Southeast University

Department of Computer Science and Engineering

Project Report

Project Name: Arduino Home Security and Fire Smoke Detection System with Automatic Water Pump Activation

Course Code and Section	CSE382.2
Course Title	Introduction to Embedded Systems Lab
Program	B.Sc. in CSE
Department	Computer Science and Engineering
Semester	Fall 2025
Assignment Posted on	13/12/2025
Assignment Submission Date	22/01/2026

Submitted By

SL	Name	Student ID
1	Sheikh Arman Karim Aditto	2022100000031
2	Md. Jahidul Islam	2022100000048
3	Md. Zubayer Ahmad Shibly	2022100000052
4	Shahrin Kabir Tashin	2022100000055

Contents

1 Project Overview

This project builds a low-cost home security and safety prototype using an Arduino Uno R3 as the central controller. The system combines two important functions: (1) password-based door access control and (2) continuous hazard monitoring for flame and smoke/gas events. A 4x4 keypad allows the user to enter a 4-digit password, while a 16x2 I2C LCD provides real-time feedback such as prompts, warnings, and alert messages. When a correct password is entered, a door-lock servo rotates to unlock/open the door and then automatically returns to the locked position after a fixed delay.

For safety, the controller continuously reads a flame sensor and an MQ-2 smoke/-gas sensor (digital detection). If either sensor indicates danger, the system immediately switches to a high-priority FIRE ALERT state, activates a buzzer and LED warning pattern, and turns on a relay output intended for an emergency device such as a fan or (with proper driving) a small water pump. A security lockout feature is also included: repeated wrong-password attempts trigger a locked state until the correct password is entered. An optional sweep/scan servo can be used to rotate a sensor mount or demonstrate automation movement.

2 Objectives

- Design a complete Arduino-based prototype that combines access control and hazard monitoring in a single system.
- Implement password-protected door unlocking using a 4x4 keypad, on-screen LCD feedback, and a door servo actuator.
- Provide clear user interaction using a 16x2 I2C LCD (status prompts, masked password entry, warnings, and alerts).
- Detect flame and smoke/gas hazards using a flame sensor and MQ-2 sensor (digital output) and prioritize safety events over access control.
- Trigger emergency signaling using both an audible buzzer and a visible LED pattern that is easily noticeable in a room.
- Control an external emergency device through a relay output (fan or pump) during hazard events, with correct electrical isolation.
- Add a lockout mechanism after repeated wrong password attempts to reduce brute-force guessing.
- Demonstrate an optional sweep servo to support sensor scanning or automation movement for future extensions.
- Test the full system using realistic scenarios and document behavior through an observation table and analysis.

3 Components and Tools

The following hardware and tools were used to build and test the prototype (the list matches the project design and wiring plan).

Item	Component / Tool
1	Arduino Uno R3
2	4x4 Matrix Keypad
3	16x2 LCD with I2C backpack (typical address 0x27)
4	Servo Motor SG90 (door lock servo)
5	Servo Motor (sweep/scan servo, optional)
6	Flame Sensor Module (digital output)
7	MQ-2 Smoke/Gas Sensor Module (digital output; analog optional)
8	Buzzer, LED indicator, Relay module, breadboard, jumper wires
9	Submersible 3V Mini DC Water Pump, 5V power supply

Power and safety note: If servos draw high current, a separate 5V supply should be used for the servos, and all grounds must be connected together for stable operation.

4 System Design and Architecture

The system is organized into three coordinated subsystems. The **Access Control** subsystem handles keypad input, password verification, LCD feedback, and door servo actuation. The **Hazard Detection** subsystem continuously reads the flame and MQ-2 digital outputs. The **Emergency Response** subsystem drives the buzzer and LED warning signals and activates the relay output to power an emergency device.

Priority logic is critical: hazard detection has the highest priority. When a hazard is detected, the controller enters an alert loop and keeps the alarms and relay active until the environment becomes safe. During this period, keypad processing and sweep-servo updates are paused to ensure the alert behavior is immediate and consistent.

5 Circuit Connections and Wiring Summary

The following pin mapping summary is used to connect each module to the Arduino Uno. All modules share a common ground.

Module / Device	Signal	Arduino Pin	Notes
Flame Sensor	Digital OUT	D9	INPUT_PULLUP, active LOW detection
MQ-2 Sensor	Digital OUT	D10	INPUT_PULLUP, active LOW detection
MQ-2 Sensor	Analog OUT	A2	Optional, declared for future thresholding
Buzzer	Control	D3	ON during alarms/alerts
LED Indicator	Control	A3	Blinks during lockout and alerts
Relay Module	IN	D11	Active LOW (ON=LOW, OFF=HIGH)

Module / Device	Signal	Arduino Pin	Notes
Door Servo (SG90)	Signal	D5	Open=90 deg, Closed=0 deg
Sweep Servo	Signal	D6	Sweeps 40-120 deg
Keypad (4x4)	Row pins	A1, A0, D2, D4	Row inputs
Keypad (4x4)	Column pins	D7, D8, D12, D13	Column inputs
LCD (I2C)	SDA / SCL	A4 / A5	I2C data / clock

Safety notes: Use a stable 5V supply for Arduino and sensors. If servos draw high current, use a separate 5V supply for servos and connect grounds together. Do not connect a pump or other high-current load directly to an Arduino pin; use the relay output or a proper MOSFET/transistor driver and an external supply.

6 Working Procedure

Hardware setup: Assemble the Arduino, keypad, LCD, sensors, buzzer, LED, relay module, and servos on a breadboard or prototype board. Connect every module using the wiring summary table, then double-check the power rails (5V and GND) before powering on. For the LCD, connect SDA to A4 and SCL to A5; confirm the I2C address (commonly 0x27) and adjust in code if required.

Normal operation: When the system powers on, the LCD displays an "Enter Pass:" prompt. The user enters four digits using the keypad; the LCD masks the digits with asterisks to avoid exposing the password. If the password matches the stored value (default 1234), the LCD shows a welcome message and the door servo rotates to the open position. After a fixed time window (60 seconds), the controller automatically returns the servo to the locked position and resets the display back to the initial prompt.

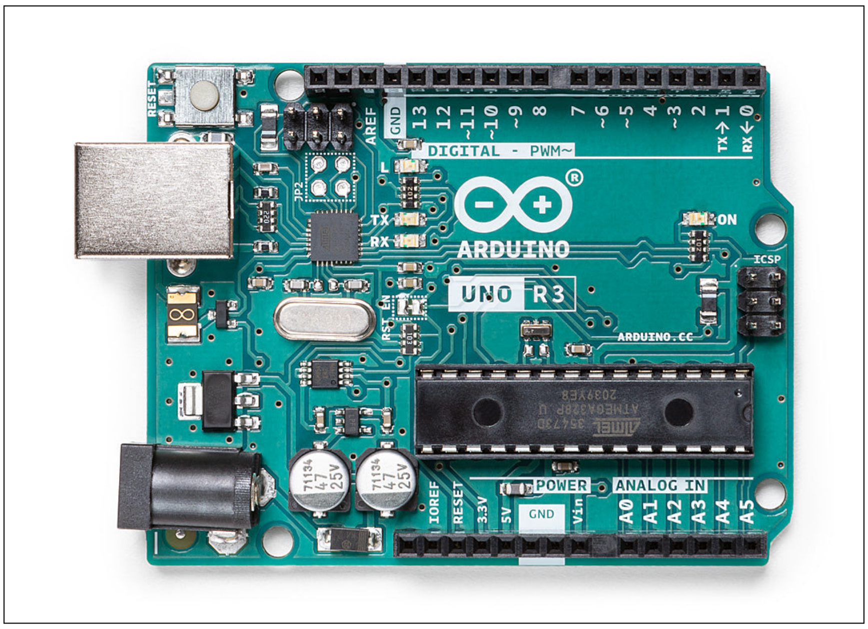
Security behavior: Each incorrect password attempt increases a failed-attempt counter. The system displays a warning with remaining attempts to guide the user. After five consecutive failures, the controller enters a LOCKED state and activates an alarm pattern (buzzer on and LED blinking) until a correct password is entered, which resets the counter and returns the system to normal mode.

Hazard behavior: Flame and smoke/gas sensors are read continuously. If either sensor reports danger, the system immediately switches to FIRE ALERT, displays an alert message on the LCD, blinks the LED, sounds the buzzer, and activates the relay output for an emergency device. The alert behavior continues until both sensors return to a safe state. This priority design ensures that safety events override door access actions while danger is present.

7 Images and Component Description

This section contains the key project images. Images 1–7 are placeholders that you can replace with real photos of your components. Image 8 is included from the provided circuit diagram showing the complete wiring.

7.1 Image 1: Arduino Uno R3 Microcontroller Board

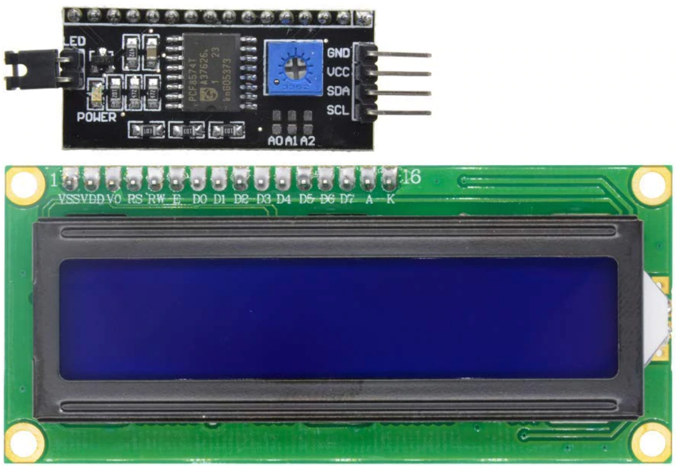


The Arduino Uno R3 is the main controller that reads sensors, processes keypad input, and drives the outputs. Digital pins are used for the keypad matrix, buzzer, relay, and sensors, while analog pins support I2C (A4/A5) and other signals. The Uno executes the control logic continuously so it can respond quickly to hazards while still handling access control. In the project, the Uno is powered from a regulated 5V supply to keep readings stable and reduce random resets.

7.2 Image 2: 4x4 Matrix Keypad

The 4x4 keypad is wired as a matrix of rows and columns to reduce the number of required pins. The firmware scans the keypad and records a 4-digit password while showing masked characters on the LCD. Special keys can be used to clear an entry and restart input when the user makes a mistake. This keypad interface enables a simple, low-cost access control method without needing biometric sensors.

7.3 Image 3: 16x2 LCD with I2C Backpack (User Display)

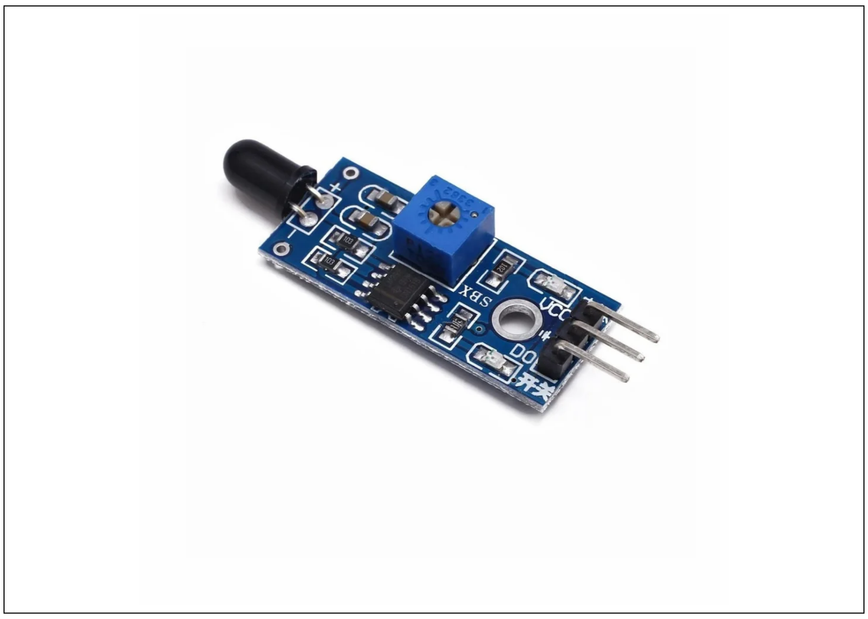


The I2C LCD displays short messages such as “Enter Pass:”, “Welcome Home”, “Wrong Password”, “LOCKED”, and “FIRE ALERT”. Using I2C reduces wiring complexity because only two signal wires (SDA and SCL) are required. The LCD improves usability by providing clear feedback during password entry and during alarm states. If the LCD address differs from 0x27, it can be updated in the code to match the installed I2C backpack.

7.4 Image 4: Servo Motor (Door Lock / Door Actuator)

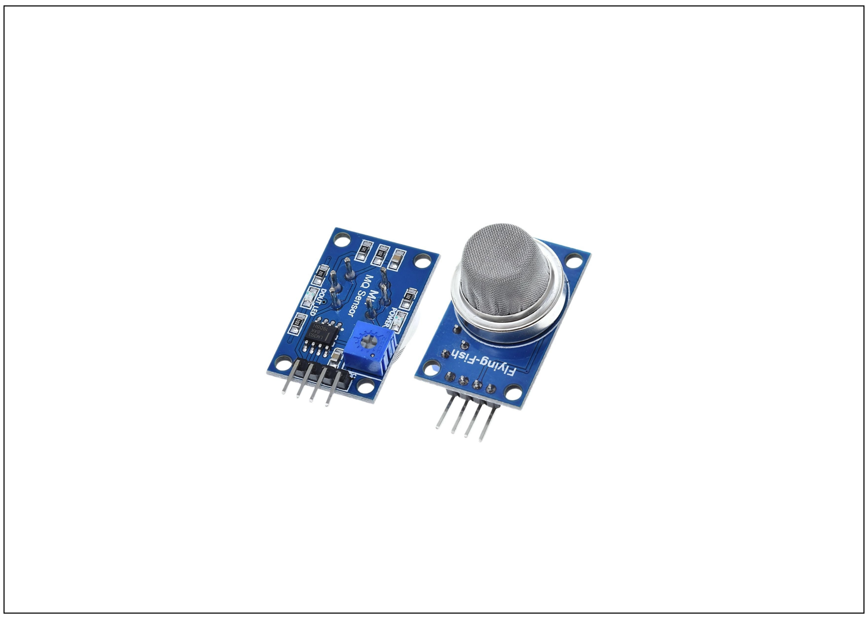
A small servo motor (such as SG90) is used as a mechanical actuator for the door lock mechanism. When the correct password is entered, the servo rotates to an open position (for example 90 degrees). After the unlock window, the servo returns to the closed position (0 degrees) to re-lock automatically. For reliable motion, the servo should be powered from a stable 5V supply and share ground with the Arduino.

7.5 Image 5: Flame Sensor Module (Digital Hazard Input)



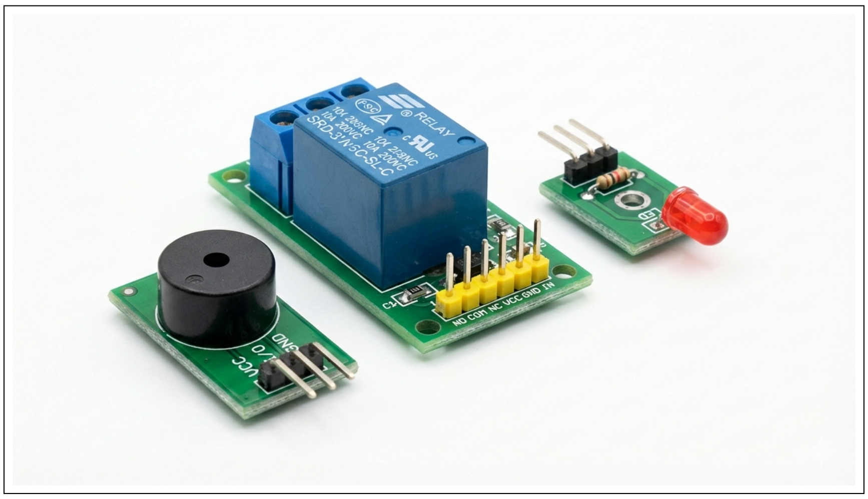
The flame sensor provides a digital output that changes state when flame or strong infrared light is detected. In this design the input is read using INPUT_PULLUP, so an active LOW signal indicates detection. When triggered, the system immediately enters the FIRE ALERT state and activates alarms and the relay output. Sensitivity is adjustable on many sensor modules and should be tuned to reduce false alarms.

7.6 Image 6: MQ-2 Smoke/Gas Sensor Module (Digital Output)



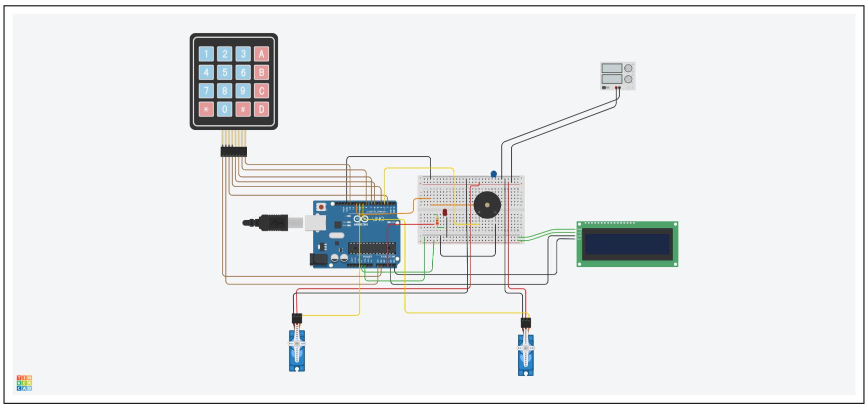
The MQ-2 module detects smoke and combustible gases and can provide both digital and analog signals. This project uses the digital output for simple detection logic; an active LOW indicates smoke/gas presence. The analog output can be used later for calibrated thresholds and improved sensitivity control. Because MQ sensors require warm-up, the system should be given a short stabilization time after power-on.

7.7 Image 7: Buzzer, LED Indicator, and Relay Module (Emergency Output)



The buzzer and LED provide immediate audible and visible alerts during lockout and hazard events. The relay module is used to switch an external emergency load such as a fan or pump, providing electrical isolation from the Arduino. Many relay modules are active LOW; the firmware handles this by writing LOW to enable the relay and HIGH to disable it. For safety, external loads should use appropriate power supplies and insulated wiring, especially when switching higher voltages.

7.8 Image 8: Complete Circuit Diagram (All Components Connected)



This circuit diagram shows the full hardware integration around the Arduino Uno. The 4x4 keypad provides the password input interface and the LCD is used for user prompts, warnings, and alert messages. The buzzer and LED provide audible and visible feedback for wrong-password attempts and safety alerts. Two servos are included: one for the door lock mechanism and another for an optional sweep/scan movement. A stable 5V supply is used to power the modules; in real hardware, servos may require a separate 5V source with a shared ground.

8 Source Code

The following Arduino program implements password entry, door servo control, lockout behavior, hazard detection, alarm signaling, relay control, and a continuous sweep/scan servo. The code is presented without comments as requested.

```
#include <Servo.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <Keypad.h>

const int FLAME_SENSOR_PIN = 9;
const int MQ2_DIGITAL_PIN = 10;
const int MQ2_ANALOG_PIN = A2;
const int BUZZER_PIN = 3;
const int SWEEP_SERVO_PIN = 6;
const int DOOR_SERVO_PIN = 5;
const int LED_PIN = A3;
const int FAN_RELAY_PIN = 11;

const int SWEEP_MIN_POS = 40;
const int SWEEP_MAX_POS = 120;

const int FAN_ON = LOW;
const int FAN_OFF = HIGH;

LiquidCrystal_I2C lcd(0x27, 16, 2);

const byte ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] = {
{'1', '2', '3', 'A'},
{'4', '5', '6', 'B'},
{'7', '8', '9', 'C'},
{'*', '0', '#', 'D'}
};

byte rowPins[ROWS] = {A1, A0, 2, 4};
byte colPins[COLS] = {7, 8, 12, 13};

Keypad customKeypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

Servo myServo;
Servo doorServo;
const int DOOR_OPEN_POS = 90;
const int DOOR_CLOSED_POS = 0;

unsigned long previousServoTime = 0;
const long servoInterval = 15;
int servoPos = SWEEP_MIN_POS;
int servoDirection = 1;

#define PASSWORD_LENGTH 5
char masterPassword[PASSWORD_LENGTH] = "1234";
char enteredPassword[PASSWORD_LENGTH];
byte passwordIndex = 0;
bool unlocked = false;
unsigned long unlockTime = 0;
const int MAX_FAILED_ATTEMPTS = 5;
int failedAttempts = 0;

bool fireAlarmActive = false;
unsigned long previousBlinkMillis = 0;
const long blinkInterval = 200;

void checkPassword();
void keypadActivateAlarm();
void keypadDeactivateAlarm();
void displayInitialScreen();
void handleServoSweep();

void displayInitialScreen() {
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Enter Pass:");
lcd.setCursor(0, 1);
lcd.print(" ");
}
```

```

void keypadDeactivateAlarm() {
failedAttempts = 0;
digitalWrite(BUZZER_PIN, LOW);
digitalWrite(LED_PIN, LOW);
fireAlarmActive = false;
doorServo.write(DOOR_CLOSED_POS);
displayInitialScreen();
}

void keypadActivateAlarm() {
fireAlarmActive = true;
digitalWrite(BUZZER_PIN, HIGH);
}

void checkPassword() {
enteredPassword[passwordIndex] = '\0';

if (strcmp(enteredPassword, masterPassword) == 0) {
unlocked = true;
unlockTime = millis();
keypadDeactivateAlarm();

lcd.setCursor(0, 0);
lcd.print("Welcome Home");

doorServo.write(DOOR_OPEN_POS);
} else {
failedAttempts++;

if (failedAttempts >= MAX_FAILED_ATTEMPTS) {
    keypadActivateAlarm();
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("!!! LOCKED !!!");
    lcd.setCursor(0, 1);
    lcd.print("                ");
} else {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Wrong Password!");
    lcd.setCursor(0, 1);
    lcd.print("Try again (");
    lcd.print(MAX_FAILED_ATTEMPTS - failedAttempts);
    lcd.print(" left)");
    delay(2000);
    displayInitialScreen();
}
}

passwordIndex = 0;
memset(enteredPassword, 0, PASSWORD_LENGTH);
}

void handleServoSweep() {
if (millis() - previousServoTime >= servoInterval) {
previousServoTime = millis();

servoPos += servoDirection;

if (servoPos >= SWEEP_MAX_POS) {
    servoDirection = -1;
    servoPos = SWEEP_MAX_POS;
} else if (servoPos <= SWEEP_MIN_POS) {
    servoDirection = 1;
    servoPos = SWEEP_MIN_POS;
}

myServo.write(servoPos);
}
}

void setup() {
```

```
pinMode(BUZZER_PIN, OUTPUT);
pinMode(LED_PIN, OUTPUT);
pinMode(FAN_RELAY_PIN, OUTPUT);
pinMode(FLAME_SENSOR_PIN, INPUT_PULLUP);
pinMode(MQ2_DIGITAL_PIN, INPUT_PULLUP);

digitalWrite(FAN_RELAY_PIN, FAN_OFF);

myServo.attach(SWEEP_SERVO_PIN);
myServo.write(SWEEP_MIN_POS);

doorServo.attach(DOOR_SERVO_PIN);
doorServo.write(DOOR_CLOSED_POS);

lcd.begin();
lcd.backlight();

Serial.begin(9600);
Serial.println("--- Merged System Initialized ---");

displayInitialScreen();
}

void loop() {
unsigned long currentMillis = millis();

int flameState = digitalRead(FLAME_SENSOR_PIN);
int smokeState = digitalRead(MQ2_DIGITAL_PIN);

bool hazardDetected = (flameState == LOW) || (smokeState == LOW);

if (hazardDetected) {
digitalWrite(FAN_RELAY_PIN, FAN_ON);

lcd.clear();
lcd.setCursor(0, 0);
lcd.print("!!! FIRE ALERT !!!");

while (digitalRead(FLAME_SENSOR_PIN) == LOW || digitalRead(MQ2_DIGITAL_PIN) == LOW) {
    if (flameState == LOW) Serial.println("FLAME DETECTED");
    if (smokeState == LOW) Serial.println("SMOKE/GAS DETECTED");

    digitalWrite(BUZZER_PIN, HIGH);
    digitalWrite(LED_PIN, HIGH);
    delay(200);

    digitalWrite(BUZZER_PIN, LOW);
    digitalWrite(LED_PIN, LOW);
    delay(200);

    flameState = digitalRead(FLAME_SENSOR_PIN);
    smokeState = digitalRead(MQ2_DIGITAL_PIN);
}

digitalWrite(BUZZER_PIN, LOW);
digitalWrite(LED_PIN, LOW);
digitalWrite(FAN_RELAY_PIN, FAN_OFF);

displayInitialScreen();
} else {
digitalWrite(FAN_RELAY_PIN, FAN_OFF);

if (fireAlarmActive) {
    if (currentMillis - previousBlinkMillis >= blinkInterval) {
        previousBlinkMillis = currentMillis;
        digitalWrite(LED_PIN, !digitalRead(LED_PIN));
    }
}

if (unlocked && (currentMillis - unlockTime >= 60000)) {
    unlocked = false;
    doorServo.write(DOOR_CLOSED_POS);
    displayInitialScreen();
}

char key = customKeypad.getKey();

if (key && !unlocked) {
    if (isdigit(key) && passwordIndex < 4) {
        enteredPassword[passwordIndex] = key;
        lcd.setCursor(passwordIndex, 1);
        lcd.print('*');
        passwordIndex++;

        if (passwordIndex == 4) {
            delay(500);
            checkPassword();
        }
    } else if (key == '*' || key == '#') {
        passwordIndex = 0;
        memset(enteredPassword, 0, PASSWORD_LENGTH);
        displayInitialScreen();
    }
}

handleServoSweep();
}
}
```

9 Result and Analysis

Functional testing was performed to confirm that the system behaves correctly under normal access control operation, wrong-password attempts, lockout scenarios, and hazard detection events. The tests verify both the user interface (LCD messages) and the physical/electrical outputs (door servo motion, buzzer and LED patterns, and relay activation). Overall, the system produced consistent behavior across repeated trials and matched the intended control logic, with hazard detection always overriding access-control actions when danger is present.

9.1 Test Results Table

Test	Input / Condition	Observed Behavior	Status
T1	Power ON (no hazard)	LCD shows Enter Pass; door locked; relay OFF; alarms OFF	Pass
T2	Enter correct password 1234	LCD Welcome; door servo opens; failed counter resets	Pass
T3	Wait 60 seconds after unlock	Door servo returns to locked; LCD returns to Enter Pass	Pass
T4	Enter incorrect code	LCD Wrong Password; attempts decrease; door stays locked	Pass
T5	Five consecutive wrong attempts	LCD LOCKED; buzzer ON; LED blinks; door locked	Pass
T6	Enter correct password during lockout	Alarm clears; door opens; LCD Welcome	Pass
T7	Flame or smoke digital input active	LCD FIRE ALERT; buzzer/LED blink; relay ON	Pass
T8	Hazard removed	Buzzer OFF; LED OFF; relay OFF; LCD returns to Enter Pass	Pass
T9	No hazard, idle operation	Sweep servo moves smoothly between limits (40-120 deg)	Pass

Test	Input / Condition	Observed Behavior	Status
T10	Trigger hazard during password entry	Alert immediately interrupts keypad workflow until safe	Pass

9.2 Analysis Highlights

The lockout mechanism prevents repeated brute-force entry by switching to a persistent alarm state after five failures. The hazard-detection logic is implemented as a high-priority state that keeps alarms and relay active until both sensors return safe, which reduces the chance of missing a real event. The relay output is active LOW, which matches common relay modules and is handled directly by the firmware. Because the alert behavior uses a loop and delay-based blinking, keypad input is intentionally paused during FIRE ALERT to keep the emergency response simple and reliable; this behavior can be made non-blocking in future upgrades for advanced features.

9.3 System State Summary

Condition	LCD	Buzzer	LED	Relay	Door Servo
Idle (safe)	Enter Pass	OFF	OFF	OFF	Locked
Correct password	Welcome Home	OFF	OFF	OFF	Open
Auto re-lock	Enter Pass	OFF	OFF	OFF	Locked
Wrong password	Wrong Password	OFF	OFF	OFF	Locked
Locked after 5	LOCKED	ON	Blink	OFF	Locked
Hazard detected	FIRE ALERT	Blink	Blink	ON	State unchanged

From the observation table, it is clear that door access and alerts do not conflict: access control works during safe conditions, while hazard events force the system into a dedicated alert mode. This separation improves reliability and user safety, especially in cases where fire/smoke occurs while a user is attempting to unlock the door.

10 Discussion

This project demonstrates that multiple smart-home functions can be integrated into one Arduino-based system using inexpensive modules. The keypad and LCD combination creates a simple but effective user interface, while the servo motor makes the design suitable for physical door-lock prototypes. The hazard detection approach uses digital outputs for straightforward logic; however, analog readings from the MQ-2 could be used in the future to reduce false triggers and allow calibration for different environments.

Power stability is an important practical issue: servo motors may draw peak current that can reset the Arduino if powered from the same weak supply. A separate servo supply with a shared ground is recommended for robust operation. Another limitation is the blocking alert loop used during FIRE ALERT, which pauses keypad processing. For a more advanced version, the alert behavior can be rewritten using millis-based non-blocking timing so that the system can log events, store timestamps, or send notifications while still keeping alarms active.

11 Conclusion

An Arduino-based home security and automation prototype was successfully designed and implemented to provide password-protected door access, real-time LCD feedback, and continuous hazard monitoring for flame and smoke/gas events. The integrated design demonstrates reliable servo-based door actuation with automatic re-locking, as well as a lockout mechanism that prevents repeated password guessing attempts.

Testing confirmed that, under hazard conditions, the system correctly prioritizes safety by activating alarms and switching a relay output for an emergency device until the environment becomes safe again. The observation tables show consistent mapping between system conditions and outputs, indicating stable control logic. Overall, the project provides a strong foundation for future smart-home extensions such as remote notifications (Wi-Fi/GSM), EEPROM-based password updates, additional sensors (door reed switch, PIR motion), and non-blocking alert logic for richer multi-task behavior.

12 References

[1] Arduino Uno Rev3 Documentation. https://docs.arduino.cc/hardware/uno-rev3/

[2] Arduino Servo Library Reference. https://www.arduino.cc/reference/en/libraries/servo/

[3] Arduino Keypad Library Reference. https://www.arduino.cc/reference/en/libraries/keypad/

[4] LiquidCrystal_I2C Library (common implementation). https://github.com/johnrickman/LiquidCrystal_I2C

[5] MQ-2 Gas Sensor Datasheet (example). https://www.sparkfun.com/datasheets/Sensors/Biometric/MQ-2.pdf

[6] Arduino Language Reference (digitalRead/digitalWrite). https://www.arduino.cc/reference/en/

[7] Full Project Simulation Model (Tinkercad). [https://www.tinkercad.com/things/]

13 Contributions

All contributors participated equally in the overall work (approximately 25% each), including planning, hardware integration, firmware development, testing, and report writing.

Contributor	**Contribution (Equal Share)**
Sheikh Arman Karim Aditto	Focused on overall system architecture and Arduino pin allocation. Assembled and verified keypad, LCD, and servo wiring with stable power distribution and a common ground. Led end-to-end functional tests for correct/incorrect password flows, auto relocking, and alert triggering, and improved report formatting.
Md. Jahidul Islam	Focused on integration planning, including module selection and user-interface behaviour using the 16x2 I2C LCD, including prompts, status messages, and alert displays. Implemented and tested door-servo positioning for locking/unlocking and coordinated timing with LCD feedback. Supported sensor testing and helped document the working procedure and results observations.
Md. Zubayer Ahmad Shibly	Focused on core firmware logic, including password handling, attempt counter, lockout behavior, and safe-to-alert state transitions. Integrated flame and MQ-2 sensor inputs with buzzer/LED indications and relay control logic. Assisted debugging, verification runs, and final proofreading and reference formatting.
Shahrin Kabir Tashin	Focused on hardware module setup and verification for safety outputs (buzzer, LED indicator, and relay module) and sensor modules. Validated hazard response behavior by running repeated flame/smoke scenarios and confirming safety priority over access control. Supported wiring cross-checking, results interpretation, and final report edits.