**Fire Alarm System using Arduino Uno**

**✅ Project Title:**

**Fire Alarm System Using Arduino Uno, Flame Sensor, Buzzer, and 16x2 I2C LCD**

**🧾 Description:**

This project detects the presence of fire using a **flame sensor**. If fire is detected:

* A **buzzer beeps for 10 seconds** (alternating on/off every 0.5 seconds),
* A **16x2 I2C LCD** displays a warning message,
* The **Serial Monitor** logs system activity.

Otherwise, the system monitors continuously, showing "No Fire Detected" on the LCD and logs status via Serial Monitor.

**🔌 Components Required:**

| **Component** | **Quantity** |
| --- | --- |
| Arduino Uno | 1 |
| Flame Sensor (Digital) | 1 |
| Buzzer (5V) | 1 |
| 16x2 LCD with I2C module | 1 |
| Breadboard | 1 |
| Jumper Wires | As needed |

**🔗 Circuit Diagram / Pin Connections:**

**(**[**URL:-https://app.cirkitdesigner.com/project/9766b2a3-d406-4661-80c3-595b55a8355d**](URL:-https://app.cirkitdesigner.com/project/9766b2a3-d406-4661-80c3-595b55a8355d)**)**

| **Arduino Pin** | **Connected To** | **Description** |
| --- | --- | --- |
| D2 | Flame Sensor OUT | Digital signal from flame sensor |
| D5 | Buzzer +ve pin | Controls buzzer sound |
| GND | Flame Sensor GND, Buzzer GND, I2C GND | Common Ground |
| 5V | Flame Sensor VCC, I2C VCC | 5V power supply |
| A4 (SDA) | I2C LCD SDA | I2C data line |
| A5 (SCL) | I2C LCD SCL | I2C clock line |

**🖼️ Pinout Diagram Reference:**

**LCD I2C Module**:

* SDA → Arduino **A4**
* SCL → Arduino **A5**
* VCC → Arduino **5V**
* GND → Arduino **GND**

**Flame Sensor (Digital version)**:

* VCC → Arduino **5V**
* GND → Arduino **GND**
* OUT → Arduino **D2**

**Buzzer**:

* Positive pin → Arduino **D5**
* Negative pin → Arduino **GND**

**🧠 Logic of Operation:**

| **Flame Sensor Output** | **Condition** | **Action** |
| --- | --- | --- |
| LOW (0) | Fire Detected | Buzzer beeps 10 sec, LCD alerts |
| HIGH (1) | No Fire | Buzzer off, LCD shows normal status |

**🧾 Libraries You Need**

* LiquidCrystal\_I2C

***Install them via the Arduino Library Manager (Sketch → Include Library → Manage Libraries).***

**🧾 Full Code:**

You already have the working code, but here it is again for documentation:

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

// Create LCD object

LiquidCrystal\_I2C lcd(0x27, 16, 2); // Use 0x3F if 0x27 doesn't work

const int flameSensorPin = 2;

const int buzzerPin = 5;

void setup() {

pinMode(flameSensorPin, INPUT);

pinMode(buzzerPin, OUTPUT);

Serial.begin(9600);

Serial.println("🔥 Fire Alarm System Starting...");

lcd.init();

lcd.backlight();

lcd.setCursor(0, 0);

lcd.print("Fire Alarm Ready");

delay(2000);

lcd.clear();

}

void loop() {

int flameStatus = digitalRead(flameSensorPin);

if (flameStatus == LOW) { // Flame detected

Serial.println("🚨 FIRE DETECTED! Starting alarm sequence...");

lcd.setCursor(0, 0);

lcd.print("\*\* FIRE ALERT \*\*");

lcd.setCursor(0, 1);

lcd.print("Evacuate Now!");

for (int i = 0; i < 10; i++) {

digitalWrite(buzzerPin, HIGH);

Serial.print("🚨Alarm ON ("); Serial.print(i+1); Serial.println("/10)");

delay(500);

digitalWrite(buzzerPin, LOW);

Serial.print("Beep OFF ("); Serial.print(i+1); Serial.println("/10)");

delay(500);

}

Serial.println("🔄 Alarm sequence complete. Cooling down...");

lcd.setCursor(0, 0);

lcd.print("Waiting... ");

lcd.setCursor(0, 1);

lcd.print("Rechecking... ");

delay(2000);

lcd.clear();

} else {

digitalWrite(buzzerPin, LOW);

Serial.println("✅ No fire detected. Monitoring...");

lcd.setCursor(0, 0);

lcd.print("No Fire Detected");

lcd.setCursor(0, 1);

lcd.print("System Normal ");

delay(500);

}

}

**🧪 Testing Procedure:**

1. Connect all components as per the table.
2. Upload code to Arduino Uno.
3. Open **Serial Monitor** at **9600 baud**.
4. Place a flame source near the sensor (e.g., matchstick) at safe distance.
5. Observe:
   * Buzzer beeps for 10 seconds
   * LCD shows fire alert
   * Serial Monitor logs activity

**⚠️ Safety Note:**

* Do **not hold a real flame close** for too long.
* Test briefly and extinguish quickly.
* Avoid overheating the sensor.

**📝 Optional Improvements:**

* Use an **LED** as a visual alert.
* Add a **relay module** to trigger an exhaust fan.
* Add **GSM or IoT module (like ESP32)** to send SMS/email alerts.
* Use **EEPROM** to log fire events.

Let me know if you'd like:

* **A fritzing circuit diagram**
* **.docx or PDF format** of the documentation
* **Extension to IoT/cloud alert**

I'm here to help!