```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <DHT.h>
#include <SoftwareSerial.h>

// --- Pin definitions ---
#define DHTPIN             6
#define DHTTYPE            DHT11
#define SMOKE_PIN          A1
#define HEATER_RELAY       5
#define FAN_RELAY          4
#define BUZZER             13

#define HEATER_BUTTON      2   // INT0
#define FAN_BUTTON         3   // INT1
#define AUTOMATION_BUTTON  10  // polled/debounced

// --- Thresholds ---
#define TEMP_LOW           25
#define TEMP_HIGH          30
#define SMOKE_THRESHOLD    300

// --- Components ---
DHT dht(DHTPIN, DHTTYPE);
LiquidCrystal_I2C lcd(0x27,16,2);
SoftwareSerial gsm(7,8); // TX, RX for SIM800L

// --- State ---
volatile bool heaterManualState = false;
volatile bool fanManualState    = false;
bool automationMode             = false;
bool smokeAlertSent             = false;
bool lowTempAlertSent           = false;
bool highTempAlertSent          = false;

// --- Debounce timing ---
const unsigned long debounceDelay = 50;
unsigned long lastHeaterIRQ     = 0;
unsigned long lastFanIRQ        = 0;
unsigned long lastAutoBounce    = 0;
bool lastAutoBtnState           = HIGH;

// --- Flags for main loop logging ---
volatile bool heaterToggled = false;
```

```
volatile bool fanToggled    = false;

void setup() {
  Serial.begin(9600);
  gsm.begin(9600);
  dht.begin();
  lcd.init();
  lcd.backlight();

  pinMode(HEATER_RELAY, OUTPUT);
  pinMode(FAN_RELAY,    OUTPUT);
  pinMode(BUZZER,       OUTPUT);
  pinMode(SMOKE_PIN,    INPUT);

  pinMode(HEATER_BUTTON,     INPUT_PULLUP);
  pinMode(FAN_BUTTON,        INPUT_PULLUP);
  pinMode(AUTOMATION_BUTTON, INPUT_PULLUP);

  digitalWrite(HEATER_RELAY, LOW);
  digitalWrite(FAN_RELAY,    LOW);
  digitalWrite(BUZZER,       LOW);

  attachInterrupt(digitalPinToInterrupt(HEATER_BUTTON), handleHeaterInterrupt, FALLING);
  attachInterrupt(digitalPinToInterrupt(FAN_BUTTON),    handleFanInterrupt,    FALLING);

  lcd.setCursor(0,0);
  lcd.print("Poultry System");
  lcd.setCursor(0,1);
  lcd.print(" Initializing ");
  delay(2000);
  lcd.clear();

  Serial.println(F("System Initialized."));
}

void loop() {
  // 1) Read sensors
  float temp    = dht.readTemperature();
  float hum     = dht.readHumidity();
  int   smoke   = analogRead(SMOKE_PIN);
  int   smokeP  = map(smoke, 0, 1023, 0, 100);
        smokeP  = constrain(smokeP, 0, 100);

  // 2) Handle manual toggles logged from IRQ
```

```
if (heaterToggled) {
  Serial.print(F("Heater Manual: "));
  Serial.println( heaterManualState ? "ON":"OFF" );
  heaterToggled = false;
}
if (fanToggled) {
  Serial.print(F("Fan Manual: "));
  Serial.println( fanManualState ? "ON":"OFF" );
  fanToggled = false;
}

// 3) Debounce & toggle automation
if (!digitalRead(AUTOMATION_BUTTON)) {
  delay(100);
  while (!digitalRead(AUTOMATION_BUTTON)) {}
  automationMode = !automationMode;
  // Reset alerts when switching mode
  lowTempAlertSent  = false;
  highTempAlertSent = false;
}

// 4) Drive relays
if (automationMode) {
  // Heater control
  if (temp < TEMP_LOW) {
    digitalWrite(HEATER_RELAY, HIGH);
    if (!lowTempAlertSent) {
      sendSMS("ALERT! Temp below threshold >> Heater turned ON!");
      lowTempAlertSent = true;
    }
  } else {
    digitalWrite(HEATER_RELAY, LOW);
    lowTempAlertSent = false;
  }

  // Fan control
  if (temp > TEMP_HIGH) {
    digitalWrite(FAN_RELAY, HIGH);
    if (!highTempAlertSent) {
      sendSMS("ALERT! Temp above threshold >> Fan turned ON!");
      highTempAlertSent = true;
    }
  } else {
    digitalWrite(FAN_RELAY, LOW);
```

```
    highTempAlertSent = false;
  }
} else {
  digitalWrite(HEATER_RELAY, heaterManualState ? HIGH : LOW);
  digitalWrite(FAN_RELAY,    fanManualState    ? HIGH : LOW);
}

// 5) Smoke alarm + SMS
if (smoke > SMOKE_THRESHOLD) {
  digitalWrite(BUZZER, HIGH);
  if (!smokeAlertSent) {
    sendSMS("ALERT! Smoke detected in poultry house!");
    Serial.println(F("SMOKE ALERT! SMS sent."));
    smokeAlertSent = true;
  }
} else {
  digitalWrite(BUZZER, LOW);
  smokeAlertSent = false;
}

// 6) Update LCD
lcd.setCursor(0,0);
lcd.print(F("T:"));
lcd.print(temp,1);
lcd.print(F("C H:"));
lcd.print(hum,0);
lcd.print(F("%   "));

lcd.setCursor(0,1);
lcd.print(F("Smoke:"));
lcd.print(smokeP);
lcd.print(F("% M:"));
lcd.print( automationMode ? 'A':'M' );
lcd.print("   ");

// 7) Periodic debug
static unsigned long lastLog = 0;
if (millis() - lastLog > 1000) {
  Serial.println(F("---- Readings ----"));
  Serial.print(F("Temp: "));
    Serial.print(temp,1);
    Serial.println(F(" C"));
  Serial.print(F("Hum : "));
    Serial.print(hum,0);
```

```
      Serial.println(F(" %"));
    Serial.print(F("Smoke: "));
      Serial.print(smoke);
      Serial.print(F(" ("));
      Serial.print(smokeP);
      Serial.println(F("%)"));
    Serial.print(F("Heater: "));
      Serial.print(digitalRead(HEATER_RELAY) ? "ON":"OFF");
      Serial.print(F("  Fan: "));
      Serial.println(digitalRead(FAN_RELAY) ? "ON":"OFF");
    Serial.print(F("Mode  : "));
      Serial.println(automationMode ? "AUTO":"MANUAL");
    Serial.println(F("------------------"));
    lastLog = millis();
  }

  delay(100);
}

// --- Interrupt routines ---
void handleHeaterInterrupt() {
  unsigned long now = millis();
  if (now - lastHeaterIRQ > debounceDelay) {
    heaterManualState = !heaterManualState;
    automationMode     = false;
    heaterToggled      = true;
    lastHeaterIRQ      = now;
  }
}

void handleFanInterrupt() {
  unsigned long now = millis();
  if (now - lastFanIRQ > debounceDelay) {
    fanManualState   = !fanManualState;
    automationMode   = false;
    fanToggled       = true;
    lastFanIRQ       = now;
  }
}

void sendSMS(const String &msg) {
  Serial.println("sedig message");
  gsm.println(F("AT+CMGF=1"));
  delay(500);
```

```
  gsm.println(F("AT+CMGS=\"+256748332550\"")); // your number
  delay(500);
  gsm.println(msg);
  gsm.write(26);  // Ctrl+Z
  delay(2000);
}
```