**Project Report:**

Syed Yousaf Shah (FA21-BCE-028)

Syed M.Shahir Ul Haq (FA21-BCE-014)

**Lab Title :**

**“Smart Agriculture Control”**

1. **Abstract:**

The Smart Farming System automates the monitoring and control of key environmental parameters for agriculture, including temperature, soil moisture and humidity. Using an Arduino Uno as the central controller, the system improves efficiency and conserves resources. This report details the design, implementation, and results of the project.

**Contents Of Reports:**

1. **Background**
2. **Introduction**
3. **Methodology**
   * **Software Components**
   * **Circuit Diagram**
   * **Implementation**
4. **Results**

**Background:**

Agriculture is the backbone of the economy in many countries, playing a crucial role in providing food, raw materials, and employment. However, traditional farming methods face numerous challenges such as inefficient resource usage, unpredictable weather patterns, and labor shortages. To address these issues, modern technology is being integrated into agricultural practices, leading to the emergence of "smart agriculture."our project also aim to contribute to the idea of smart agriculture system.

**Introduction:**

Smart Agriculture Control, aims to develop a system that automates and optimizes various agricultural processes using modern technology. The system is designed to monitor critical environmental parameters such as temperature, soil moisture, and humidity, and to control devices like fans and water pumps based on the real-time data collected.

**Problem Statement:**

The primary objective is to design a system that can monitor and control environmental conditions in a farm, optimizing water use and maintaining optimal growing conditions for crops.

**Objectives:**

* Monitor soil moisture, temperature, humidity, and rainfall.
* Automate the irrigation process based on real-time data.
* Display environmental data on an LCD screen.

**Scope:**

The system is intended for small to medium-sized farms. It will help farmers monitor and control environmental conditions like soil moisture, temperature, and humidity. In the future, the system could be upgraded to allow farmers to monitor and control their farms remotely using smartphones or computers, making it even easier to manage their crops from anywhere.

**Why we choose this project:**

We chose the Smart Agriculture Control project because it helps farmers in important ways. It uses technology to make farming more efficient and productive. The system saves water and energy, cutting down on waste and costs. It helps grow more and better crops by keeping the best conditions for plants. Automation reduces the need for manual labor, saving farmers time and effort. It also supports sustainable farming, which is good for the environment. Plus, it gives real-time data on conditions, making it easier for farmers to monitor and manage their crops.

**Methodology:**

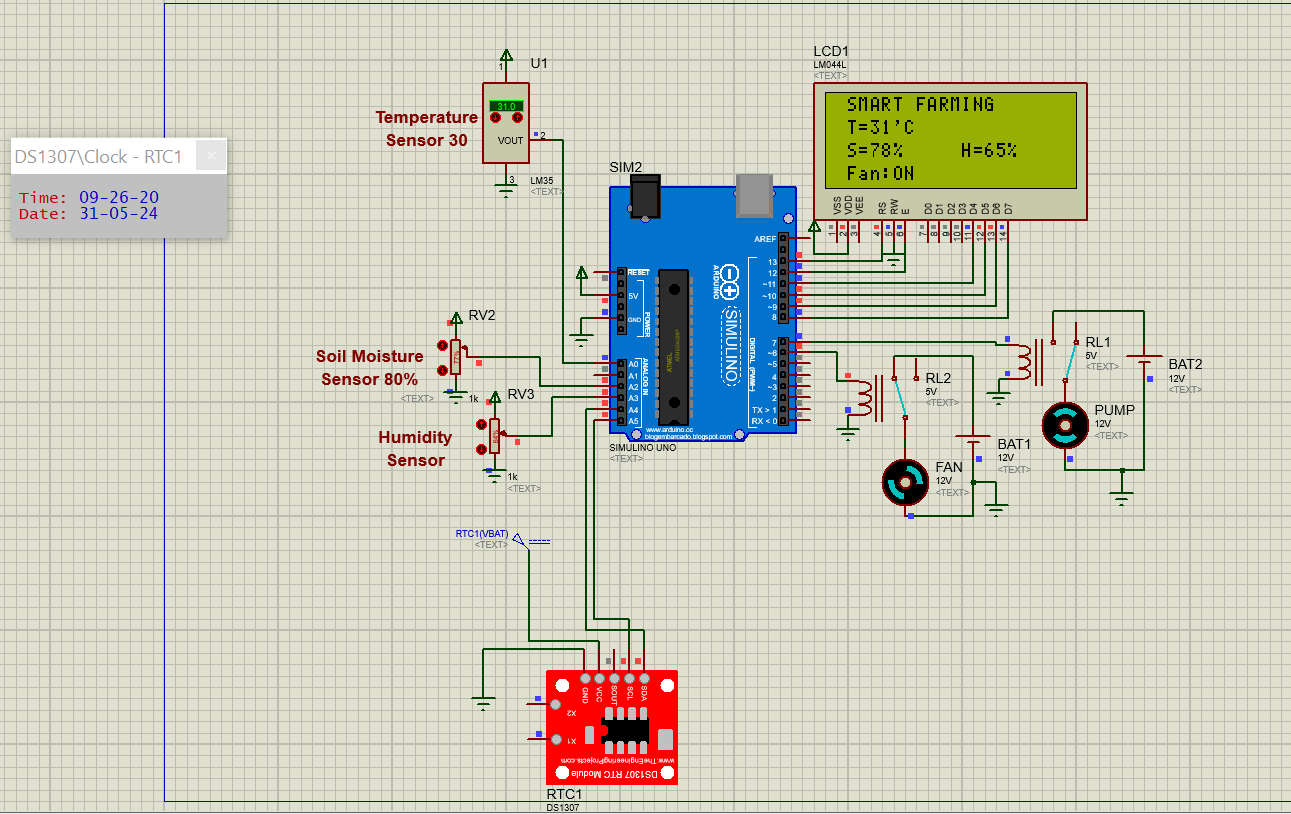
This is simulation-based project for which the Proteus version 8.13 is used for simulation for coding Arduino IDE is used.

* For simulation Proteus 8.13 is used
* For coding Arduino IDE is used

**For Hardware the following components can be utilized:**

* **Arduino Uno:** Central processing unit
* **LM35 Temperature Sensor:** Measures temperature
* **Soil Moisture Sensor:** Measures soil moisture(LM393 based design)
* **Humidity Sensor:** Monitors humidity(DHT11 can be used)
* **DS1307 RTC Module:** Real-time clock for scheduling
* **LCD Display (20x4):** Shows data
* **Relays (5V):** Controls water pump and fan
* **12V DC Fan:** Provides ventilation
* **12V Water Pump:** Pumps water
* **Power Sources:** 12V batteries

**Simulation Diagram:**



**Assembling the Circuit:**

Humidity Sensor, Soil Moisture sensor and temperature sensor output will be given as input to Arduino via Analog pin. The RTC (Real Time Clock) Module is connected using 12c Interface,20x4 Display is connected with Arduino using 4 pin configuration and the motors to be controlled are connected to Relay and that relay will be controlled by Digital pins.

Pin numbers can be seen the above simulation diagram

**Working of the Smart Farming System**

The project is a smart farming system using Arduino Uno that monitors and controls the temperature, soil moisture, and humidity. Here's how the system works:

**Step 1. Sensors**:

* **Temperature Sensor (LM35)**: Measures the temperature and sends the data to the Arduino.
* **Soil Moisture Sensor**: Measures the moisture level in the soil and sends the data to the Arduino.
* **Humidity Sensor**: Measures the humidity level in the environment and sends the data to the Arduino.

**Step 2. Real-Time Clock (RTC)**:

* The DS1307 RTC module provides the current date and time to the system, ensuring time-based operations are accurate.

**Step 3. Arduino Uno**:

* Receives data from the sensors.
* Processes the data to determine the actions to be taken.
* Controls the relays for the fan and the water pump based on the sensor data.

**Step 4. Relays and Actuators**:

* **Relay for Fan**: Activates the fan if the temperature exceeds a predefined threshold.
* **Relay for Water Pump**: Activates the water pump if the soil moisture level falls below a predefined threshold.

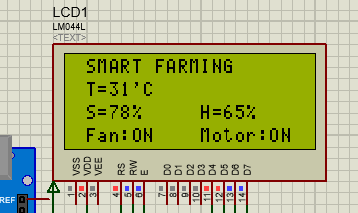
**Step 5. LCD Display**:

* Displays the current temperature, soil moisture level, humidity, and the status of the fan and pump.

**Code:**

|  |
| --- |
| #include <LiquidCrystal.h>  #include <Wire.h> // Include the Wire library for I2C communication  #include "RTClib.h" // Include the RTClib for DS137   * #include <LiquidCrystal.h>: This includes the library for interfacing with the LCD display. * #include <Wire.h>: This includes the library for I2C communication, used for interfacing with the RTC module. * #include "RTClib.h": This includes the library for the RTC (Real Time Clock) DS3231.   RTC\_DS3231 rtc; // Create an instance of the RTC\_DS3231 class   * RTC\_DS3231 rtc: Creates an instance of the RTC\_DS3231 class to interact with the DS3231 RTC module. |
| **LCD Setup and Pin Definitions**  LiquidCrystal lcd(13, 12, 11, 10, 9, 8);   * LiquidCrystal lcd(13, 12, 11, 10, 9, 8): Initializes the LCD with the specified pin connections.   cpp  Copy code  const int Fan\_Pin = 6;  const int Pump\_Pin = 7;  const unsigned long motorInterval = 5UL \* 1000UL; // Motor runs every 24 seconds  const unsigned long motorDuration = 2UL \* 1000UL; // Motor runs for 2 seconds  unsigned long lastMotorRunTime = 0;  bool motorRunning = false;   * const int Fan\_Pin = 6: Defines the pin connected to the fan. * const int Pump\_Pin = 7: Defines the pin connected to the pump. * const unsigned long motorInterval = 5UL \* 1000UL: Defines the interval between motor runs, set to 5 seconds (note the comment should be updated to 5 seconds). * const unsigned long motorDuration = 2UL \* 1000UL: Defines the duration the motor runs, set to 2 seconds. * unsigned long lastMotorRunTime = 0: Stores the last time the motor was run. * bool motorRunning = false: Tracks whether the motor is currently running. |
| **Setup Function**  void setup() {  lcd.begin(20, 4); // set up the LCD's number of columns and rows:  lcd.setCursor(0, 0); // set the cursor position:  lcd.print(" SMART FARMING ");  pinMode(Fan\_Pin, OUTPUT);  pinMode(Pump\_Pin, OUTPUT);  pinMode(A1, INPUT);  Wire.begin(); // Start the I2C communication  rtc.begin(); // Initialize the DS137  // Uncomment the following line if the DS137 lost power and you want to set the time manually.  // rtc.adjust(DateTime(F(\_\_DATE\_\_), F(\_\_TIME\_\_)));  }   * lcd.begin(20, 4): Initializes the LCD with 20 columns and 4 rows. * lcd.setCursor(0, 0): Sets the cursor to the first column of the first row. * lcd.print(" SMART FARMING "): Prints "SMART FARMING" on the LCD. * pinMode(Fan\_Pin, OUTPUT): Sets the fan pin as an output. * pinMode(Pump\_Pin, OUTPUT): Sets the pump pin as an output. * pinMode(A1, INPUT): Sets pin A1 as an input (not used in this code). * Wire.begin(): Initializes the I2C communication. * rtc.begin(): Initializes the RTC module. |
| **Loop Function**  void loop() {  DateTime now = rtc.now();  static unsigned long lastUpdateTime = 0;  unsigned long currentTime = millis();   * DateTime now = rtc.now(): Gets the current date and time from the RTC. * static unsigned long lastUpdateTime = 0: A static variable to store the last update time (not used in this code). * unsigned long currentTime = millis(): Gets the current time in milliseconds since the program started. |
| **Temperature Sensing**  int S1 = analogRead(A0); // Read Temperature  float mV = (S1 / 1023.0) \* 5000; // Storing value in millivolts  int Temp = mV / 10; // Converting millivolts to degree Celsius  lcd.setCursor(0, 1);  lcd.print(" T=");  lcd.print(Temp);  lcd.print("'C ");   * int S1 = analogRead(A0): Reads the analog value from the temperature sensor. * float mV = (S1 / 1023.0) \* 5000: Converts the analog value to millivolts. * int Temp = mV / 10: Converts millivolts to degrees Celsius. * lcd.setCursor(0, 1): Sets the cursor to the first column of the second row. * lcd.print(" T="), lcd.print(Temp), lcd.print("'C "): Displays the temperature on the LCD. |
| **Soil Moisture Sensing**  int S3 = analogRead(A2); // Read Soil Moisture  int SM = S3 / 10;  lcd.setCursor(0, 2);  lcd.print(" S=");  lcd.print(SM);  lcd.print("% ");   * int S3 = analogRead(A2): Reads the analog value from the soil moisture sensor. * int SM = S3 / 10: Converts the analog value to a percentage. * lcd.setCursor(0, 2): Sets the cursor to the first column of the third row. * lcd.print(" S="), lcd.print(SM), lcd.print("% "): Displays the soil moisture on the LCD. |
| **Air Humidity Sensing**  int S4 = analogRead(A3); // Read Air Humidity  int H = S4 / 10;  lcd.setCursor(10, 2);  lcd.print(" H=");  lcd.print(H);  lcd.print("% ");   * int S4 = analogRead(A3): Reads the analog value from the air humidity sensor. * int H = S4 / 10: Converts the analog value to a percentage. * lcd.setCursor(10, 2): Sets the cursor to the 11th column of the third row. * lcd.print(" H="), lcd.print(H), lcd.print("% "): Displays the air humidity on the LCD. |
| **Fan Control**  if (Temp > 30) {  digitalWrite(Fan\_Pin, HIGH);  lcd.setCursor(0, 3);  lcd.print(" Fan:ON ");  } else {  digitalWrite(Fan\_Pin, LOW);  lcd.setCursor(0, 3);  lcd.print(" Fan:OFF ");  }   * if (Temp > 30): Checks if the temperature is greater than 30°C. * digitalWrite(Fan\_Pin, HIGH): Turns on the fan if the temperature is above 30°C. * lcd.setCursor(0, 3): Sets the cursor to the first column of the fourth row. * lcd.print(" Fan:ON "): Displays "Fan   " on the LCD.   * digitalWrite(Fan\_Pin, LOW): Turns off the fan if the temperature is 30°C or below. * lcd.print(" Fan:OFF "): Displays "Fan   " on the LCD. |
| **Motor Control**  if (motorRunning) {  if (currentTime - lastMotorRunTime >= motorDuration) {  motorRunning = false;  digitalWrite(Pump\_Pin, LOW);  lcd.setCursor(10, 3);  lcd.print(" Motor:OFF ");  }  } else {  if (currentTime - lastMotorRunTime >= motorInterval) {  if (SM <= 80) { // Only turn on the motor if soil moisture is less than or equal to 80%  motorRunning = true;  digitalWrite(Pump\_Pin, HIGH);  lastMotorRunTime = currentTime;  lcd.setCursor(10, 3);  lcd.print(" Motor:ON ");  } else {  lcd.setCursor(10, 3);  lcd.print(" Motor:OFF ");  }  }  }  }   * if (motorRunning): Checks if the motor is currently running. * if (currentTime - lastMotorRunTime >= motorDuration): Checks if the motor has been running for the specified duration (2 seconds).   + motorRunning = false: Sets the motor state to not running.   + digitalWrite(Pump\_Pin, LOW): Turns off the motor.   + lcd.setCursor(10, 3): Sets the cursor to the 11th column of the fourth row.   + lcd.print(" Motor:OFF "): Displays "Motor   " on the LCD.   * else if (currentTime - lastMotorRunTime >= motorInterval): Checks if the interval between motor runs has passed (5 seconds).   + if (SM <= 80): Checks if the soil moisture is 80% or less.     - motorRunning = true: Sets the |

**Results:**

****