#include <DHT.h>           // Include DHT library

// Pin definitions

#define DHTPIN D1       // DHT11 sensor pin

#define RELAY\_PIN D2    // Relay control pin

#define MQ2\_PIN A0      // MQ-2 gas sensor pin

#define FLAME\_PIN D5    // Flame sensor pin

#define IR\_PIN D6       // IR sensor pin

// DHT sensor type

#define DHTTYPE DHT11   //  Define the type of DHT sensor you are using. DHT11 or DHT22

// Function Declarations

void setupSensors();

void readSensors();

void controlSolenoid(); // Renamed for clarity

// Global Variables

DHT dht(DHTPIN, DHTTYPE);

float temperature = 0.0;

float humidity = 0.0;

int mq2Value = 0;

bool flameDetected = false;

bool irDetected = false;

// Thresholds for sensors

const int mq2Threshold = 600;       // Example threshold, adjust as needed.  Higher value, less sensitive.

const float temperatureThreshold = 35.0;  // Example threshold, adjust as needed.  In Celsius

const float humidityThreshold = 80.0;     // Example threshold, adjust as needed. Percentage

const bool  flameThreshold = false; //example threshold, the flame sensor returns either true or false

const bool irThreshold = true;

void setup() {

  Serial.begin(115200);  // Start serial communication

  setupSensors();       // Initialize the pins and DHT sensor

  Serial.println("Setup complete.");

}

void loop() {

  readSensors();    // Read sensor data

  controlSolenoid(); // Control the solenoid lock

  delay(1000);    // Small delay for stability. Consider non-blocking techniques.

}

void setupSensors() {

  // Initialize the pins as INPUT or OUTPUT

  pinMode(RELAY\_PIN, OUTPUT);

  pinMode(FLAME\_PIN, INPUT);

  pinMode(IR\_PIN, INPUT);

  // Initialize DHT sensor

  dht.begin();

  // Set initial state of relay.  Initialize to the state where current can flow.

  digitalWrite(RELAY\_PIN, HIGH); // Or LOW, depending on your relay module's active state.  If HIGH allows current flow, use HIGH.

  //  Most relay modules are active low, meaning you'd use digitalWrite(RELAY\_PIN, HIGH); to allow current

}

void readSensors() {

  // Read data from the sensors

  temperature = dht.readTemperature();

  humidity = dht.readHumidity();

  mq2Value = analogRead(MQ2\_PIN);

  flameDetected = (digitalRead(FLAME\_PIN) == LOW); // Flame sensor is often active low

  irDetected = (digitalRead(IR\_PIN) == HIGH);    //  IR sensor logic depends on the module.

  // Check for errors reading DHT sensor

  if (isnan(temperature) || isnan(humidity)) {

    Serial.println("Failed to read from DHT sensor!");

    temperature = 0.0; // Or some default value

    humidity = 0.0;    // Or some default value

  }

  Serial.print("Temperature: ");

  Serial.print(temperature);

  Serial.print(" °C, Humidity: ");

  Serial.print(humidity);

  Serial.print(" %, MQ-2 Value: ");

  Serial.print(mq2Value);

  Serial.print(", Flame: ");

  Serial.print(flameDetected ? "Detected" : "Not Detected");

  Serial.print(", IR: ");

  Serial.println(irDetected ? "Detected" : "Not Detected");

}

void controlSolenoid() {

  // Control the solenoid lock based on sensor readings

  // 1.  First, ensure the relay is ON (current flow allowed).

  digitalWrite(RELAY\_PIN, HIGH); // Or LOW, depending on your relay module. HIGH = ON.

  Serial.println("Solenoid UNLOCKED (Default).");

  // 2. Then,     check the sensor conditions to see if it should be LOCKED.

  // Lock if ANY of the following conditions are true:

  if (flameDetected ||

      mq2Value > mq2Threshold ||

      temperature > temperatureThreshold ||

      humidity > humidityThreshold ||

      irDetected) {

    digitalWrite(RELAY\_PIN, LOW); // Or HIGH, depending on your relay module.  LOW = OFF (LOCKED)

    Serial.println("DANGER! Solenoid LOCKED!");

  }

  // No 'else' is needed.  The relay is defaulted to the UNLOCKED state.

}