/\*

 \* ESP32-C6 Ultimate Clock System (English Version)

 \* Hardware: ESP32-C6 WROOM1 + DS3231 + 0.96" OLED(128x64)

 \* Date: 2025-10-21

 \*/

#include <Wire.h>

#include <RTClib.h>

#include <Adafruit\_GFX.h>

#include <Adafruit\_SSD1306.h>

#include <WiFi.h>

#include <time.h>

// ==================== Configuration ====================

// WiFi Settings

const char\* ssid = "XIUYUAN";           // Change to your WiFi name

const char\* password = "92511111";   // Change to your WiFi password

// NTP Server Configuration

const char\* ntpServer1 = "ntp.aliyun.com";

const char\* ntpServer2 = "ntp1.aliyun.com";

const char\* ntpServer3 = "cn.ntp.org.cn";

const long gmtOffset\_sec = 8 \* 3600;        // UTC+8 Beijing Time

const int daylightOffset\_sec = 0;

// I2C Pins

#define I2C\_SDA 6

#define I2C\_SCL 7

// OLED Configuration

#define SCREEN\_WIDTH 128

#define SCREEN\_HEIGHT 64

#define OLED\_RESET -1

#define OLED\_ADDR 0x3C

// Page Switch Interval (seconds)

#define PAGE\_SWITCH\_INTERVAL 5

// ==================== Global Objects ====================

RTC\_DS3231 rtc;

Adafruit\_SSD1306 display(SCREEN\_WIDTH, SCREEN\_HEIGHT, &Wire, OLED\_RESET);

// Weekday Names

const char\* weekDays[] = {"Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"};

// Page Status

int currentPage = 0;

unsigned long lastPageSwitch = 0;

// ==================== Lunar Calendar Data ====================

// Lunar data table (2000-2050)

const unsigned int lunarInfo[] = {

    0x04bd8, 0x04ae0, 0x0a570, 0x054d5, 0x0d260, 0x0d950, 0x16554, 0x056a0, 0x09ad0, 0x055d2,

    0x04ae0, 0x0a5b6, 0x0a4d0, 0x0d250, 0x1d255, 0x0b540, 0x0d6a0, 0x0ada2, 0x095b0, 0x14977,

    0x04970, 0x0a4b0, 0x0b4b5, 0x06a50, 0x06d40, 0x1ab54, 0x02b60, 0x09570, 0x052f2, 0x04970,

    0x06566, 0x0d4a0, 0x0ea50, 0x06e95, 0x05ad0, 0x02b60, 0x186e3, 0x092e0, 0x1c8d7, 0x0c950,

    0x0d4a0, 0x1d8a6, 0x0b550, 0x056a0, 0x1a5b4, 0x025d0, 0x092d0, 0x0d2b2, 0x0a950, 0x0b557

};

const char\* lunarMonths[] = {"Jan", "Feb", "Mar", "Apr", "May", "Jun",

                             "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"};

// Lunar Date Structure

struct LunarDate {

    int year;

    int month;

    int day;

    bool isLeap;

};

// ==================== Lunar Calculation Functions ====================

int getLunarYearDays(int year) {

    int i, sum = 348;

    for(i = 0x8000; i > 0x8; i >>= 1) {

        sum += (lunarInfo[year - 2000] & i) ? 1 : 0;

    }

    return sum + getLeapDays(year);

}

int getLeapDays(int year) {

    if(getLeapMonth(year)) {

        return (lunarInfo[year - 2000] & 0x10000) ? 30 : 29;

    }

    return 0;

}

int getLeapMonth(int year) {

    return lunarInfo[year - 2000] & 0xf;

}

int getLunarMonthDays(int year, int month) {

    return (lunarInfo[year - 2000] & (0x10000 >> month)) ? 30 : 29;

}

// Solar to Lunar Conversion

LunarDate solarToLunar(int year, int month, int day) {

    LunarDate lunar;

    int i, leap = 0, temp = 0;

    DateTime base(2000, 1, 31, 0, 0, 0);

    DateTime target(year, month, day, 0, 0, 0);

    int offset = (target.unixtime() - base.unixtime()) / 86400;

    for(i = 2000; i < 2051 && offset > 0; i++) {

        temp = getLunarYearDays(i);

        offset -= temp;

    }

    if(offset < 0) {

        offset += temp;

        i--;

    }

    lunar.year = i;

    leap = getLeapMonth(i);

    lunar.isLeap = false;

    for(i = 1; i < 13 && offset > 0; i++) {

        if(leap > 0 && i == (leap + 1) && lunar.isLeap == false) {

            --i;

            lunar.isLeap = true;

            temp = getLeapDays(lunar.year);

        } else {

            temp = getLunarMonthDays(lunar.year, i);

        }

        if(lunar.isLeap == true && i == (leap + 1)) {

            lunar.isLeap = false;

        }

        offset -= temp;

    }

    if(offset == 0 && leap > 0 && i == leap + 1) {

        if(lunar.isLeap) {

            lunar.isLeap = false;

        } else {

            lunar.isLeap = true;

            --i;

        }

    }

    if(offset < 0) {

        offset += temp;

        --i;

    }

    lunar.month = i;

    lunar.day = offset + 1;

    return lunar;

}

// ==================== WiFi and NTP Functions ====================

bool connectWiFi() {

    Serial.print("Connecting WiFi");

    WiFi.begin(ssid, password);

    int attempts = 0;

    while(WiFi.status() != WL\_CONNECTED && attempts < 20) {

        delay(500);

        Serial.print(".");

        attempts++;

    }

    if(WiFi.status() == WL\_CONNECTED) {

        Serial.println("\nWiFi Connected!");

        Serial.print("IP: ");

        Serial.println(WiFi.localIP());

        return true;

    } else {

        Serial.println("\nWiFi Failed!");

        return false;

    }

}

bool syncTimeWithNTP() {

    Serial.println("Syncing NTP...");

    configTime(gmtOffset\_sec, daylightOffset\_sec, ntpServer1, ntpServer2, ntpServer3);

    struct tm timeinfo;

    int attempts = 0;

    while(!getLocalTime(&timeinfo) && attempts < 10) {

        Serial.print(".");

        delay(1000);

        attempts++;

    }

    if(attempts >= 10) {

        Serial.println("\nNTP Sync Failed!");

        return false;

    }

    rtc.adjust(DateTime(timeinfo.tm\_year + 1900, timeinfo.tm\_mon + 1, timeinfo.tm\_mday,

                        timeinfo.tm\_hour, timeinfo.tm\_min, timeinfo.tm\_sec));

    Serial.println("\nNTP Sync Success!");

    Serial.printf("Time: %04d-%02d-%02d %02d:%02d:%02d\n",

                  timeinfo.tm\_year + 1900, timeinfo.tm\_mon + 1, timeinfo.tm\_mday,

                  timeinfo.tm\_hour, timeinfo.tm\_min, timeinfo.tm\_sec);

    WiFi.disconnect(true);

    WiFi.mode(WIFI\_OFF);

    return true;

}

// ==================== Animation Functions ====================

void slideLeftAnimation() {

    for(int x = 0; x < SCREEN\_WIDTH; x += 8) {

        display.clearDisplay();

        display.setCursor(-x, 20);

        display.setTextSize(2);

        display.print("Loading...");

        display.display();

        delay(10);

    }

}

void pageTransition() {

    for(int x = 0; x <= SCREEN\_WIDTH; x += 16) {

        display.clearDisplay();

        display.fillRect(0, 0, SCREEN\_WIDTH - x, SCREEN\_HEIGHT, SSD1306\_BLACK);

        display.display();

        delay(10);

    }

}

// ==================== Display Page Functions ====================

// Page 0: Large Digital Clock

void displayPage0(DateTime now) {

    display.clearDisplay();

    // Date

    display.setTextSize(1);

    display.setCursor(10, 0);

    display.printf("%04d-%02d-%02d", now.year(), now.month(), now.day());

    // Weekday

    display.setCursor(90, 0);

    display.print(weekDays[now.dayOfTheWeek()]);

    // Large Time

    display.setTextSize(3);

    display.setCursor(10, 20);

    display.printf("%02d:%02d", now.hour(), now.minute());

    // Seconds

    display.setTextSize(2);

    display.setCursor(100, 28);

    display.printf("%02d", now.second());

    // Temperature

    float temp = rtc.getTemperature();

    display.setTextSize(1);

    display.setCursor(25, 55);

    display.printf("Temp: %.1fC", temp);

    display.display();

}

// Page 1: Lunar Calendar Display

void displayPage1(DateTime now) {

    display.clearDisplay();

    // Solar Date

    display.setTextSize(1);

    display.setCursor(0, 0);

    display.printf("Solar:%04d-%02d-%02d", now.year(), now.month(), now.day());

    // Time

    display.setTextSize(2);

    display.setCursor(20, 15);

    display.printf("%02d:%02d:%02d", now.hour(), now.minute(), now.second());

    // Lunar Date

    LunarDate lunar = solarToLunar(now.year(), now.month(), now.day());

    display.setTextSize(1);

    display.setCursor(0, 38);

    display.printf("Lunar: %d Year", lunar.year);

    display.setCursor(0, 50);

    if(lunar.isLeap) display.print("Leap ");

    display.printf("M%d D%d", lunar.month, lunar.day);

    display.display();

}

// Page 2: Detailed Information

void displayPage2(DateTime now) {

    display.clearDisplay();

    // Title

    display.setTextSize(1);

    display.setCursor(30, 0);

    display.print("System Info");

    // Time

    display.setCursor(0, 15);

    display.printf("Time: %02d:%02d:%02d", now.hour(), now.minute(), now.second());

    // Date

    display.setCursor(0, 27);

    display.printf("Date: %02d/%02d/%04d", now.month(), now.day(), now.year());

    // Weekday

    display.setCursor(0, 39);

    display.print("Week: ");

    display.print(weekDays[now.dayOfTheWeek()]);

    // Temperature

    float temp = rtc.getTemperature();

    display.setCursor(0, 51);

    display.printf("Temp: %.2f C", temp);

    display.display();

}

// Page 3: Analog Clock Face

void displayPage3(DateTime now) {

    display.clearDisplay();

    int centerX = 64;

    int centerY = 36;

    int radius = 26;

    // Draw clock circle

    display.drawCircle(centerX, centerY, radius, SSD1306\_WHITE);

    display.drawCircle(centerX, centerY, radius - 1, SSD1306\_WHITE);

    // Draw hour markers

    for(int i = 0; i < 12; i++) {

        float angle = (i \* 30 - 90) \* PI / 180;

        int x1 = centerX + (radius - 4) \* cos(angle);

        int y1 = centerY + (radius - 4) \* sin(angle);

        int x2 = centerX + (radius - 1) \* cos(angle);

        int y2 = centerY + (radius - 1) \* sin(angle);

        display.drawLine(x1, y1, x2, y2, SSD1306\_WHITE);

    }

    // Hour hand

    float hourAngle = ((now.hour() % 12) \* 30 + now.minute() \* 0.5 - 90) \* PI / 180;

    int hx = centerX + (radius - 12) \* cos(hourAngle);

    int hy = centerY + (radius - 12) \* sin(hourAngle);

    display.drawLine(centerX, centerY, hx, hy, SSD1306\_WHITE);

    display.drawLine(centerX + 1, centerY, hx + 1, hy, SSD1306\_WHITE);

    // Minute hand

    float minAngle = (now.minute() \* 6 - 90) \* PI / 180;

    int mx = centerX + (radius - 6) \* cos(minAngle);

    int my = centerY + (radius - 6) \* sin(minAngle);

    display.drawLine(centerX, centerY, mx, my, SSD1306\_WHITE);

    // Second hand

    float secAngle = (now.second() \* 6 - 90) \* PI / 180;

    int sx = centerX + (radius - 2) \* cos(secAngle);

    int sy = centerY + (radius - 2) \* sin(secAngle);

    display.drawLine(centerX, centerY, sx, sy, SSD1306\_WHITE);

    // Center dot

    display.fillCircle(centerX, centerY, 2, SSD1306\_WHITE);

    // Digital time at top

    display.setTextSize(1);

    display.setCursor(35, 0);

    display.printf("%02d:%02d:%02d", now.hour(), now.minute(), now.second());

    display.display();

}

// ==================== Main Program ====================

void setup() {

    Serial.begin(115200);

    delay(1000);

    Serial.println("\n=== ESP32-C6 Ultimate Clock System ===");

    // Initialize I2C

    Wire.begin(I2C\_SDA, I2C\_SCL);

    // Initialize OLED

    if(!display.begin(SSD1306\_SWITCHCAPVCC, OLED\_ADDR)) {

        Serial.println("OLED Init Failed!");

        while(1);

    }

    Serial.println("OLED Init OK");

    display.clearDisplay();

    display.setTextColor(SSD1306\_WHITE);

    display.setTextSize(1);

    display.setCursor(15, 10);

    display.println("Ultimate Clock");

    display.setCursor(20, 25);

    display.println("Starting...");

    display.display();

    delay(1000);

    // Initialize DS3231

    if(!rtc.begin()) {

        Serial.println("DS3231 Init Failed!");

        display.clearDisplay();

        display.setCursor(10, 20);

        display.println("RTC Error!");

        display.display();

        while(1);

    }

    Serial.println("DS3231 Init OK");

    // NTP Time Sync

    display.clearDisplay();

    display.setCursor(5, 20);

    display.println("Connecting WiFi...");

    display.display();

    if(connectWiFi()) {

        display.clearDisplay();

        display.setCursor(15, 20);

        display.println("Syncing NTP...");

        display.display();

        if(syncTimeWithNTP()) {

            display.clearDisplay();

            display.setCursor(15, 20);

            display.println("Time Synced!");

            display.display();

            delay(1500);

        } else {

            display.clearDisplay();

            display.setCursor(10, 15);

            display.println("NTP Sync Failed");

            display.setCursor(10, 30);

            display.println("Using RTC Time");

            display.display();

            delay(2000);

        }

    } else {

        display.clearDisplay();

        display.setCursor(15, 15);

        display.println("WiFi Failed");

        display.setCursor(10, 30);

        display.println("Using RTC Time");

        display.display();

        delay(2000);

    }

    // Startup Animation

    slideLeftAnimation();

    Serial.println("System Ready!");

    lastPageSwitch = millis();

}

void loop() {

    DateTime now = rtc.now();

    // Auto page switching

    if(millis() - lastPageSwitch > PAGE\_SWITCH\_INTERVAL \* 1000) {

        pageTransition();

        currentPage = (currentPage + 1) % 4;

        lastPageSwitch = millis();

    }

    // Display current page

    switch(currentPage) {

        case 0:

            displayPage0(now);

            break;

        case 1:

            displayPage1(now);

            break;

        case 2:

            displayPage2(now);

            break;

        case 3:

            displayPage3(now);

            break;

    }

    // Serial output

    if(now.second() % 10 == 0) {

        LunarDate lunar = solarToLunar(now.year(), now.month(), now.day());

        Serial.printf("Date: %04d-%02d-%02d %s %02d:%02d:%02d | Lunar: M%d D%d | Temp: %.1fC\n",

                      now.year(), now.month(), now.day(), weekDays[now.dayOfTheWeek()],

                      now.hour(), now.minute(), now.second(),

                      lunar.month, lunar.day, rtc.getTemperature());

        delay(1000);

    }

    delay(100);

}