#include <WiFi.h>

#include "time.h"

/////////////////////////////////////lcd

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

// Set the LCD address to 0x27 for a 16 chars and 2 line display

LiquidCrystal\_I2C lcd(0x27, 16, 2);

/////////////////////////////////temp humidity libraries

#include <Adafruit\_Sensor.h>

#include <DHT.h>

#include <DHT\_U.h>

//////////////////////////////////temp and humidity

// Define the GPIO pin where the DHT22 data pin is connected

#define DHTPIN 4

// Define the type of DHT sensor

#define DHTTYPE DHT22

// Create a DHT object

DHT dht(DHTPIN, DHTTYPE);

////////////////////////////////////////////soil

// Define the GPIO pin where the soil moisture sensor is connected

#define SOIL\_MOISTURE\_PIN 33

// Define the minimum and maximum analog values for dry and wet soil

#define DRY\_SOIL 3000

#define WET\_SOIL 4095

////////////////////////////////////////////////////////relay

// Define GPIO pins for the relay module

#define RELAY\_FAN 14

#define RELAY\_PELTIER 15

//#define RELAY\_FANtwo 15

#define RELAY\_humidifire 18

#define RELAY\_waterPump 13

#define RELAY\_light 19

int fanAc = 0; //if fan and Ac on value is 1

int fanhum = 0; //if fan2 and humidifire on value is 1

int waterPump = 0;

int fanAcCount = 0;

int fanhumCount = 0;

int waterPumpCount = 0;

//////////////////////////////////////////////////

int lightOn = 0;

int tempCount = 0;

int humiCount = 0;

int soilCount = 0;

int needTempHumi = 0;

int needSoil = 0;

float humi = 0;

float interruptHumi;

float temp = 0;

float interruptTemp;

float soil = 0;

float interruptSoil;

float interruptSoilpercent;

float avghumi;

float avgtemp;

float avgsoil;

// Replace with your network credentials

const char\* ssid = "Dhanuja123";

const char\* password = "uncy1765";

// NTP server to request time from

const char\* ntpServer = "pool.ntp.org";

const long gmtOffset\_sec = 5 \* 3600; // GMT +5 hours

const int daylightOffset\_sec = 30 \* 60; // +30 minutes for Sri Labuznka

unsigned long msgSentTime = 0;

unsigned long waterMotorTime = 0;

//////////////////////////////////ultrasonic

#define trigPin 26

#define echoPin 25

long distance, duration;

int ultrasonicDetects = 0;

//////////////////////////////////////////////sms

#include <SoftwareSerial.h>

#define RX\_PIN 3 // RX of ESP32 to TX of SIM800C

#define TX\_PIN 1 // TX of ESP32 to RX of SIM800C

SoftwareSerial sim800(RX\_PIN, TX\_PIN);

const char\* phoneNumber = "+94714536866"; // Predefined phone number

////////////////////////////////////////////////

const int buzzerPin = 27; // Pin where the buzzer is connected

void handleWaterPumpRelay() {

struct tm timeinfo;

if (!getLocalTime(&timeinfo)) {

Serial.println("Failed to obtain time");

return;

}

int hour = timeinfo.tm\_hour;

int minute = timeinfo.tm\_min;

// Check if the current time is 2 AM or 2 PM

if ((hour == 2 || hour == 14) && minute == 0) {

digitalWrite(RELAY\_waterPump, LOW); // Turn on the water pump

delay(1500); // Wait for 1500 milliseconds

digitalWrite(RELAY\_waterPump, HIGH); // Turn off the water pump

}

}

void setup() {

// put your setup code here, to run once:

Serial.begin(115200);

delay(1000);

dht.begin();

// Connect to Wi-Fi

Serial.printf("Connecting to %s ", ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println(" connected");

// Initialize and obtain time

configTime(gmtOffset\_sec, daylightOffset\_sec, ntpServer);

// Wait for time to be set

struct tm timeinfo;

if (!getLocalTime(&timeinfo)) {

Serial.println("Failed to obtain time");

return;

}

// Initialize the DHT sensor

//////////////////////////////////////////////

// Initialize relay pins as outputs

pinMode(RELAY\_FAN, OUTPUT);

pinMode(RELAY\_PELTIER, OUTPUT);

// pinMode(RELAY\_FANtwo, OUTPUT);

pinMode(RELAY\_humidifire, OUTPUT);

pinMode(RELAY\_waterPump, OUTPUT);

pinMode(RELAY\_light, OUTPUT);

// Turn off all relays initially

digitalWrite(RELAY\_FAN, HIGH);

digitalWrite(RELAY\_PELTIER, HIGH);

//digitalWrite(RELAY\_FANtwo, LOW);

digitalWrite(RELAY\_humidifire, HIGH);

digitalWrite(RELAY\_waterPump, HIGH);

digitalWrite(RELAY\_light, HIGH);

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

//////////////////////////////////////////

// Initialize the LCD

lcd.init();

lcd.backlight(); // Turn on the backlight

///////////////////////////////////////////sms

// Initialize SoftwareSerial with chosen pins and baud rate

sim800.begin(9600);

delay(1000); // Wait for 1 second to ensure SIM800C initializes

////////////////////////////////////buzzer

// Initialize the buzzer pin as an output

pinMode(buzzerPin, OUTPUT);

displayAndGetTempHumi();

}

void loop() {

struct tm timeinfo;

if(!getLocalTime(&timeinfo)){

Serial.println("Failed to obtain time");

delay(1000);

return;

}

// put your main code here, to run repeatedly:

displayAndGetTempHumi();

delay(1000);

getSoil();

delay(1000);

if(tempCount>7){

checkAvgTempHumi();

}

if(soilCount>7){

checkAvgSoil();

}

handleWaterPumpRelay();

lightOnOff();

//relayOff();

Serial.println(tempCount);

Serial.println(soilCount);

for (int count = 0; count < 5; count++) {

Serial.println("checking ultrasonic");

delay(500);

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10); // transmit waves for 10 micro seconds

digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH); //recieve reflected waves

distance = duration / 58.2; //distance in centimetres," velocity=distance/time " used here with unit conversion, and also dividing by 2 also used since reflection happens (wave hits and reflected back in same path)

Serial.print("distance = ");

Serial.println(distance);

if (distance <= 11 && distance >= 3) {

ultrasonicDetects++;

}

}

if (ultrasonicDetects > 3) {

if(millis()>waterMotorTime+1000\*3600\*12){

digitalWrite(RELAY\_waterPump, LOW);

Serial.println("water pump onnnnnnnnnnn for 3 sec");

delay(2000);

digitalWrite(RELAY\_waterPump, HIGH);

waterMotorTime=millis();

}

//waterPump = 1;

//delay(20);

}else{

tone(buzzerPin, 1000); // Send 1kHz sound signal

if(millis()>msgSentTime+10000){

sendMessage("please fill the water tank..");

// Read and print any final responses

while (sim800.available()) {

Serial.write(sim800.read()); // Print final response to Serial Monitor

}

msgSentTime=millis();

}

// Wait for 3 second

delay(1000);

// Stop the tone

noTone(buzzerPin);

}

ultrasonicDetects = 0;

delay(3000);

}

void displayAndGetTempHumi(){

// Read temperature as Celsius

float tempC = dht.readTemperature();

delay(300);

// Read humidity

float humidity = dht.readHumidity();

// Check if any reads failed and exit early (to try again).

if (isnan(tempC)) {

Serial.println("Failed to read temp!");

temp += 31;

lcd.setCursor(0, 0);

lcd.print("Temp in C: ");

lcd.setCursor(11, 0);

lcd.print("31.00");

} else {

Serial.print("temp in C:");

Serial.println(tempC);

temp += tempC;

lcd.setCursor(0, 0);

lcd.print("Temp in C: ");

lcd.setCursor(11, 0);

lcd.print(tempC);

}

if (isnan(humidity)) {

Serial.println("Failed to read humidity!");

humi += 85;

lcd.setCursor(0, 1);

lcd.print("Humd in %: ");

lcd.setCursor(11, 1);

lcd.print("85.00");

} else {

Serial.print("Humd in %: ");

Serial.println(humidity);

humi += humidity;

lcd.setCursor(0, 1);

lcd.print("Humd in %: ");

lcd.setCursor(11, 1);

lcd.print(humidity);

if(humidity>90){

digitalWrite(RELAY\_humidifire,HIGH);

}

}

tempCount++;

humiCount++;

}

void getSoil(){

// Read the analog value from the soil moisture sensor

int soilMoistureValue = analogRead(SOIL\_MOISTURE\_PIN);

Serial.print("soilMoistureValue =");

Serial.println(soilMoistureValue);

// Convert the analog value to a percentage

float soilMoisturePercent = map(soilMoistureValue, DRY\_SOIL, WET\_SOIL, 0, 100);

Serial.print("soilMoisturePercent =");

Serial.println(soilMoisturePercent);

soil+=soilMoisturePercent;

soilCount++;

}

void checkAvgTempHumi() {

//if counts are not zero

avgtemp=temp/tempCount;

avghumi=humi/humiCount;

if(avgtemp>30){

Serial.println("Ac fan onn");

fanAcCount++;

digitalWrite(RELAY\_FAN, LOW);

digitalWrite(RELAY\_PELTIER, LOW);

fanAc=1;

}

else{

digitalWrite(RELAY\_FAN, HIGH);

digitalWrite(RELAY\_PELTIER, HIGH);

Serial.println("Ac fan off");

fanAc=0;

}

if(avghumi<90){

fanhumCount++;

digitalWrite(RELAY\_humidifire, LOW);

delay(5000);

digitalWrite(RELAY\_humidifire, HIGH);

Serial.println("humidifire fan on");

fanhum=1;

}

else{

digitalWrite(RELAY\_humidifire, HIGH);

Serial.println("humidifire fan off");

fanhum=1;

}

temp=0;

humi=0;

tempCount=0;

humiCount=0;

}

void checkAvgSoil(){

// avgsoil=soil/soilCount;

// if(avgsoil<20){

// waterPumpCount++;

// digitalWrite(RELAY\_waterPump, LOW);

// Serial.println("water pump onnnnnnnnnnnnnnnnnnnnnnnn for 1 sec");

// delay(1000);

// digitalWrite(RELAY\_waterPump, HIGH);

// waterPump=1;

// }

// soil=0;

// soilCount=0;

}

void relayOff(){

if(needTempHumi==1){

if(fanAcCount>4){

digitalWrite(RELAY\_FAN, HIGH);

digitalWrite(RELAY\_PELTIER, HIGH);

fanAcCount=0;

fanAc=0;

}

if(fanhumCount>4){

digitalWrite(RELAY\_humidifire, HIGH);

fanhumCount=0;

fanhum=0;

}

}

if(needSoil==1){

if(waterPumpCount>4){

digitalWrite(RELAY\_waterPump, HIGH);

waterPumpCount=0;

waterPump=0;

}

}

}

void lightOnOff(){

struct tm timeinfo;

if(!getLocalTime(&timeinfo)){

Serial.println("Failed to obtain time");

return;

}

int hour = timeinfo.tm\_hour;

if (hour >= 6 && hour < 15 || hour<12) { // Change this condition based on your requirement

digitalWrite(RELAY\_light, HIGH); // Turn off the light during the day

Serial.println("light off");

} else {

digitalWrite(RELAY\_light, LOW); // Turn on the light during the night

Serial.println("light on");

}

// if(hour>0){

// digitalWrite(RELAY\_light, LOW); // Turn on the light during the night

// Serial.println("light on");

// delay(60000);

// digitalWrite(RELAY\_light, HIGH); // Turn off the light during the day

// Serial.println("light off");

// delay(10000);

// }

}

void sendMessage(const char\* message) {

// sim800.print("AT+CMGF=1\r"); // Set SMS mode to text

// delay(1000);

// sim800.print("AT+CMGS=\"");

// sim800.print(phoneNumber); // Add your phone number here

// sim800.print("\"\r");

// delay(1000);

// sim800.print(message);

// delay(1000);

// sim800.print((char)26); // ASCII code of CTRL+Z

// delay(1000);

// Serial.println("Message sent");

// Set SMS mode to text

sim800.print("AT+CMGF=1\r");

delay(2000);

// Check response from the module

while (sim800.available()) {

String response = sim800.readString();

Serial.println("Response: " + response);

}

// Set recipient phone number

sim800.print("AT+CMGS=\"");

sim800.print(phoneNumber); // Add your phone number here

sim800.print("\"\r");

delay(2000);

// Check response from the module

while (sim800.available()) {

String response = sim800.readString();

Serial.println("Response: " + response);

}

// Send the message

sim800.print(message);

delay(2000);

// Check response from the module

while (sim800.available()) {

String response = sim800.readString();

Serial.println("Response: " + response);

}

// Send the Ctrl+Z character to indicate the end of the message

sim800.print((char)26); // ASCII code of CTRL+Z

delay(5000); // Wait for a response

// Check final response from the module

while (sim800.available()) {

String response = sim800.readString();

Serial.println("Final Response: " + response);

}

Serial.println("Message sent");

}