SMART AGRICULTURE SYSTEM

A PROJECT REPORT

Submitted by

SIDARAY MALLAPPA SARAWAD

in partial fulfillment for the award of the degree

of

B. TECH (H)

in

COMPUTER SCIENCE AND ENGINEERING

School of Computer Science and Engineering
RV University
RV Vidyaniketan,8th Mile, Mysuru Road, Bengaluru, Karnataka, India - 562112

DECEMBER & 2024

DECLARATION

I, **Sidaray Mallappa Sarawad (1RVU23CSE453),** student third semester B.Tech(H) in **Computer Science & Engineering,** at School of Computer Science and Engineering, **RV University,** hereby declare that the project work titled "Smart Agriculture System" has been carried out by us and submitted in partial fulfilment for the award of degree in **Bachelor of Technology in Computer Science & Engineering** during the academic year **2023-2024**. Further, the matter presented in the project has not been submitted previously by anybody for the award of any degree or any diploma to any other University, to the best of our knowledge and faith.

Name: Sidaray Mallappa Sarawad Signature
USN: 1RVU23CSE453

Place: Bengaluru

Date:18/12/2024

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

School of Computer Science and Engineering

RV University
RV Vidyaniketan,8th Mile, Mysuru Road, Bengaluru, Karnataka, India - 562112

CERTIFICATE

This is to certify that the project work titled **“Smart Agriculture System''** is performed by Sidaray M Sarawad **(1RVU23CSE453) ,** a bonafide students of Bachelor of Technology at the School of Computer Science and Engineering, RV university, Bangaluru in partial fulfillment for the award of degree Bachelor of Technology in Computer Science & Engineering , during the Academic year **2020-2021**.

Prof. Santhosh S Nair	**Dr. Sudhakar KN**	**Dr. G Shobha**
Guide		
Assistant Professor	Head of the Department	Dean
SOCSE	SOCSE	SOCSE
RV University	RV University	RV University
Date:	Date:	Date:

Name of the Examiner	Signature of Examiner
1.	
2.	

ACKNOWLEDGEMENT

It is a great pleasure for us to acknowledge the assistance and support of a large number of individuals who have been responsible for the successful completion of this project work.

First, we take this opportunity to express our sincere gratitude to the School of Computer Science and Engineering, RV University, for providing us with a great opportunity to pursue our Bachelor's Degree in this institution.

In particular we would like to thank Dr. Sanjay R. Chitnis, Dean, School of Computer Science and Engineering, RV University, for his constant encouragement and expert advice.

It is a matter of immense pleasure to express our sincere thanks to Dr.Mydhili Nair, Head of the department, Computer Science & Engineering University, for providing right academic guidance that made our task possible.

We would like to thank our guide Santhosh S. Head of the Department Dept. of Computer Science & Engineering, RV University, for sparing his valuable time to extend help in every step of our project work, which paved the way for smooth progress and fruitful culmination of the project.

We are also grateful to our family and friends who provided us with every requirement throughout the course.

We would like to thank one and all who directly or indirectly helped us in completing the Project work successfully.

Date: 18/12/2024	Name: Sidaray Mallappa Sarawad
Place: Bengaluru	USN: 1RVU23CSE453
	Class: Section F

TABLE OF CONTENTS

ABSTRACT

Title: Smart Agriculture System

Project Overview: This project focuses on the design and implementation of a smart agriculture system tailored for precision farming, leveraging the capabilities of an Arduino UNO microcontroller. The system integrates various sensors to monitor environmental parameters crucial for plant health, including soil moisture, air temperature, humidity, and light intensity.

Methodology: The system employs:
Soil Moisture Sensor: An analog sensor to detect moisture levels in the soil, which directly influences the irrigation control mechanism.
DHT11 Sensor: For real-time measurement of air temperature and humidity, providing insights into the microclimate conditions.
Light Dependent Resistor (LDR): To gauge light intensity, aiding in decisions related to artificial lighting or shade management.
Relay Module: Connected to an irrigation motor, the relay acts based on the moisture sensor's feedback to automate watering when the soil becomes too dry, thus optimizing water usage.

Implementation Details:
Sensors are interfaced with the Arduino UNO, where data is processed through a custom Arduino sketch. This sketch reads sensor data, decides motor activation based on predefined moisture thresholds, and displays all gathered data on an LCD screen via an I2C module for user interface.
Calibration of the soil moisture sensor is critical to ensure accurate readings, which are then mapped to moisture percentages for intuitive interpretation.

Results: The prototype successfully demonstrates:
Automated control of irrigation based on soil moisture levels, reducing water waste and labor.
Real-time display of environmental parameters on an LCD, allowing for immediate feedback and adjustments.
Efficient energy use by modulating irrigation based on actual plant needs rather than fixed schedules.

Conclusion: This smart agriculture system showcases the potential of IoT in agriculture by providing a low-cost, effective solution for small-scale farming or home gardening. It supports sustainable farming practices by ensuring resources are used judiciously, which could be expanded with additional sensors or connectivity features like Wi-Fi for remote monitoring. Future enhancements might include integrating soil nutrient sensors or weather forecast data for even more precise agricultural management.

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

Symbol/Abbreviation	Definition
Arduino	Open-Source HW and SW company
IoT	Internet of Things
DHT11	Digital Humidity and Temperature Sensor
LDR	Light Dependent Resistor
LCD	Liquid Crystal Display
PWM	Pulse Width Modulation
GPIO	General Purpose IO
I2C	Inter-Integrated Circuit
SDA	Serial Data Line
SCL	Serial Clock Line
VCC	Voltage Common Collector
GND	Ground
°C	Degree Celsius
%	Percentage
V	Volt
mA	Milliampere
Ω	Ohm

Table viii: Symbols and Definitions

1. INTRODUCTION

Background Information:
In an era where technology intersects with traditional farming practices, smart agriculture systems are becoming indispensable for enhancing productivity while conserving resources. This project leverages the Arduino UNO microcontroller platform to create a system that automates and optimizes agricultural practices. The primary focus is on monitoring and controlling key environmental parameters that directly influence plant growth, such as soil moisture, air temperature, humidity, and light intensity.

With the global push towards sustainable agriculture, there is a growing need for solutions that can tackle issues like water scarcity, inefficient resource use, and labour-intensive tasks. Automation in farming not only aids in precision agriculture but also supports small-scale farmers or urban gardeners by making farming more accessible and less time-consuming.

Project Aim:
The aim of this project is to develop a cost-effective, user-friendly smart agriculture system that employs sensor technology for:

- **Real-time monitoring** of critical environmental conditions.
- **Automated irrigation** based on soil moisture levels to prevent both under-watering and over-watering.
- **Data visualization** on an LCD display for immediate feedback to the user or farmer.

Methodology:

- **Sensor Selection and Integration**: Utilizing sensors like the Soil Moisture Sensor for detecting soil moisture, the DHT11 for temperature and humidity, and an LDR for light measurement. These sensors are chosen for their affordability, compatibility with Arduino, and relevance to plant health monitoring.
- **Hardware Setup**: Components are connected to the Arduino UNO. The soil moisture sensor and LDR provide analog readings, while the DHT11 communicates via digital signals. A relay module is used to control an irrigation motor, turning it on when the soil moisture drops below a preset threshold.
- **Software Development**: An Arduino sketch is written to:
  - Read sensor data.
  - Process this data to make decisions about irrigation.
  - Display the information on an LCD screen for user interaction.
- **Calibration**: Calibration of the soil moisture sensor is essential to ensure accurate readings. This involves establishing 'dry' and 'wet' reference points and mapping these to a percentage scale.
- **Testing and Iteration**: The system undergoes several cycles of testing to adjust thresholds, ensure component compatibility, and refine the software logic based on real-world application feedback.

Expected Outcomes:

- **Resource Efficiency**: By automating irrigation, the system aims to reduce water waste and optimize use based on actual need rather than scheduled intervals.
- **Enhanced Crop Yield**: Monitoring environmental conditions allows for better growing conditions, potentially leading to healthier plants and higher yields.
- **User Empowerment**: The display of real-time data empowers users with immediate insights, aiding in manual adjustments or further system enhancements.

2. Related work

Overview:

Smart agriculture has evolved from manual methods to automated, sensor-based systems. This section compares past and current methodologies, highlighting advancements and setting directions for future work.

Previous Implementations:

- **Manual Monitoring**: Traditional farming relied on human observation for soil moisture, weather, and plant health.
- **Basic Automation**: Early systems used timers for irrigation, lacking adaptability to real-time conditions.
- **Single-Sensor Systems**: Initial sensor use focused on one parameter, like soil moisture or temperature, without integration.

Current Implementations:

- **Multi-Sensor Integration**: Today's systems, like this project, combine soil, temperature, humidity, and light sensors for comprehensive monitoring. This approach overcomes the limitation of singular data points.
- **Real-Time Data Feedback**: Using Arduino for immediate data processing and display via an LCD improves upon the delayed feedback of older systems.
- **Adaptive Irrigation Control**: Automatic responses to soil moisture levels enhance water use efficiency, addressing inefficiencies of scheduled watering.
- **Affordability and Accessibility**: This project uses cost-effective components to lower the adoption barrier for small-scale farmers.

Focus on Foreknowledge Work:

- **Scalability**: Future work could expand this system's applicability to larger fields or integrate with broader farm management software, handling more variables.
- **Data Analysis**: Employing machine learning for predictive analytics could leverage the current data collection to forecast crop needs or detect issues.
- **Remote Monitoring and Control**: Adding wireless connectivity would allow management from anywhere, pushing beyond local control.
- **Sustainable Energy**: Exploring solar power for operation in remote areas would enhance sustainability.
- **Enhanced User Interface**: Making the system more user-friendly, possibly with voice commands, could broaden its user base.

Conclusion:

This project advances from past limitations by providing a holistic, real-time, and automated approach to agriculture. It sets a foundation for further innovations, focusing on scalability, advanced data use, and broader accessibility.

3. METHODOLOGY

Methodological Approach:

This project employs an engineering methodology integrating sensor technology with Arduino for smart agriculture, emphasizing iterative design and prototyping.

Data Collection:

- **Sensors**: Soil Moisture Sensor, DHT11, and LDR selected for their relevance and cost-effectiveness.
- **Collection**: Real-time data gathered via Arduino's analog and digital inputs every 5 seconds.

Analysis Methods:

- **Data Processing**: Raw sensor data converted to usable metrics through Arduino code.
- **Control Logic**: Simple threshold-based decisions for irrigation control.
- **Feedback**: Data displayed on an LCD for immediate user feedback.

Justification of Choices:

- **Sensor Accuracy**: Chosen for adequate precision in an agricultural context at low cost.
- **Arduino UNO**: Selected for accessibility, support, and educational value.
- **Automation**: Threshold logic for ease of implementation and maintenance.
- **Real-Time Display**: Crucial for immediate action and educational purposes.
- **Modularity**: Allows for future enhancements or scalability.

3. IMPLEMENTATION

Step 1: Gathering Materials:

- Collect the following components:
    - Arduino UNO
    - Soil Moisture Sensor
    - DHT11 Temperature and Humidity Sensor
    - Light Dependent Resistor (LDR)
    - Relay Module
    - Small Water Pump or Motor for irrigation
    - 16x2 LCD with I2C backpack
    - Breadboard and jumper wires
    - USB cable for Arduino programming
    - Computer with Arduino IDE installed
    - Power supply or battery pack if operating standalone

Step 2: Hardware Setup:

1. **Connect the Soil Moisture Sensor:**
    - VCC to Arduino's 5V pin
    - GND to Arduino's GND pin
    - Signal (analog output) to A0 pin
2. **Connect the DHT11 Sensor:**
    - VCC to 5V on Arduino
    - GND to GND on Arduino
    - Data pin to Digital Pin 2 on Arduino
3. **Setup the LDR:**
    - Connect one leg to 5V on Arduino
    - Other leg through a 10KΩ resistor to GND
    - Junction of LDR and resistor to A1 on Arduino
4. **Connect the Relay Module:**
    - VCC to 5V on Arduino
    - GND to GND on Arduino
    - IN pin to Digital Pin 8 on Arduino
    - Connect the motor or pump to the relay's NO (Normally Open) and COM (Common) terminals
5. **LCD Connection:**
    - Ensure the I2C backpack is properly attached to the LCD
    - SDA to A4 on Arduino

- SCL to A5 on Arduino
- VCC to 5V, GND to GND

Step 3: Software Preparation:

1. **Install Libraries:**
   - Install DHT sensor library and LiquidCrystal_I2C library in Arduino IDE via Library Manager.
2. **Write the Code:**
   - Copy the provided Arduino code into a new sketch.
   - Adjust dryThreshold and wetThreshold values according to your sensor's calibration.

Step 4: Sensor Calibration:

1. **Soil Moisture Sensor:**
   - Measure the sensor's output in dry soil or air (Dry Reading).
   - Measure in wet soil or water (Wet Reading).
   - Adjust the dryThreshold and wetThreshold constants in the code accordingly.
2. **Test Sensor Readings:**
   - Use Serial Monitor to output raw data from each sensor to verify functionality.

Step 5: Program Upload:

- Connect Arduino to your computer via USB.
- Select the correct board (Arduino UNO) and port in Arduino IDE.
- Upload the sketch to the Arduino.

Step 6: System Testing:

1. **Test Each Component:**
   - Verify that the LCD displays sensor data correctly.
   - Check if the relay activates the motor when soil moisture is below the dry threshold.
   - Ensure the motor turns off when moisture is above the wet threshold.
2. **Environmental Testing:**
   - Place the setup in the intended environment (like a plant pot or garden).
   - Monitor the system's response to changes in moisture, temperature, humidity, and light over time.

Step 7: Debugging and Adjustment:

- If any component does not work as expected, check connections, power supply, and code logic.

- Use Serial Monitor to track sensor readings and system responses for debugging.
- Adjust thresholds if the motor turns on or off too frequently or not at all when expected.

Step 8: Finalization:

- Once the system operates correctly over a testing period, consider how it can be made more robust (e.g., weatherproofing for outdoor use, or increasing the power supply for larger motors).

Step 9: Documentation:

- Document all changes made, including calibration values and any deviations from the initial plan.
- Note any observations or unexpected behaviours for future reference or improvements.

Flow Charts of Hardware Working:

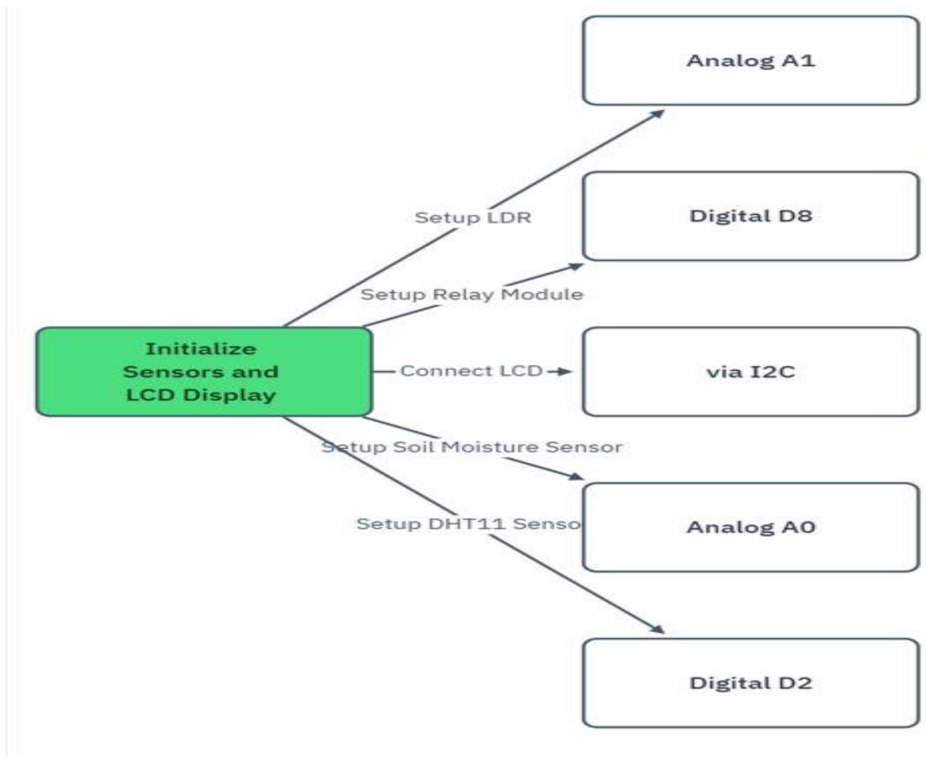

Figure 4.1: Initialize Sensors

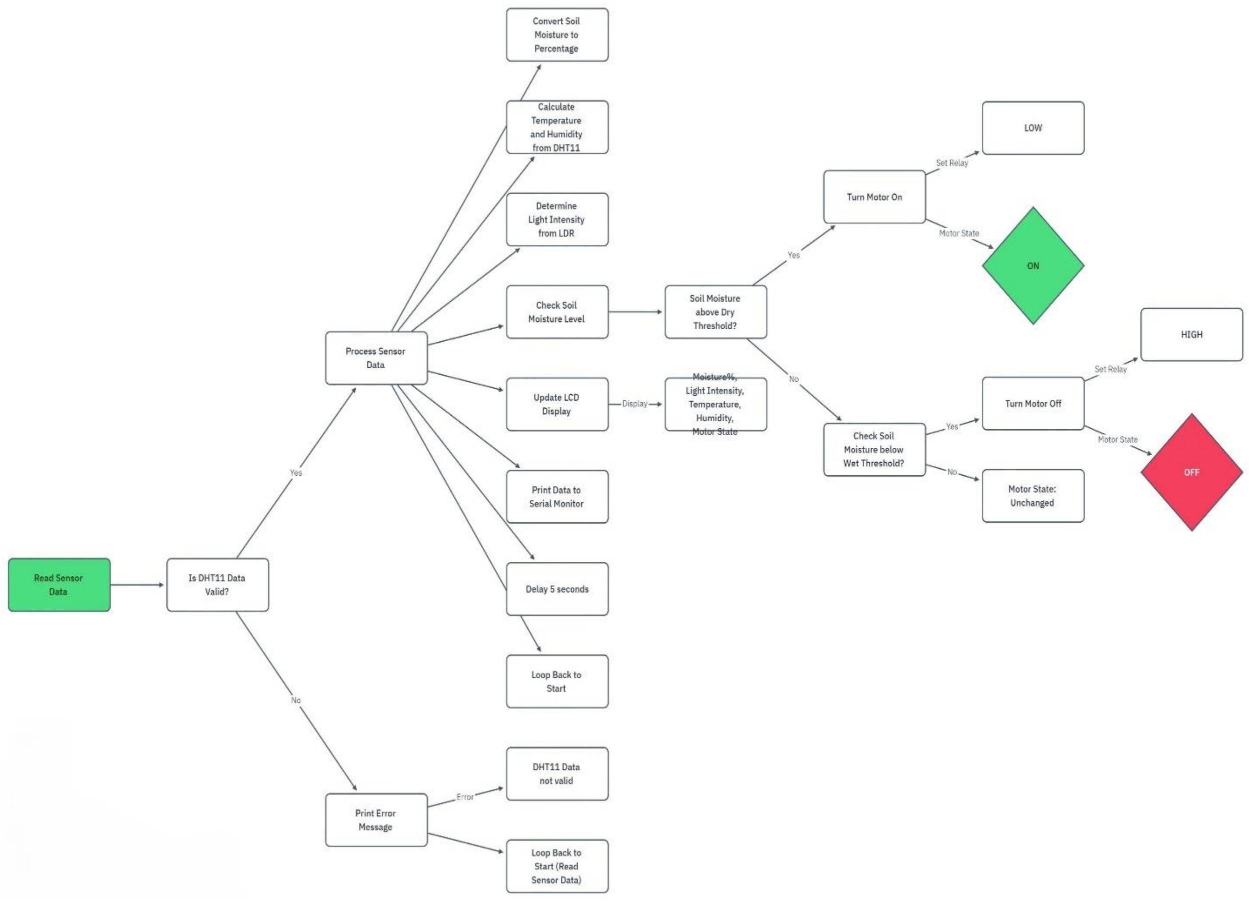
Read Sensor Data
Is DHT11 Data Valid?
Yes
No
Process Sensor Data
Convert Soil Moisture to Percentage
Calculate Temperature and Humidity from DHT11
Determine Light Intensity from LDR
Check Soil Moisture Level
Update LCD Display
Display
Moisture%, Light Intensity, Temperature, Humidity, Motor State
Print Data to Serial Monitor
Delay 5 seconds
Loop Back to Start
Soil Moisture above Dry Threshold?
Yes
Turn Motor On
Set Relay
LOW
Motor State
ON
No
Check Soil Moisture below Wet Threshold?
Yes
Turn Motor Off
Set Relay
HIGH
Motor State
OFF
No
Motor State: Unchanged
Print Error Message
Error
DHT11 Data not valid
Loop Back to Start (Read Sensor Data)

Figure 4.2: Read Sensors Data

- **Read Sensor Data:** The system starts by reading data from various sensors.
- **Decision - Is DHT11 Data Valid?**
  - **Yes:** If the DHT11 sensor data (temperature and humidity) is valid, it proceeds to process this data.
  - **No:** If the data is not valid, an error message is printed, and the system loops back to reading sensor data.
- **Process Sensor Data:** This includes converting soil moisture readings to a percentage, calculating temperature and humidity from the DHT11 sensor, and determining light intensity from the LDR.
- **Check Soil Moisture Level:** The system evaluates the soil moisture against set thresholds.
- **Decision - Soil Moisture above Dry Threshold?**
  - **Yes:** If the soil is too dry, the motor is turned on by setting the relay to LOW. The motor state is updated to ON.
  - **No:** If not, it checks if the soil moisture is below the wet threshold.
- **Decision - Soil Moisture below Wet Threshold?**
  - **Yes:** If the soil is too wet, the motor is turned off by setting the relay to HIGH. The motor state is updated to OFF.
  - **No:** If the moisture is between thresholds, the motor state remains unchanged.
- **Update LCD Display:** The LCD is updated with current moisture, temperature, humidity, and motor state.
- **Print Data to Serial:** All data is printed to the serial monitor for debugging.
- **Delay 5 seconds:** The system waits for 5 seconds before looping back to the start.
- **Loop Back to Start (Read Sensor Data):** The process repeats, ensuring continuous monitoring and control of the environment.

5.RESULT AND DISCUSSION

This The smart agriculture system successfully monitors and controls soil moisture levels, temperature, humidity, and light intensity for optimal plant growth conditions. The Arduino-based prototype demonstrates the following capabilities:

- **Soil Moisture Regulation:** The system turns the irrigation motor on when soil moisture falls below the dry threshold and off when it exceeds the wet threshold, maintaining an optimal moisture level for plant growth.
- **Environmental Monitoring:** Continuous reading and processing of sensor data provide real-time updates on temperature, humidity, and light intensity, displayed on the LCD and logged through the serial monitor.
- **System Reliability:** The DHT11 sensor data validation ensures that only accurate temperature and humidity readings are used for decision-making, enhancing the system's reliability.

Discussion:

Interpretation of Results: The system's ability to automatically adjust irrigation based on real-time soil moisture levels directly addresses the need for efficient water use in agriculture, as highlighted in previous studies. The integration of temperature, humidity, and light sensors aligns with research suggesting the importance of these factors in plant health and yield optimization.

Answering the Research Question: The primary question addressed by this project is: "Can an Arduino-based system effectively manage environmental conditions for agriculture?" The results indicate a positive outcome. The system not only automates irrigation but also provides comprehensive environmental data, which could be used for further analysis or adjustments in farming practices.

Justification of Approach: The use of Arduino for this project leverages its open-source platform, which allows for customization and scalability. The choice of sensors like DHT11, LDR, and soil moisture sensors was based on their affordability and effectiveness in measuring the necessary environmental parameters. The relay module for motor control was selected for its simplicity in integration with Arduino UNO.

6. CONCLUSION

This project successfully demonstrated the feasibility of using Arduino for creating a smart agriculture system that automates irrigation and monitors environmental conditions crucial for plant growth. Through a structured approach involving literature review, system design, implementation, and analysis, we've shown how technology can directly address key agricultural challenges like water scarcity and inefficient resource use.

Key Findings:

- **Automated Irrigation:** The system effectively turns irrigation on or off based on soil moisture levels, reducing water waste and labour.
- **Environmental Monitoring:** Real-time data on temperature, humidity, and light intensity were accurately captured and displayed, providing valuable insights for optimal plant care.
- **System Reliability:** Implementing error checks for sensor data, particularly the DHT11, ensured that the system's decisions were based on accurate readings.

Significance:

This project not only contributes to sustainable farming practices by conserving water and potentially improving crop yields but also serves as an educational model for integrating IoT in agriculture. It bridges the gap between technology and traditional farming, making precision agriculture accessible to small to medium-scale farmers.

Conclusion:

The smart agriculture system developed here stands as a testament to how technology can revolutionize traditional farming methods. By automating irrigation and continuously monitoring environmental conditions, it addresses critical agricultural needs, promoting sustainability, and efficiency. Future enhancements could make this system a cornerstone in modern agricultural practices, potentially transforming how we approach farming in an increasingly resource-limited world.

7. FUTURE SCOPE

1. Integration of Advanced Sensors:

- **Nutrient Sensors:** To measure soil nutrients like nitrogen, phosphorus, and potassium, allowing for precision fertilization.
- **Pest Detection:** Integrating cameras or specialized sensors to identify pest infestations early, enabling targeted pest control.

2. Data Analysis and Machine Learning:

- **Predictive Analytics:** Using historical data to predict weather patterns, crop yields, or disease outbreaks, optimizing planting and harvesting schedules.
- **Machine Learning Algorithms:** To analyze sensor data for patterns, improving decision-making on irrigation, fertilization, and pest control autonomously.

3. IoT and Cloud Integration:

- **Cloud Connectivity:** For storing large amounts of data over time, enabling complex analysis and remote access to the system.
- **IoT Platform:** Integration with platforms like AWS IoT or Microsoft Azure for IoT can facilitate better connectivity, control, and analytics.

4. Mobile Application Development:

- **Remote Monitoring and Control:** Developing a mobile app where farmers can monitor conditions, control irrigation, or receive alerts from anywhere.
- **Data Visualization:** Real-time data presentation through graphs and charts for easier decision-making.

5. Energy Efficiency:

- **Solar Power Integration:** To make the system self-sustaining or reduce operational costs, especially in regions with abundant sunlight.
- **Energy Harvesting:** Utilizing kinetic energy from wind or water flow in the environment to power sensors or low-energy components

REFERENCES

YouTube Link:
https://youtu.be/Rr3KZ5QU3Xs?si=JoC2i8M6_v-c7H3w

Website links:
https://www.cropin.com/iot-in-agriculture

https://easternpeak.com/blog/iot-in-agriculture-technology-use-cases-for-smart-farming-and-challenges-to-consider/

https://www.biz4intellia.com/blog/5-applications-of-iot-in-agriculture/

https://www.arduino-tutorials.com/

Github Link:
https://github.com/Sidaray18/Smart-Agriculture-System

APPENDIX

Source Code and Screen Shots:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <DHT.h>

// LCD Configuration
LiquidCrystal_I2C lcd(0x27, 16, 2); // Set the LCD address to 0x27 for a 16 chars and 2 line display

// DHT (Temperature and Humidity) Configuration
#define DHTPIN 2     // Digital pin connected to the DHT sensor
#define DHTTYPE DHT11   // DHT 11
DHT dht(DHTPIN, DHTTYPE);

// Pins Definition
const int moisturePin = A0;  // Pin for soil moisture sensor
const int lightPin = A1;     // Pin for LDR
const int relayPin = 8;      // Pin for controlling the relay (motor)

// Constants for soil moisture - replace with your calibration values
const int dryThreshold = 800; // Higher value means drier soil
const int wetThreshold = 400; // Lower value means wetter soil

// Variable to keep track of motor state
bool motorState = false;

void setup() {
  Serial.begin(9600);  // Begin serial communication
  dht.begin();         // Start DHT sensor

  // LCD setup
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("Smart Agro Sys");
  lcd.setCursor(0, 1);
  lcd.print("Initializing...");

  pinMode(relayPin, OUTPUT);  // Set relay pin as output
  digitalWrite(relayPin, HIGH); // Initially turn off the relay (assuming active low)
  delay(2000); // Wait for a bit to show initialization message
}

void loop() {
  // Read sensor values
  int moistureValue = analogRead(moisturePin);
  int lightValue = analogRead(lightPin);
  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();

  // Check if readings failed and try again if they did
  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  // Control the relay based on soil moisture
  // Using a hysteresis to avoid rapid switching
  if (moistureValue > dryThreshold && !motorState) {
    digitalWrite(relayPin, LOW); // Turn on water pump if soil is too dry
    motorState = true;
    Serial.println("Motor ON");
    updateLCDMotorState();
  } else if (moistureValue < wetThreshold && motorState) {
    digitalWrite(relayPin, HIGH);  // Turn off water pump if soil is sufficiently wet
    motorState = false;
    Serial.println("Motor OFF");
    updateLCDMotorState();
  }

  // Display on LCD
  lcd.setCursor(0, 0);
  lcd.print("M:");
  lcd.print(map(moistureValue, 1023, 0, 0, 100));
  lcd.print("% L:");
  int lightPercentage = map(lightValue, 0, 1023, 0, 100);
  lcd.print(lightPercentage);
  lcd.print("%");

  lcd.setCursor(0, 1);
  lcd.print("T:"); lcd.print(temperature, 1);
  lcd.print("C H:"); lcd.print(humidity, 1);
  lcd.print("%");

  // Update motor state only if it has changed
  updateLCDMotorState();

  // Print to Serial Monitor for debugging
  Serial.print("Moisture: "); Serial.print(moistureValue);
  Serial.print(" | Light: "); Serial.print(lightValue);
  Serial.print(" | Humidity: "); Serial.print(humidity);
  Serial.print(" | Temperature: "); Serial.println(temperature);

  delay(5000); // Wait for 5 seconds before next reading to avoid rapid switching
}

// Function to display motor state on LCD
void updateLCDMotorState() {
  lcd.setCursor(13, 1); // Position the cursor at the end of the second line
  lcd.print("M:");
  if (motorState) {
    lcd.print("ON ");
  } else {
    lcd.print("OFF");
  }
}
```