.*, 2018). Lora was therefore used in this project for communication between the gateway and the node due to it's long range transmission. Bluetooth can efficiently communicate within a range of 50 feet without extenders, while Wi-Fi can communicate within 200 feet. LoRa's modulation occurs at the physical layer and implements the Chirp Spread Spectrum (CSS). The CSS modulation is derived from broad spectrum scheme. The CSS modulation involves converting a single bit of information into several other bits, which are spread and transmitted across the entire spectrum. Owing to its modulation technique that allocates the whole bandwidth to broadcast (data and information signal), LoRa provides security for both the network and data (Zourmand *et al.*, 2019). The LoRa module, shown in Fig. 14, is designed to work efficiently with long-range, limited-energy smart devices that do not require high-frequency communication establishment, which is usually the case with IoT implementations. LoRa can transmit up to 10 km in rural areas that need low power, which is the project's setting. Table 6 shows the comparison between LoRa and the popular IoT transmission protocols. While ZigBee and Bluetooth offer a higher channel bandwidth, uplink peak rate, and downlink peak rate, LoRa offers the longest transmission range.

Figure 14: LoRa module

Table 6: Comparison between LoRa, Bluetooth, and Zigbee protocols

Parameter	LoRa	Bluetooth LE	Zigbee
Channel bandwidth	125000 HZ	$2*10^6$ MHZ	$6*10^6$ HZ-$5*10^6$ MHZ
Transmission range	10000-15000 M	10 M	70-100 M
Transmission mode	Half	Half/ Full	Half
Downlink peak rate	300 – 50,000 bps	$125*10^3$ bps, $500*10^3$ bps, $1*10^6$ bps,$2*10^6$ bps	$20*10^3$-$250*10^3$ bps
Uplink peak rate	300 bps - 50 Kbps	125 Kbps, 500 Kbps, 1 Mbps,2 Mbps	20 Kbps-250 Kbps
Encryption	128bit NwKSkey/ AppSkey	AES-CCM, HMAC, SHA256	AES128
Maximum transmission power	22 dBm	0,4,20 dBm	Case dependent

(vii) The PLA standard 1.75 mm

Polylactic acid (PLA) standard 1.75 mm filament, shown in Fig. 15, was used for packaging case printing. The PLA 1.75 mm is more durable than regular PLA, providing excellent layer adhesion and a good finish free from bubbles. Despite its compatibility with most printers, PLA standard is non-toxic and environmentally friendly since it is bio-degradable. The PLA standard also has minimal shrinkages, and its printing temperature ranges between 70℃ and 80℃.

Figure 15: The PLA standard 1.75 mm

(viii) Ferric Chloride acid

Ferric chloride acid, shown in Fig. 16, is a mixture of iron and chloride in the ratio of one part of iron to three parts of chloride. In this project, ferric chloride was used for PCB etching. PCB etching is the process of removing exposed copper from the printed board.

Figure 16: Ferric chloride acid

(ix) Solenoid valve

A solenoid valve, shown in Fig 17, is a unidirectional liquid valve where liquid flows only in one direction, usually indicated by an arrow at the bottom. The solenoid valve used in this project has the specifications in Table 7. The solenoid valve is the main actuator that is used to control the flow of water in the system.

Table 7: Features of the solenoid valve

Parameter	Specification
Current	250 mA
Valve Type	Diaphragm(servo operated)
Diameter of Connector	14 mm
Voltage	12 V
Usage	Water and low-viscosity fluids
Pressure	0-0.02 MPa
Operation Mode	Normally closed
Fluid Temperature	1-100→
Maximum Size	84 x 53 x 41 mm
Length of Screw	17 mm
Diameter of Screw	19 mm

Figure 17: Solenoid valve

(x) The OLED 128 * 32 display

We used a 128*32 OLED display in Fig. 18 majorly to display the system's status. This LCD is used in the coordinator module. The display is made of an I2C interface and requires 3.3 V to 5 V power at its VCC terminal. The OLED display performance is better than the traditional LCD, uses lower power, and doesn't require a backlight.This component was used for display at the system’s node.

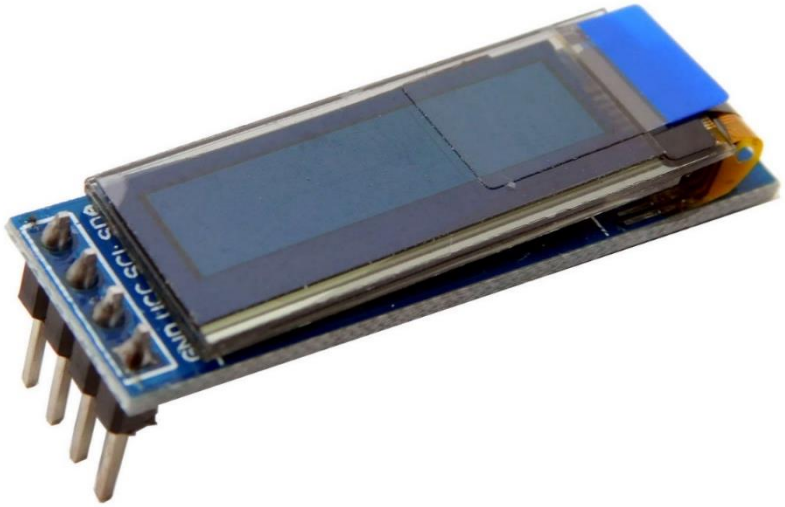


Figure 18: The OLED 128*32 pinout diagram

3.8.2 Hardware Development

Hardware development was done after the hardware design phase. The development involved printing and assembling the printed circuit board (PCB). Inkscape, a vector graphics editor which enables the printing of PCB layouts for transfer to the copper board, was used. The steps to achieving board development are explained below.

(i) Designing the circuit and PCB on Kicad software.

(ii) Checking for errors in the design.

(iii) Board preparation involved the following steps:

(a) Exporting PCB as an SVG file to Inkscape for further processing. An image of the two boards' PCB output is shown in Figs. 19, 20, 21, and 22.

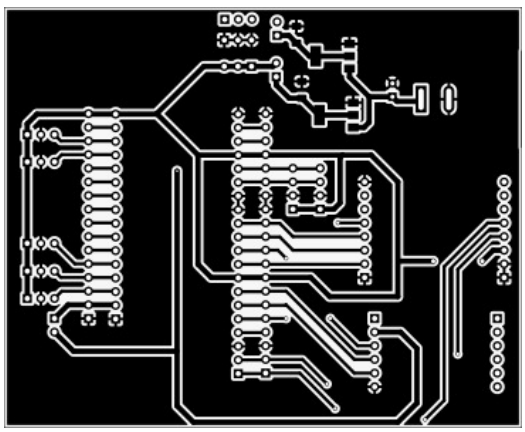


Figure 19: Gateway node PCB Top

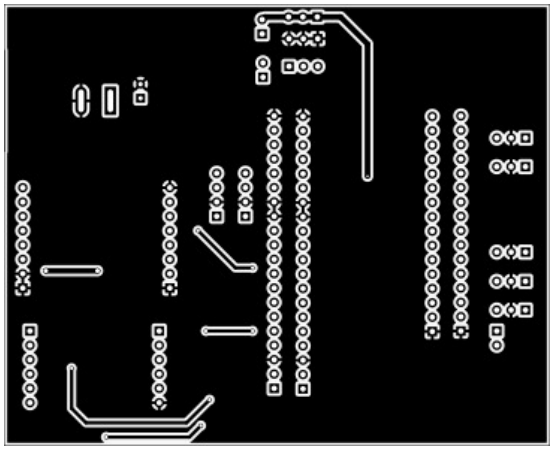


Figure 20: Gateway node PCB bottom

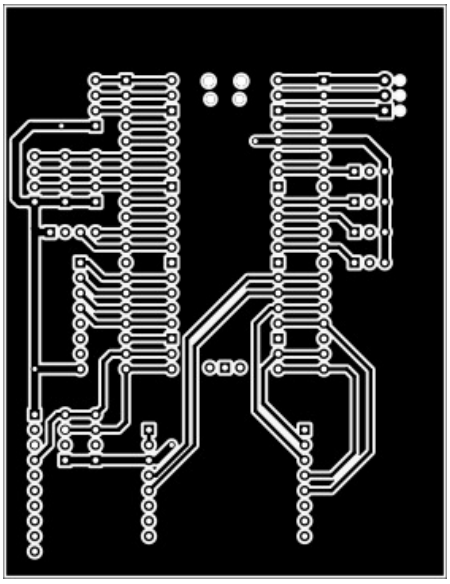


Figure 21: The PCB bottom layer

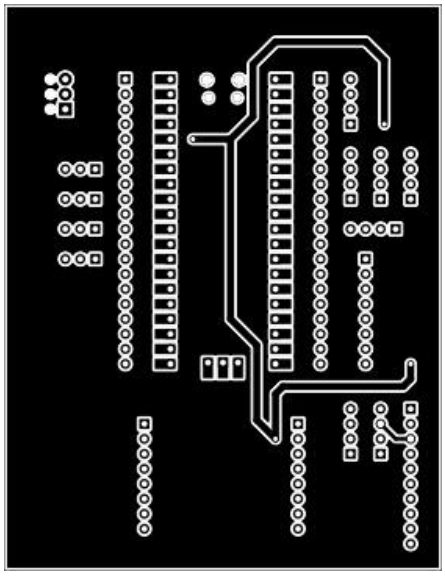


Figure 22: Node PCB Top layer

(b) printing on paper the PCB tracks for toner transfer to a copper clad.

(c) Preparing copper clad by polishing to avoid toner transfer defects.

(d) Transferring the toner to copper clad using a hot iron box.

(e) Etching the copper clad by submerging it in ferric chloride (FeCl3).

(f) Rinsing and drying the copper clad to remove the remaining FeCl3 chemical.

(g) Drilling the vias, mounting holes, and through holes.

(h) Applying finishing by tinning the PCB.

(i) Riveting the vias.

(j) Cutting the PCB into its respective sizes.

(iv) Board assembly involved the following key steps

(a) Using tweezers to place service mount devices (SMD) on the solder paste.

(b) Blowing hot air using a hot air gun to solder the SMD components into place.

(c) Placing through-hole components on the drilled holes and soldering them into place.

(d) Cutting the excess lead wires

(e) Testing to ascertain that the components are working well. This step is done using a multimeter in continuity mode.

(v) Powering the device

The device was powered externally by 12 V input, which was then regulated to 3.3 V and 5 V using low-dropout (LDO) voltage regulators. The 3.3V was fed into the sensors, PICO2040, and other low-power devices. The 5 V power was fed into the GSM module in the gateway module, whereas the 12 V goes directly to the solenoid valve, which is switched to ground by Darlington pair transistors. The power schematic is shown in Fig. 6 in the system design section of this document shows the power schematic.

Table 8 shows how LoRa was interfaced with ESP32, while Table 9 shows how LoRa was interfaced with Raspberry pi Pico.

Table 8: Interfacing LoRa and ESP32

LoRa Ra-02	ESP32
MISO	PIN19
NSS	PIN5
RST	PIN2
DIO0	PIN15
MOSI	PIN23
Ground (GND)	GND
Power (3.3V)	Power (3.3 V)
SCK	PIN18

Table 9: Interfacing LoRa and Raspberry Pi Pico

LoRa Ra-02	Raspberry pi Pico
MISO	GPIO16
NSS	GPIO17
RST	GPIO21
DIO0	GPIO20
MOSI	GPIO19
Ground (GND)	GND
Power (3.3V)	Power (3.3 V)
SCK	GPIO18

3.8.3 Software Requirements and Tools

(i) Arduino IDE

The Arduino IDE (Integrated Development Environment) is an official Arduino-based software used as a text editor and a compiler (Fezari & Al-Dahoud, 2018). Arduino IDE can also be used for error checking and for uploading codes to the microcontroller unit in IoT projects. Arduino IDE was used in this project's initial design and development phases. The

Arduino IDE was used to test the sensors and other components before development to ensure they functioned as desired.

(ii) MicroPython

MicroPython is a full Python compiler and an efficient implementation of the C-python language. It runs on bare metal and provides an interactive prompt for command execution. Figure 23 compares MicroPython-based architecture with standard development platform architecture. MicroPython aims to achieve compatibility with normal Python as much as possible. In addition to implementing a selection of core Python libraries, MicroPython includes modules such as "machine" for accessing low-level hardware (*MicroPython*, 2021). According to Gaspar *et al.* (2020), MicroPython applications on modern-day microcontrollers provide precise support for IoT applications development. By using proven and debugged libraries, MicroPython applications implement a regular standard high-level peripheral operation. Figure 23 compares MicroPython architecture and a classical development architecture.

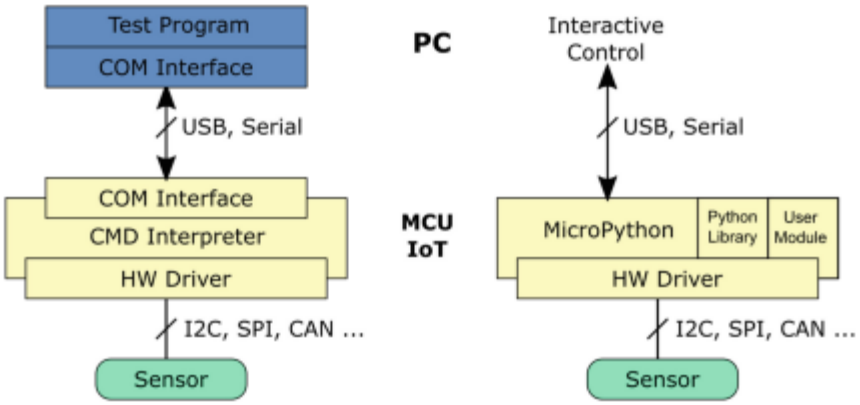


Figure 23: Comparison between MicroPython architecture and classical development platform (Gaspar *et al.*, 2020)

(iii) KiCad, Freecad, and Cura

The KiCad is an open-source PCB board development software that supports up to thirty-two wires and another thirty-two technical layers. It provides for schematic editing and PCB editing. Kicad was used to develop the printed circuit boards. Kicad circuits can be printed using lab-level economic printers (Kanagachidambaresan, 2021). The KiCad's 3D viewer makes inspection of the designed finished PCB easy. Freecad was used to design the PCB case

package, whereas Cura software was used to slice and generate Gcode, which was later transferred to a 3D printer to print out the case.

(iv) Things board cloud

ThingsBoard is an open-source IoT cloud platform for collecting, visualizing, and processing data, with features including device management and telemetry data access via access tokens. Although ThingSpeak was considered, it was not chosen due to its limited visualization support. Compared to Google Cloud IoT Core, ThingsBoard offers greater customization, deployment flexibility, and cost-effectiveness. It supports both on-premises and cloud deployments, providing better control over data and infrastructure, and benefits from continuous updates and innovations driven by its strong open-source community. Figure 24 shows an overview of the Thingsboard cloud platform.

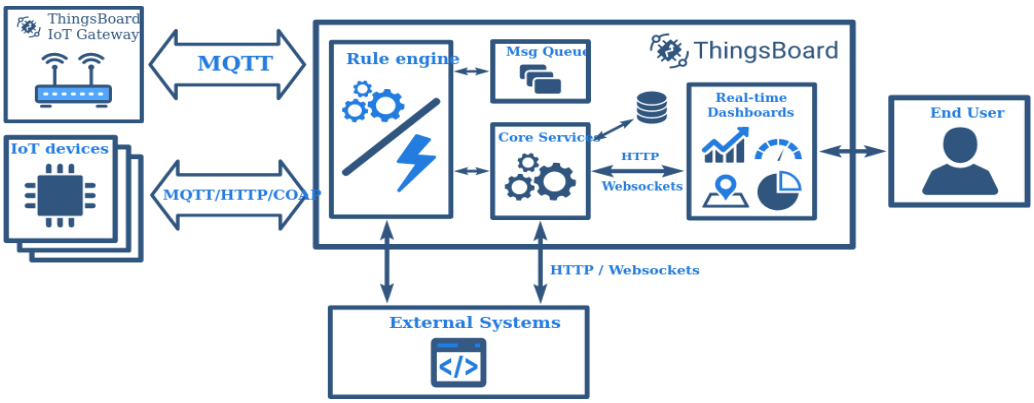


Figure 24: Overview of Thingsboard cloud platform

Some of the telemetry protocols that the platform supports include Hypertext Transfer Protocol (HTTP), Constrained Application Protocol (CoAP), and Message Queuing Telemetry Protocol (MQTT). The MQTT protocol was used for this project, and its details are covered later in this section. The Rule Engine is the primary logical system that allows the developers to filter, transform or enrich messages from their IoT devices. The rule engine is highly customizable and configurable to implement the processing of complex events. The rule Engine contains the rule engine message, Rule Node, Rule node Relation, and Rule Chain. The rule engine message is a data structure for representing various messages in the system. The rule node is the rule engine's main logical unit that filters incoming messages, performs an action, or communicates with an external system. Each rule node has a relation that generates message routes to other nodes. A rule chain is a logical group of rule nodes together with their relations. A rule chain processes incoming messages and can also forward the incoming messages to other node chains

for further processing. The things board cloud platform allows users to design and configure dashboards according to their desires. These dashboards are also customizable; hence, users can customize them to fit their desires (Henschke *et al.*, 2020). It provides widgets that can be modified according to the user's requirements. The widgets can be modified in shape, color, or form. Thingsboard also provides alarm settings, which is a noble feature since there is a high chance of collecting bad data during data collection. Such data include data that is out of range. In such instances, Thingsboard provides email messaging to the user's email address. Thingsboard platform also allows for custom REST API calls to the rule engine, which helps extend existing REST APIs and enrich REST API calls with other details such as device attributes to ensure fetching or richer data, and also provides custom API for widgets.

(v) Message Queuing Telemetry Protocol

The MQTT is a lightweight telemetry protocol that was used in this project. Telemetry is a technology that allows things to be measured and or monitored from a distance. Other telemetry protocols include web socket, Hypertext Transfer Protocol (HTTP), and Constrained Application Protocol (CoAP). The most popular among these protocols are HTTP since it is widely available and installed in almost all devices with TCPIP stack. While HTTP uses the request and response model, which is the most used model, MQTT uses the publish-subscribe model. Table 10 below shows a detailed comparison of MQTT and HTTP.

Table 10: Comparison between MQTT and HTTP

Parameter	MQTT	HTTP
Design orientation	Transfers data content as a byte array without checking what content it is.	Supports (MIME) to define a content type which is an advanced feature that is not required by constrained devices
Message model	It uses the publish-subscribe model, which is a loose coupling model. This is because users do not need to know of the existence of other devices but need to be aware of the content to be delivered or received.	It uses a request-response model, where users know all devices' addresses which is undesired for IoT devices.
Service levels	Implements three levels of Quality of Service (QOS) to ensure message delivery.	Doesn't implement QOS. Developers must implement other features for QOS.
Message size	A small size that includes only critical features with a minimal header. The smallest header message packet size is two bytes.	It is text-based, and the minimum packet header size is eight bytes.
Data distribution	Supports one-to-one (1-1), one-to-zero (1-0), and one-to-many (1-many) distribution models. This is made possible by MQTT's built-in distribution mechanism.	It is a point-to-point, and developers must create their mechanisms for distribution should they need it.
Complexity	It is less complex since it has a few message types: CONNECT, SUBSCRIBE, PUBLISH, DISCONNECT, and UNSUBSCRIBE.	It is more complex as it uses POST, GET, PUT, DELETE, TRACE, HEAD, and CONNECT codes.
Power consumption	High power consumption	Relatively low power consumption.

(vi) Remote procedural call (RPC) protocol

With RPC, one device can request a service from another device in the network without knowing the network details. A computer can call procedures in a different computer in the network using RPC calls. It is the most uncomplicated and protected web service. The Thingsboard platform form allows us to send remote RPC from server-side APPs to devices and vice-versa. We can send commands to devices and receive responses. As shown in Fig. 25,

client-side RPC originates from the device to the cloud. In contrast, the server side originates from the Thingsboard platform to the client.

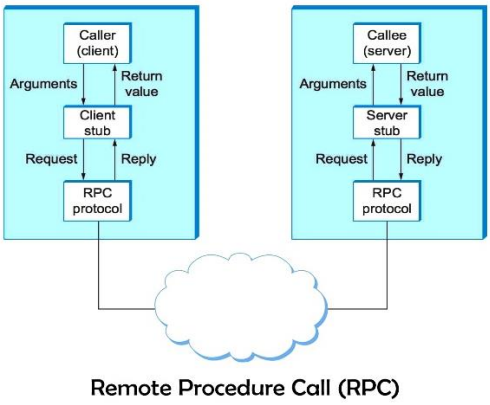


Figure 25: Representation of a client-side RPC

Server-side RPCs can be two-way or one-way, depending on the command sent. Two-way RPC calls require a response from the client, whereas one-way calls do not require responses. An example of one way RPC calls to a device is an RPC call to set a value temperature to a specific value. Here, the server doesn't need a response, while in the case of two ways where the server is requesting temperature values from the device, the server will require a response containing the requested temperature values.

The RPC requests are a JSON string whose root element is a Methodcall element which contains a MethodName element and a Params element. An RPC request's body should be a valid JSON object. The RPC request's MethodName identifies the method to be called, whereas the params element is the method's name. The method contains the method name and the JSON string, whereas the params contain method parameters and JSON objects.

Whenever a response is successful, and the procedure is executed correctly and returns results, the RPC response is similar to the request except that the "method call" is replaced with "methodresponse" element.

(vii) Open weather Application Programming Interface

An application programming interface (API) is a software interface that helps establish an online connection between computer programs. The connected computer systems usually comprise the data provider and end-user programs. The IoT developers have used weather forecasting APIs to get data for various parameters needed, including temperature and

humidity, among others. Numerous weather forecast APIs are freely available to developers for use. In this project, we used Open weather API to collect weather forecast information. OpenWeather API provides weather data, including current analysis and forecasts. This API uses machine learning technology to enhance its numerical models for weather prediction (Saleh *et al.*, 2016). The datasets from Openweather API are fetched as JSON and fed into the Thingsboard cloud server. In their experimental study on scheduled irrigation systems, Ajaz *et al.* (2022) reported that weather information improved the irrigation system's efficiency.

3.8.4 Firmware Development

The system is made up of three parts, the node, the Gateway, and the Thingsboard cloud platform**.** The firmware was developed using MicroPython. Before programming the boards, a test was performed to ensure they could connect to the computer and be debugged and programmed. The first step was to test individual sensors to ascertain that they worked as desired. The ADCs were configured to give accurate readings. Temperature sensors used the I2C line so that it could be read while executing other functions concurrently. The I2C is a synchronous protocol that switches packets and is a serial communication bus. Since the Gateway and the node communicate using LoRa, we used an asynchronous function (interrupt) to execute whenever there is an oncoming package. This ensured that there were no missed LORA data packets. The pinout diagram of WeMosESP32 (Gateway) and Raspberry Pi Pico (node) shown in Fig. 26 were obtained for cross reference as programming took place.

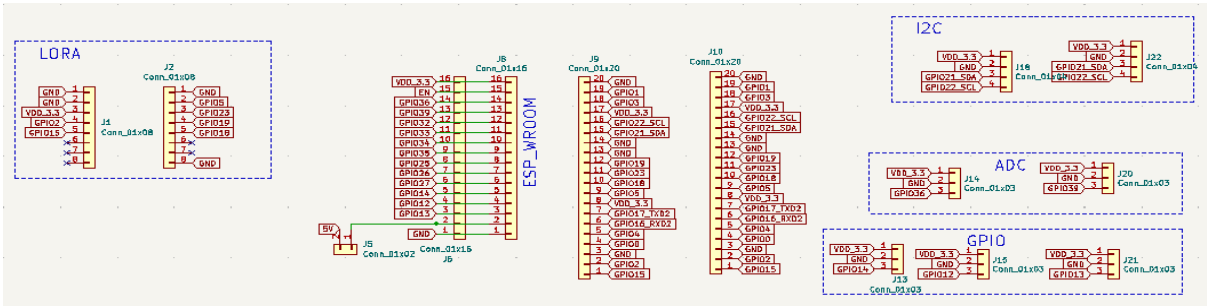


Figure 26: LoRa to ESP gateway pinout

The node collects the sensor data at intervals set by the user. For this project, the set interval was 10 seconds. The data is then packaged with variable names, e.g., humidity, soil moisture, and temperature, so the variables and values are identified and saved to respective database fields. Upon receiving packets by the Gateway, it displays the packet, the sending node, and the signal strength. This is to estimate the distance of a node to the Gateway. It also adds to it

the Received Signal Strength Indication (RSSI), Signal Noise Ratio (SNR), and ambient temperature, which is measured by an AHT10 (Humidity and temperature) sensor connected to its I2C line are all added to the packet and, after that, sent to the server. The Gateway also receives RPC (Remote Procedural Calls) calls to turn on the actuators; hence, farm irrigation takes place. These RPC calls are forwarded to the specific node address to turn ON or OFF the solenoid valves. When the solenoid valve is ON, the blue LED is turned on, while the Red LED is turned on whenever the solenoid valve is OFF. After the actuation of the solenoid valve, it sends back the response to the Gateway, which is then forwarded to the cloud server to ascertain that the event has been executed successfully. The system's code is attached in Appendix 1; Commenting on the different code pieces has been done to offer more insight into what each code performs.

The different protocols used in sending and receiving packets by the system's three parts, namely, the node, Gateway, and server, are LoRa and MQTT. The first protocol that enables data to be sent from the node to the Gateway is LoRa (Long Range Radio). This physical proprietary radio communication technique is based on spread spectrum modulation techniques derived from chirp spread spectrum technology. After that, the packets are added to data as captured earlier and sent to the server using the MQTT protocol (Message Queueing Telemetry Transport); this is a lightweight, publish-subscribe, machine-to-machine network protocol. It is designed for connections with remote locations that have devices with resource constraints or limited network bandwidth. Both of these protocols are two-way communication. Detailed information about MQTT and LoRa protocols are provided in this report's hardware and software requirements sections, respectively.

Figure 27 shows the flow of processing in the system's node. Once the system starts, the communication protocol (LoRa) is initialized. The system checks if LoRa is initialized; if not, LoRa has initialized again, and this cycle repeats itself until LoRa is initialized. Once LoRa is initialized, the node enters waiting mode, where it waits for data from the Gateway. As soon as data is received from the Gateway, the received data is processed depending on the message. The node checks this received data and acts on it by either turning on or off the solenoid valve. If there isn't any data received from the Gateway, the node enters delay mode, where it waits for ten seconds. After the ten seconds delay period lapses, the node collects and processes data from the sensor and, after that, sends the collected data to the Gateway. This event loops endlessly unless the system is powered off.

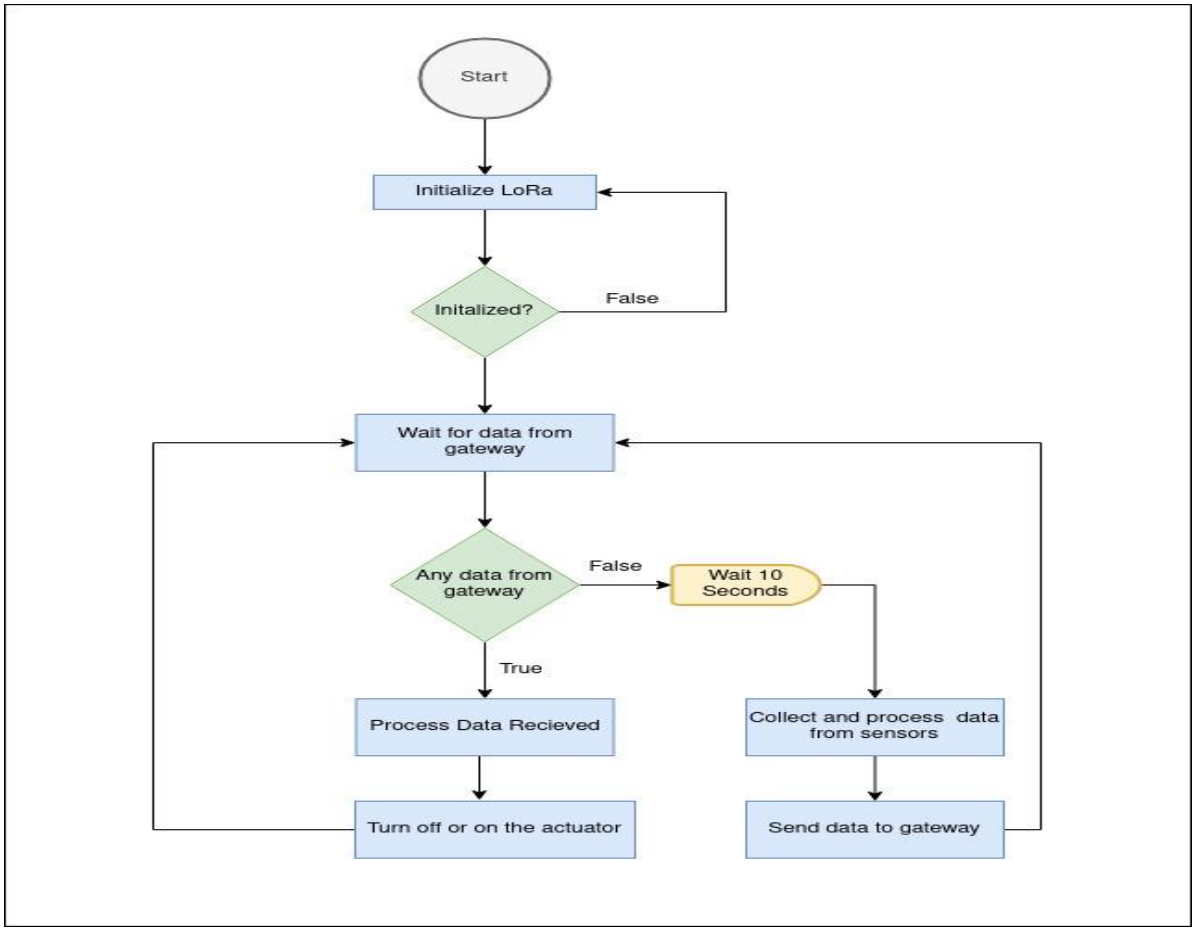


Figure 27: The system node's activities flow chart

Figure 28 shows the flow of processes in the Gateway of the system. Once the system starts, the Gateway attempts to connect to the Thingsboard cloud, and the Gateway repeats the attempt to connect until a connection to the cloud is established. Once a connection is established, LoRa is initialized, and this initialization attempt is repeated until LoRa is initialized. As soon as LoRa is initialized, the Gateway waits for data from both the cloud and the nodes. Whenever the Gateway receives data packets, it checks the data, whether it is from the node or the cloud. If the data doesn't originate from the cloud, the Gateway forwards it to the cloud, which acts as the server (indicated as the server in the flowchart). If the data originates from the cloud (server), the Gateway forwards the data to the specific node address.

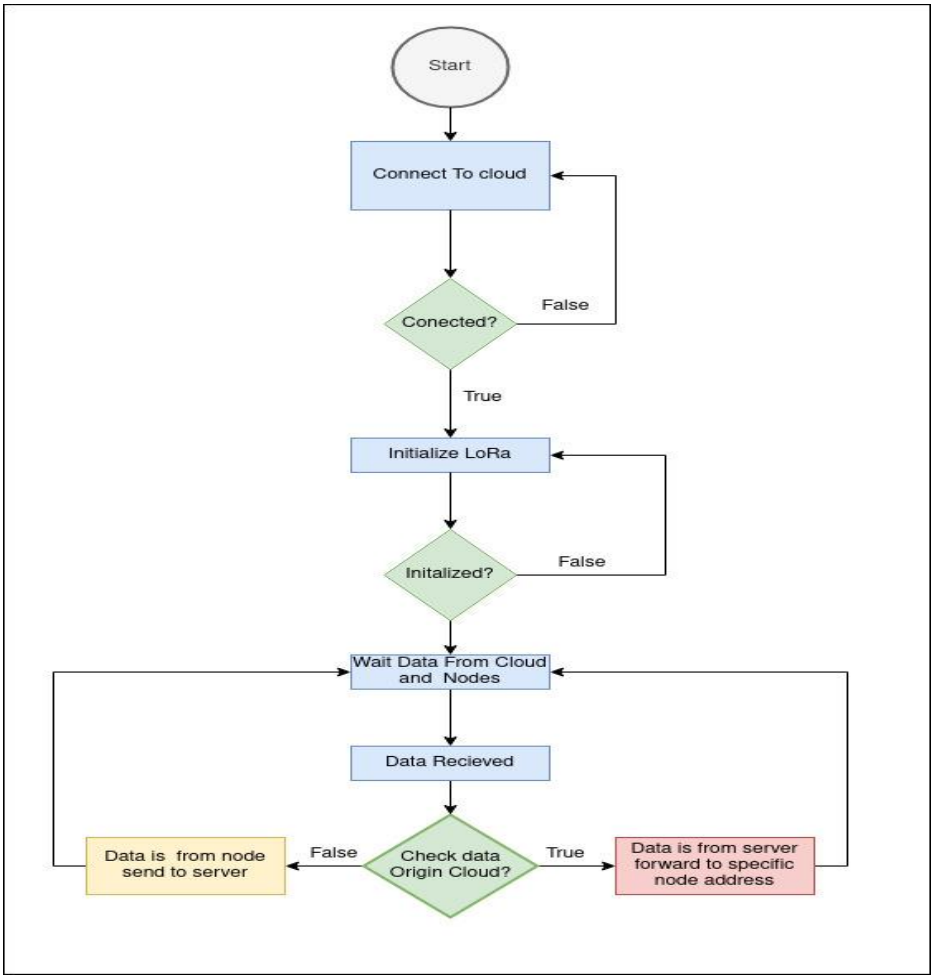


Figure 28: Gateway activities flowchart

Figure 29 above shows the flow of processes in the cloud platform. Immediately after the system starts, the cloud waits for data from the Gateway and the Openweather Application Programming Interface (API). When the cloud receives data, it stores it in a database and checks soil moisture value whether it is low. If the soil moisture level is low, the cloud checks data from the API; otherwise, it continues receiving and storing data. If the API data received indicates that there will be no rain within one hour, the cloud sends an open valve message to the Gateway. Suppose it will rain in the next hour; the cloud checks if the previous raincheck is less than three times. The soil moisture content is continuously checked every time the raincheck value is compared. If the raincheck value is three and the soil moisture level is low, the cloud server sends an open valve message to the Gateway. If the soil moisture value is high, the cloud server continues receiving data; this is the same case when the soil moisture is low; the API data indicates that there will be rain within an hour, and the rain check value is below three times.

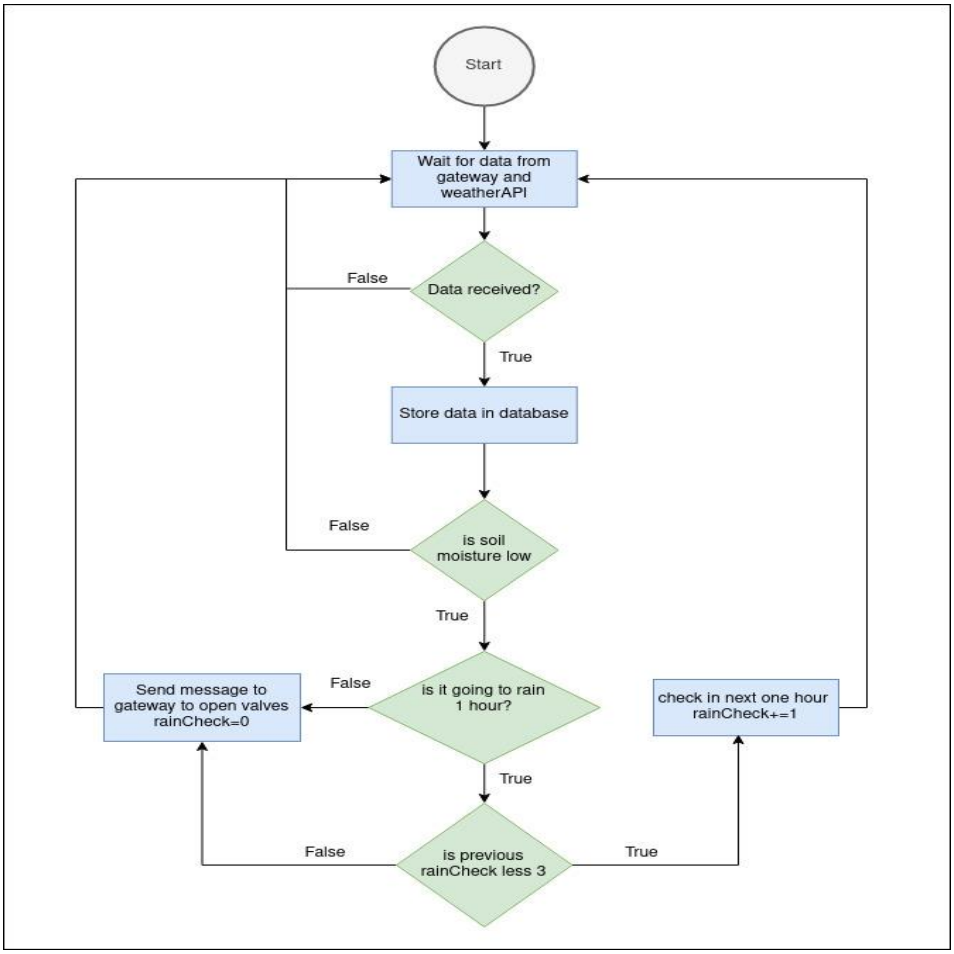


Figure 29: Cloud activities flow chart

3.8.5 Thingsboard Cloud Configuration

(i) Communication with the gateway

The Thingsboard cloud platform acts as the system's server, where all optimal controls are done. The cloud receives and sends data to the gateway through WIFI by implementing the MQTT protocol. The system also comprises a GSM, a backup communication module that can be used in areas where WIFI is unavailable. The MQTT was chosen for this system due to its lightweight characteristic, among others discussed in the software requirement section. The MQTT protocol functions by subscribing and publishing specific topics. Subscribing is arranging to receive data from a particular topic unto which each device subscribed receives data published to that topic. On the other hand, publishing is where messages or instructions are sent to subscribed devices on a topic. A topic is a placeholder where messages are published to be sent to subscribed devices.

(ii) Communication with the system nodes

The server (cloud) communicates with the system nodes using Remote Procedural Calls (RPC). The RPC message protocol has two structures: call and reply messages. Each node makes an RPC request to the server and receives a response (reply) containing the results of the procedure's execution. The RPC can match a reply message to a specific RPC (or request) message by providing a unique specification for the RPC message. The server-side cloud applications also send RPC calls to the nodes. The server-side RPC calls in our system include two-way RPC to turn ON or OFF the solenoid valves. The two-way RPC was implemented to ensure the node sends a response on the valve's status, which is a crucial visualization feature.

(iii) Implementation of controls

Thingsboard implements control using its rule engine. The rule engine provides for optimal control implementation by configuring rule chains. In a rule chain, one specifies the rules against which incoming data shall be passed to decide on actions to be implemented. Within the rule chains, thresh-hold values are set; these values that processed data received shall be compared to determine whether the irrigation valves shall be turned ON or OFF. In our case, the soil moisture value is compared with the threshold value, then the rainfall prediction from the OpenWeather API is also checked. With the soil moisture and rainfall prediction data, the rule chain decides whether the solenoid valve shall be turned ON or OFF. The overall system rule chain is shown in Fig. 30.

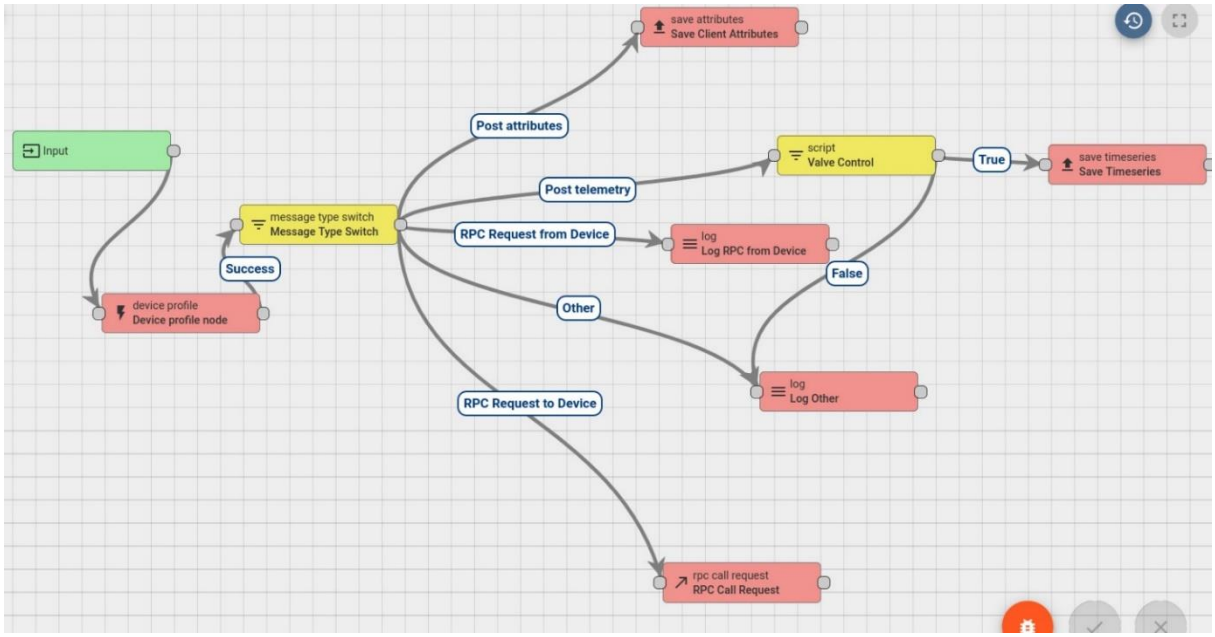


Figure 30: System control rule chain

The blocks represent rule nodes where the rules are specified using either JavaScript or Thingsboard scripting language (TSL). We used JavaScript to write codes to define the rules for each node, and Thingsboard translates the JavaScript codes to TSL and then implements them.

The input node is the endpoint in which data from the nodes is consumed. The device profile node processes device messages based on the device profile. The message Type Switch Node Routes incoming messages by Message Type and Sends messages with message types "Post attributes," "Post telemetry," "RPC Request," etc. via the corresponding chain; otherwise, the Other chain is used. The “message-type-switch” rule node modifies incoming data before storing it in the database. It also filters messages containing “post telemetry requests,” which are modified using “transformation” rule nodes.

(iv) Fetching weather data from OpenWeather API

Thingsboard cloud provides for interaction with RESTFUL APIs by implementing built-in transport control protocols. For this system, we implemented HTTP API reference. The server (cloud) receives weather data from OpenWeather API by implementing GET command and specifying the data to be received and the coordinates of the node's location encoded in its URL. The rule chain shown in Fig. 31 is defined for fetching data from the API.

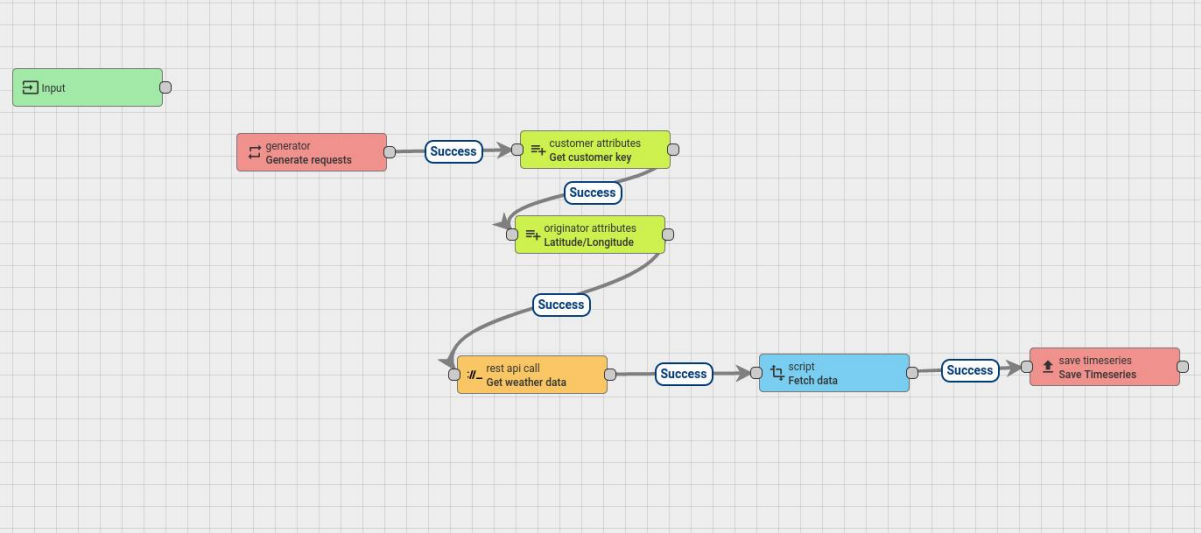


Figure 31: Rule chain for fetching weather data from OpenWeather API

The rule chain begins with generating requests node where the request is generated, and after that, the customer key is requested. After the validation of the customer key, the location attributes of the request message generator are specified. The API request call is then executed, and the resulting response will be the data of the specific location. The script node runs the script in appendix 4 to filter information of interest from the response in JSON file format, and forwards it to be saved in the database. The data fetched include weather temperature, rainfall, rain probability humidity.

(v) Data storage

Thingsboard cloud provides three databases, and subscribers specify the database they implement in their projects. The three databases are PostgreSQL, Timescale, and Cassandra. We implemented Cassandra because it supports Time to Live (TTL) with a maximum TTL of five years. The TTL value is specified for every entry recorded in a row. This TTL feature of Cassandra gives it an edge over the other databases as it enables subscribers to optimize storage consumption.

(vi) Dashboard configuration

The dashboard was populated with widgets to display various weather and soil parameters. The dashboard also contains a logical button widget to control the solenoid valve remotely. Upon clicking on this widget, the solenoid valve can either be turned ON or OFF, and this action overrides the automatic functionality of the system. The dashboard was also populated with a widget to switch from automatic mode to remote control mode. Other widgets include a rain

probability widget that displays the probability of rainfall, a valve status widget that displays the ON/OFF status of the solenoid valve, and light intensity, soil humidity, weather humidity, and soil temperature widgets display the respective parameters.

3.9 System Testing

The system was continuously tested as it was being developed. V-model was adopted for testing, where the system components and software units were tested incrementally following the steps in Fig. 32.

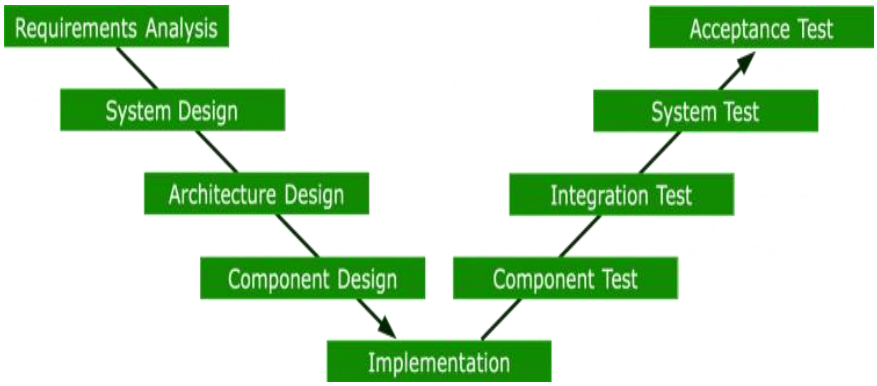


Figure 32: V-model diagram

3.9.1 Unit/ Component Testing

Unit testing, also called component testing, was performed during the firmware development phase. During the firmware development phase, individual hardware components were programmed and tested using Arduino IDE to ensure they produced the desired output. Unit testing was done by writing short test codes on Arduino IDE and implementing them on the components. Unit testing aims to validate that every unit of the developed software code gives the desired output for a given test input.

3.9.2 Integration Testing

Integration testing was done every time one or more modules were connected. During the firmware development phase, the pinouts of all components were obtained and tested for compatibility. After the firmware and hardware modules development, an integration test was done to ascertain that the modules communicated as expected. The Thingsboard configuration of different modules, including rule chains, nodes, and RESTFUL APIS, was continuously

tested during development. Integration testing aimed to expose faults in the interaction between integrated units.

3.9.3 System Testing

System Testing is the software testing performed on the complete integrated system to evaluate the system's compliance with the corresponding requirements. The aim was to detect any irregularity between the integrated units. System testing detects defects within both the integrated units and the whole system.

3.10 System Validation

A user acceptance testing exercise was conducted to ascertain that the system met the requirements of the end users. After completing the system demonstration exercise, the users present were asked to give their opinions according to each requirement listed through a google form. Ten passion fruit farmers participated in the validation exercise. The user acceptance testing form is attached in Appendix 3.

CHAPTER FOUR

RESULTS AND DISCUSSION

4.1 Results

This chapter presents the results of this project. The chapter begins by presenting results from the data collection phase and lists the requirements compiled from data analysis. It is in this chapter that we also present the results of system design and system development. Images of various system parts are shown together with the final prototype images. The chapter ends by presenting results from system testing and system validation testing.

4.1.1 Results From Focus Group Discussion

As indicated in the data collection section of this document, five respondents were involved in a focus group discussion. The focus group discussion comprised six sections. Among the questions in the transition section was, “Do you suppose the development of an automated irrigation system could solve the above challenges?”

All the respondents replied “YES” to this question, representing 100% agreement.

A summary of the responses to the question “with your experience, which are the major challenges facing farmers practicing irrigation in your jurisdiction?” are shown in Table 11.

As for the question, “What crops should we prioritize when developing an automated irrigation system for farmers in Uasin Gishu county?” Four of the five respondents mentioned passion fruit as their preferred crop to be considered, while one preferred tomatoes. This was an 80% preference for passion fruit against 20% for tomatoes.

The question “What are the major features of the automated system that should be considered when developing an automated irrigation system?” was meant to collect some information on some desired features of the proposed system. Among the most mentioned features were:

(i) The system should be safe for usage across the population.

(ii) The system should be easy to use.

(iii) The system should be reliable.

(iv) The system should be affordable.

(v) The system should provide remote control.

(vi) The system should provide information on the system status.

(vii) The system should automatically turn the valves ON or OFF depending on the soil moisture level.

(viii) Data from the focus group discussion was captured and analyzed according to the individual responses from the five respondents. Two of the five respondents for the focus discussion group were females, while three were male; this transitions to 40% female representation and 60% male representation.

Table 11: Analysis of challenges facing irrigation farmers captured from focus group discussion

Challenges facing irrigation farmers.	Frequency
Irrigating crops is time-consuming.	100%
Irrigation labor costs are high.	40%
Farmers do not have a way of knowing when their crops need water or not.	100%
Some farmers practice small-scale irrigation due to the scarcity of water	60%
Management of water in the irrigated farm requires farmers' Physical presence	80%

4.1.2 Results from the Questionnaire Administered Through Google Forms

The questionnaire attached in Appendix 2 was administered through the google platform through purposive sampling, and eleven responses were received. Seven of the eleven respondents to the questionnaire were selected from registered youth and women groups that practice open field irrigation farming,while five were farmers who practise open field irrigation farming independently without joining any group.

(i) Demographic representation of the respondents

Figure 33 shows the demographic representation of respondents as captured in google forms. 90.9% of the respondents were male, while 9.1% were female. This high representation of

males owes to the patriarchal cultural land ownership practice (Musangi, 2017), where the majority of land owners in homesteads are males.

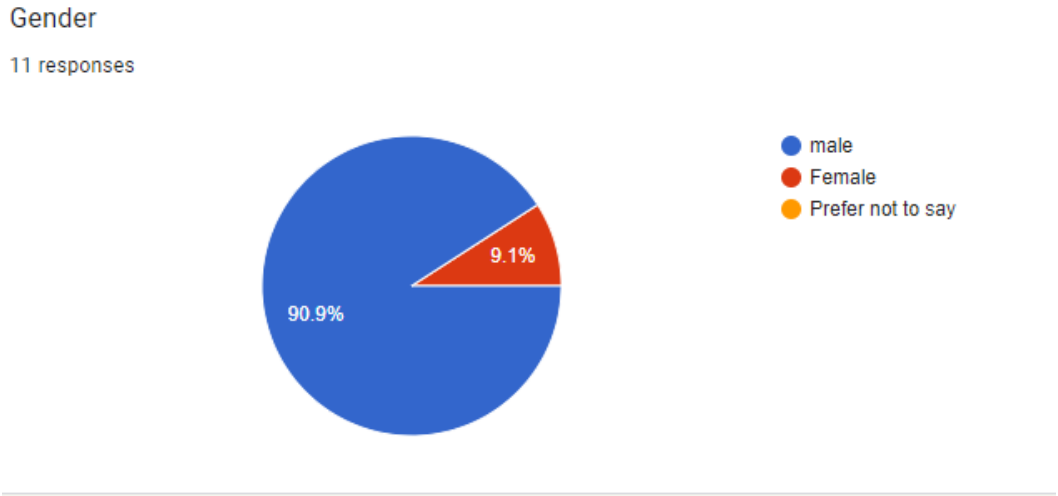


Figure 33: Gender representation of the questionnaire respondents

(ii) Sub-county representation

Respondents were drawn from all six sub-counties, as shown in Fig. 34.

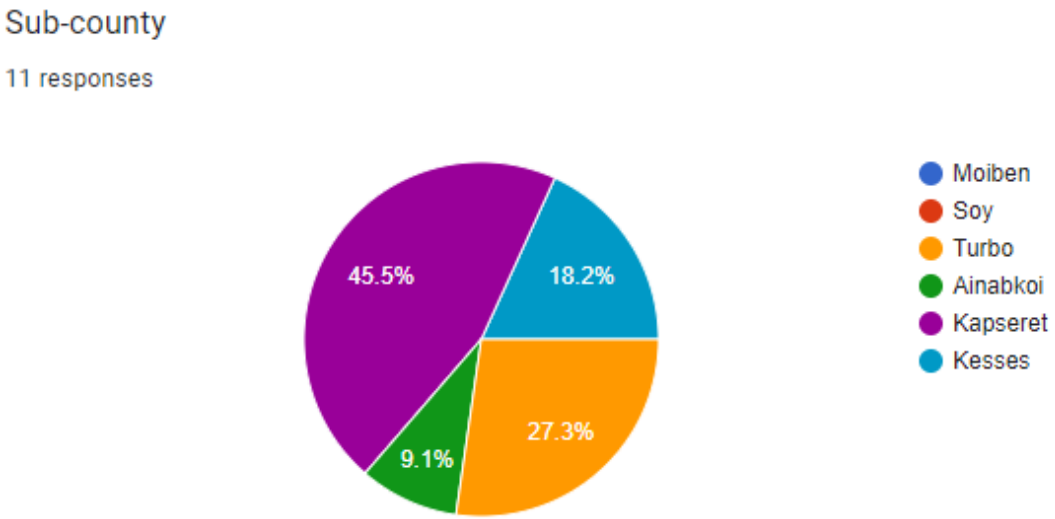


Figure 34: Respresentation of sub-counties of respondents

(iii) Type of irrigation practiced by respondents

The 54.5% of the respondents practiced open-field irrigation. This was the leading representation against 27.3% and 18.2% practicing mixed (open field and greenhouse) and

greenhouse irrigation, respectively. Figure 35 shows responses on the type of irrigation practice as captured from Google Forms.

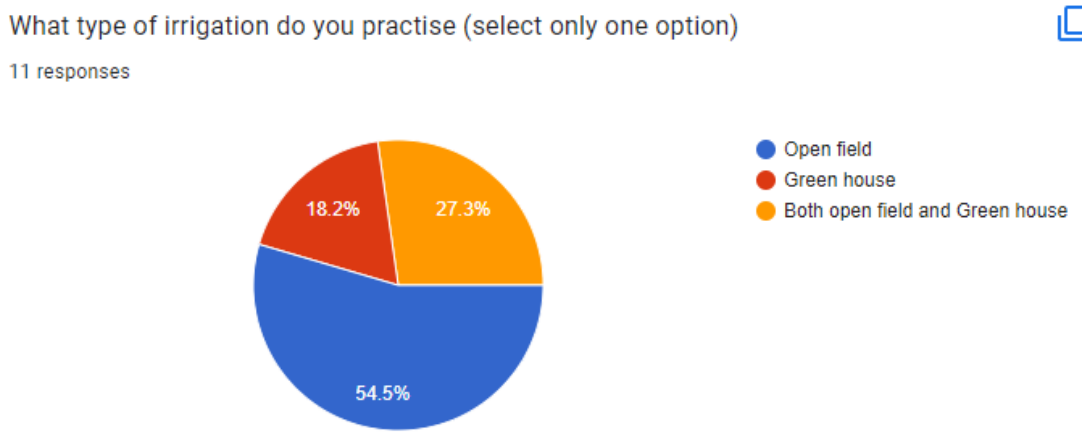


Figure 35: Response on type of irrigation practiced

(iv) When farmers irrigate their crops

Figure 36 below is a screenshot of the online questionnaire captured from google forms. The image shows an analysis of when farmers irrigate their crops, which indicates low precision.

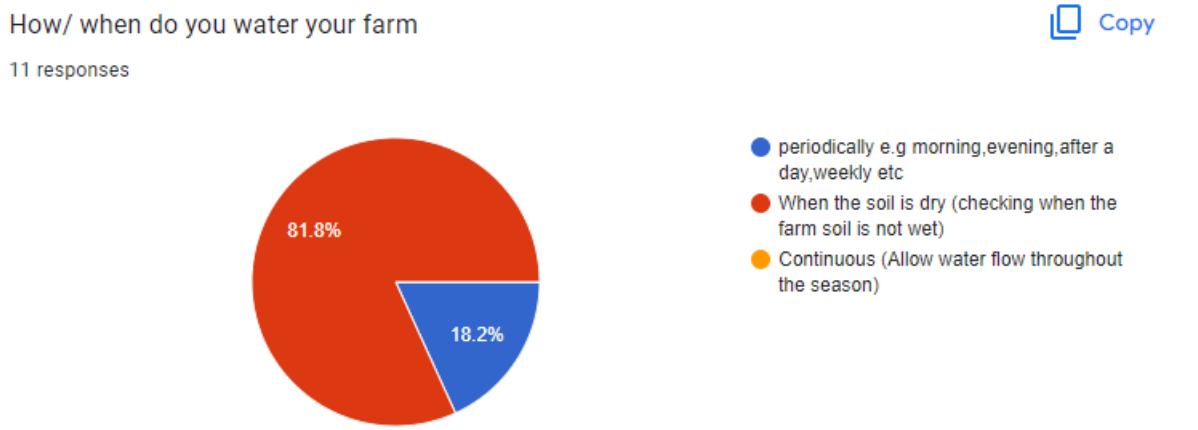


Figure 36: Analysis of when farmers irrigate their crops

(v) Challenges facing irrigation farmers

We used a five-point Likert scale to gather information from the respondents on various challenges faced in their day-to-day irrigation activities. Each statement had five options ranging from Agree, Strongly Agree, Not Sure, Disagree and Strongly Disagree.

We posed a question targeting farmers' need to know precisely when their crops needed water. Table 12 summarizes the responses to the opinions on the statement, "I do not have a way of knowing exactly when my crops need water." The percentage representation is rounded off to one decimal value.36.4% strongly agreed with the statement, while another 1% agreed. 27.3% of the respondents were unsure, while 27.3% disagreed.

Table 12: Results of the opinions on the statement "I do not have a way of knowing exactly when my crops need water"

Answer	Respondent's frequency	Percentage representation
Agree	1	9.0%
Strongly Agree	4	36.4%
Not Sure	3	27.3%
Disagree	3	27.3%
Strongly Disagree	0	0%

Another question was set to collect information on the demand for the physical presence of farm managers to ensure water is well managed. We presented the statement, "Management of water in the irrigated land requires my physical presence," where 45.5% strongly agreed, and 27.3% agreed. One (1) % of the respondents were not sure, while 18.2% disagreed. Table 13 presents a summary of responses to opinions on this statement.

Table 13: Results from opinion on statement "Management of water in the irrigated land requires my physical presence"

Answer	Respondent's frequency	Percentage representation
Agree	5	45.5%
Strongly Agree	3	27.3%
Not Sure	1	9.0%
Disagree	2	18.2%
Strongly Disagree	0	0%

We also asked for an opinion on real-time monitoring of the irrigated land. The results of the responses on views on real-time monitoring of the farm are presented in Table 14.

The statement "I do not have a way to ensure real-time monitoring of my irrigated farm" was included in the questionnaire.

Table 14: Results on the opinion on the statement "I do not have a way ensuring real-time monitoring of my irrigated farm"

Answer	Respondent's frequency	Percentage representation
Agree	6	54.6%
Strongly Agree	2	18.2%
Not Sure	2	18.2%
Disagree	1	9.0%
Strongly Disagree	0	0%

Another statement was presented to gather information about the over- and under-irrigation of land; the responses are summarized in Table 15. In the questionnaire, we included the statement "sometimes I over-irrigate or under-irrigate my farm" 45.5% of the respondents strongly agreed to the statement, while 27.3% also agreed to the statement 18.2% were not sure, whereas 9% disagreed with the statement.

Table 15: Results on opinions of the statement "sometimes I over-irrigate or under-irrigate my farm"

Answer	Respondent's frequency	Percentage representation
Agree	5	45.5%
Strongly Agree	3	27.3%
Not Sure	2	18.2%
Disagree	1	9%
Strongly Disagree	0	0%

(vi) Respondents' opinions on suggested system features

We also used a five-point Likert scale to determine respondents' views on proposed features. A summary of the results is presented in Table 16.

Table 16: Results showing respondents' opinions on some proposed features

Statement		Agree	Strongly Agree	Not Sure	Disagree	Strongly Disagree
I would like the system to open and close taps according to soil moisture content automatically.	Respondent's frequency	11	0	0	0	0
	Percentage representation	100%	0%	0%	0%	0%
I would like the system to send a notification in case of a malfunction	Respondent's frequency	9	2	0	0	0
	Percentage representation	81.8%	18.2%	0%	0%	0%
The system should record daily information for future reference and prediction	Respondent's frequency	9	2	0	0	0
	Percentage representation	81.8%	18.2%	0%	0%	0%
I would like to be able to access the system remotely	Respondent's frequency	10	1	0	0	0
	Percentage representation	91%	9.0%	0%	0%	0%
I would like to be able to control irrigation taps remotely	Respondent's frequency	11	0	0	0	0
	Percentage representation	100%	0%	0%	0%	0%

4.2 Identified Requirements

From research question 1 and specific objective 1, the system requirements were collected and identified using the focus discussion guide in Appendix 1 and the questionnaire in Appendix 2. The gathered data were analyzed, and system requirements were derived. The system requirements were divided into functional requirements and non-functional requirements. Functional requirements describe what a system or a system component must do, whereas non-functional requirements describe performance attributes of a system that determine the system's performance or quality.

4.2.1 Functional Requirements

Table 17 presents a list of essential functional requirements and their descriptions.

Table 17: Functional requirements

S/N	Functional requirement	Description
1.	Automatic valve operation.	The system shall automatically control valves.
2.	Remote control.	The system shall enables remote valve control via mobile phones.
3.	Rainfall prediction.	The system shall provide rainfall prediction statistics.
4.	Notification of critical events.	The system shall send users an email notification whenever a critical event occurs.
5.	Real-time soil parameter measurement.	The system shall measure soil parameters and provide real-time information visualization.
6.	Real-time soil parameter visualization.	The system shall provide real-time visualization of soil parameters.
7.	Authentication.	The system shall provide user login authentication.
8.	Visualization of irrigation valve status.	The system shall visualize the real-time status of the irrigation valves
9.	User registration.	The system shall register users.
10.	Real-time weather visualization.	The system shall display real-time weather events.
11.	Rainfall prediction.	The system shall predict and display rainfall probability.
12.	User removal.	The system shall be able to delete a user.

4.2.2 Non-Functional Requirements

The non-functional requirements of the system are presented in Table 18.

Table 18: Non-Functional requirements

S.N	Non-Functional requirements	Description
1.	Security.	The system shall be secure.
2.	Accuracy.	The system shall provide accurate real-time data.
3.	Power consumption.	The system shall consume low power.
4.	Reliability.	The system shall respond instantly upon turning the valves ON or OFF.
5.	Ease of use.	The system shall provide an easy-to-use dashboard.
6.	Safety.	The system shall be safe for use across the population.

4.3 System Design Results

This section covers the results of design stage of the second objective of this project. it presents the final designs of the developed system's printed circuit boards.

The figures below show the final design of the system's printed circuit boards. Figure 37 shows the final PCB design for the system node, while Fig. 38 shows the final design for the system's gateway.

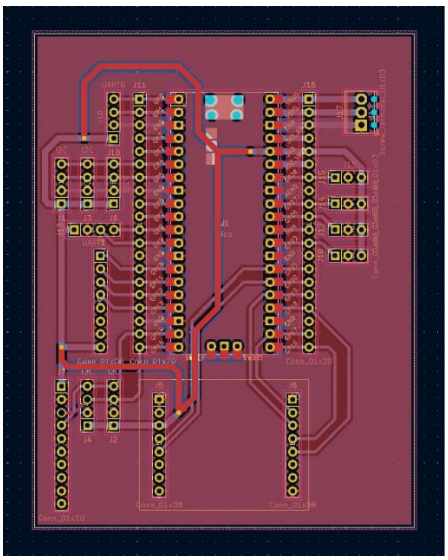


Figure 37: System's node PCB design

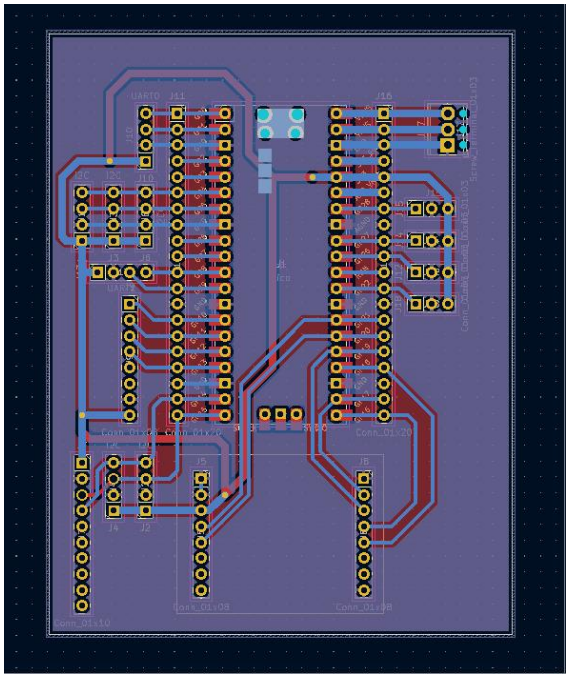


Figure 38: System's gateway PCB design

The general system architecture is shown in Fig. 5.

4.4 System Development Results

The system was developed from specific objective two and research question two, and the results are covered here. The system hardware was developed, as well as the firmware, and the Thingsboard cloud platform was configured to meet the system's requirements. The images of the developed system are also presented.

4.4.1 Hardware Development Results

The PCB boards were printed, and the hardware was assembled, as shown in Fig. 39 and Fig. 40. The gateway comprised an ESP32 microcontroller, LoRa module, OLED display, and a GSM module. The GSM module is a backup for Wi-Fi transmission in areas where WI-FI communication is unavailable. Each irrigation node comprised Raspberry pi Pico, LoRa module AHT20 sensors, DS18B20 temperature sensor, solenoid valve actuator, and LDR.

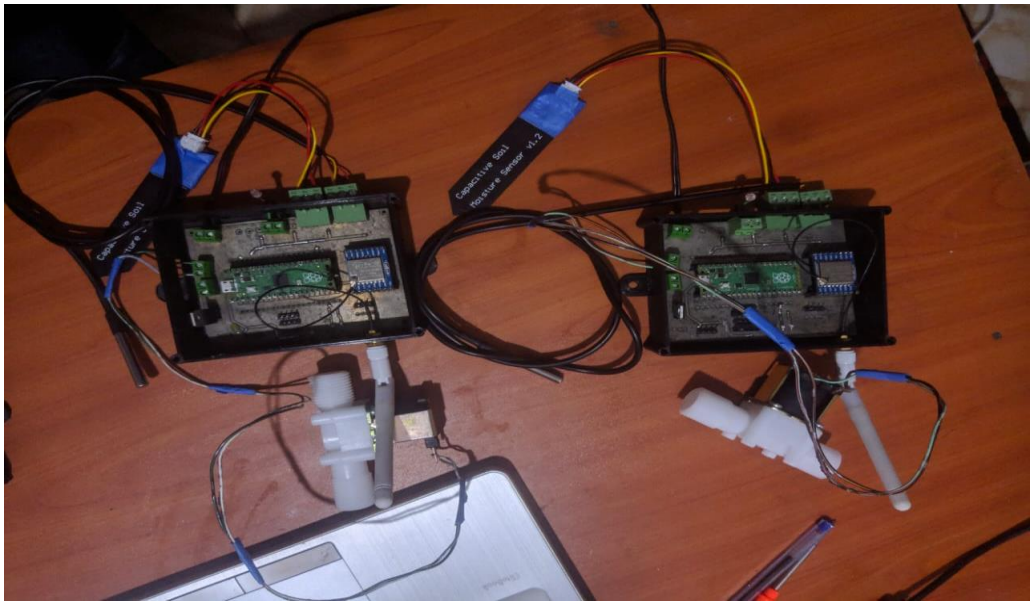


Figure 39: Assembled node hardware

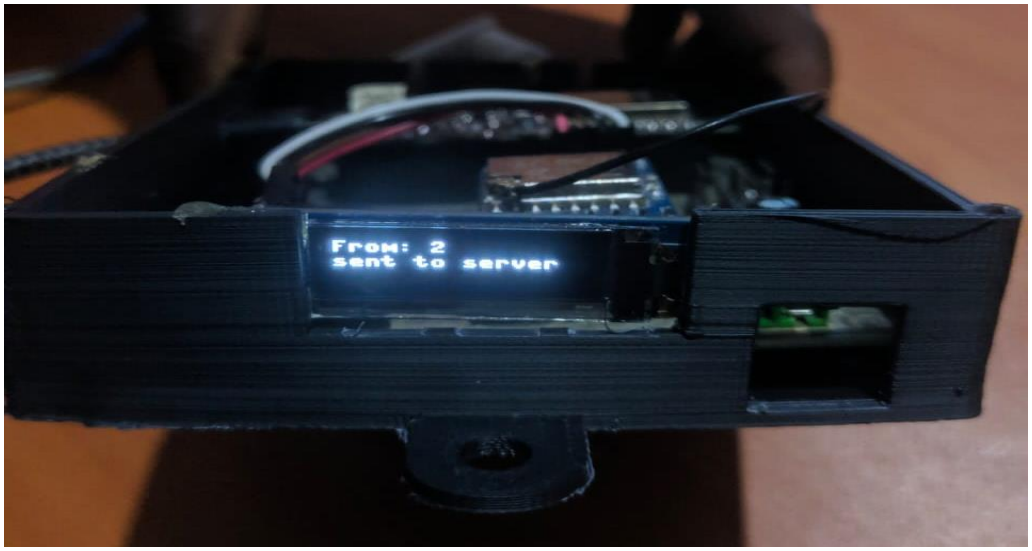


Figure 40: Assembled gateway hardware

(i) Hardware system when the solenoid valve is in OFF state

Figure 41 shows the hardware components when the solenoid valve is in an OFF state. When the solenoid valve is turned OFF, and there is no water flow, the system turns on the red LED indicator. This is specific to each node. In Fig. 41, the two nodes had their solenoid valves turned off. The turning ON and OFF of the valves can happen automatically and can also be done manually on the Thingsboard cloud interface.

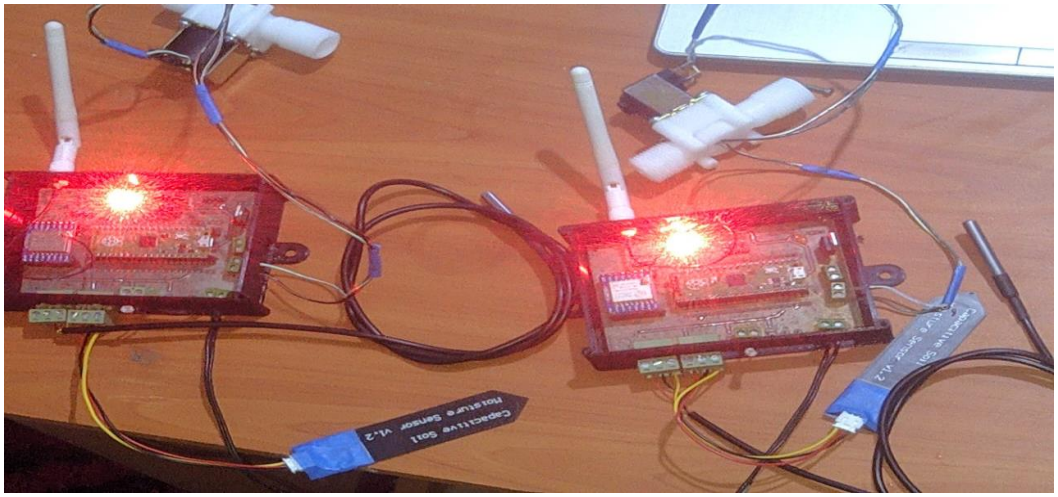


Figure 41: System hardware when the solenoid valve is in the OFF state

(ii) Hardware system when the solenoid valve is in ON state

Figure 42 shows the nodes when the valves are turned ON. The valves can be controlled from a mobile device by logging in to the user account. The ON status is also displayed in the Thingsboard dashboard live application, as shown in the mobile phone in Fig. 42, where the green icon indicator is turned ON.

Figure 42: System hardware when the solenoid valve is in the ON state

4.4.2 System Cloud Platform Results

This section presents results of Thingsboard cloud configuration. It presents the results of both mobile view and web cloud platform views.

(i) Administrator interface

The Thingsboard cloud platform was configured to meet the system requirements. Figure 43 shows the administrator screen of the Thingsboard configuration. An administrator can add tenants, who, in this case, are the farmers. These accounts are the platform from which all their devices and information are managed. In this project, we created an account for farmer "Winnie" that we used to demonstrate the results of the entire system development. An administrator can manage users, assets, and dashboards and even delete a user.

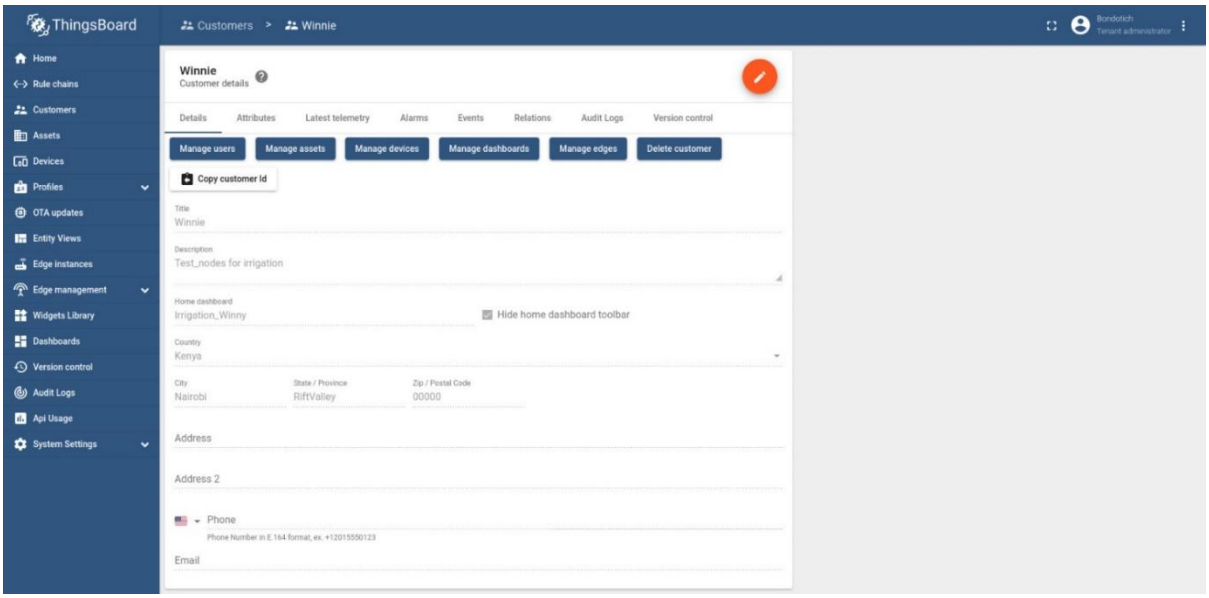


Figure 43: Creating tenant account in the Thingsboard cloud platform

(ii) System's Homepage

Figure 44 shows the homepage that is presented once the farmer logs in to the Thingsboard cloud account. The page displays two nodes marked as node one and node two that were used in this project. The location of the nodes is provided for in Thingsboard maps, where we keyed in the longitude and latitudes. When one clicks on a specific node, it gives a pop-up window showing the details of the particular node. One can also click on the "details" in the pop-up to access the dashboard.

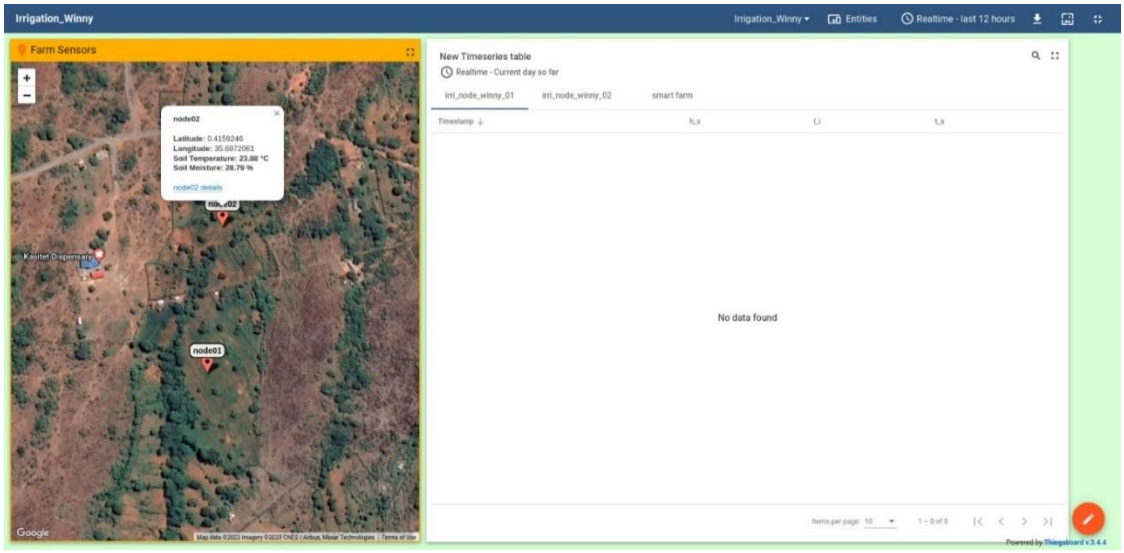


Figure 44: System's Thingsboard homepage

(iii) System Node Dashboard of the developed system

The dashboard details are specific to each node. Figure 45 shows the dashboard of node 1 in our system when the valve is OFF, while Fig. 46 shows node two dashboard when the system is ON. The dashboard provides visualization of valve status where the "valve status icon is gray whenever the solenoid valve is OFF and turns green whenever it is turned ON.

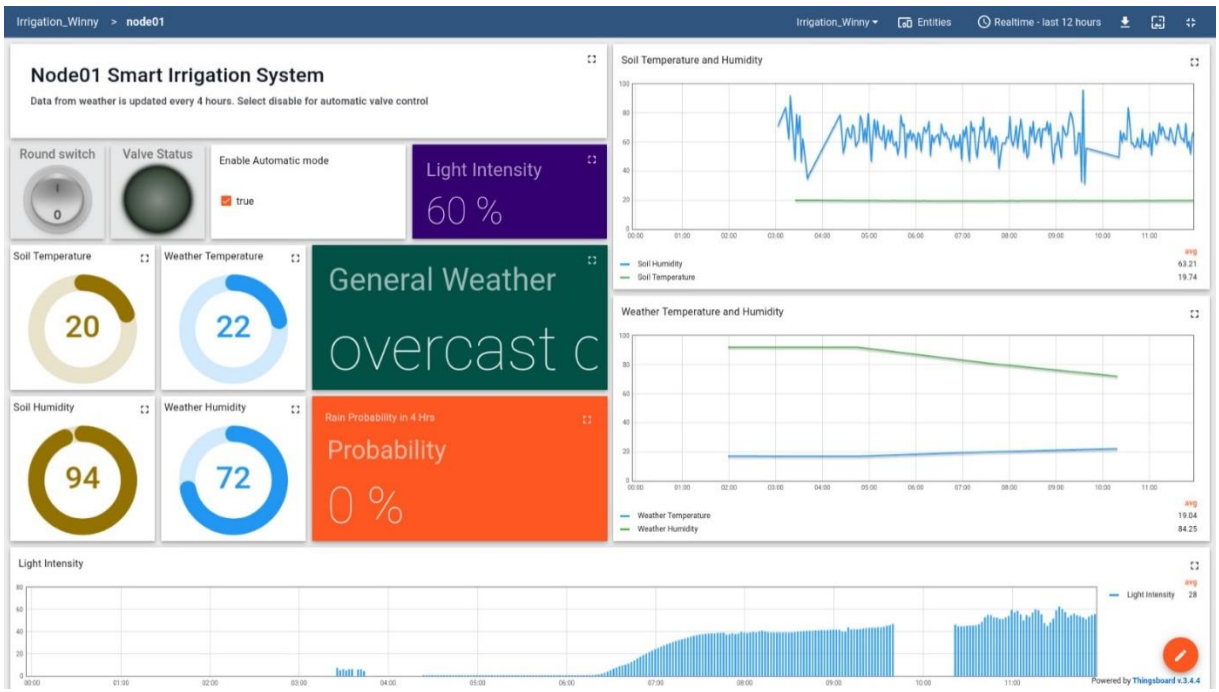


Figure 45: Node one dashboard when the valve is OFF image taken from administrator's account

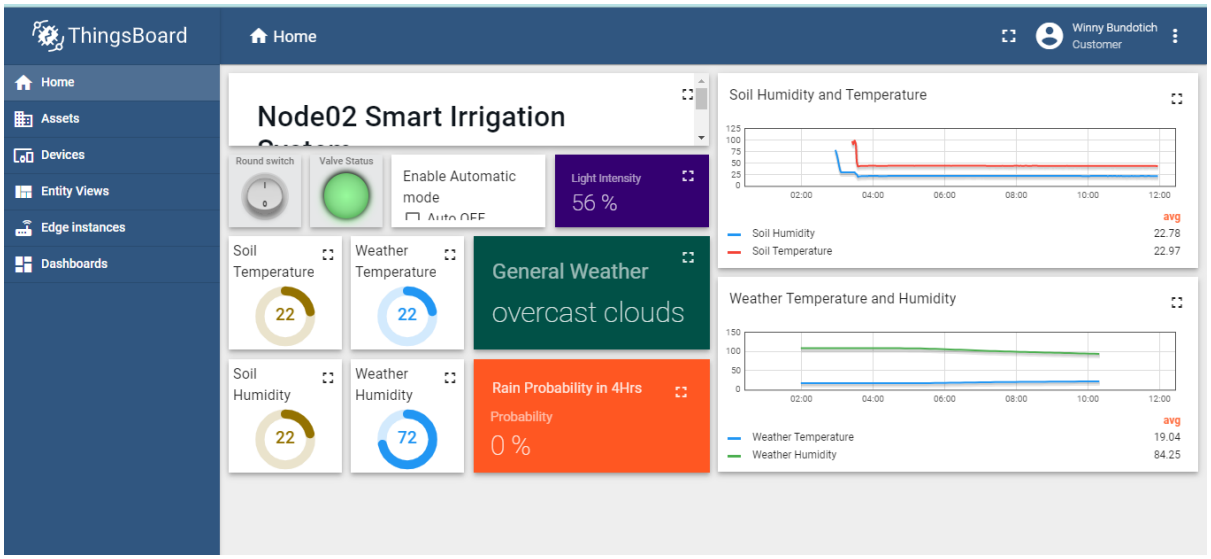


Figure 46: Node two Dashboard when the valve is ON image taken from user's account

The general weather data is fetched from OpenWeather API, while the soil data is fetched from the sensors. Graphical displays of parameters with time are also displayed in the dashboard.

(iv) Thingsboard Mobile view

A user can turn ON or OFF the solenoid valves remotely. This can be done through mobile phones by logging into their Thingsboard accounts. Figure 47 shows the Thingsboard mobile view with the valve round switch of the ON and OFF icon.

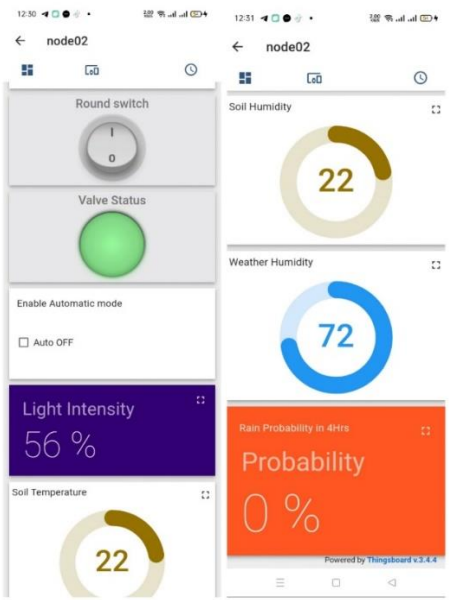


Figure 47: Node 2 mobile view switch on Thingsboard's Android live mobile application image taken from user's account

4.5 System Testing Results

This section presents the results from system testing as part of objective number two. It covers unit, intergration and system system testing results.

4.5.1 Unit Testing Results

Unit testing was done on each component; the results are presented in Table 19.

Table 19: Unit testing results

S/N	Test description	Results
1.	ESP32 shall successfully upload and debug code	PASS
2.	Raspberry pi Pico shall upload and debug code	PASS
3.	The Solenoid valve shall be switched ON and OFF	PASS
4.	AHT20 shall measure real-time humidity	PASS
5.	Soil moisture sensors shall take real-time soil moisture measurements	PASS
6.	The OLED display shall display data	PASS
7.	DS18B20 shall take real-time temperature measurements.	PASS

4.5.2 Integration Testing Results

Integration testing was done on integrated components and software, and the results are presented in Table 20.

Table 20: Integration testing results

S/N	Test description	Results
1.	LoRa shallconnect to ESP 32	PASS
2.	LoRa shall connect to Raspberry pi Pico	PASS
3.	ESP32 shall connect to Thingsboard through MQTT	PASS
4.	ESP 32 shall communicate with Raspberry pi Pico	PASS
5.	Raspberry pi Pico shall communicate with the soil moisture sensor	PASS
6.	Raspberry pi Pico shall communicate with soil AHT20	PASS
7.	Raspberry pi Pico shall communicate with the soil moisture sensor	PASS
8.	Raspberry pi Pico shall communicate with DS18B20	PASS
9.	Raspberry pi Pico shall communicate with a solenoid valve	PASS
10.	There shall be power continuity in the node PCB	PASS
11.	There shall be power continuity in the gateway PCB	PASS
12.	The OLED display shall display data at the gateway	PASS
13.	Thingsboard cloud shall receive data from OpenWeather API	PASS
14.	ESP32 shall turn LEDs ON/OFF	PASS
15.	The gateway shall send and receive data from the node	PASS
16.	The node shall send and receive data from the gateway	PASS
17.	The gateway shall send and receive data from the cloud	PASS
18.	The cloud shall send and receive data from the gateway	PASS

4.5.3 System testing Results

System testing was done on the entire system, and the results are presented in Table 21.

Table 21: System testing results

S/N	Test description	Results
1.	The system shall turn ON or OFF the irrigation valves automatically.	PASS
2.	The system shall provide remote control where users control the valves using a mobile phone.	PASS
3.	The system shall provide rainfall prediction statistics.	PASS
4.	The system shall send users an email notification whenever a critical event occurs.	PASS
5.	The system shall measure soil parameters and provide real-time information visualization.	PASS
6.	The system shall provide real-time visualization of soil parameters.	PASS
7.	The system shall provide user login authentication.	PASS
8.	The system shall visualize the real-time status of the irrigation valves status.	PASS
9.	The system shall register users.	PASS
10.	The system shall display real-time weather events.	PASS
11.	The system shall predict and display rainfall probability.	PASS
12.	The system shall be able to delete a user.	PASS

4.6 System Validation Results

After the successful development of the system, a user acceptance test exercise was done. The exercise pulled together farmers and the technical staff in the department of agriculture to participate in a demonstration exercise. After the demonstration of the prototype functions, the participants interacted with the system and gave their experience on the developed system through a Google survey questionnaire attached in Appendix 3. The results of the validation exercise are summarized below.

4.6.1 Gender Representation of Respondents

Twenty-seven respondents participated in the exercise where the system functionalities were demonstrated . Seven respondents (25.9 %) were females, while twenty (74.1%) were males, as shown in Fig. 48.

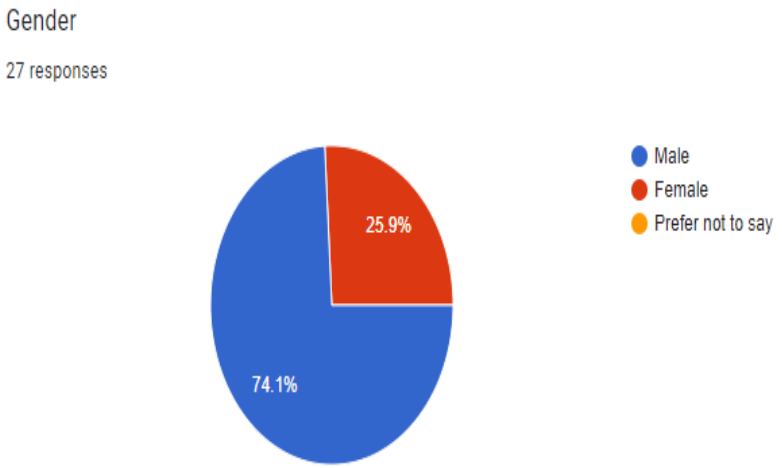


Figure 48: Gender representation of user acceptance testing respondents

4.6.2 Results from Validation Ease of Use, Weather Conditions Display Soil Moisture Level Display, Automatic System Functionalities

Figure 49 presents a summary of responses captured from the google survey questionnaire. Among the statements in our validation questionnaire was "the system is easy to use." This statement targeted to collect opinions on the system's ease of use. Five respondents agreed to this statement, and another twenty strongly agreed to the statement, while an insignificant one person was neutral and another disagreed.

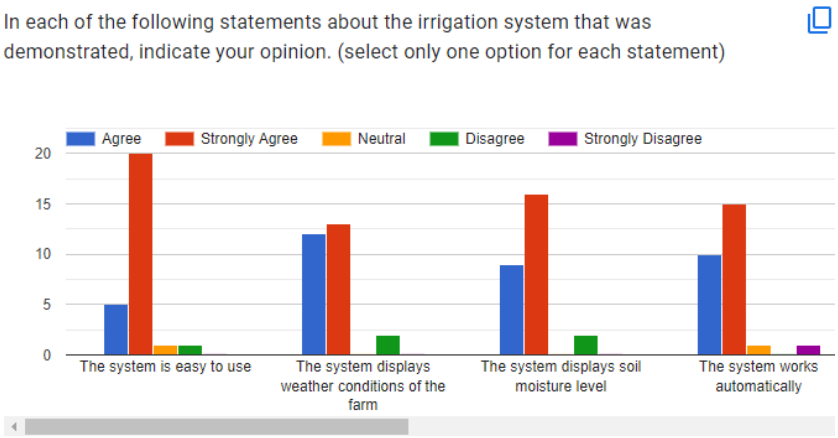


Figure 49: Statistic of system validation responses for four test statements

Another statement was set to inform on respondents' opinion about the system displaying weather conditions. On this statement, twelve respondents agreed, thirteen strongly agreed, and two disagreed.

On the statement that the system displays soil "moisture-level," nine and sixteen respondents agreed and strongly agreed, respectively, while two insignificant respondents disagreed.

On the automatic functioning of the system, ten and fifteen respondents agreed and strongly agreed, respectively. One respondent was neutral, while another respondent disagreed.

4.6.3 Validation Statistics Results for Other Features and Functionalities

Table 22 shows the results from respondents on other statements in the questionnaire.

Table 22: Validation questionnaire results statistics

SN	Test statement	Agree	Strongly Agree	Neutral	Disagree	Strongly Disagree
1.	The system shall be controlled remotely	9	16	1	1	0
2.	The system shall be safe for use	7	19	1	0	0
3.	The system shall reliable	7	17	3	0	0
4.	The dashboard interface shall be easy to interact with	10	16	0	1	0
5.	The system shall display rainfall probability	10	15	2	0	0

4.7 Discussion

The developed IoT-based smart irrigation system for efficient water management tried to satisfy the requirements of such a system. Requirements were collected analyzed and classified into functional and non-functional requirements which achieved objective one. To achieve the second objective, the system was developed where users can easily maneuver through the system with the rich, easy-to-use Thingsboard dashboard. The system's provision of remote control and visualization enriches the system's usability and applicability. The remote operation mode of the system required a user to have the email and password they registered with. The system satisfactorily controlled irrigation valves automatically and remotely. The system also satisfactorily displayed critical weather and soil parameters remotely. The system tried to satisfy all functional user requirements while correcting identified shortcomings during testing throughout the system development. The third objective was achieved thorough

validation that was carried out to ascertain the satisfaction of system users. The responses confirmed that the system fulfilled the user requirements and was received positively by the users. The system is therefore considered to have met the objectives of its development. The system provides a basis for implementation of artificial intelligence too analyze data collected and provide predictions among other AI opportunities.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

There have been significant advances in agricultural technologies in the recent past. Among them is the adoption and implementation of IoT in irrigation systems, as shown in the empirical literature review. Globally, irrigation systems have been automated to achieve precision agriculture, where every input gives the maximum value output. Most of these systems have focused on controlled environments, such as greenhouse setups leaving behind open-field irrigation. The developed IoT-based smart irrigation system for water efficiency addressed gaps in water-efficient smart systems for open-field irrigation where rainfall is critical. The system maps irrigation nodes and gets weather prediction data from OpenWeather RESTFUL API, used alongside soil moisture content to control irrigation valves. The system provides rich remote visualization of the farm environment through Thingsboard live android mobile application. The system was tested by farmers and technical staff of the agriculture department of Uasin Gishu County to confirm its user-friendliness, usability, reliability, and other core system functionalities. Economically, the system reduces labor costs involved in repetitive tasks and also improves crop yieds as it prevents waterlogging and under-irrigation. The saved water can also be utilized in expansion of irrigable land.

5.2 Recommendations

5.2.1 Implications on the Policy Developers

This project demonstrated how IoT should be implemented in irrigated farms across the county. The developed system contributes to the value chain by providing precise water input. The system can be advanced to include other essential features needed to monitor and evaluate irrigated farms within the county. This system can also be implemented for fertilizer control where liquid fertilizer is used. Policies can be developed to help manage IoT irrigation systems and encourage adoption by farmers.

5.2.2 Implications for Practitioners

Whereas this system demonstrated the viability of IoT-based smart farming, there are a lot of other gaps to tap into. The project leaves much room for developing holistic agricultural

precision systems that can monitor all input requirements and compare with the farm output, providing accurate analysis. Artificial intelligence can also be tapped into to give a prediction of the food situation within the county.

5.2.3 Future Work

The project leaves vast space for advancement. Artificial intelligence and machine learning can be used to analyze patterns in weather and soil parameters to predict farm harvests and provide early warning whenever there is a likelihood of severe drought.

Other sensors can also be included to measure soil nutrients content such as Nitrogen Phosphorus Potassium (NPK) ratio, plant growth, detect diseases, and detect pests. With more sensor data and machine learning, crop diseases can be predicted in advance and precautions taken.

REFERENCES

Ajaz, A., Berthold, T. A., Xue, Q., Jain, S., & Masasi, B. Scientific Irrigation Scheduling by Utilizing Weather Api and Open Source Modeling in Texas High Plains and Rio Grande Basin. *Available at SSRN 4104204*.

Aggarwal, N., & Singh, D. (2021). Technology assisted farming: Implications of IoT and AI. In *IOP Conference Series: Materials Science and Engineering*, 1022(1), 012080.

Al-Saqqa, S., Sawalha, S., & AbdelNabi, H. (2020). Agile software development: Methodologies and trends. *International Journal of Interactive Mobile Technologies*, *14*(11), 13269.

Bannon, L. (1997). Activity theory. *Retrieved October 24*, 2005.

Birch, I. (2018). *Agricultural productivity in Kenya: Barriers and opportunities*.

Brown, E. (2021). Raspberry Pi Goes MCU with Open-Spec Pico. *Circuit Cellar*. https://circuitcellar.com/newsletter/raspberry-pi-goes-mcu-with-open-spec-pico.

Choi, W., Chang, Y. S., Jung, Y., & Song, J. (2018). Low-Power LoRa signal-based outdoor positioning using fingerprint Algorithm. *ISPRS International Journal of Geo-Information*, *7*(11), 440.

Climate risk profile Uasingishu County. (2017). The Ministry of Agriculture, Livestock and Fisheries (MoALF).

Davis, F. D. (1989). Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly*, 319–340.

Evstigneev, V. P., Kuznetsov, P. N., Voronin, D. Y., & Naumova, V. A. (2022). Variant analysis of measurement components in environmental engineering. *IOP Conference Series: Earth and Environmental Science*, *981*(3), 032030.

Fezari, M., & Al Dahoud, A. (2018). Integrated development environment "IDE" for Arduino. *Wireless Sensors Network Applications*, 1–12.

Food and Agriculture Organization of the United Nations. (2018). *World food and agriculture: Statistical pocketbook 2018.*

Kreische, F., Ullrich, A., & Ziemann, K. (2015). Internet of Things. Using Sensors for Good: How the Internet of Things Can Improve Lives. *Federal Ministry for Economic Cooperation and Development*, 1-25.

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.

Gaspar, G., Fabo, P., Kuba, M., Dudak, J., & Nemlaha, E. (2020). Micropython as a development platform for IoT applications. *Intelligent Algorithms in Software Engineering: Proceedings of the 9th Computer Science on-line Conference*, 388-394.

Haule, J., & Michael, K. (2014). Deployment of wireless sensor networks (WSN) in automated irrigation management and scheduling systems: A review. In *Proceedings of the 2nd Pan African International Conference on Science, computing and Telecommunications*, 86-91.

Haule, J., & Michael, K. (2014). Designing and Simulation of an Automated Irrigation Management System Deployed by using Wireless Sensor Networks (WSN) (Dectoral Dissertation, NM-AIST).

Henschke, M., Wei, X., & Zhang, X. (2020). Data visualization for wireless sensor networks using ThingsBoard. *29th Wireless and Optical Communications Conference*, 1–6.

Hobby, S. (2021). Capacitive soil moisture sensor v1.2 Arduino code | Step by step instructions. *SriTu Hobby*. https://srituhobby.com/capacitive-soil-moisture-sensor-v1-2-arduino-code.

Hollan, J., Hutchins, E., & Kirsh, D. (2000). Distributed cognition: Toward a new foundation for human-computer interaction research. *ACM Transactions on Computer-Human Interaction*, *7*(2), 174–196.

Kanagachidambaresan, G. R. (2021). Introduction to KiCad Design for Breakout and Circuit Designs. In *Role of Single Board Computers in Rapid IoT Prototyping*, 165–175.

Karina, F. Z., & Mwaniki, A. W. (2011). *Impact of the Economic Stimulus Programme and Longterm Prospects for Food Security in an Era of Climate Change.*

Kelly, T., & Dunand, E. (2021). *Overview of digital development in the Horn of Africa*.

LANGAT, J. (2019). *Modeling Optimal Yield Of Maize Under Deficit Irrigation And Nutrients Levels In Uasin Gishu County, Kenya* (Doctoral dissertation, University of Eldoret).

Lang'at, J. (2019). *Modeling optimal yield of maize under deficit irrigation and nutrients levels in Uasin Gishu county, Kenya* (Doctoral Thesis, University of Eldoret.

Maier, A., Sharp, A., & Vagapov, Y. (2017). Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things. *Internet Technologies and Applications*, 143–148.

Mavrommati, I., Birbilis, G., & Darzentas, J. (2013). A conceptual framework for the design of IoT architectures that support end-user development. *Networking Science*, *3*(1), 71–81.

Mbandi, J., & Kisangari, M. (2020). Data Collection Using Wireless Sensor Networks and Online Visualization for Kitui Kenya. *African Handbook of Climate Change Adaptation*, 1–13.

MicroPython. (2021). MicroPython. https://micropython.org.

Mirza, M. S., & Datta, S. (2019). Strengths and Weakness of Traditional and Agile Processes-A Systematic Review. *Journal of Software*, *14*(5), 209–219.

Gaafar, R. (2014). Women's land and property rights in Kenya. *Center for Women's Land Rights. www. landesa. org*.

Mwaniki, E. B. (2010). *Quantifying Available Water Resources in Kenya*.

Nigussie, E., Olwal, T., Musumba, G., Tegegne, T., Lemma, A., & Mekuria, F. (2020). IoT-based irrigation management for smallholder farmers in rural sub-Saharan Africa. *Procedia Computer Science*, *177*, 86–93.

Njuu, K. (2016). *Sensor based method for water monitoring in smallholder irrigated agriculture Tanzania* (Doctoral dissertation, NM-AIST).

Olatunji, K. A., Oguntimilehin, A., & Adeyemo, O. A. (2020). A Mobile Phone Controllable Smart Irrigation System. *International Journal of Advanced Trends in Computer Science and Engineering*, *9*(1), 279–284.

Rawal, S. (2017). IOT-based smart irrigation system. *International Journal of Computer Applications*, *159*(8), 7–11.

Saleh, M., Elhajj, I. H., Asmar, D., Bashour, I., & Kidess, S. (2016). Experimental evaluation of low-cost resistive soil moisture sensors. *2016 IEEE International Multidisciplinary Conference on Engineering Technology*, 179–184.

Silva, P. (2015). Davis' technology acceptance model (TAM)(1989). *Information Seeking Behavior and Technology Adoption: Theories and Trends*, 205–219.

Togneri, R., Kamienski, C., Dantas, R., Prati, R., Toscano, A., Soininen, J. P., & Cinotti, T. S. (2019). Advancing IoT-based smart irrigation. *IEEE Internet of Things Magazine*, *2*(4), 20–25.

Ullah, R., Abbas, A. W., Ullah, M., Khan, R. U., Khan, I. U., Aslam, N., & Aljameel, S. S. (2021). EEWMP: An IoT-Based Energy-Efficient Water Management Platform for Smart Irrigation. *Scientific Programming*, *2021*(1), 5536884.

Zourmand, A., Kun Hing, A. L., Wai Hung, C., & AbdulRehman, M. (2019). Internet of Things (IoT) using LoRa technology. *2019 IEEE International Conference on Automatic Control and Intelligent Systems*, 324–330.

APPENDICES

Appendix 1: Data Collection Questionnaire

Development of an automated irrigation system for efficient water management data collection questionnaire.

Dear respondent,

I am a Masters's student at Nelson Mandela Institution of Science and Technology (NM-AIST) and currently developing an automatic smart irrigation system for passion fruit farmers in Uasin Gishu county as part of my research studies. The system aims to assist passion fruit farmers in achieving more efficiency in water management in their irrigated farms by automating irrigation activities. Some of the activities the proposed system seeks to automate include automatic turning on and off irrigation taps depending on the amount of water in the soil (Soil moisture), temperature, humidity, and rainfall probability.

Kindly help by spending some minutes of your time responding to this questionnaire. This survey is conducted among farmers within Uasin Gishu county.

NB This survey questionnaire does not collect/record any personal information. The information gathered in this questionnaire will be used only for this research and not for any other purpose.

Thank you for your participation and contribution to this research.

1. Gender (Mark only one.)
   - Male
   - Female
   - I prefer not to say
2. Sub-county * (Mark only one.)
   - Moiben
   - Soy
   - Turbo
   - Ainabkoi
   - Kapseret

- Kesses

3. What type of irrigation do you practice? (select only one option)

- Open field
- Greenhouse
- Both open field and Greenhouse

4. What type of passion fruit do you cultivate (Select all that apply).

- Yellow passion fruit
- Purple passion fruit

5. How long have you practiced passion fruit farming (select only one option)

- < 1 Year (under one year)
- 1 - 5 Years (Between one and five years)
- 6 -10 Years (Between six and ten years)
- >10 Years (More than ten years)

6. Where is your farm located (select only one option)?

- Urban (Near town)
- Rural (Far from town)

7. What is your preferred type of irrigation (select only one option)? *

- Drip
- Sprinkler
- Other_____

8. How/ when do you water your farm *

- periodically, e.g., morning, evening, after a day, weekly, etc.
- When the soil is dry (checking when the farm soil is not wet)
- Continuous (Allow water flow throughout the season)

9. In each of the statements below, indicate how irrigation of passion fruit has enhanced your farming practice (select only one option for each statement)

	Agree	Strongly Agree	Neutral	Disagree	Strongly Disagree
Practising Irrigation has improved passion fruit yield	⬭	⬭	⬭	⬭	⬭
Practising Irrigation has improved farmers' living standards	⬭	⬭	⬭	⬭	⬭
Practising Irrigation has improved overall soil fertility in irrigated land	⬭	⬭	⬭	⬭	⬭
Practising Irrigation ensures easy weed control	⬭	⬭	⬭	⬭	⬭
Practising Irrigation has enabled farming throughout the year without depending on the rainfall	⬭	⬭	⬭	⬭	⬭

10. In each of the following statements about the current irrigation system that you * practice, indicate your opinion. (select only one option for each statement)

	Agree	Strongly Agree	Neutral	Disagree	Strongly Disagree
Irrigating crops is time-consuming, and sometimes, Idon't manage to irrigate my plants	⬭	⬭	⬭	⬭	⬭
Irrigation labor costs are high	⬭	⬭	⬭	⬭	⬭
I do not have a way of knowing exactly when my crops need water or not	⬭	⬭	⬭	⬭	⬭
Management of water in the irrigated farm requires my Physical presence	⬭	⬭	⬭	⬭	⬭
I do not have a way of ensuring real-time irrigation of my farm	⬭	⬭	⬭	⬭	⬭
I do small-scale irrigation due to the scarcity of water	⬭	⬭	⬭	⬭	⬭
	⬭	⬭	⬭	⬭	⬭

11. In each of the following statements, indicate features you would like to be included in an automated irrigation system. (select only one option for each statement)

	Yes	No
I would like to get an SMS notification when Irrigation valves (taps)are on or off or when a system malfunction is detected	⬭	⬭
I would like the system to open and close taps according to soil moisture content automatically	⬭	⬭
I would like to receive email notifications whenever taps are on/off and when there is a system malfunction	⬭	⬭
The system should record daily information for future reference and prediction	⬭	⬭
I would like to be able to access the system remotely	⬭	⬭
I would like the system to send my farm's information to the county agricultural extension officers	⬭	⬭
I would like the system to be able to predict rainfall	⬭	⬭
I would like to be able to control irrigation taps remotely	⬭	⬭

12. List any reasons you would not prefer an automated irrigation system. (List all reasons that apply)

Appendix 2: Focus Group Discussion Guide

DEVELOPMENT OF AN IoT-BASED SMART IRRIGATION SYSTEM FOR EFFICIENT WATER MANAGEMENT IN UASIN GISHU COUNTY

Focus group discussion Questions

Section A: Opening question

Kindly provide me with your details, including your name, department, position in your department, and current roles in your office.

Section B: Introductory question

What is your opinion on using automation of irrigation systems in Uasin Gishu county?

Section C: Transition question

a) With your experience, which are the major challenges facing farmers practicing irrigation in your area of jurisdiction?
b) Could the development of an automated irrigation system solve the above challenges?

Section D: Key questions

a) What are the standards and guidelines to be followed during the development and implementation of an automated irrigation system in Uasin Gishu county?
b) What crops should we prioritize when developing an automated irrigation system for farmers in Uasin Gishu county?
c) What are the major features of the automated system that should be considered when developing an automated irrigation system?

Section E: Concluding questions

a) What advice would you give the stakeholders developing the automated irrigation system?

Section F: Final question

Is there anything you can add to the discussion we have had?

Appendix 3: User Acceptance Testing Questionnaire

Validation of the developed "IoT-based smart irrigation system for efficient water management."

Dear respondent,

I am a Master's student at the Nelson Mandela Institution of Science and Technology (NM-AIST). As part of my research studies, I developed an automatic smart irrigation system for passion fruit farmers in Uasin Gishu county. The system aimed to assist passion fruit farmers in achieving more efficiency in water management in their irrigated farms by automating irrigation activities. Some of the activities the proposed system automated include the automatic turning ON and OFF of irrigation taps depending on the amount of water in the soil (Soil moisture) and the probability of rainfall. The system also visualizes other farm information, such as temperature, light intensity, and weather humidity. After a physical demonstration of the system functionalities, kindly help by spending some minutes responding to this questionnaire. NB This survey questionnaire does not collect/record any personal information. The information collected in this questionnaire will be used only for this research and not for any other purpose. Thank you for your participation and contribution to this research.

5. Gender *(Mark only one.)

- o Male
- o Female
- o I prefer not to say

6. Sub-county * (Mark only one.)

- o Moiben
- o Soy
- o Turbo
- o Ainabkoi
- o Kapseret

- Kesses

7. In each of the following statements about the demonstrated irrigation system, indicate your opinion. (select only one option for each statement)

	Statement	Agree	Strongly Agree	Neutral	Disagree	Strongly Disagree
1	The system is easy to use	⬭	⬭	⬭	⬭	⬭
2	The system displays the weather conditions of the farm	⬭	⬭	⬭	⬭	⬭
3	The system displays the soil moisture level	⬭	⬭	⬭	⬭	⬭
4	The system works automatically	⬭	⬭	⬭	⬭	⬭
5	The system can be controlled remotely	⬭	⬭	⬭	⬭	⬭
6	The dashboard interface is easy to interact with	⬭	⬭	⬭	⬭	⬭
7	The system is safe for use	⬭	⬭	⬭	⬭	⬭
8	The system is reliable	⬭	⬭	⬭	⬭	⬭
9	The system can display rainfall probability	⬭	⬭	⬭	⬭	⬭
10	The system can send an email notification whenever there is an alarm The system can send an email notification whenever there is an alarm	⬭	⬭	⬭	⬭	⬭

Appendix 4: System Codes

Gateway code

```
from time import sleep
from ulora import LoRa, ModemConfig, SPIConfig
import json
import machine
import time
from machine import Pin, PWM ,SoftI2C
from umqtt.robust import MQTTClient
from time import sleep
import random
import network,sys
import ssd1306

##################MQTT###################

WIFI_SSID = 'Winny'
WIFI_PASS = 'ac21301963'
'''
WIFI_SSID = 'Winny'
WIFI_PASS = 'Mamba0ut'
'''

# WIFI_SSID = 'Winny'
# WIFI_PASS = ' Mamba0ut '

state = 0
```

```
# Lora Parameters
RFM95_RST = 2
RFM95_SPIBUS = SPIConfig.esp32_2
RFM95_CS = 5
RFM95_INT = 15
RF95_FREQ = 433.0
RF95_POW = 20
CLIENT_ADDRESS1 = 0x01
CLIENT_ADDRESS2 = 0X03

SERVER_ADDRESS = 0xff

i2c = SoftI2C(scl=Pin(22), sda=Pin(21))
oled_width = 128
oled_height = 32
oled = ssd1306.SSD1306_I2C(oled_width, oled_height, i2c)

count = 0
# initialise radio
lora = LoRa(RFM95_SPIBUS, RFM95_INT, SERVER_ADDRESS, RFM95_CS,
reset_pin=RFM95_RST, freq=RF95_FREQ, tx_power=RF95_POW, acks=True)

Mqtt_CLIENT_ID = "1"
broker=  "demo.thingsboard.io"

username1="node01"
PASSWORD1="node01234"

username2="node02"
PASSWORD2="node02345"
```

```
topic = "v1/devices/me/telemetry"
topic1 = "v1/devices/me/rpc/response/+"
topic2 = "v1/devices/me/rpc/request/+"
topic3 = "v1/devices/me/attributes"

client1 = MQTTClient(client_id=Mqtt_CLIENT_ID, server=broker, port=1883,
user=username1, password=PASSWORD1, keepalive=10000)
client2 = MQTTClient(client_id=Mqtt_CLIENT_ID, server=broker, port=1883,
user=username2, password=PASSWORD2, keepalive=10000)

def on_recv(payload):
    print("From:", payload.header_from)
    print("Received:", payload.message)
    print("RSSI: {}; SNR: {}".format(payload.rssi, payload.snr))

    t1 = {"RSSI": str(payload.rssi) , "SNR" : str(payload.snr)}
    t1 = json.dumps(t1)
    #print(t1)
    text=payload.message
    text = text.decode()
    text = json.loads(text)
    #t_a,h_a=read_aht10()
    kk = {'RSSI' : str(payload.rssi), 'SNR' : str(payload.snr)}
    text.update(kk)

    print(text)
    data=json.dumps(text)
    Topic = topic.encode()
    if (payload.header_from == 2 ):
        client1.publish(Topic , data)
```

```
    else:
        client2.publish(Topic , data)
    oled.fill(0)
    oled.show()
    oled_message1 = "From: " + str(payload.header_from)
    oled.text(oled_message1 , 0, 0)
    oled.text('sent to server', 0, 10)
    oled.show()
    lora.set_mode_rx()


def callback(resp_id, data):
    global state
    print('Response {id}: {data})'.format(id=resp_id, data=data))
    data = data.decode()
    data = json.loads(data)
    if (data['method'] == "setValue"):
        if (data['params'] == True) :
            print("Node 1 :ON")
            state = 1
            message = "valve.on"
            lora.send(message,CLIENT_ADDRESS1)
            rpc_reply(client1)

        elif (data['params'] == False ):
            print("Node 1 :OFF")
            state = 0
            message = "valve.off"
            lora.send(message,CLIENT_ADDRESS1)
            rpc_reply(client1)
```

```
        else:
            print("error")

    elif (data['method'] == "setValue2"):
        if (data['params'] == True) :
            print("Node 2 :ON")
            state = 1
            message = "valve.on"
            lora.send(message,CLIENT_ADDRESS2)
            rpc_reply(client2)

        elif (data['params'] == False ):
            print("Node 2 :OFF")
            state = 0
            message = "valve.off"
            lora.send(message,CLIENT_ADDRESS2)
            rpc_reply(client2)
        else:
            print("error")

def do_connect():
    import network
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    if not wlan.isconnected():
        print('connecting to network...')
        wlan.connect(WIFI_SSID,WIFI_PASS)
        while not wlan.isconnected():
            pass
    print('network config:', wlan.ifconfig())
```

```
def rpc_reply(client):
    if (state == 1):
        value = True

    else:
        value = False

    kush= {"value" : value}
    jango = json.dumps(kush)
    jango = jango.encode()
    Topic3 = topic3.encode()
    client.publish(Topic3 , jango)


def connect():

  try:
    client1.connect()
    client2.connect()

  except OSError:
    print('Connection failed')
    #sys.exit()


  client1.set_callback(callback)
  client1.subscribe(topic1)
  client1.subscribe(topic2)
  client1.subscribe(topic3)
```

```
  client2.set_callback(callback)
  client2.subscribe(topic1)
  client2.subscribe(topic2)
  client2.subscribe(topic3)

  lora.on_recv = on_recv

  while 1:
      lora.set_mode_rx()
      client1.check_msg()
      client2.check_msg()

do_connect()
connect()
```

NODES CODE

```
from utime import sleep_ms
from machine import Pin, I2C,SoftI2C,ADC
import ahtx0,ssd1306,onewire, ds18x20,ujson
from ulora import LoRa, ModemConfig, SPIConfig
from time import sleep

#----------------------
# Lora Parameters
RFM95_RST = 21
RFM95_SPIBUS = SPIConfig.rp2_0
RFM95_CS = 17
```

```
RFM95_INT = 20
RF95_FREQ = 433.0
RF95_POW = 20
CLIENT_ADDRESS = 0x01
SERVER_ADDRESS = 0xff
#----------------------
#i2c for aht10
i2c_aht10 = i2c = I2C(1, scl=Pin(15), sda=Pin(14), freq=400_000)
# Create the sensor object using I2C
#aht10 = ahtx0.AHT10(i2c_aht10)

#i2c for aht10
i2c_oled = SoftI2C(scl=Pin(5), sda=Pin(4))
#oled = ssd1306.SSD1306_I2C(128, 32, i2c_oled)

#solenoid valve
s_valve = Pin(11, Pin.OUT)

#led
red_led = Pin(12, Pin.OUT)

blue_led = Pin(13, Pin.OUT)
#ADC_ldr
adc_ldr = ADC(Pin(27))

#ADC_Moisture sensor
adc_soil = ADC(Pin(26))

#Temperature probe(DS18b20)
ds_pin = machine.Pin(22)
```

```
ds_sensor = ds18x20.DS18X20(onewire.OneWire(ds_pin))

"""
def read_aht10():
    t=aht10.temperature
    h=aht10.relative_humidity
    return t,h
"""
state=0
def display(string):
    oled.fill(0)
    oled.text(str(string), 0, 1, 1)
    oled.show()
def read_ds18b20():
    rom = ds_sensor.scan()
    ds_sensor.convert_temp()
    sleep_ms(750)
    t=ds_sensor.read_temp(rom[0])
    return t
def ldr():
    ldr=adc_ldr.read_u16()
    v = (3.3*ldr)/65535
    val=(v/3.3)*100
    return val

def soil_m():
    value=adc_soil.read_u16()
    #print(value)
    v = 3.3-((3.3*value)/65535)
    val=(v/3.3)*100
```

```
    return val


def solenoid(v):
    s_valve.value(v)
def on_recv(payload):
    global state
    print("From:", payload.header_from)
    print("Received:", payload.message)
    print("RSSI: {}; SNR: {}".format(payload.rssi, payload.snr))
    text=payload.message
    text=text.decode()
    print(text)
    if (text=='valve.on'):
        print("data recieved ok")
        solenoid(1)
        red_led.value(0)
        blue_led.value(1)
        state=1


    elif(text=='valve.off'):
        print("data recieved ok")
        solenoid(0)
        red_led.value(1)
        blue_led.value(0)
        state=0
#-----------------------
# initialise radio
lora = LoRa(RFM95_SPIBUS, RFM95_INT, CLIENT_ADDRESS, RFM95_CS,
reset_pin=RFM95_RST, freq=RF95_FREQ, tx_power=RF95_POW, acks=True)
lora.on_recv = on_recv
#-----------------------
```

```
count=0
while True:
    count+=1
    #ambient temp and hum-------
    #t_a,h_a=read_aht10()
        #ambient temp and hum-------
    #t_s="{:.2f}".format(read_ds18b20())
    h_s="{:.2f}".format(soil_m())
    #ldr
    l_i="{:.2f}".format(ldr())
    #msg2={"t_s":str(t_s),"h_s":str(h_s),"l_i":str(l_i),"c":str(count),"s":str(state)}
    msg2={"h_s":str(h_s),"c":str(count),"s":str(state)}
        msg2=ujson.dumps(msg2)
    print(msg2)
    lora.send_to_wait(msg2 ,SERVER_ADDRESS)
    lora.set_mode_rx()
    sleep(10)
```

Thingsboard cloud scripts

Script to filter specific required information from response JSON file

```
var newMsg = {
    "outsideTemp": msg.list[0].main.temp,
    "outsideMaxTemp": msg.list[0].main.temp_max,
    "outsideMinTemp": msg.list[0].main.temp_min,
    "outsideHumidity": msg.list[0].main.humidity,
    "rain_pop": msg.list[0].pop,
    "rain": msg.list[0].weather[0].description,
};
return {
    msg: newMsg,
```

```
    metadata: metadata,
    msgType: msgType
};
```