**Includes and Object Instantiations**

cpp

Copy 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.

cpp

Copy code

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**

cpp

Copy code

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**

cpp

Copy code

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**

cpp

Copy code

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**

cpp

Copy code

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**

cpp

Copy code

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**

cpp

Copy code

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**

cpp

Copy code

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**

cpp

Copy code

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