AUTOMATED FARM IRRIGATION SYSTEM-rUrban

By

JOSEPH CALEB JOHNSTON (9406419)

ACQUAH GODWIN EKPAHOOMAH (9391519)

Under the supervision of

PROF. J B HAYFRON ACQUAH

KWAME NKRUMAH UNIVERSITY OF SCIENCE AND TECHNOLOGY

COLLEGE OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

TABLE OF CONTENTS

TABLE OF FIGURES

DECLARATION

We solemnly declare that the project work titled "Automated Irrigation System-rUrban" presented in this document is a bona fide work undertaken by us and is being submitted to Kwame Nkrumah University of Science and Technology, department of Computer Science in requirement fulfillment of Bachelor of Science in Computer Science degree award under the supervision of Prof. J B Hyfron Acquah. All sources of information, data, and references used in this project have been duly acknowledged through proper citations.

---------- ----------

JOSEPH CALEB JOHNSTON DATE

---------- ----------

ACQUAH GODWIN EKPAHOMAH DATE

SUPERVISOR

---------- ----------

PROF. J.B. HAYFRON-ACQUAH DATE

ABSTRACT

The contemporary progress in technology has catalyzed a paradigmatic transformation within the agricultural sector, ushering it into a phase of enhanced efficacy and ecological responsibility. A notable example of such progressive initiatives is the creation of an automated irrigation system. This project amalgamates cutting-edge sensor technology, the Internet of Things (IoT), and sophisticated automation techniques, with the overarching aim of optimizing water allocation and elevating crop yields to unprecedented levels.

DEDICATION

We humbly dedicate this endeavor to our cherished parents, whose unwavering love and steadfast support have been the guiding light that has led us to this juncture. Our heartfelt appreciation extends to the esteemed faculty members of the Department of Computer Science, whose imparted knowledge and expertise have empowered us to navigate this journey with confidence and competence.

ACKNOWLEGMENT

It is with a profound sense of gratitude that we extend our sincere appreciation to Professor J. B. Hayfron-Acquah, our esteemed project supervisor. His erudition, guidance, and unwavering support have indelibly shaped the trajectory of this endeavor, underscoring the symbiotic relationship between mentorship and academic growth.

Furthermore, we wish to convey our heartfelt thanks to our knowledgeable friends for their willingness to share their knowledge and expertise with us.

Finally, a paramount acknowledgement is due to God Almighty, whose benevolence graced us with robust health and acuity of mind throughout this project.

CHAPTER 1 - INTRODUCTION

1.1 OVERVIEW/ BACKGROUND

Agriculture is a key sector in our economy and a vital source of food. The yield and productivity of farms are essential to the success of this sector. Irrigation is an essential component of farming and when managed properly, can significantly improve crop yield and quality. This automated irrigation system aims to address the problems associated with farm irrigation by providing an automated solution that ensures efficient water usage.

The system is composed of sensors to monitor soil moisture levels, temperature and humidity, an Arduino microprocessor to analyze the data and control automatic regulation of the system components depending on the soil moisture levels and weather conditions. An IOT module is integrated to enable data communication with a cloud platform for remote controlling and to visualize the data.

1.2 MOTIVATION OF THE PROJECT

Agriculture has always been the backbone of human civilization, providing sustenance and nourishment for generations. It is therefore imperative to preserve and improve farming amidst growing technologies as we face rising challenges that threaten our food security and water supply as growing global population puts a strain on our agricultural systems.

To ensure efficient water usage and agricultural productivity, we are motivated to automate the crop irrigation process.

1.3 PROBLEM DEFINITION

With increasing global population, agricultural yield and productivity is in high demand. Farmers are faced with time constraints, unpredictable weather and inconsistencies that inhibit efficient irrigation of crops. To overcome these challenges, an automated irrigation system is an ideal solution.

1.4 AIM OF PROJECT

Our goal is to design and implement a cost-effective automatic farm irrigation system that will help farmers save time and resources while increasing crop yield. The system will be able to:

- Monitor soil moisture levels and irrigate where necessary to ensure optimal water usage.
- Monitor weather conditions to adjust irrigation schedules and prevent crop damage.
- Provide remote access to farmers to monitor and control the irrigation system from their internet-enabled devices.
- Provide a remote USSD (Unstructured Supplementary Service Data) system for farmers without smart devices to access the system.
- Improve crop quality: The system should be able to ensure that crops receive the correct amount of water, which will help to improve crop quality. This will lead to higher-quality produce and better prices for farmers.

1.5 SPECIFIC OBJECTIVES

1. The system would automate the process of farm irrigation based on the soil and atmospheric (temperature and humidity) conditions.
2. Provide internet-enabled remote access allowing farmers to view real-time data on soil moisture and weather conditions on farm.

3. Provide a remote USSD (Unstructured Supplementary Service Data) system for farmers without smart devices to access the system.
4. Ability to trigger manual irrigation via web interface.

1.6 DEFINITION OF TECHNOLOGIES

1. **Internet of Things (IoT) Technology:**

   The Internet of Things (IoT) is a system of interrelated, internet-connected objects that collect and exchange data. These objects are embedded with electronics, software, sensors, and actuators to collect and exchange data. The Internet of Things allows these objects to be controlled and monitored remotely across a network.

2. **ESP32 microcontroller**

   The ESP32 is a low-cost, low-power system on a chip (SoC) microcontroller with integrated Wi-Fi and dual-mode Bluetooth. It is developed and manufactured by Espressif Systems, a Shanghai-based Chinese company. It is programmable using the Arduino IDE (Integrated Development Environment). The ESP32 is a successor to the ESP8266 microcontroller, and it is one of the most popular IoT development boards available today.

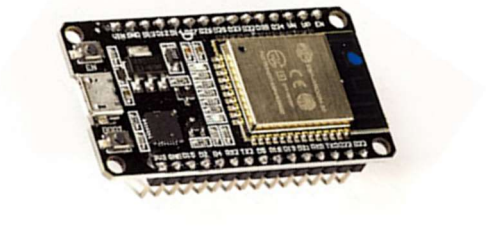

Figure 1 ESP32 microcontroller

3. Soil Moisture Sensor

The capacitive soil moisture sensor v1.2 is a small, low-cost sensor that can be used to measure the moisture level in soil. It is a capacitive sensor, which means that it measures the change in capacitance between two electrodes as the moisture content of the soil changes.

Figure 2 Capacitive soil moisture sensor

The capacitive soil moisture sensor v1.2 has three pins:

- VCC: This pin is connected to a 3.3 to 5V power supply.
- GND: This pin is connected to ground.
- AOUT: This pin outputs the sensor's analog signal, which can be read by an Arduino or other microcontroller.

The sensor's analog signal is proportional to the moisture content of the soil. A dry soil will have a low capacitance, and a wet soil will have a high capacitance. The sensor's analog signal can be calibrated to provide a reading in percentage of moisture content.

4. Water Level Sensor (HC-RO4 Ultrasonic Sensor

The HC-SR04 sensor is a small, rectangular device with four pins. The two pins on the left are the power pins (VCC and GND). The pin in the middle is the trigger pin, and the pin on the right is the echo pin.

Figure 3 HC-R04 Ultrasonic Sensor

The trigger pin is used to send the ultrasonic sound wave. The echo pin is used to receive the echo from the object. The time it takes for the echo to return is used to calculate the distance to the object.

Ultrasonic sensors are used in automated irrigation systems to measure the water level in a reservoir or tank. This information is then used to control the irrigation system, ensuring that plants are only watered when needed.

Here is how it works:

The ultrasonic sensor is placed in the reservoir or tank.

- The sensor emits a high-frequency ultrasonic sound wave and listens for the echo.
- The time it takes for the echo to return is used to calculate the water level.
- The sensor sends the water level information to the irrigation controller.
- The irrigation controller turns on the pump when the water level drops below a certain level.
- The pump turns off when the water level reaches a certain level.

This process repeats itself as needed, ensuring that plants are always watered at the right time and in the right amount.

Ultrasonic sensors are a good choice for automated irrigation systems because they are accurate, reliable, and relatively inexpensive. They are also easy to install and maintain.

5. Temperature and Humidity Sensors

The DHT 22 digital temperature and humidity sensor is a composite sensor that contains a calibrated digital signal output for both temperature and humidity. It uses a dedicated digital modules collection technology and temperature and humidity sensing technology to ensure that the product has high reliability and excellent long-term stability.

Figure 4 DHT 22

The sensor consists of a humidity sensing component, a NTC temperature sensor, and an IC on the back side of the sensor. The humidity sensing component has two electrodes with a moisture-holding substrate between them. As the humidity changes, the conductivity of the substrate changes, or the resistance between the electrode changes. This change in resistance is measured and processed by the IC, which makes it ready to be read by a microcontroller.

The NTC temperature sensor, or thermistor, is a variable resistor that changes its resistance with change in temperature. These sensors are made by sintering of semiconductive materials such as ceramics or polymers in order to provide larger changes in the resistance with just small changes in temperature. The term "NTC" means "Negative Temperature Coefficient", which means that the resistance decreases with increase of the temperature.

The DHT22 is a digital temperature and humidity sensor that measures both atmospheric temperature and moisture. The relative humidity is expressed as a percentage, and the humidity sensing component is a HS2200 sensor. The output of the sensor is in terms of frequency, with a range of 5 kHz to 10 kHz.

6. Relay Module

A relay module is an electronic device used to control high-power circuits with low-power control signals, typically from a microcontroller or other digital logic circuits. It acts as an electrical switch that can turn on or off a separate circuit when an input signal is applied to it. The primary function of a relay module is to isolate the control circuit from the high-power circuit it is switching, providing safety and protection to the controlling device.

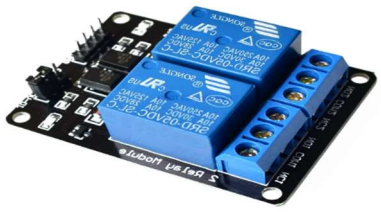

Figure 5 Relay Module

Key features and components of a typical relay module include:

Relay: The heart of the module is an electromechanical relay. It consists of a coil, an armature, and one or more sets of contacts. When a current is applied to the coil, it generates a magnetic field that attracts the armature, causing the contacts to close or open, depending on the relay type.

Input Control Signal: The relay module is triggered by an input control signal, which is usually a voltage or digital signal from a microcontroller, sensor, or any other digital logic circuit. When the control signal is applied, it energizes the relay coil, allowing current to flow through the relay's contacts and activate the connected load.

Output Contacts: Relay modules have one or more output contacts that act as the switching element. These contacts can be normally open (NO), normally closed (NC), or both (changeover relay). Depending on the relay type and its configuration, these contacts can either complete or interrupt the circuit when the relay is energized.

Opto-isolation (Optional): Some relay modules include opto-isolators between the control circuit and the relay coil. Opto-isolators use light-emitting diodes (LEDs) to trigger a phototransistor,

providing complete electrical isolation between the control circuit and the high-power circuit, protecting the control circuit from voltage spikes and noise.

Status Indicators: Relay modules may come equipped with status indicators such as LEDs that show the activation status of the relay. These LEDs help to visualize whether the relay is energized or not, providing useful feedback during testing and troubleshooting.

Screw Terminals: The relay module usually has screw terminals or other connectors for easy and secure connection of wires from both the control circuit and the high-power circuit.

7. Micro Servo (Tower Pro SG90).

Figure 6 Tower Pro SG90

A micro servo is a small, lightweight servo motor that is typically used in robotics and other applications where space is limited. Micro servos are typically less powerful than larger servo motors, but they are also much more affordable and easier to use.

The purpose of the micro servo is to control the opening and closing of the valve to control water supply depending on soil moisture threshold signals received from the soil moisture sensor connected to the system.

8. Submersible pump

The submersible pump is a type of pump that is designed to be submerged in water. It is typically used to pump water from a well or reservoir to the surface.

Submersible pumps are often used in this irrigation system to pump water from reservoir to the field through tubes.

Fig 7. pump

1.7 PROJECT BENEFICIARIES

The target audience for the automated farm irrigation system include both large scale farmers and rural small-scale farmers as it is cost effective and scalable. It can also be tailored toward households with compound flowering for automated watering.

1.8 SCOPE OF PROJECT

The automated farm irrigation system is capable of:

- Sensing soil moisture levels to determine the irrigation requirements
- Controlling water pumps to deliver appropriate amount of water to crops
- Transmitting data to the IOT platform for real-time remote monitoring and analysis.
- Enable USSD based remote access to irrigation system

The system utilizes Arduino, an open-source electronics platform, for sensing and controlling irrigation components. An IoT module is integrated to enable data communication with the cloud platform for remote monitoring. Soil moisture sensors are deployed in the fields to measure soil moisture levels. Based on the sensor data, the Arduino triggers the water pump through relay modules to irrigate the crops as needed.

1.9 DELIVERABLES

The main deliverable of this project is a Printed Circuit Board (PCB) connecting all electronic components and a dedicated cloud server for the system that fully executes all defined requirements. Other deliverables include a simple web application and simple USSD interface to remotely access and control some components of this PCB below.

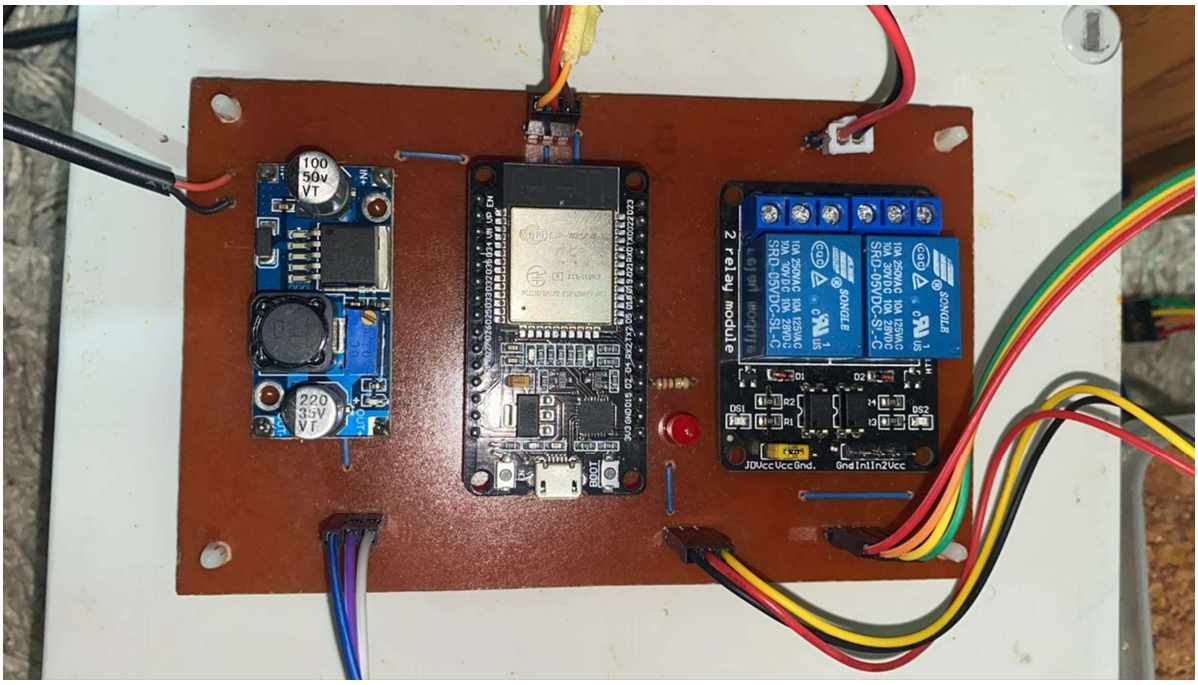

Figure 7 Printed Circuit Board connecting components

Below is the blue print of the Printed Circuit board for the system above:

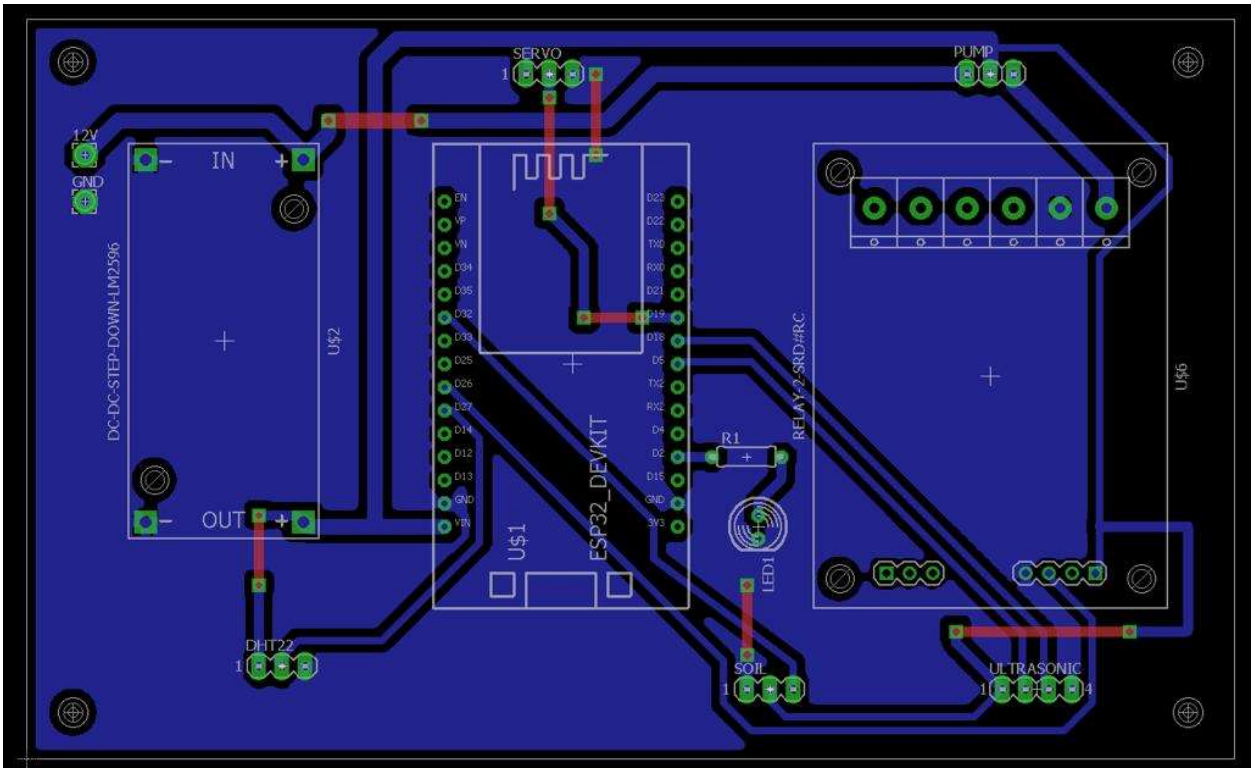

Figure 8 Blueprint of PCB

CHAPTER 2 - LITERATURE REVIEW

2.1 REVIEW OF SIMILAR SYSTEM

An overview of smart irrigation systems using IoT

Khaled Obaideen [a], Bashria A.A. Yousef [b], Maryam Nooman AlMallahi [b], Yong Chai Tan [c], Montaser Mahmoud [a d], Hadi Jaber [e]

… The study is based on a qualitative design along with focusing on secondary data collection method. Automated irrigation systems are essential for conservation of water, this improvement could have a vital role in minimizing water usage. Agriculture and farming techniques is also linked with IoT and automation…

Solar powered smart irrigation system

S Harishankar, RS Kumar, KP Sudharsan… - Advance in Electronic …, 2014 - researchgate.net

… 2. Literature Survey and Background Study According to the survey conducted by the Bureau of …

1981. Small-scale solar powered irrigation pumping systems: technical and economic review. UNDP Project GLO/78/004.Intermediate Technology Power, London, UK …

Irrigation of Water by Automatic Sprinkler System (Nelson, 2021).

R Nelson et al 2021 IOP Conf. Ser.: Mater. Sci. Eng. 1145 012107

…This automated system of irrigation is fabricated to constantly monitor the moisture level of soil. This system works accurately by supplying required amount of water to the soil and shutting off the supply of water when the necessary soil moisture rate is reached [11]. Corrosion resistant materials are used in fabricating irrigation per unit time was calculated by taking the water pump's capacity and the water sprinkler's …

2.2 PROPOSED SYSTEM

We proposed a cost-effective smart irrigation system, employing IOT technology to connect data to a cloud server and visualize the data. We included USSD system access for areas with limited network coverage. The proposed system allows farmers a peace of mind as they can remotely monitor their farm irrigation system from any internet enabled device or just any mobile device with 2G network coverage.

CHAPTER 3 - METHODOLOGY

3.1 PROJECT METHODS

In this project, we used the waterfall model of the Software Development Life Cycle (SDLC). Here is the step-by-step methodology of the process:

1. **Gathering & Analysis of Requirement**:

   This is where we defined the scope of our project as state in SCOPE OF PROJECT

   We then conducted a thorough assessment of the irrigation requirements, such soil moisture monitoring, and weather conditions.

2. **Research and Component Selection**:
   We researched available technologies, sensors, and irrigation systems suitable for our project. Then we selected appropriate sensors for measuring soil moisture and weather conditions (temperature, humidity).

3. **System Design and Architecture:** This is where we created a system architecture that outlines how different components will interact with each other. We chose to use a printed circuit board to interconnect the electronic components for its durability, reliability and high current carrying capacity. We then assembled the components and

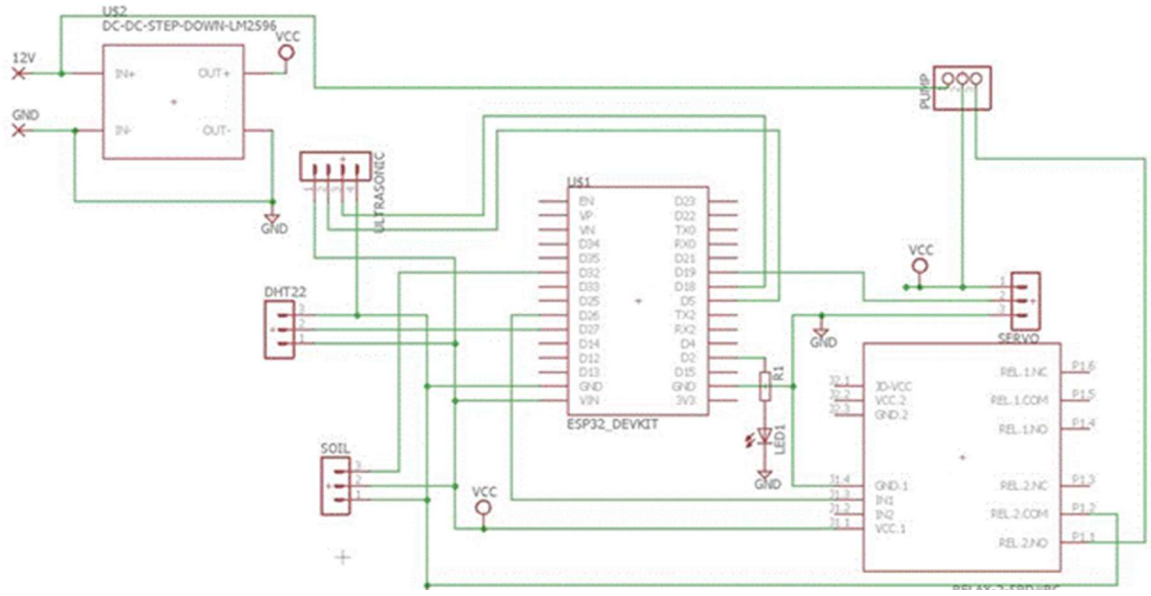

Figure 9 Circuit design

built the circuit.

The schematic above show how the IoT devices are connected, forming a complex system. The figure above shows how these devices communicate with each other, allowing them to exchange data and automate tasks. The lines and nodes in the diagram represent the connections between devices, showing the advanced architecture and design.

4. **Coding/Implementation:**
   Here, we programmed the ESP32 microcontroller into being the brain of the system using Arduino IDE.

5. **User Interface**
   We created a web-based user interface for data visualization and remote control using IoT technology. Subsequently, we created a USSD based system for farmers without smartphones and also to supplement remote control in areas with limited internet connectivity.

6. **Testing and Deployment.**
   We tested the system under different circumstances using XAMPP server for local hosting for the web-based system. Later, we deployed the site to a cloud hosting platform.

3.2 DESIGN CONSIDERATIONS

1. A Printed Circuit Board (PCB) was used for its durability, reliability and high current capacity over the Breadboard circuit.

2. Budget: We tried to minimize expenses as possible without sacrificing quality.

3.2.1 USER INTERFACE DESIGN (FRONT-END)

3. **Html, CSS and JavaScript (Bootstrap Liberary) for web interface design:**
   We incorporated html, css, and bootstrap for the web interface to enable the creation of responsive and intuitive user interfaces. Some of the design considerations of the user interface are below:
   - **Simplicity**- use of clear labels and visual aids and grouping related features or action's together.
   - **Responsiveness**- the interface should provide a timely feedback and load times should be minimal.
   - **Accessibility**- Font sizes should be readable, and there should be a contrast between text and background.
   - **Dashboard overview**- the dashboard gives a glance view of all sensor readings.
   - **Mobile and Desktop Variants**- Since farmers might access the system from any device, the UI is designed responsibly to adapt to various screen sizes.

The figures below represent the web interface design:

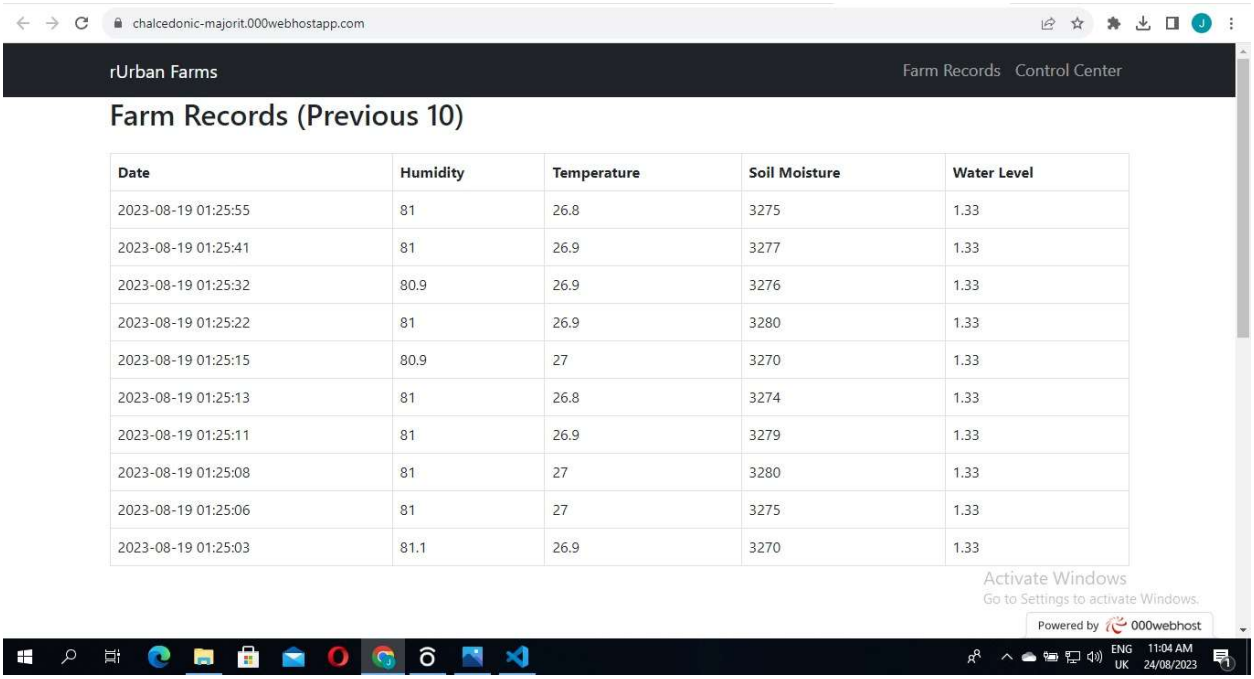

Figure 10. Web interface showing farm records

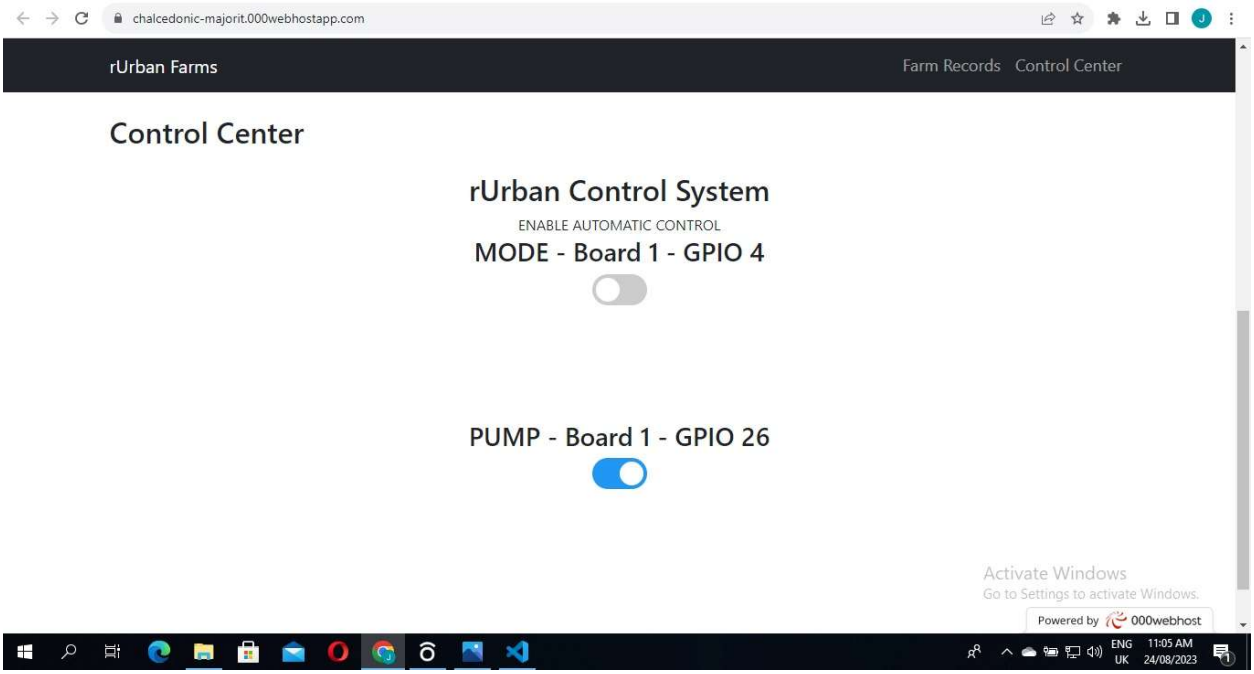

Figure 11. Web interface showing control center

USSD SYSTEM DESIGN

The next stage involves developing a USSD application that will enable farmers to access readings from their farms and control their irrigation pumps. This USSD application will be designed to receive information from farm recordings database through a call-back URL. Additionally, the USSD application will allow farmers to start and stop their irrigation pumps, ensuring that farmers without internet devices or smartphones can manage their farm irrigation.

To integrate the USSD API into the website, we have developed a logic that will ensure the operation of the USSD application. This logic will enable farmers to interact with the USSD application through prompts and responses, allowing them to check farm readings and control their irrigation pumps. The USSD interface just like the web interface will be user-friendly, with clear instructions and options for farmers to easily navigate the different functionalities.

The figures below represent the USSD interfaces:

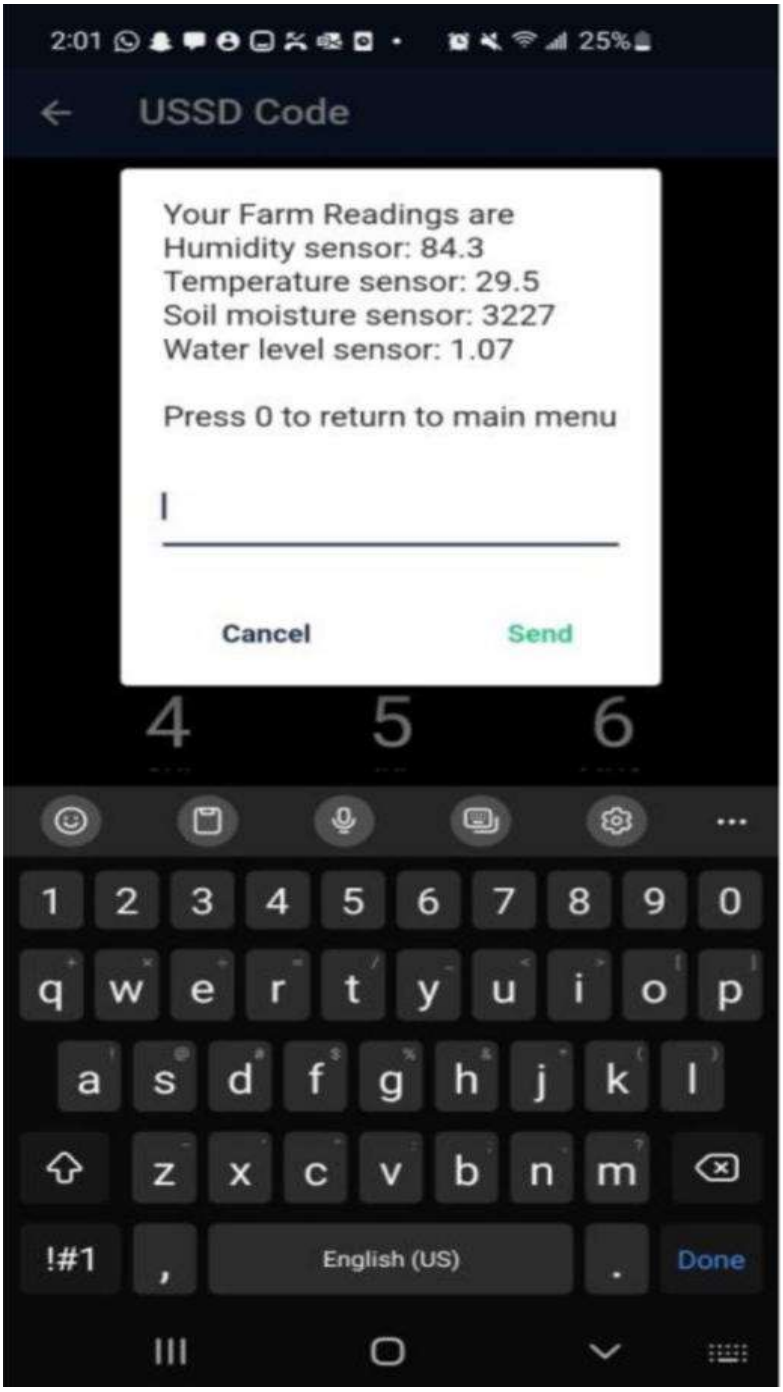

Figure 12. USSD interface showing farm readings

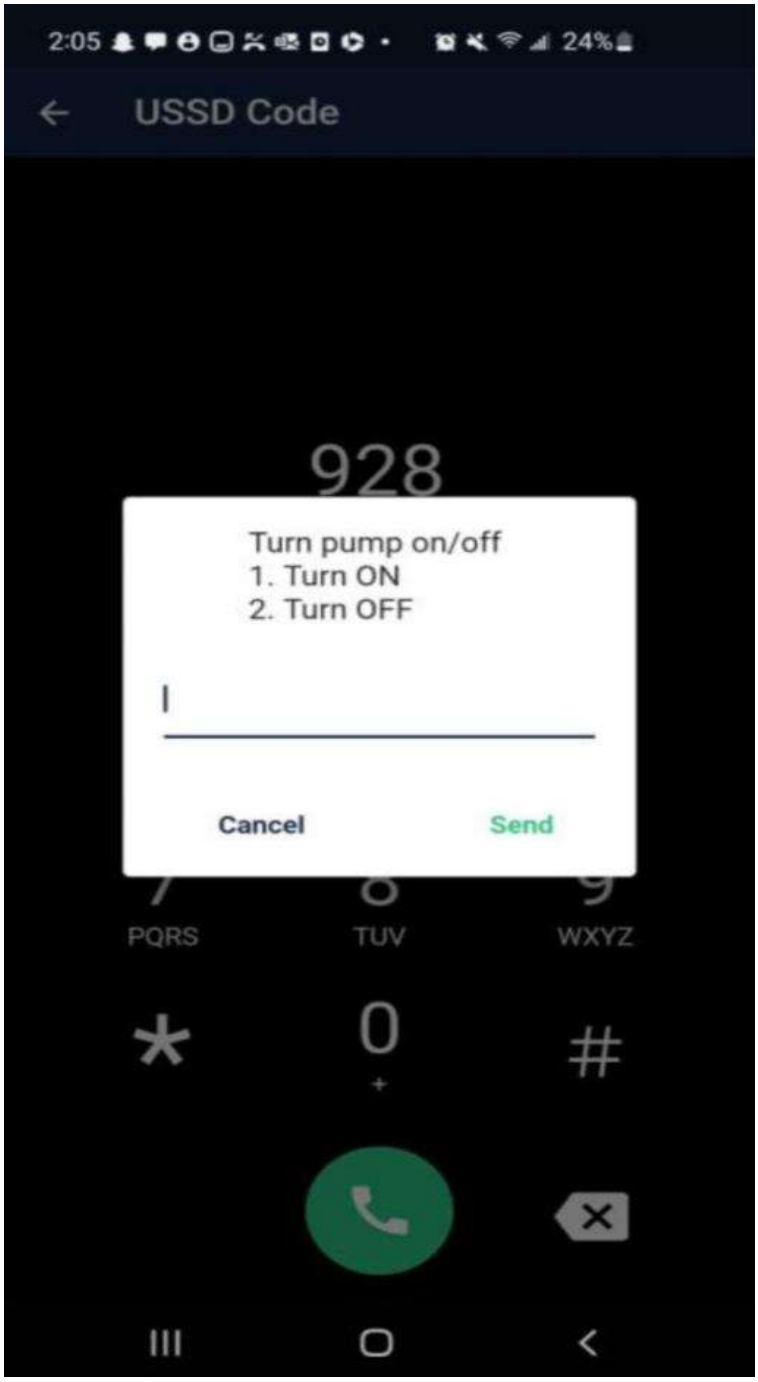

Figure 13. USSD control interface

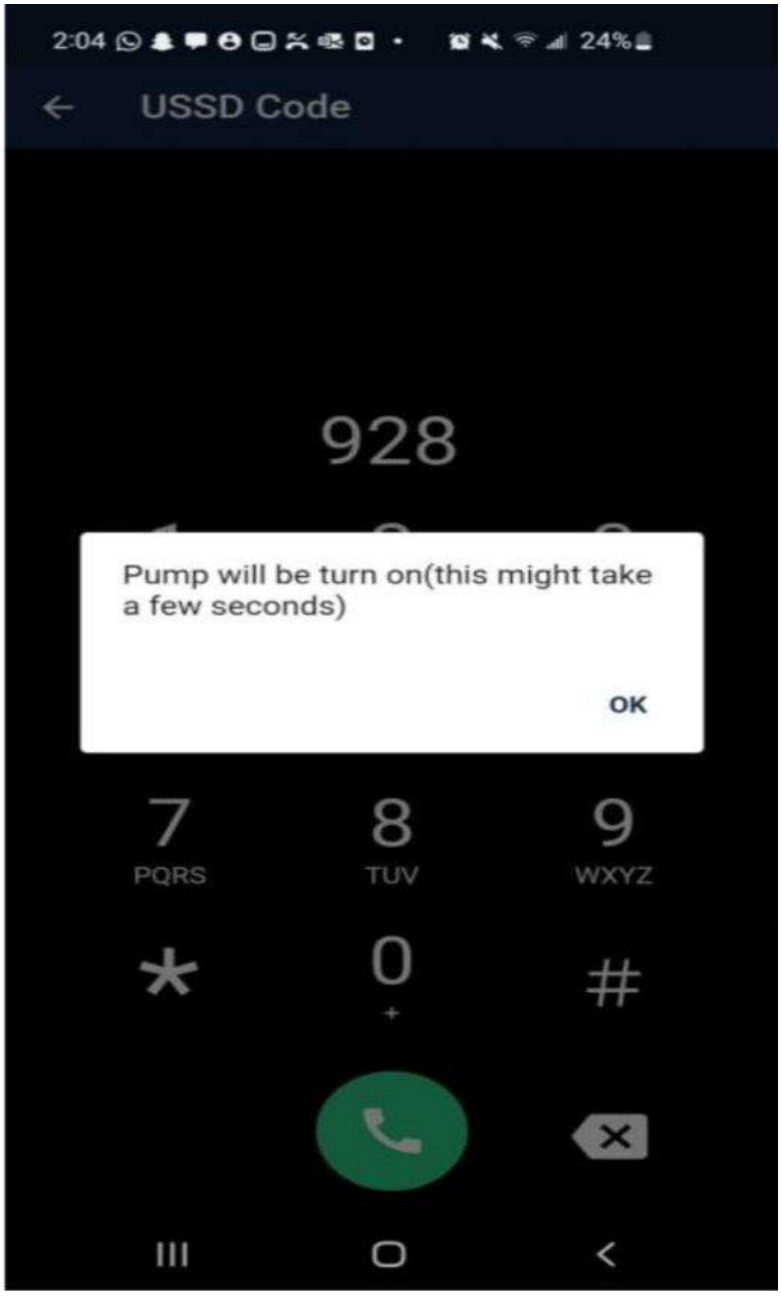

Figure 14. Ussd Confirmation Screen

BACK-END DESIGN

We used PHP and SQL for the backend scripting to ensure seamless communication between the User Interfaces (Web and USSD), the database and the IoT components.

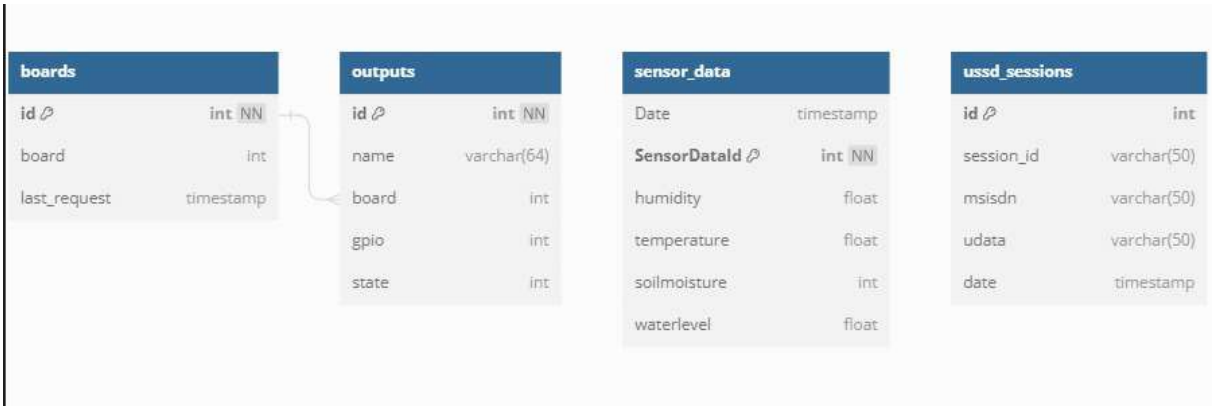

Figure 15. Database Schema

These are some backend design considerations we implemented:

- Performance and Scalability: We ensured database tables are indexed correctly to speed up queries.
- state management: sessions are typically stateful. Using a session management system to maintain the context for each user is crucial.
- Security: We always validate and sanitize inputs coming from both the USSD and Web interface to prevent SQL injection and other attacks.
- Modularity: Since we want to integrate other services into our system later, we designed our PHP backend to be modular, making future integrations easier.
- Database design: Normalization of database to eliminate data redundancy.

3.3 REQUIREMENTS SPECIFICATION

3.3.1 FUNCTIONAL REQUIREMENTS

- Automated Irrigation: The system should be able to automate irrigation based on soil moisture levels, humidity and temperature thresholds.

- Soil Moisture Monitoring: The system should incorporate soil moisture sensors to measure the moisture content in the soil. This data is crucial for determining when and how much to irrigate.

- Weather Monitoring and Integration: The system should be able to access real-time weather data and forecasts to adjust irrigation schedules accordingly. It should consider factors like rainfall, temperature, humidity, and evaporation rates.

- Remote Monitoring and Control: The system should offer remote access through an associated web interface and USSD, allowing farmers to monitor the system's performance and make adjustments from anywhere.

3.3.2 NON-FUNCTIONAL REQUIREMENTS

- **Reliability**: The system should be highly reliable and available, ensuring that it operates as intended without frequent downtime or disruptions.
- **Scalability**: The system should be able to scale its capabilities to accommodate an increasing number of crops, fields, or sensors as the farm expands.
- **Performance**: The system should respond quickly to changing conditions and adjust irrigation schedules promptly to meet crop water requirements.
- **Usability**: The user interface should be intuitive and easy to navigate, allowing farmers to configure and manage the system with minimal training.

- **Security**: The system should implement robust security measures to protect against unauthorized access, data breaches, and tampering.
- **Interoperability:** If the irrigation system integrates with other farm management tools or third-party devices, it should ensure smooth data exchange and compatibility.
- **Data Privacy:** The system should adhere to data privacy regulations and protect sensitive information collected from the farm and its operations.
- **Environmental Impact**: The system should be designed to minimize water wastage and energy consumption, promoting eco-friendly irrigation practices.
- **Accuracy**: Soil moisture sensors and weather forecasting should be accurate to ensure precise irrigation decisions.
- Maintenance: The system should be easy to maintain, with clear instructions for troubleshooting, repairs, and component replacements.
- **Adaptability**: The system should be adaptable to different types of soil, crops, and local climate conditions.
- **Power Efficiency**: The system relies on electricity; hence it should strive for power efficiency to reduce operational costs and environmental impact.
- **Durability**: The components of the system, such as sensors, valves, and pumps, should be designed to withstand outdoor conditions and have a long service life.
- Adaptability to Low Connectivity: In rural areas with limited internet connectivity, the system should still function effectively and store data locally if needed.

3.3.3 USER REQUIREMENTS

- **Easy Installation**: Farmers should be able to install and set up the irrigation system with minimal technical expertise and without the need for specialized knowledge.
- **Real-time Monitoring:** Farmers should be able to monitor the status of the irrigation system in real-time, including soil moisture levels, water flow, and weather data

- **Intuitive User Interface:** The system should have a user-friendly interface that is easy to navigate and understand, enabling users to configure irrigation schedules, monitor performance, and make adjustments effortlessly.
- **Remote Control:** Users should have the ability to remotely start, stop, or adjust the irrigation process as needed.

3.3.4 SYSTEM REQUIREMENTS

- **Soil Moisture Sensors**: These sensors measure the moisture content in the soil and provide real-time data to determine the irrigation needs of the crops.
- **Temperature and humidity sensors**: These sensors should provide atmospheric temperature and humidity to contribute to irrigation factors.
- **Software System:** The system should include a software system that is able to receive data from sensors, process it, and control the irrigation system accordingly.

- **User Interface:** The system includes a user interface that allows farmers to view, monitor and control the system.

- **Power Supply:** The system requires constant electricity supply to function reliably.

- **Internet enabled device or phone with 2G network accessibility:** This is required to enable remote connectivity to the system.

3.4 UML DIAGRAMS

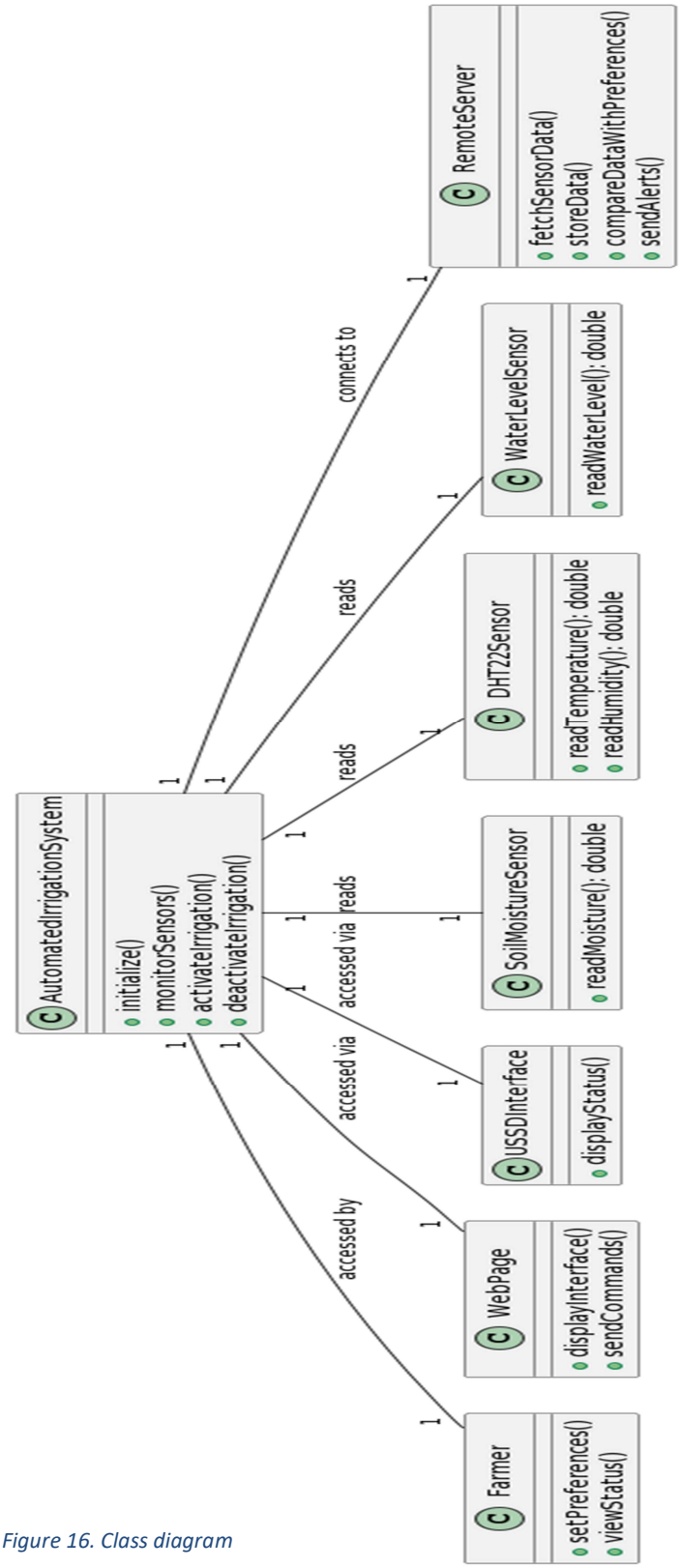

Figure 16. Class diagram

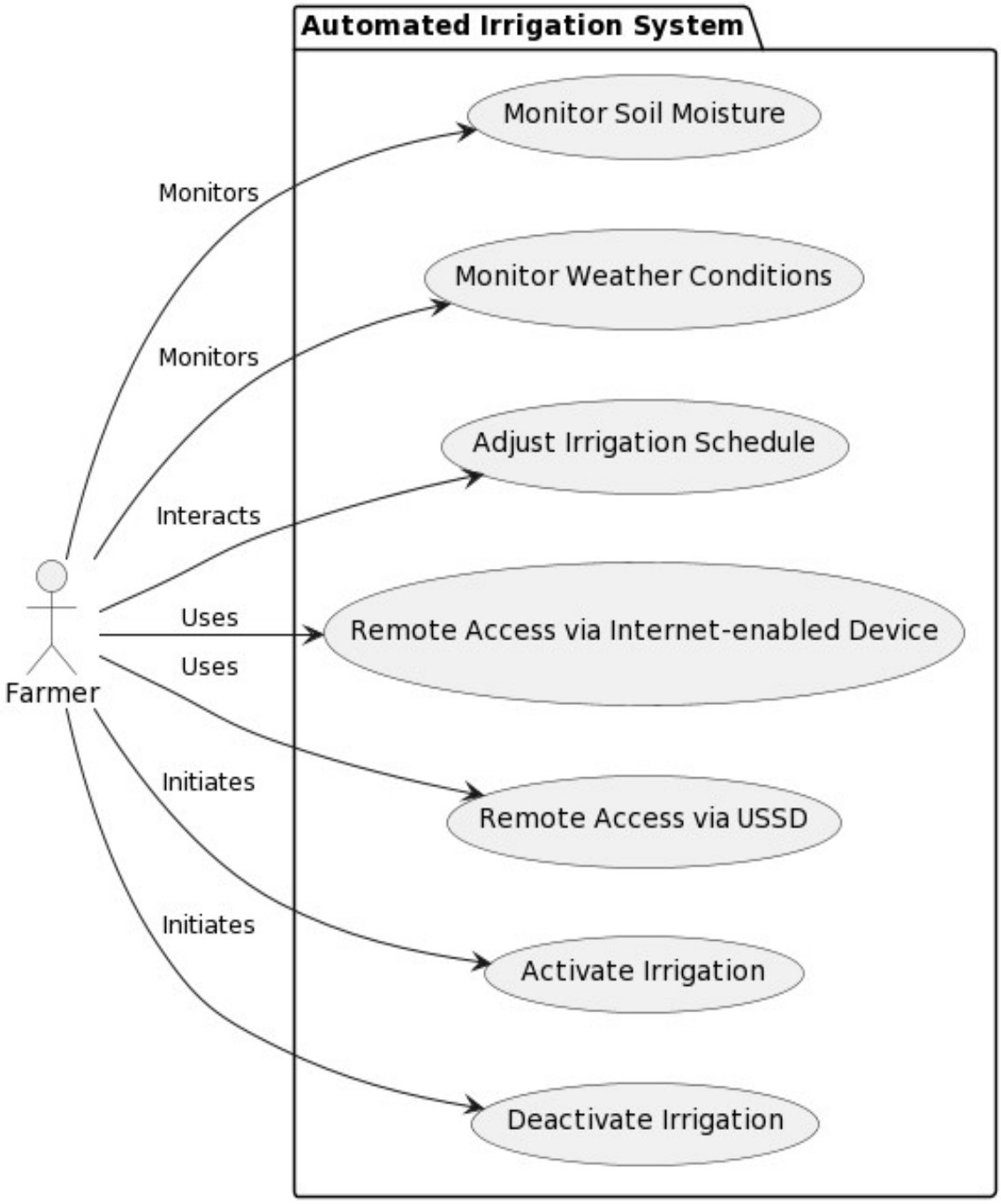

Figure 17. Use case diagram

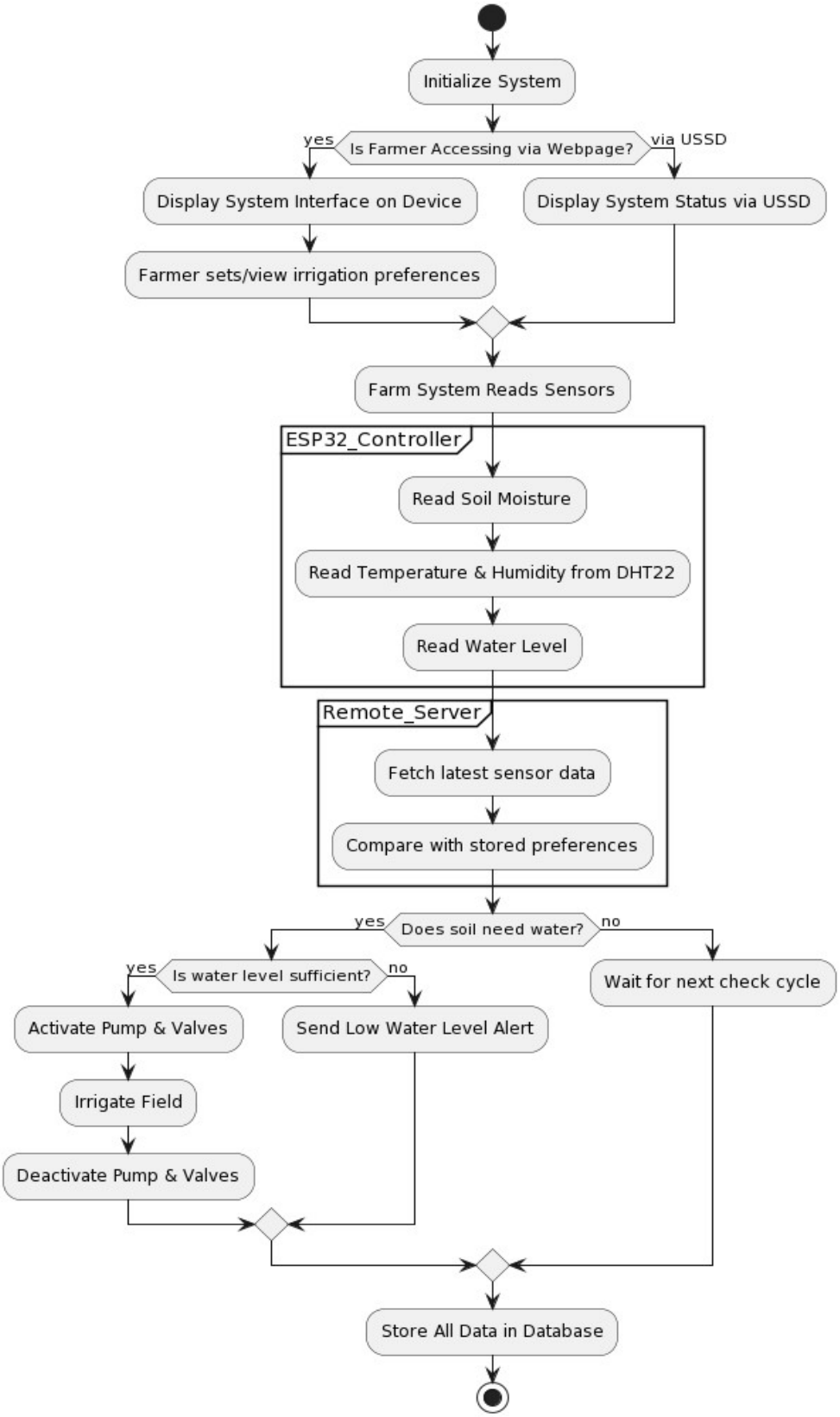

Figure 18. Activity diagram

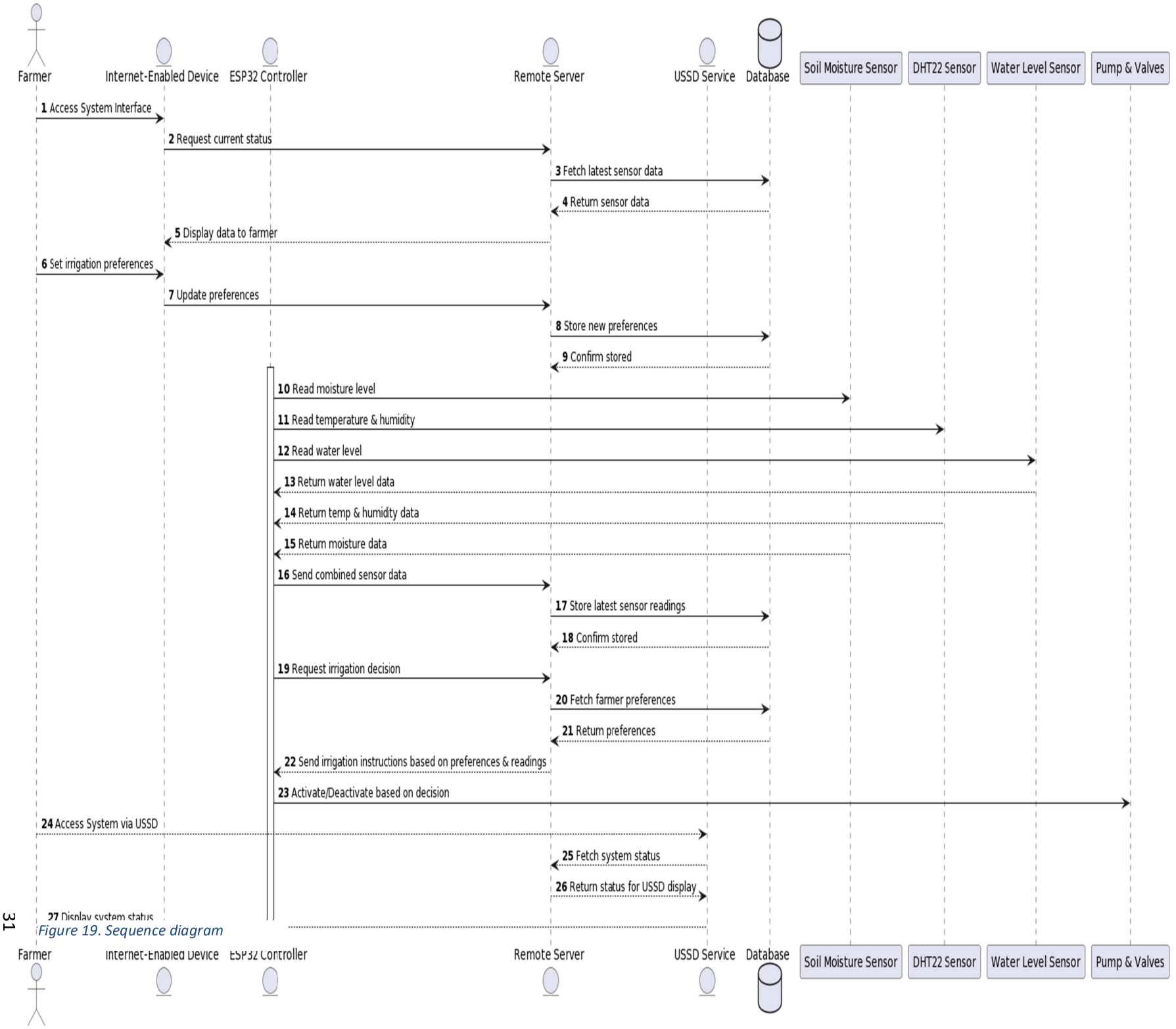

Figure 19. Sequence diagram

3.5 FLOWCHART OF THE SYSTEM

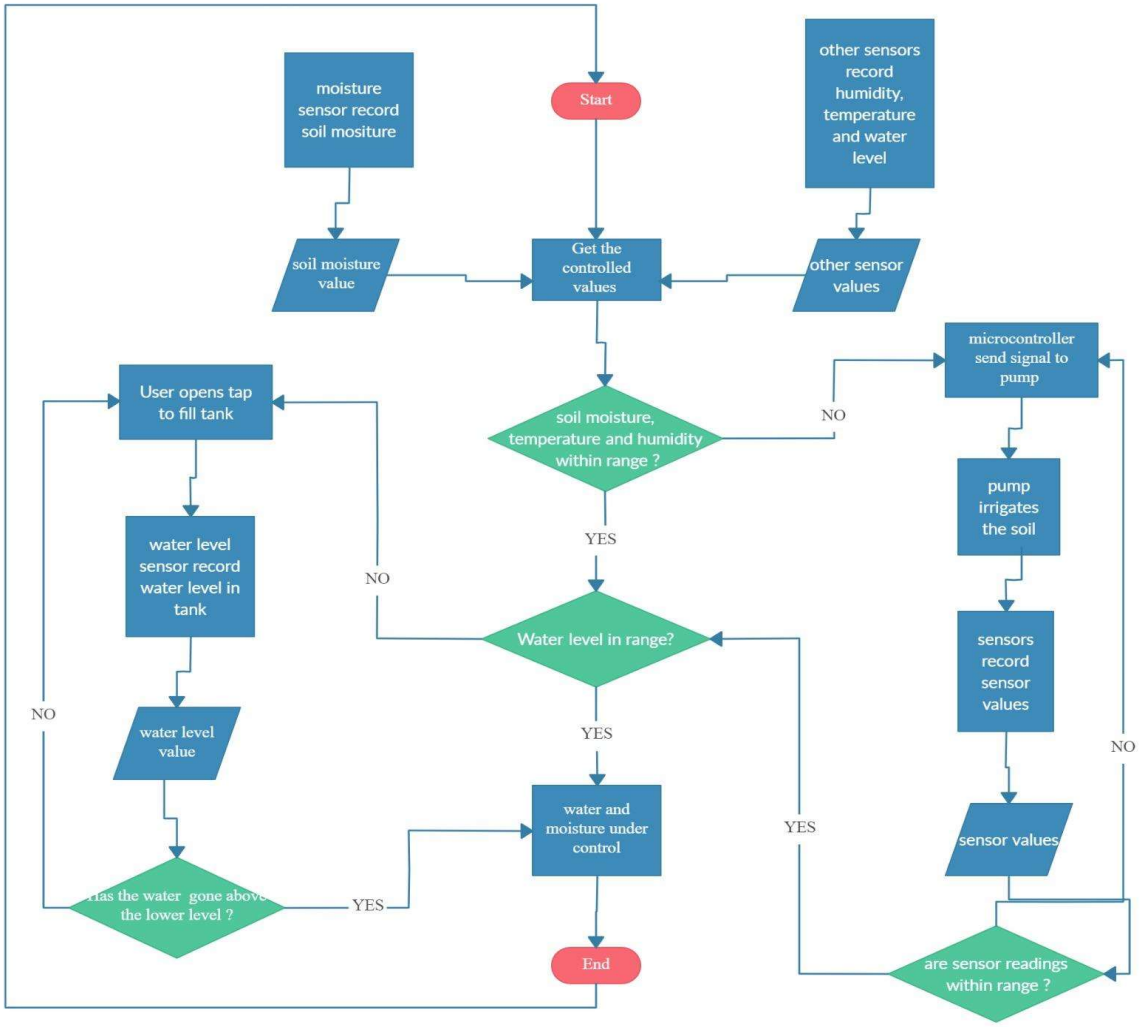

Figure 21 Flowchart diagram of the system

Figure 22 Flowchart diagram of the system

3.6 CODE SNIPPETS

Complete codes will be found at the public github repository below:

https://github.com/jcalebjohnston/finalyear_KNUST_rUrban

ESP32 Uploaded code snippet

This code snippet manages an automated irrigation system using an ESP32. The ESP32 connects to a WiFi network and communicates with a remote server to fetch system operation modes and update sensor readings. The primary functionalities of the system include:

Initialization: The ESP32 sets up WiFi communication and initializes pins for the DHT sensor (to measure temperature and humidity), an ultrasonic sensor (to measure water level), a relay (to control a water pump), and a servo motor.

Server Communication: The ESP32 periodically fetches operation mode settings (either manual or automatic) from the server. In manual mode, the water pump and servo's operations are dictated by the server's commands. In automatic mode, sensor readings determine the pump and servo actions.

Sensors Readings:

- The system measures soil moisture to determine when to water the plants.
- An ultrasonic sensor checks the water level, ensuring there's enough water before activating the pump.
- A DHT sensor monitors ambient temperature and humidity, which can also influence the decision to irrigate.

Pump & Servo Control:

- In manual mode, the pump and servo operations are directly controlled by the server's commands.
- In automatic mode, the system turns on the pump when the soil is too dry and turns it off when sufficiently moist. If conditions, like the water level being too low or the ambient temperature being too high combined with low humidity, are met, the pump will not be activated. The servo's rotation signifies the system's mode of operation.

Data Upload: The ESP32 periodically sends sensor readings (humidity, temperature, soil moisture, and water level) to the remote server.

1. **libraries and Definitions**

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <Arduino_JSON.h>
#include <ESP32Servo.h>
#include "DHT.h"
#define DHTPIN 27
#define DHTYPE DHT22
...
```

WiFi - For connecting the ESP32 to a WiFi network.

HTTPClient - Helps the ESP32 make HTTP requests.

Arduino_JSON - Used to parse JSON payloads from the server.

ESP32Servo - Provides support for controlling servo motors.

DHT - Library for the DHT sensor which measures the temperature and humidity of the environment

2. **WiFi Credentials and Server Endpoint**

```
const char* ssid = "WIFI SSID";
const char* password = "WIFI PASSWORD";
const char* serverName = "http://.../esp-outputs-
action.php";
...
```

The above are the credentials for the WiFi connection and the endpoint of the server where the ESP32 will make its HTTP requests.

3. **Sensor Initialization and Servo Configuration**

```
DHT dht(DHTPIN, DHTTYPE);
Servo myservo;
...
```

An instance for the DHT object is created for the temperature and humidity sensor and a servo object is also initialized.

4. WiFi Connection Setup

```
void setup() {
    ...
    WiFi.begin(ssid, password);
    while(WiFi.status() != WL_CONNECTED) {...}
    ...
}
```

This connects the ESP32 to the specified WiFi network and waits until a connection is made.

5. Data Acquisition and Control Logic

```
void loop() {
    ...
    int soilmoisture = analogRead(AOUT_PIN);
    float humidity = dht.readHumidity();
    float temperature = dht.readTemperature();
    …
   if(not automatic) {
  if(pumpOn) {...}
  else {...}
   }
    if(automatic) {
        ...
        if (temperature > temperatureThreshold &&
humidity < humidityThreshold) {...}
        if (soilmoisture >= soilMoistureThresholdOn)
{...}
    }
    ...
}
```

In the main loop, the ESP32 reads values from the various sensors. Based on the readings and the mode (automatic or manual), it decides whether to activate pumps or adjust servos.

6. HTTP GET Request Function

```
String httpGETRequest(const char* serverName) {...}
```

This allows the ESP32 to make an HTTP GET request to the server and retrieve the latest 'outputs' state from the database.

7. Server Communication with IoT

The script does the following:

GET Request Handling:

- The script checks if it has received a GET request containing the parameters 'humidity', 'temperature', 'soilmoisture', and 'waterlevel'.
- If all parameters are present, their values are extracted and stored in respective variables.

Database Insertion:

- It forms an SQL INSERT query to add the received sensor values into the sensor_data table in the database.
- The script then executes this query. If successful, it displays a success message. Otherwise, an error message is shown.

```
//Handle GET request from IoT device
if(isset($_GET['humidity']) && isset($_GET['temperature']) &&
isset($_GET['soilmoisture']) && isset($_GET['waterlevel'])) {
    //Extract data from GET request
    $humidity = $_GET['humidity'];
    $temperature = $_GET['temperature'];
    $soilmoisture = $_GET['soilmoisture'];
    $waterlevel = $_GET['waterlevel'];
  }
// Query for latest irrigation data
```

```
$sql = "INSERT INTO sensor_data (humidity, temperature,
soilmoisture, waterlevel) VALUES
($humidity,$temperature,$soilmoisture,$waterlevel)";
```

Web Interface and Web control center

The provided code showcases a section of a web-based interface that interacts with an automated irrigation system.

HTML Structure:

- <section id="farm-records">: This section displays the last ten farm records in a table format. These records show the humidity, temperature, soil moisture, and water level.
- <section id="control-center">: This section is the control center that lets users toggle between automatic and manual control of certain functionalities. The toggles are represented by checkboxes, and their states (checked/unchecked) are fetched from the database.

Server-side PHP:

- The code interacts with two primary PHP files, esp-database.php and farm_records.php.
- The esp-database.php is included to provide functions that fetch data from the database.
- In the given PHP block, two sets of data are retrieved: outputs (which correspond to control toggles) and boards (which represent different control boards and their last request time).
- Outputs fetched from the database determine the checked state of checkboxes in the control center.

JavaScript & AJAX:

- The <script> section contains JavaScript that fetches and displays the farm records.
- An AJAX fetch request is made to farm_records.php, which retrieves the latest ten farm records. This data is then populated in the "Farm Records" table.
- There's also a function updateOutput(element), which sends an AJAX request to update the state of a particular output in the database based on the checkbox's state.

Data Fetching from farm_records.php:

- This PHP file connects to the database and fetches the last ten records from the sensor_data table.

- After fetching the records, the data is encoded as a JSON object and sent as a response to the AJAX request.

```
<section id="farm-records" class="py-5">
    <div class="container">
        <h2 class="mb-4" id="farmrecords">Farm Records (Previous
10)</h2>
        <table class="table table-bordered">
            <thead>
                <tr>
                    <th>Date</th>
                    <th>Humidity</th>
                    <th>Temperature</th>
                    <th>Soil Moisture</th>
                    <th>Water Level</th>
                </tr>
            </thead>
            <tbody id="farm-records-content">
                <!--farm records are inserted here-->
            </tbody>
        </table>
    </div>
</section>
<?php
include_once('esp-database.php');

$result = getAllOutputs();
$html_buttons = null;
if ($result) {
    while ($row = $result->fetch_assoc()) {
```

```
        if ($row['state'] == "0") {
            $button_checked = "checked";
        } else {
            $button_checked = "unchecked";
        }
        $html_buttons .= '<h3>' . $row["name"] . ' - Board ' .
$row["board"] . ' - GPIO ' . $row["gpio"] .
        ' </h3><label class="slider-checkbox"><input
type="checkbox" onchange="updateOutput(this)" id="' . $row["id"]
. '" ' .
        $button_checked . '><span class="slider"></span></label>
<br><br><br><br><br><br>';
    }
}

$result2 = getAllBoards();
$html_boards = null;
if ($result2) {
    $html_boards .= '<h3>Boards</h3>';
    while ($row = $result2->fetch_assoc()) {
        $row_reading_time = $row["last_request"];
        $html_boards .= '<p><strong>Board ' . $row["board"] .
'</strong> - Last Request Time: ' . $row_reading_time . '</p>';
    }
}
?>

<section id="control-center" class="py-5">
    <div class="container">
        <h2 class="mb-4">Control Center</h2>
```

```
        <center><h2>rUrban Control System</h2></center>
        <center>
            ENABLE AUTOMATIC CONTROL
            <?php echo $html_buttons; ?>
           <?php echo $html_boards; ?>
            <center>
                <br><br>
    </div>
</section>



<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstr
ap.bundle.min.js"></script>
<script>
    // AJAX is used  to fetch farm records from the PHP file here
    fetch('farm_records.php')
        .then(response => response.json())
        .then(data => {
            // Process the farm records and create the HTML
content
            let farmRecordsContent = '';
            data.forEach(record => {
                farmRecordsContent += `
                    <tr>
                        <td>${record.Date}</td>
                        <td>${record.humidity}</td>
                        <td>${record.temperature}</td>
                        <td>${record.soilmoisture}</td>
                        <td>${record.waterlevel}</td>
```

```
                    </tr>
                `;
            });

            // Insert the generated HTML content into the farm
records table body
            document.getElementById('farm-records-
content').innerHTML = farmRecordsContent;
        })
        .catch(error => console.error('Error fetching farm
records:', error));



    function updateOutput(element) {
    var xhr = new XMLHttpRequest();
    if (element.checked) {
        xhr.open("GET", "esp-outputs-
action.php?action=output_update&id=" + element.id + "&state=0",
true);
    }
    else {
        xhr.open("GET", "esp-outputs-
action.php?action=output_update&id=" + element.id + "&state=1",
true);
    }
    xhr.send();
    }
</script>
```

how data is fetched from farm_records.php

```
// Fetch farm records from the 'sensor_data' table
$sql = "SELECT * FROM `sensor_data` ORDER BY `Date` DESC LIMIT
10";
```

```
$result = $conn->query($sql);
$farm_records = array();
if ($result->num_rows > 0) {
    while ($row = $result->fetch_assoc()) {
        $farm_records[] = $row;
    }
}
// Close the database connection
$conn->close();
// Output the farm records as JSON
header('Content-Type: application/json');
echo json_encode($farm_records);
```

USSD Interface and Control Center

The code is part of a USSD (Unstructured Supplementary Service Data) interface for the "rUrban" irrigation system. Users can interact with the system via mobile dialing and receive immediate feedback through textual menus and prompts. Here's a breakdown:

Initial Setup:

- The code starts by including a PHP file named pump_func.php, which presumably contains functions related to controlling the pump.

New Session:

- If it's a new session ($newSession is true), the system checks whether the user input is the USSD code or another predefined value to present the main menu options.
- If valid, a welcome message is displayed with two primary options: "Farm Sensor Readings" and "Irrigation Control".
- The system then stores this session using the save_session function.

Ongoing Session:

- If it's an ongoing session ($newSession is false), the system retrieves the user's previous interactions using the get_session function.

- If the user inputs "0", they are taken back to the main menu.
- Depending on the previously selected options (retrieved from $stages), the system processes the new user input.

Farm Sensor Readings:

- If the user selects option "1" from the main menu, the system fetches the latest sensor readings from the database and presents them to the user.
- Information about humidity, temperature, soil moisture, and water level is retrieved and displayed.

Irrigation Control:

- If the user selects option "2" from the main menu, they are provided with options to either turn the pump ON or OFF.
- Based on the user's choice, the system either starts or stops the pump using startPump() and stopPump() functions respectively.

Session Management:

- Throughout the process, the system maintains session data using the save_session function. This ensures that the user's progress and choices are remembered, allowing for multi-level USSD menus and interactions.
- If there are any errors in the process, especially during session management, the user is informed with an error message.

```
require_once "pump_func.php";

if ($newSession) {

    if ($i_ussd || (!$i_ussd && ($userData == "" || $userData == 
"{$ussd_code}"))) {

        $message = "Welcome to Akuafo, what would you like to 
check?\n";

        $message .= "1) Farm Sensor Readings\n";

        $message .= "2) Irrigation Control";


        $continueSession = true;

        $session_fb = save_session($sessionID, $msisdn, '', 
$conn);

        if (!$session_fb) {
```

```
            $message = "An error occured! Please try again.";
            $continueSession = false;
        }
    } else {
        $message = $userData;
        $message .= " is not identified!\n";
        $message .= "Please dial {$ussd_code} instead";
    }
}
if (!$newSession) {
    // Get the previous session record
    $session = get_session($sessionID, $msisdn, $conn);
    $stages = explode('*', $session['udata']);

    if ($userData == "0") {
        $message = "Welcome to rUrban, what would you like to
check?\n";
        $message .= "1) Farm Sensor Readings\n";
        $message .= "2) Irrigation Control";
        $continueSession = true;
        $session_fb = save_session($sessionID, $msisdn,
$userData, $conn);
        if (!$session_fb) {
            $message = "An error occured! Please try again.";
            $continueSession = false;
        }
    } else {
        if ($stages[0] == "" || $stages[0] == null) {
            //  Userdata is from first level Menu
```

```
            if ($userData == "1")
                // Business logic for first level response
                $readings = "SELECT * FROM sensor_data ORDER BY
SensorDataId DESC LIMIT 1";
                $results = mysqli_query($conn, $readings);
                $results->data_seek(0);
                $res = $results->fetch_array(MYSQLI_ASSOC);
                $humidity = $res['humidity'];
                $temperature = $res['temperature'];
                $soilmoisture = $res['soilmoisture'];
                $waterlevel = $res['waterlevel'];

                $message = "Your Farm Readings are";
                $message .= "\nHumidity sensor: " . $humidity;
                $message .= "\nTemperature sensor: " .
$temperature;
                $message .= "\nSoil moisture sensor: " .
$soilmoisture;
                $message .= "\nWater level sensor: " .
$waterlevel;
                $message .= "\n\nPress 0 to return to main
menu\n";
                $continueSession = true;

                $session_fb = save_session($sessionID, $msisdn,
$userData, $conn);
                if (!$session_fb) {
                    $message = "An error occured! Please try
again.";
                    $continueSession = false;
                }
```

```
            } else if ($userData == "2") {
                $message = "Turn pump on/off \n";
                $message .= "1. Turn ON \n";
                $message .= "2. Turn OFF \n";
                $continueSession = true;
                $session_fb = save_session($sessionID, $msisdn,
$userData, $conn);
                if (!$session_fb) {
                    $message = "An error occured! Please try
again.";
                    $continueSession = false;
                }
            } else {
                $message = "Wrong input!";
            }
        } else if ($stages[0] == "2") {
            // Userdata is from option2 from initail menu
            if ($userData == "1") {
                // function to start pump
                startPump();

                $message = "Pump will be turned on(This might
take a few seconds)";
                $session_fb = save_session($sessionID, $msisdn,
$userData, $conn);
                if (!$session_fb) {
                    $message = "An error occured! Please try
again.";
                    $continueSession = false;
                }
```

```
            } else if ($userData == "2") {
                // function to close pump
                stopPump();
                $message = "Pump will be turned off(This might
take a few seconds)";
                $session_fb = save_session($sessionID, $msisdn,
$userData, $conn);
                if (!$session_fb) {
                    $message = "An error occured! Please try
again.";
                    $continueSession = false;
                }
            }
        } else {
            $message = "Wrong input!";
        }
    }
}
```

pump_func.php

The provided code has three PHP functions designed to control an irrigation system:

checkmanual():

- Checks if the system is in manual mode. If not, switches it to manual mode.

startPump():

- Ensures the system is in manual mode using checkmanual().
- Activates the irrigation pump by updating the pump's state in the database.

stopPump():

- Ensures the system is in manual mode using checkmanual().
- Deactivates the irrigation pump by updating the pump's state in the database.

Note: These functions assume an existing database connection ($conn).

```
function checkmanual() {
```

```
    $check_mode = "SELECT state FROM outputs WHERE name =
'MODE'";
    $results = $conn->query($check_mode);
    $results->data_seek(0);
    $state = $results->fetch_array(MYSQLI_ASSOC)['state'];


    if ($state == 0){
        //$update_mode = "UPDATE outputs SET state = '1' WHERE
name = 'MODE
        $update_mode = "UPDATE `outputs` SET `state`='1' WHERE
`name`='MODE'";
        $conn->query($update_mode);
    }
}
function startPump() {
    checkmanual();
    $sql ="UPDATE `outputs` SET `state`='0' WHERE `name`='PUMP'";
    $conn->query($sql);
    $conn->close();
}
function stopPump(){
    checkmanual();
    $sql ="UPDATE `outputs` SET `state`='1' WHERE `name`='PUMP'";
    $conn->query($sql);
    $conn->close();
}
```

CHAPTER 4 - SYSTEM IMPLEMENTATION AND TESTING

4.1 SYSTEM IMPLEMENTATION

rUrban has been designed to offer intelligent solutions to irrigation, combining hardware and software components. Below is a vivid look at the system's architecture and its implementation:

1. **Hardware Components:**

a. ESP32 Microcontroller:

- Acts as the brain of the operation, processing data from sensors and sending commands to actuators.

b. Sensors:

- Humidity and Temperature Sensor (DHT22): Measures humidity and temperature.
- Soil Moisture Sensor: Provides readings on soil's water content.
- Ultrasonic Sensor: Determines the water level in the tank.

c. Actuators:

- Relay Module: Controls the water pump based on the data from the sensors.
- Servo Motor: Used for controlling valves or other mechanical aspects of the system.

2. **Software Components**

a. ESP32 Software:

- Written in C++ for the Arduino platform.
- Collects data from sensors at regular intervals.
- Sends the collected data to the server via Wi-Fi.
- Listens for commands from the server to activate/deactivate the pump.

b. Backend System:

- Hosted on a web server(00webhost) with PHP and MySQL support.
- Receives data from the ESP32 and stores it in the database.
- Provides an API for the frontend system to fetch and display data.
- Processes commands from the frontend and USSD interfaces to control the system.

c. Frontend Web Interface:

- Designed with HTML, CSS, and JavaScript (Bootstrap library).
- Displays real-time data fetched from the backend.
- Allows both automatic and manual control of the irrigation system.

d. USSD Interface:

- Provides mobile interaction for users without internet access.
- Enables users to check sensor readings and control the irrigation system via USSD commands.

3**. Workflow:**

- Sensors send readings to the ESP32 at regular intervals.
- The ESP32 processes the data and sends it to the server.
- The server stores the data in the MySQL database.
- Users access the web interface to view data and control the system.
- Alternatively, users without smartphones utilize the USSD interface for control and monitoring.
- Commands from users are processed by the server and sent to the ESP32.
- The ESP32 activates/deactivates the pump based on user commands or automatic conditions.

4.2 SYSTEM TESTING

FUNCTIONAL TESTING

a. Sensor Data Retrieval

Objective: To ensure the system can read data from sensors.

Test: Trigger each sensor manually and check if the ESP32 captures the changes and if the server updates the data.

Expected Outcome: The ESP32 should capture the changes and the server should reflect the manual triggers accurately.

Actual Outcome:

Test 1 - ESP32 captures the changes but the server could not update the data.

Test 2 - ESP32 captures the changes and the server updated with manual trigger sensor values correctly.

b. Pump Activation/Deactivation

Objective: to ensure that the irrigation system can be controlled both manually and automatically.

Preparation:

- Make sure the system is fully operational.
- Have a mechanism to adjust soil moisture manually (e.g., dry soil and water).
- Make sure the interface (web or USSD) is accessible.

Manual Testing via Interface:

Procedure:

- Access the web interface.
- Look for the control section.
- Activate the pump using the slider.
- Listen or feel for the pump to ensure it starts.
- Deactivate the pump using the interface.
- Listen or feel for the pump to ensure it stops.

Expected Outcome: The pump should start and stop almost instantly upon receiving commands from the interface.

Actual Outcome:

Test 1 - Though there was a little delay, the pump responded correctly.

Automatic Activation Testing:

Procedure:

- Make the soil very dry (below the threshold for automatic activation).
- Monitor the system's response.
- Once the pump activates, add water to the soil to simulate increased moisture.
- Monitor the system to see if the pump deactivates upon reaching the moisture threshold.

Expected Outcome: The pump should automatically start when the soil is dry and stop when the soil reaches adequate moisture.

Actual Outcome:

Test 1 - The pump started as expected when the soil was dry and the pump stopped after adding the water.

INTEGRATION TESTING

a. ESP32 to Server Communication:

Objective: to ensure that the ESP32 communicates correctly with the server.

Test: we sent mock data from ESP32 to the server.

Expected Outcome: The server database should update with the mock data received.

Actual Outcome:

Test 1 - The server data database updated with the mock data received.

b. Web Interface, USSD Interface and Backend Integration:

Objective: to ensure that the frontend fetches and displays data correctly.

Test: Compare the data displayed on the web interface and USSD interface with the data stored in the database.

Expected Outcome: The data should match.

Actual Outcome:

Test 1 - There was an error in the query to display the data hence the data didn't display.

Test 2 - The data displayed on the interfaces matches with the data in the server database.

USABILITY TESTING

a. Web and USSD User Interface (UI) Efficiency:

Objective: Ensure that the UI is user-friendly.

Test: We asked a group of users unfamiliar with the system to perform specific tasks, like viewing sensor readings or activating the pump.

Expected Outcome: Users should be able to complete tasks without confusion or excessive delays.

Actual Outcome: All users were able to complete tasks without any confusion or excessive delays.

PERFORMANCE TESTING

a. System Response Time:

Objective: to ensure that there's minimal delay between a user's action and the system's response.

Test: measuring the time it takes for the system to respond after a command has been issued from the web or USSD interface.

Expected Outcome: The response time should be within acceptable limits.

Actual Outcome:

Test 1 - The response time was fast enough but optimization would make it better.

Test 2 - The second time has response time better than the first test.

RELIABILITY TESTING

a. Continuous Operation:

Objective: to ensure the system can operate continuously without failures.

Test: Letting the system run continuously, capturing and logging data for an extended period.

Expected Outcome: The system should operate without interruptions or data lost.

Actual Outcome: There were minor interruptions anytime the WiFi network was not strong enough but there wasn't any data loss.

CHAPTER 5 – CONCLUSIONS AND FUTURE WORKS

5.1 FUTURE PROJECTIONS

1. Using the foundational framework established by this study, future endeavors have the exciting potential to expand the scope of automated irrigation systems. Building upon the existing architecture, forthcoming research could seamlessly integrate a sophisticated fertilizer application system. This innovative augmentation would enable a comprehensive approach to crop management, synchronizing precise irrigation with the judicious application of nutrients.

2. The integration of a dedicated mobile application holds immense promise in terms of system control and user accessibility. This forward-looking extension could empower farmers with an intuitive interface through which they can dynamically adjust irrigation parameters, monitor soil conditions, and review historical data. Such interactivity not only streamlines decision-making but also fosters a deeper understanding of crop-water dynamics.

3. Incorporating SMS alerts into the system's repertoire of features is yet another avenue ripe for exploration. By coupling the system with real-time notifications, stakeholders would receive immediate updates on critical events such as low soil moisture levels or impending irrigation cycles. This proactive communication mechanism would effectively bridge the temporal gap between automated system operations and human oversight, ensuring timely interventions and optimizing resource utilization.

In essence, the potential enhancements outlined here signify a progression towards a comprehensive and holistic agricultural management system. The envisioned integration of a fertilizer application system, a user-friendly mobile application, and SMS alerts builds upon the groundwork laid by this study, heralding a new era of precision-driven, technologically empowered farming practices.

5.2 CONCLUSION

The automated farm irrigation system using ESP32 microcontroller and IoT technology has experimentally proven to work satisfactorily. We successfully automated the irrigation process based on soil moisture level, temperature and humidity conditions while implementing remote control through a web and Ussd system interface using IoT technology; thus, allowing farmers to remotely and automatically monitor irrigation of their farm.

5.3 REFERENCES

1. Journal Article:
   Smith, A. B., Johnson, C. D., & Martinez, E. F. (2020). Advances in automated irrigation systems: A review of sensor technologies and control strategies. Agricultural Engineering Journal, 15(3), 123-140.

2. Conference Paper:
   Garcia, M. L., Rodriguez, J. R., & Kim, S. H. (2019). Integration of wireless sensor networks for automated irrigation control in precision agriculture. In Proceedings of the International Conference on Agricultural Innovation (ICAI 2019) (pp. 78-85).

3. Book:
   Thompson, R. L., & Johnson, L. M. (2018). Modern Irrigation Systems: Design, Management, and Applications. Springer.

4. Thesis/Dissertation:
   Davis, P. H. (2017). Development and Evaluation of an IoT-Based Automated Irrigation System for Sustainable Agriculture (Doctoral dissertation). University of Delhi.

5. Online Resource:
   World Bank. (2021). Smart Irrigation: Harnessing Technology for Sustainable Water Use in Agriculture. https://www.worldbank.org/en/topic/water/brief/smart-irrigation

6. Surendran, K., & Murugesan, S. (2019). Automated irrigation system: A review. Journal of Irrigation and Drainage Engineering, 145(8), 04019003. https://doi.org/10.1061/(ASCE)IR.1943-4774.0001392

7. Research on automatic irrigation control: State of the art and recent results - ScienceDirect