#include <Wire.h>

#include <RTClib.h>

#include <LiquidCrystal\_I2C.h>

#include <EEPROM.h>

#define RED\_LED 8

#define GREEN\_LED 9

#define BUZZER 10

#define BTN\_SET\_START\_HOUR 6

#define BTN\_SET\_START\_MINUTE 5

#define BTN\_SET\_END\_HOUR 7

#define BTN\_SET\_END\_MINUTE 4

RTC\_DS3231 rtc;

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

int startHour, startMinute, endHour, endMinute;

unsigned long lastButtonPressTime = 0;  // To track debounce time

void setup() {

    Serial.begin(9600);

    pinMode(RED\_LED, OUTPUT);

    pinMode(GREEN\_LED, OUTPUT);

    pinMode(BUZZER, OUTPUT);

    pinMode(BTN\_SET\_START\_HOUR, INPUT\_PULLUP);

    pinMode(BTN\_SET\_START\_MINUTE, INPUT\_PULLUP);

    pinMode(BTN\_SET\_END\_HOUR, INPUT\_PULLUP);

    pinMode(BTN\_SET\_END\_MINUTE, INPUT\_PULLUP);

    lcd.init();

    lcd.backlight();

    if (!rtc.begin()) {

        Serial.println("RTC not found!");

        while (1);

    }

    // Set RTC to computer time (Only needed once)

    rtc.adjust(DateTime(F(\_\_DATE\_\_), F(\_\_TIME\_\_)));

    // Load saved Makruh times from EEPROM

    startHour = EEPROM.read(0);

    startMinute = EEPROM.read(1);

    endHour = EEPROM.read(2);

    endMinute = EEPROM.read(3);

    Serial.println("Makruh Time Loaded:");

    Serial.print("Start: "); Serial.print(startHour); Serial.print(":"); Serial.println(startMinute);

    Serial.print("End: "); Serial.print(endHour); Serial.print(":"); Serial.println(endMinute);

    Serial.println("\n========== INSTRUCTIONS ==========");

    Serial.println("To set Makruh time, enter:");

    Serial.println("startHour startMinute endHour endMinute");

    Serial.println("Example: 5 30 6 45 (Sets Makruh time from 5:30 AM to 6:45 AM)");

    Serial.println("Press '#' to reset Makruh time to 0:0");

    Serial.println("==================================\n");

}

void loop() {

    DateTime now = rtc.now();

    int hour = now.hour();

    int minute = now.minute();

    // Display current time

    lcd.setCursor(0, 0);

    lcd.print("Time: ");

    lcd.print(hour);

    lcd.print(":");

    if (minute < 10) lcd.print("0"); // Add leading zero

    lcd.print(minute);

    lcd.print("    "); // Clear remaining characters

    // Check button presses (Non-blocking)

    checkButtonPress(BTN\_SET\_START\_HOUR, &startHour, 23, 0);

    checkButtonPress(BTN\_SET\_START\_MINUTE, &startMinute, 59, 1);

    checkButtonPress(BTN\_SET\_END\_HOUR, &endHour, 23, 2);

    checkButtonPress(BTN\_SET\_END\_MINUTE, &endMinute, 59, 3);

    // Check Makruh Time Logic

    lcd.setCursor(0, 1);

    lcd.print("                "); // Clear previous text

    if (isMakruhTime(hour, minute)) {

        digitalWrite(RED\_LED, HIGH);

        digitalWrite(GREEN\_LED, LOW);

        digitalWrite(BUZZER, HIGH);

        lcd.setCursor(0, 1);

        lcd.print("Restricted Time");

    } else {

        digitalWrite(RED\_LED, LOW);

        digitalWrite(GREEN\_LED, HIGH);

        digitalWrite(BUZZER, LOW);

        lcd.setCursor(0, 1);

        lcd.print("Prayer Allowed ");

    }

    // Check for input from Serial Monitor

    checkSerialInput();

    delay(100);  // Reduced delay to keep responsiveness (better than 1000ms)

}

// \*\*Non-blocking Button Press Handling\*\*

void checkButtonPress(int buttonPin, int\* value, int maxValue, int eepromAddress) {

    if (digitalRead(buttonPin) == LOW && millis() - lastButtonPressTime > 300) {  // Debounce time of 300ms

        lastButtonPressTime = millis();  // Update last button press time

        (\*value) = ((\*value) + 1) % (maxValue + 1);  // Increment and reset after maxValue

        updateEEPROM(eepromAddress, \*value);  // Save to EEPROM

        Serial.print("Updated Value: "); Serial.println(\*value);

    }

}

// \*\*Handle Serial Input for Setting Makruh Time\*\*

void checkSerialInput() {

    if (Serial.available()) {

        String input = Serial.readStringUntil('\n');

        input.trim(); // Remove spaces or newline characters

        if (input == "#") {

            // Reset Makruh time

            startHour = 0;

            startMinute = 0;

            endHour = 0;

            endMinute = 0;

            updateEEPROM(0, startHour);

            updateEEPROM(1, startMinute);

            updateEEPROM(2, endHour);

            updateEEPROM(3, endMinute);

            Serial.println("\nMakruh time has been RESET to 0:0");

        } else {

            // Parse and set new Makruh time

            int newStartHour, newStartMinute, newEndHour, newEndMinute;

            if (sscanf(input.c\_str(), "%d %d %d %d", &newStartHour, &newStartMinute, &newEndHour, &newEndMinute) == 4) {

                if (newStartHour >= 0 && newStartHour < 24 &&

                    newStartMinute >= 0 && newStartMinute < 60 &&

                    newEndHour >= 0 && newEndHour < 24 &&

                    newEndMinute >= 0 && newEndMinute < 60) {

                    startHour = newStartHour;

                    startMinute = newStartMinute;

                    endHour = newEndHour;

                    endMinute = newEndMinute;

                    updateEEPROM(0, startHour);

                    updateEEPROM(1, startMinute);

                    updateEEPROM(2, endHour);

                    updateEEPROM(3, endMinute);

                    Serial.println("\nMakruh time UPDATED:");

                    Serial.print("Start: "); Serial.print(startHour); Serial.print(":"); Serial.println(startMinute);

                    Serial.print("End: "); Serial.print(endHour); Serial.print(":"); Serial.println(endMinute);

                } else {

                    Serial.println("Invalid input! Please enter valid hours (0-23) and minutes (0-59).");

                }

            } else {

                Serial.println("Invalid format! Use: startHour startMinute endHour endMinute");

            }

        }

    }

}

// \*\*Function to Check if Current Time is Makruh\*\*

bool isMakruhTime(int hour, int minute) {

    int startTotal = startHour \* 60 + startMinute;

    int endTotal = endHour \* 60 + endMinute;

    int currentTotal = hour \* 60 + minute;

    return (currentTotal >= startTotal && currentTotal <= endTotal);

}

// \*\*Write to EEPROM Only if the Value is Different\*\*

void updateEEPROM(int address, int value) {

    if (EEPROM.read(address) != value) {

        EEPROM.write(address, value);

    }

}