Functionality

To implement this advanced home monitoring system with the specified functionalities, we will integrate additional components like a motor for the fan and enhance the logic in the Arduino code to meet the requirements. We will also develop a mobile application dashboard using Blynk, a platform that allows easy creation of mobile apps for IoT projects.

**Components Needed:**

1. **Arduino Uno**
2. **MQ-7 Gas Sensor**
3. **LDR (Light Dependent Resistor)**
4. **PIR (Passive Infrared) Sensor**
5. **Infrared Sensor**
6. **Temperature Sensor (e.g., DHT11 or LM35)**
7. **Push Buttons**
8. **Breadboard**
9. **16x2 LCD Display**
10. **LED Bulbs (various colors)**
11. **Buzzers**
12. **Resistors (various values)**
13. **Potentiometer (for adjusting LCD contrast)**
14. **Jumper Wires**
15. **Power Supply (e.g., 9V battery or USB power)**
16. **DC Motor (for the fan)**
17. **Motor Driver Module (e.g., L298N)**
18. **Relay Module (for controlling the lights)**
19. **Bluetooth Module (e.g., HC-05)**

**Connection Design:**

**Components and Pins:**

1. **MQ-7 Gas Sensor**
   * Analog Output: Arduino Mega Analog Pin A0
2. **LDR (Light Dependent Resistor)**
   * One end: Arduino Mega 5V
   * Other end: Arduino Mega Analog Pin A1 (through 10k ohm resistor to GND)
   * Junction: Arduino Mega GND
3. **PIR Sensor**
   * Digital Output: Arduino Mega Digital Pin 7
   * Power: Arduino Mega 5V
   * Ground: Arduino Mega GND
4. **Infrared Sensor**
   * Digital Output: Arduino Mega Digital Pin 8
   * Power: Arduino Mega 5V
   * Ground: Arduino Mega GND
5. **DHT11 Temperature and Humidity Sensor**
   * Data: Arduino Mega Digital Pin 6
   * Power: Arduino Mega 5V
   * Ground: Arduino Mega GND
6. **Push Button**
   * One side: Arduino Mega GND
   * Other side: Arduino Mega Digital Pin 2 (with a 10k ohm pull-up resistor)
7. **16x2 LCD Display**
   * VSS: Arduino Mega GND
   * VDD: Arduino Mega 5V
   * VEE: Connect to a potentiometer for contrast adjustment (middle pin to VEE, one pin to 5V, and the other to GND)
   * RS: Arduino Mega Digital Pin 12
   * RW: Arduino Mega GND
   * E: Arduino Mega Digital Pin 11
   * D4: Arduino Mega Digital Pin 5
   * D5: Arduino Mega Digital Pin 4
   * D6: Arduino Mega Digital Pin 3
   * D7: Arduino Mega Digital Pin 9
8. **Green LED**
   * Anode: Arduino Mega Digital Pin 10 (with a 220-ohm resistor)
   * Cathode: Arduino Mega GND
9. **Red LED**
   * Anode: Arduino Mega Digital Pin 13 (with a 220-ohm resistor)
   * Cathode: Arduino Mega GND
10. **Buzzer**
    * Positive terminal: Arduino Mega Digital Pin 45
    * Negative terminal: Arduino Mega GND
11. **Relay Module**
    * VCC: Arduino Mega 5V
    * GND: Arduino Mega GND
    * IN: Arduino Mega Digital Pin 14
    * NO (Normally Open): Connect to the positive terminal of the DC Fan (or Motor)
    * COM (Common): Connect to the negative terminal of the DC Fan (or Motor)

Arduino code

#include <LiquidCrystal.h>

#include <DHT.h>

#include <SoftwareSerial.h>

#define DHTPIN 6

#define DHTTYPE DHT11

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

DHT dht(DHTPIN, DHTTYPE);

const int gasPin = A0;

const int ldrPin = A1;

const int pirPin = 7;

const int irPin = 8;

const int buttonPin = 9;

const int greenLed = 10;

const int redLed = 13;

const int buzzer = 13;

const int relayPin = 11;

const int motorPin = 10;

SoftwareSerial BTSerial(0, 1); // RX, TX for HC-05

void setup() {

pinMode(gasPin, INPUT);

pinMode(ldrPin, INPUT);

pinMode(pirPin, INPUT);

pinMode(irPin, INPUT);

pinMode(buttonPin, INPUT\_PULLUP);

pinMode(greenLed, OUTPUT);

pinMode(redLed, OUTPUT);

pinMode(buzzer, OUTPUT);

pinMode(relayPin, OUTPUT);

pinMode(motorPin, OUTPUT);

lcd.begin(16, 2);

dht.begin();

Serial.begin(9600);

BTSerial.begin(9600);

lcd.print("Home Monitoring");

}

void loop() {

int gasLevel = analogRead(gasPin);

int ldrValue = analogRead(ldrPin);

int pirValue = digitalRead(pirPin);

int irValue = digitalRead(irPin);

float tempValue = dht.readTemperature();

lcd.setCursor(0, 1);

lcd.print("Temp:");

lcd.print(tempValue);

lcd.print("C ");

lcd.setCursor(0, 0);

lcd.print("Gas:");

lcd.print(gasLevel);

// Control lights based on LDR value

if (ldrValue < 500) {

digitalWrite(relayPin, HIGH); // Turn on lights

} else {

digitalWrite(relayPin, LOW); // Turn off lights

}

// Alert on high temperature

if (tempValue > 30) { // Adjust temperature threshold as needed

digitalWrite(redLed, HIGH);

tone(buzzer, 1000, 500); // Play a beat sound

delay(500);

noTone(buzzer);

delay(500);

digitalWrite(motorPin, HIGH); // Turn on fan

delay(10000); // Run fan for 10 seconds

digitalWrite(redLed, LOW);

digitalWrite(motorPin, LOW); // Turn off fan

BTSerial.println("ALERT: High Temperature");

}

// Alert on motion detection

if (pirValue == HIGH) {

digitalWrite(redLed, HIGH);

for (int i = 0; i < 3; i++) {

tone(buzzer, 1500, 200); // Play a beat sound

delay(200);

noTone(buzzer);

delay(200);

}

digitalWrite(redLed, LOW);

BTSerial.println("ALERT: Motion Detected");

}

// Alert on gas leakage

if (gasLevel > 400) { // Adjust gas threshold as needed

digitalWrite(redLed, HIGH);

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

tone(buzzer, 2000, 100); // Play a different beat sound

delay(100);

noTone(buzzer);

delay(100);

}

digitalWrite(redLed, LOW);

BTSerial.println("ALERT: Gas Leakage");

}

// Send data via Bluetooth

BTSerial.print("Temperature:");

BTSerial.print(tempValue);

BTSerial.print("C, Gas Level:");

BTSerial.println(gasLevel);

delay(2000);

}

Mobile application

**Mobile Application Dashboard using MIT App Inventor**

**Step-by-Step Guide to Enhance the Mobile App:**

1. **Go to MIT App Inventor:**
   * Visit the MIT App Inventor website.
2. **Start a New Project:**
   * Click on "Start New Project" and name it "HomeMonitoringApp".
3. **Design the User Interface:**
   * **Add a Label:**
     + Drag and drop a Label component onto the screen.
     + Set the text to "Home Monitoring System".
   * **Add Labels for Sensor Data:**
     + Drag and drop four more Label components.
     + Set the text to "Temperature: ", "Gas Level: ", "Motion Detected: ", and "Light Level: ".
     + Next to each of these labels, add another Label to display the data (e.g., tempLabel, gasLabel, motionLabel, lightLabel).
   * **Add Labels for Alerts:**
     + Add three more Label components.
     + Set the text to "Temperature Alert: ", "Motion Alert: ", and "Gas Alert: ".
     + Next to each of these labels, add another Label to display the alerts (e.g., tempAlertLabel, motionAlertLabel, gasAlertLabel).
   * **Add Buttons for Bluetooth:**
     + Add a Button component and set the text to "Connect".
4. **Add Bluetooth Connectivity:**
   * From the "Connectivity" drawer, drag and drop the BluetoothClient component to the screen.
5. **Block Programming:**
   * Click on the "Blocks" tab to start coding the functionality.
   * Create blocks for connecting to the Bluetooth device, reading data from the HC-05, and displaying the data on the labels.

// When the Connect button is clicked

when ConnectButton.Click {

call BluetoothClient1.Connect(address) // Replace 'address' with your HC-05 address

}

// When Bluetooth receives data

when BluetoothClient1.ReceiveText(text) {

if contains text "ALERT:" {

if contains text "High Temperature" {

set tempAlertLabel.Text to "High Temperature"

}

if contains text "Motion Detected" {

set motionAlertLabel.Text to "Motion Detected"

}

if contains text "Gas Leakage" {

set gasAlertLabel.Text to "Gas Leakage"

}

} else {

// Split text and update labels

set tempLabel.Text to select list item (split text at ",") 1

set gasLabel.Text to select list item (split text at ",") 2

set motionLabel.Text to select list item (split text at ",") 3

set lightLabel.Text to select list item (split text at ",") 4

}

}

**Explanation of the Blocks:**

* The Connect button initiates the connection to the HC-05 module.
* The BluetoothClient1.ReceiveText block listens for incoming data.
* If the received text contains "ALERT:", it updates the corresponding alert label.
* If the received text contains sensor data, it splits the text and updates the respective labels.

**Additional Steps:**

1. **Connect Your Mobile Device:**
   * Install the MIT App Inventor Companion app on your mobile device.
   * Connect the MIT App Inventor Companion app to the MIT App Inventor on your computer using the provided QR code.
2. **Test the App:**
   * Pair your mobile device with the HC-05 Bluetooth module.
   * Open the HomeMonitoringApp on your mobile device.
   * Click the "Connect" button to start receiving data and alerts from the Arduino.

By following these steps, you'll have a functional home monitoring system that sends data and security alerts to a mobile application via Bluetooth. The mobile app will display real-time data and alerts from various sensors, and you can customize the app further based on your requirements.