DEPARTMENT OF MECHATRONICS AND ROBOTICS

MEC104 2021-2022

Open Project: Intelligent Residential System

Group Report Assignment

Group Number: A17

Student Name	Student ID
Muyang Li	2033850
Tian Li	2037785
Xiaoou Jiang	2035533

CONTENTS

1 Abstract

The purpose of this experiment is to design and develop a robust and practical project, which could work independently as a system and implement particular functions for certain scenarios in reality. In this open project, Arduino can be freely programed according to requirements and provide assistance with utilizing its advantages of compatibility and extendibility, thereby accurately operating multiple electronic components, devices and modules in various situations and conditions. Therefore, this simulation experiment will build a project about some feasible residential applications, which mainly consists of 4 independent modules coordinating with control system Arduino to implement functionality such as triggering components, reading data, receiving and writing digital and analog signals according to preset program.

In order to achieve a more intelligent residential experience, both active and passive modules are considered. Specifically, One active module could require user interaction, display prompt messages and return visual feedback in real time. Remaining three are passive modules that can automatically detect arguments by sensors and work in specific situations. Additionally, one module containing photoresistor and illuminating LEDs is specially designed for the consideration of energy-saving, which embodies the idea of sustainability. Through properly running the simulation and orderly operating components, we drew the conclusion that this simulation could normally implement appointed functions and results presented almost meet expectation. Four modules are proven to be capable of realizing their respective functions without mutual contradiction and interference.

2 Introduction

By connecting modules with different functions, which are called and operated in unison via the Arduino board, important parameters and operating states can be handled correctly and electronic components can work together according to the programmed code. The expected configuration of this design are as follows.

- 4-digit password lock: A password lock with a 4×4 touchpad and a LCD;
- Light-control LED array: The light switch will be controlled through the light detection and manual switch;
- Automatic temperature controller: A sensor detects room temperature and manipulates electric fan and heater to maintain the temperature at a normal temperature interval;
- Fire alarm: When the gas sensor detect excessive noxious gas, the connecting buzzer will emit different noise for warning and warning light will be lit up as visual message.

Components used in 2 sections are listed in Table 1 and 2 below.

Table 1: Electronic Components Used in Welcome System

Components	**Quantity**
Arduino Uno R3	1
4×4 Keypad	1
LCD 16×2	1
100Ω Resistor	1
220Ω Resistor	1
1 kΩ Resistor	1
White LED (Lighting Lamp)	5
Button (Manual Switch)	2
Photoresistor	1

Table 2: Electronic Components Used in the House

Components	**Quantity**
Arduino Uno R3	1
LCD 16×2	1
100Ω Resistor	1
200Ω Resistor	3
1 kΩ Resistor	1
Gas Sensor [MQ-2]	1
Temperature Sensor [TMP36]	1
Red LED (Warning Light)	1
Buzzer (Emergency Alarm)	1
DC Motor (Electric Fan)	1
NPN Transistor [BJT]	1
5.1V Zener Diode	1
Green LED (Heater)	1

This whole project combined with 4 fundamental modules. Although the level of size and complexity for each module is relatively basic, it still could be regarded as a typical reference to solve similar practical problems in practical application, especially in the process flow of keyboard module, LCD control, and the use of temperature, gas sensor, and photo resistance. Detailed description, interpretation and final conclusion could be found in subsequent sections below.

3 Theory

3.1 Password Lock

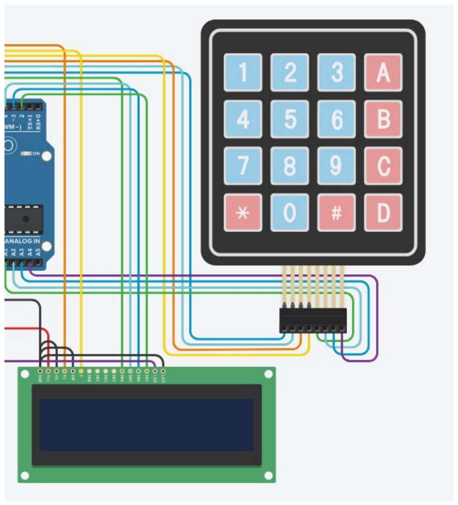

Figure 1: Password Lock

The password lock contains a 4✕4 matrix keypad and a LCD.

3.1.1 4✕4 Matrix Keypad

A 4✕4 matrix keypad is used as an input device that takes inputs from the users. It consists of 16 pins in total with four rows and four columns. On pressing a key, a connection will be established between the corresponding row and column between which the switch is placed. Working principle is pressing a button, then shorts one of the row lines to one of the column lines, which allows current to flow between them. For example, when key '4' is pressed, column 1 and row 2 are shorted.

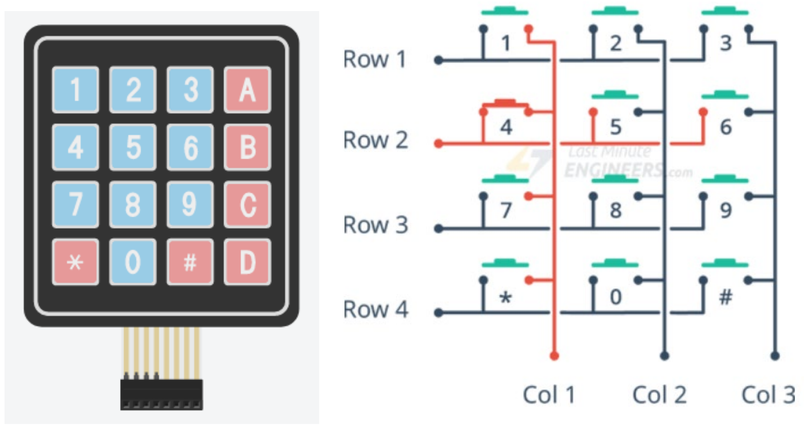

Figure 2: 4✕4 Matrix Keypad and Internal Circuit Diagram

In the coding part, call the library <Keypad.h> to use the functions of a keypad. And in our project, we have defined all of the buttons. And we define the characters just as how they appear on the keypad. Next, we create an object of keypad library.

- The constructor **Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS)** takes five parameters.
- **makeKeymap(keys)** initializes the internal keymap to be equal to the user defined keymap.
- **rowPins** and **colPins** are the Arduino pins to which rows & columns of keypad connected. **ROWS** and **COLS** are the number of rows & columns of the keypad.
- Once a keypad object is created, we can issue a simple command **getKey()** to check which key is pressed.

3.1.2 16×2 LCD Screen

Table 3: Connections between Terminals of LCD and Arduino Pins

Terminal	Connection
LED Cathode	GND
LED Anode	+5V
DB7	Pin2
DB6	Pin3
DB5	Pin4
DB4	Pin5
Enable(EN)	Pin11
Register Select (RS)	Pin12
Read/Write (RW)	GND
V0	GND

V0 here is used to control the contrast ratio of LCD, we connect it to ground as we do not use it. The liquid crystal screen can display some sentences on the position we want. In the password lock part, this LCD work as an instructor for users, which leads users to do next step.

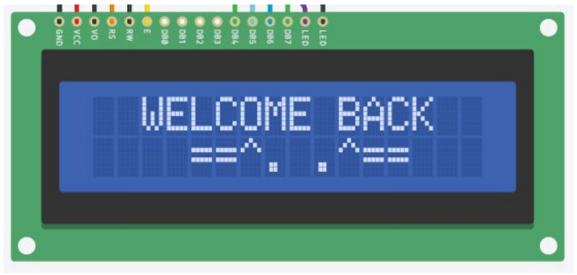

Figure 3: 16×2 LCD

In the coding part, call the library <LiquidCrystal.h> to use the functions of LCDs.

- **lcd.setCursor(x,x)** is used to locate the sentence.
- **lcd.print("xxxx")** is used to display the words we want.

3.2 Light-Control LED

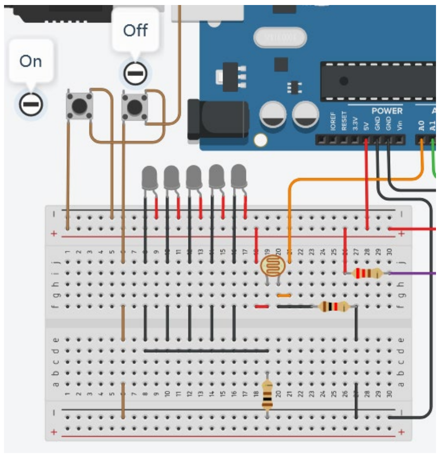

Figure 4: Light-Control LED

The light-control LED contains 4 LEDs, 2 buttons and 1 photoresistor.

3.2.1 4 LEDs

In real life, these 4 LEDs can be regarded as a LED fluorescent lamp. As it is difficult to do packages, so we use 4 LEDs to represent it. The cathode of a LED should be connected with a resistor in case that the current is large, which may damage the LED.

3.2.2 Photoresistor

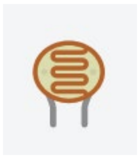

Figure 5: Photoresistor

A photoresistor is a passive component that decreases resistance with respect to receiving luminosity on the component's sensitive surface. If the luminosity density is increasing, the resistance will be decreasing. If the luminosity density is decreasing, the resistance will be increasing. In our project, a photoresistor is used to control the brightness of LEDs.

3.2.3 Button

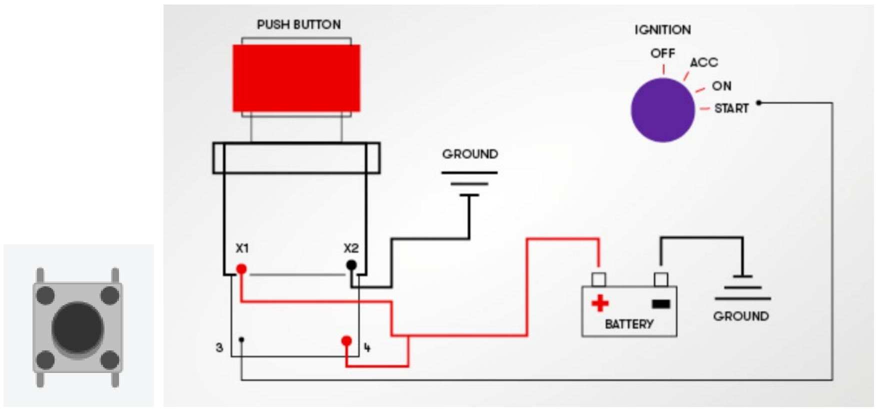

Figure 6: Button in TinkerCAD and A Use Case of Button

Pressure is placed on the button or actuator, resulting in the depression of the internal spring and contacts and the touching of stable contacts at the bottom of the switch. This process will either close or open the electrical circuit. The repeat application of pressure will cause the spring to retract and alter the status of the push button connection. In this project, pressing buttons can be regarded as making choices, that is, either HIGH or LOW will be chosen, which represents turning on or turning off the LEDs regardless of the luminosity density.

3.3 Automatic Temperature Controller & Fire Alarm

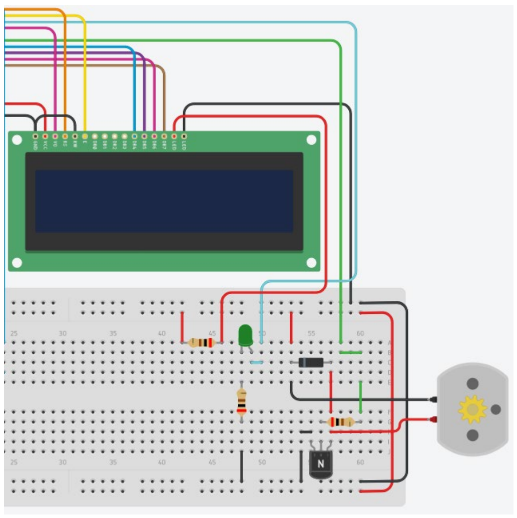

Figure 7: Automatic Temperature Controller with LCD

The Automatic Temperature Controller contains 1 LED (heater), 1 DC motor (electrical fan) and 1 LCD.

3.3.1 DC Motor & BJT & Zener Diode

In our project, the fan which is used to cool the house is simplified as a DC motor. And in this part, it is unnecessary to consider the rotating directions. The function "make the temperature lower" can be realized as long as the DC motor rotates. Connect terminal 1 of the DC motor to a digital pin and connect terminal 2 to GND. In this part, the BJT and the Zener Diode work as DC voltage regulator. The BJT is an adjustment device. Both of these two electronic components ensure the steady rotating speed.

3.3.2 Temperature Sensor

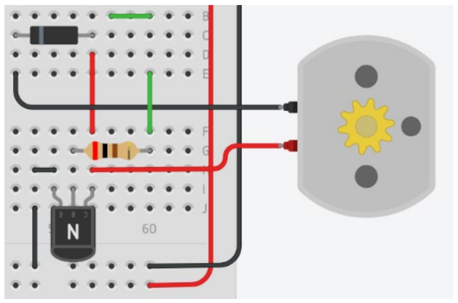

Figure 8: Temperature Sensor in the Circuit

The temperature sensor's basic operation is based on the voltage in its diode. The temperature variation is proportional to the diode's resistance. The resistance in the diode detects and sends the signal into readable values such as Fahrenheit or Centigrade. The voltage change in the MOSFET terminal is the fundamental principle of temperature sensors. The temperature and voltage are directly proportional to each other. That means if the voltage reduces the temperature also decreases and vice-versa. This happens according to the voltage drop between the MOSFET's emitter and the base sensor's terminals. In our project, the output terminal of this sensor is connected to an analog pin.

3.3.3 Gas Sensor

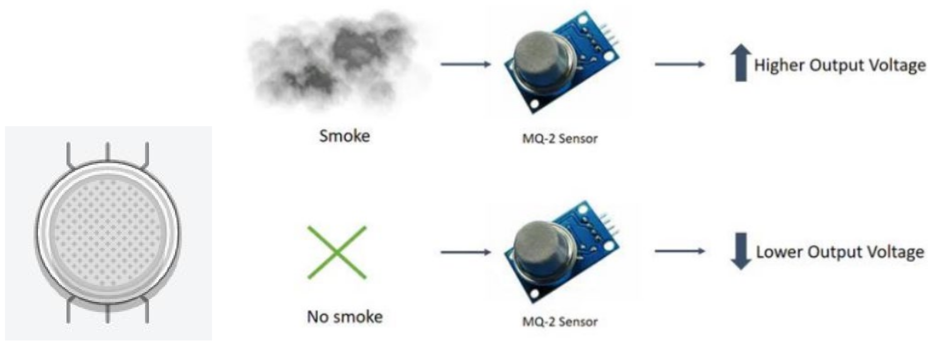

Figure 9: Gas Sensor and Basic Working Principle of the Gas Sensor

The voltage that the sensor outputs changes accordingly to the smoke/gas level that exists in the atmosphere. The sensor outputs a voltage that is proportional to the concentration of smoke/gas. The gas sensor we have used has 4 pins, the connection is shown as follows in Table 4:

Table 4: Connections between Gas Sensor and Arduino Pins

Pins	Connection
A0	Analog pins
D0	Digital pins
VCC	5V
GND	GND

3.3.4 Buzzer

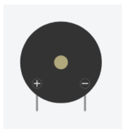

Figure 10: Buzzer Component

A buzzer is an audio signaling device. In Arduino, this device can make sounds of different tones. **tone(buzzer_pin, frequency, duration)** can help to adjust the sound type. One of its pin is connected to GND and the other is connected to the Arduino board.

4 Circuit Design

4.1 Arduino Display

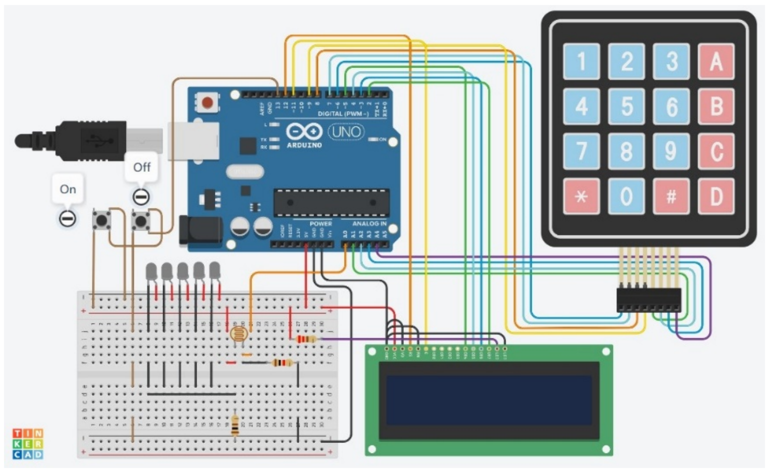

Figure 11: Arduino Display of 4-Digits Password Lock with Light-Control LED Array

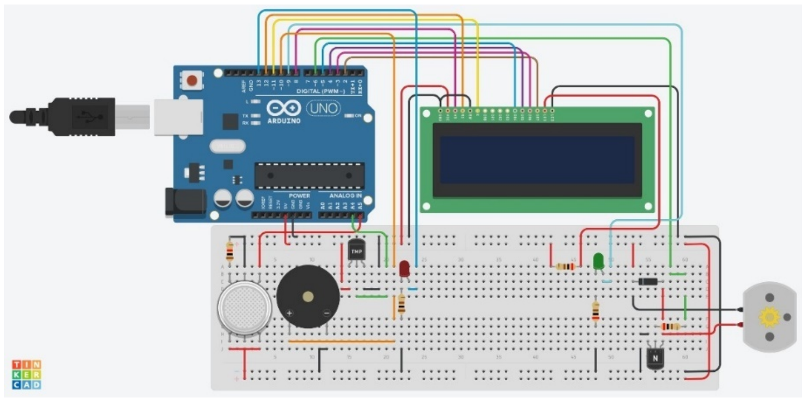
Figure 12: Arduino Display of Automatic Temperature Controller and Fire Alarm

4.2 Circuit Diagrams

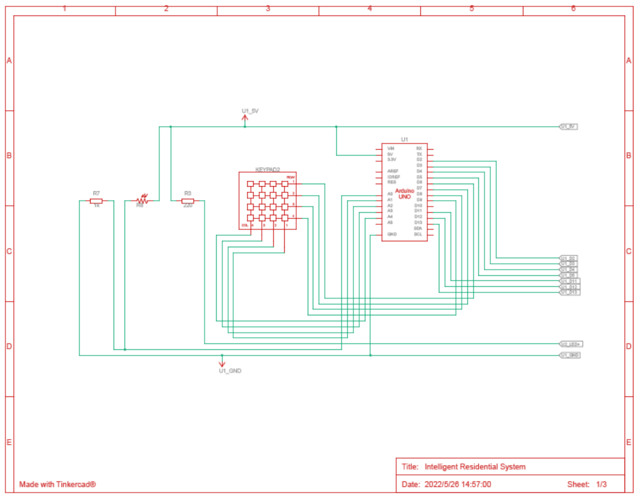

Figure 13: Circuit Diagram of 4×4 Keypad

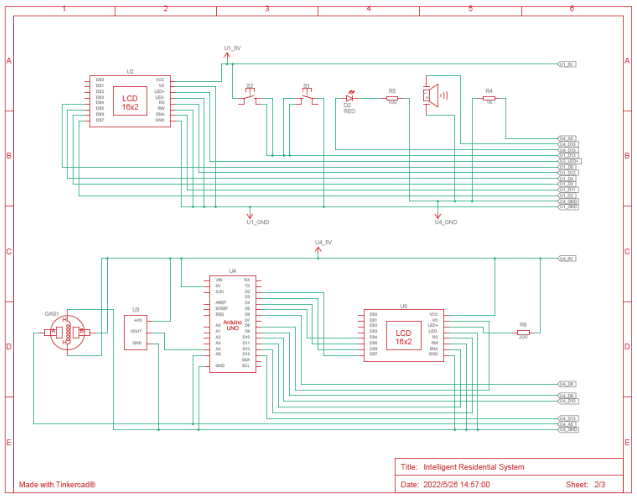

Figure 14: Circuit Diagram of 2 LCDs and Alarm

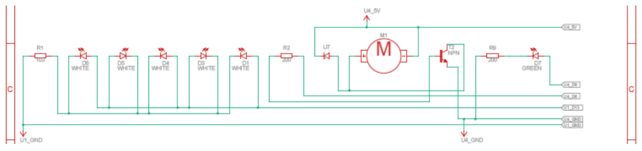

Figure 15: Circuit Diagram of DC Motor

5 Experimental Method

5.1 Flow Chart

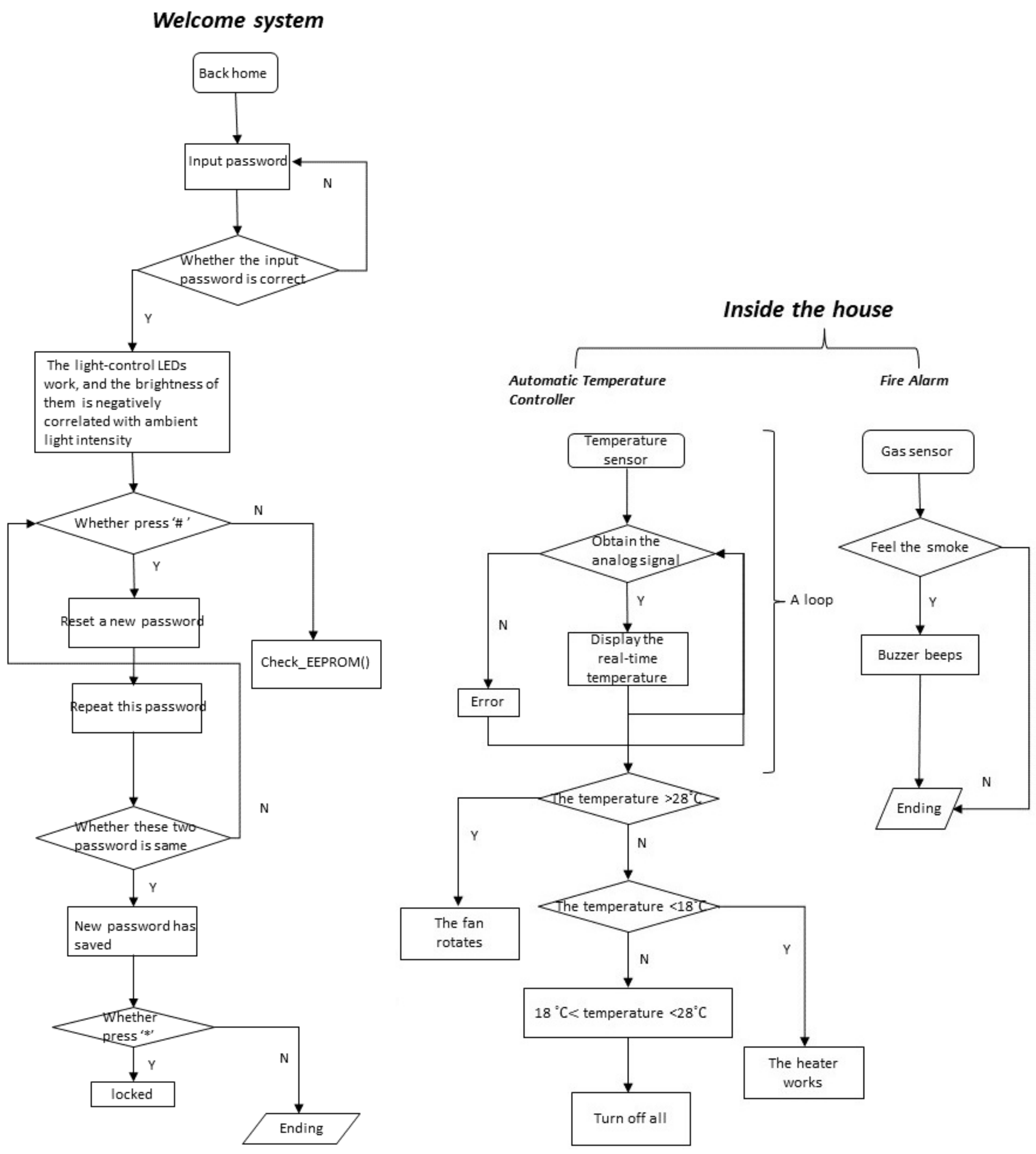

Figure 16: Flow Chart of 2 Subsystems

5.2 Debugging

During the development process we encountered a number of difficulties including, but not limited to, the inability of the LCD display to show real-time temperature changes in a timely manner and the inability of the buzzer to accurately stop beeping as the smoke disappears.

In the process of writing the code for the password lock, I wanted the code to detect if the button was successfully pressed and report an error if it was not. Looking through the documentation, I found that the function collectKey() would do the trick. However, after adding this function to my code, I found that the robustness detection part of the code did not work. Again, I checked the documentation and found that this function is applicable to realistic situations and that the TinkerCAD platform is unable to perform the detection, so the execution ends here. So I deleted the function anyway.

At first I used delay() to pause the program in the hope that the current temperature would be displayed steadily on the LCD. However, I found that sometimes I adjusted the ambient temperature and the system did not recognize it. I presume that the temperature happens to be adjusted during the delay process and therefore the adjustment is not valid. So I used a pseudo-multithreaded approach, making clever use of loops, so that each temperature change could be accurately recognized.

The problem with the buzzer was eventually detected as its ringing time being too long, changing the 'duration' to 1ms allowed it to accurately respond to the disappearance of smoke.

6 Results

Results can be obtained in the simulation and their corresponding operations are as follows.

- Module 1. 4-Digits Password Lock

Initial password was set correctly in Arduino. 4×4 keypad could receive entered digit and function keys "*" and "#" can operate the system as "Lock" and "Password Reset" command. LCD could display basic prompts for each condition, including current password state, welcome message, prompts of correct, incorrect and inconsistent password. LCD could also display visual feedback like reset new password, simulated entered digit and saving process in real time.

- Module 2. Light-Control LED Array

Photoresistor could operate the 5 illuminating LEDs based on light intensity it detected. It would only turn the lights on when natural light gradually becomes dim for the aim of energy-saving. And lights are lit linearly instead of only full brightness and off since it receives analog signals from photoresistor. 2 buttons act as manual switch, which could ignore command from photoresistor control LEDs directly.

- Module 3. Automatic Temperature Controller

Electrical fan (DC motor) and heater (green LED) could work normally according to current temperature detected by sensor and interval rules preset in Arduino. When the room temperature is between 18℃ and 28℃, both appliances are not on. If it becomes <18℃, then electrical fan will open and heater is turned off. If the room is >28℃, then heater will be turned on while fan is turned off. Temperature sensor could detect real time temperature within acceptable range of error and transmit its data to LCD for displaying. LCD is capable of displaying current temperature expressed in Celsius and Fahrenheit form and appliances starting prompt when it reaches preset temperature thresholds.

- Module 4. Fire Alarm

Fire alarm is independent from other subsystems and able to instantly emit auditory feedback and visual signal. Gas sensor can sensitively detect the concentration of harmful gas and trigger buzzer and LED when gas is near it. If gas is moved away from a certain range, gas sensor could correspondingly stop the buzzer. The buzzer acts as a waring alarm, it raises the alarm by emitting loud and continuous the sound of penetrating siren. The light of red LED can travel far and spread people warning messages.

7 Discussion

It is evident from the simulation results of this project that 4 modules can indeed finish requirements based on robust program code as a intelligent residential system. The experimental results meet the objectives as expected, which proves the feasibility, practicability and feasibility of the circuit.

Due to inevitable instability of the simulation program, it is supposed to be noted that after several reliability tests, the simulation program can only run stably and repeatedly under reasonable and natural operating conditions. Multiple improper operations might result in unknown errors. For example, pressing keys or adjusting the environment parameters too fast may cause delay or fail to work. Therefore, in order for the modules to maintain correct output and result, it is recommended to give the system sufficient reaction time between actions and commands.

8 Conclusion

From the whole simulation experiment, conclusion of the Arduino design project is that the total of four active and passive modules could operate stably as expected. Based on robust code programmed in Arduino board, all electronic components used in the project are correctly connected and manipulated in different scenarios. Additionally, the light-control LED array effectively realizes the purpose of energy-saving. We could conclude that it could really embody the idea of sustainability. Hence, these indicates that the intelligent residential system as a whole is consistent with sustainable development and is able to work properly and intelligently as designed.

Through constructing and successfully simulating the above analog circuits, we have gained a better understanding of the use of many electronic components, circuit design and connections. In addition, by addressing errors and resolving possible difficulties that might arise during software simulations, we have improved our problem-solving skills. Therefore, this simulation open project is a meaningful and significant practice.

9 Author List and Distribution

No.	Name	Contribution Description	Percentage (%)	Signature
1	Muyang Li	Coding, Video Recording, Report	38%	李沐阳
2	Tian Li	Video Caption, Report	24%	李天
3	Xiaoou Jiang	Poster Design, Report	38%	姜筱欧

10 References

[1] M. Jourden, “09 TinkerCAD Electrical Gas Sensor”, Component Tutorial.
[2] M. Jourden, “10 TinkerCAD Electrical LCD and Keypad”, Component Tutorial.
[3] M. Jourden, “11 TinkerCAD Electrical Temperature Sensor”, Component Tutorial.
[4] “ The Best Tinkercad Arduino Projects of 2022”. [Online]. Available: https://all3dp.com/2/best-tinkercad-arduino-projects/
[5] “Fire Alarm System Project by Interfacing Arduino with Temperature & Gas Sensor using TinkerCad”. [Online]. Available: https://www.learnelectronicsindia.com/post/fire-alarm-system-project-by-interfacing-arduino-with-temperature-gas-sensor-using-tinkercad

11 Appendix (Source Code in 2 Arduino Boards)

```
/* Source code of the first Arduino Board, which operates the 4-digits
password lock and light-control LED array */

#include <LiquidCrystal.h>
#include <Keypad.h>
#include <EEPROM.h>
// Password length constant
#define P_length 5

// Define each pin of LCD
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
// Define pins of LED and photoresistance
uint8_t led_pin = 13;
uint8_t ldr_pin = A0;
// Define the keypad by two-dimensional array
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] = {6, 7, 8, 9};
byte colPins[COLS] = {A1, A2, A3, A4};
// Define char array of password length for storing data
char Data[P_length];
char Data2[P_length];
char Master[P_length];
// Define data digit, entered password and mode variable
byte data_count = 0;
char key;
char last_press_key;
byte mode = 0;
// Setup first-time password
char Fst_P[] = {'1','1','1','1'};
// Initialize the keypad
Keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

void setup(){
// Enable serial communication
  Serial.begin(9600);
```

```
// Check EEPROM
  Check_EEPROM();
// Set up the number of columns and rows on the LCD.
  lcd.begin(16, 2);}

void loop(){
// Read the light intensity from the photoresistor and assign it to
light
int light = analogRead(ldr_pin);
// Scan the keyboard for the value of key
key = keypad.getKey();
if(key){
// Save the key as the last pressed key
    last_press_key = key;
// Serial port outputs key value
    Serial.println(key);}
    switch(mode){
// Mode 0: Waiting for a password to enter
case 0:{
    lcd.setCursor(1,0);
    lcd.print("Enter Password");
    if (key && key != '#') {
// "collectKey" saves the current key in the Data array
        collectKey();}
    if(data_count == P_length-1){
        lcd.clear();
// Verify correctness of entered password
// Correct: display "Welcome Back", switch mode to 3
        if(!strcmp(Data, Master)) {
            lcd.setCursor(2, 0);
            lcd.print("WELCOME BACK");
            lcd.setCursor(4, 1);
            lcd.print("==^. .^==");
            delay(2000);
            mode = 3;}else{
// Wrong: display "Incorrect" message
            lcd.setCursor(2, 0);
            lcd.print("INCORRECT !");
            lcd.setCursor(4, 1);
            lcd.print("PASSWORD");
            delay(2000);}
        delay(1000);
// Clear LCD and Data array
        lcd.clear();
        clearData();}
  break;}
case 1:{
// Mode 1: Reset new password
    lcd.setCursor(0,0);
    lcd.print("Set New Password");
    if (key && key != '#') {
// Save password in Data
    collectKey();}
    if(data_count == P_length-1){
        lcd.clear();
// Switch mode to 2
        mode = 2;
        for(int i = 0; i < P_length; i = i + 1){
// Create a copy of the password Data2 and clear Data
        Data2[i] = Data[i];}
        clearData(); }
  break;}
case 2:{
// Mode 2: New password - confirmation
    lcd.setCursor(0,0);
    lcd.print("Password Again");
    if (key && key != '#'){collectKey();}
// If the password bit is 4, verify if two new passwords are the same
    if(data_count == P_length-1){
        if(!strcmp(Data, Data2)){
// Consistent: clear LCD, and display the new password
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("New Password is ");
        lcd.setCursor(4, 1);
        lcd.print(Data);
        delay(200);
        lcd.clear();
        lcd.setCursor(4, 0);
        lcd.print("Saving...");
// Simulate save progress
        for(int i =0; i <= 100; i = i+10){
            lcd.setCursor(4, 1);
            lcd.print(i);
            lcd.setCursor(7, 1);
            lcd.print("%");
            delay(200);}
// The new password overwrites the memory and is read by the Master
        EEPROM.put(0, Data);
        EEPROM.get(0, Master);
        delay(500);}else{
// Inconsistent: display "Not Match" message
        lcd.clear();
        lcd.setCursor(4, 0);
        lcd.print("Password");
        lcd.setCursor(3, 1);
        lcd.print("Not Match!");
        delay(200);}
// Switch mode to 3 and clear LCD
        mode = 3;
        clearData();
        lcd.clear();}
    break;}
case 3:{
// Mode 3: Unlocked
// If the last key was #, switch the mode to 1
    if(last_press_key == '#' ){mode = 1;}
// If the last key was *, switch the mode to 0, clear LCD and lock the
system
    if(last_press_key == '*' ){
        mode = 0;
        lcd.clear();
        lcd.setCursor(4,0);
        lcd.print("LOCKED");
        delay(200);}
// Display "Unlocked" message
    lcd.setCursor(4,0);
    lcd.print("UNLOCKED");
// Environment light
// The map function establish a mapping and assigns values to light in
reverse direction according to the ambient light
    light = map(light,0,1023,255,0);
// Light up LED
    analogWrite(led_pin, light);
    break;}}
    }
```

```
void collectKey(){
    Data[data_count] = key;
// LCD outputs * representing entered password
    lcd.setCursor(4+data_count,1);
    lcd.print("*");
    data_count++;}

void clearData(){
// Clear password stored in Data array
    while(data_count !=0){
        Data[data_count--] = 0;
        }
}

void Check_EEPROM(){
// EEPROM.get() function allows the user to retrieve multiple bytes of
data at once
// This allows us to store floating point or integer data with a
decimal point and other data types into EEPROM
    EEPROM.get(0, Master);
// If master = 0000, there is no stored password,
// The initial password 1111 is stored and read by Master
    if(Master[0] == 0 && Master[1] == 0 && Master[2] == 0 && Master[3]
== 0){
// check if EEPROM have store password
        Serial.println("No EEPROM Password Found");
// if not found will overwrite EEPROM the first password
       EEPROM.put(0, Fst_P);
       EEPROM.get(0, Master);}
}



/* Source code of the second Arduino Board, which operates the
automatic temperature controller and fire alarm */

#include <LiquidCrystal.h>
// Set the pins of temperature sensor, smoke sensor, fan, heater,
buzzer and LED
uint8_t tmp_pin = A4, smoke_sensor_pin = A5,
Fan_pin = 6, Heater_pin = 9, buzzer_pin = 10, led_pin = 13;
float temp, MinTemp = 18, MaxTemp = 28, volts, celcius, fahrenheit;
// Define each pin of LCD
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
int analogSensor, Temp = 10000, laTemp1 = 9999, laTemp2 = 10001;
String str1, str2, str3, str4;
unsigned long currentTime = 0, previousTime = 0;

void setup(){
// Enable serial communication
    Serial.begin(9600);
// Set pin input/output mode
    pinMode(led_pin, OUTPUT);
    pinMode(buzzer_pin, OUTPUT);
    pinMode(smoke_sensor_pin, INPUT);
    pinMode(tmp_pin, INPUT);
    lcd.begin(16, 2);
    pinMode(8, OUTPUT);
    pinMode(Heater_pin, OUTPUT);
    pinMode(Fan_pin, OUTPUT);}
```

```
void loop(){
// Read the gas concentration from the smoke alarm
analogSensor = analogRead(smoke_sensor_pin);
// If gas concentration >100: Turn on the LED and start the buzzer
if (analogSensor > 100){
    digitalWrite(led_pin, HIGH);
    tone(buzzer_pin, 800, 1);}else{
    digitalWrite(led_pin, LOW);}
// Read the temperature from the temperature sensor
Temp = analogRead(tmp_pin);
if(laTemp2 == Temp){
    return;}
// Calculate voltage
volts = (Temp / 965.0) * 5;
// Calculate Celsius degree
celcius = (volts - 0.5) * 100 - 5;
// Calculate Fahrenheit degree
fahrenheit = (celcius * 9 / 5 + 32);
// Define strings which are about to be displayed on LCD
str1 = String("Tempreture is:");
str2 = String(String(celcius) + String("*C") + String(fahrenheit) +
String("*F"));
if(celcius > MaxTemp){
    str3 = "Day is hot";
    str4 = "Turn on fan";
// Turn on electric fan and turn off the heater
    analogWrite(Fan_pin, 255);
    digitalWrite(Heater_pin, LOW);}
    else if(celcius < MinTemp){
    str3 = "Day is cold";
    str4 = "Turn on heater";
// Turn on the heater and turn off electric fan
    digitalWrite(Heater_pin, HIGH);
    analogWrite(Fan_pin, 0);}else{
    str3 = "Temp is normal";
    str4 = "Turn off all!";
    digitalWrite(Heater_pin, LOW);}
if(laTemp1! = Temp){
// Display the temperature value
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print(str1);
    lcd.setCursor(0,1);
    lcd.print(str2);
    laTemp1 = Temp;}
currentTime = millis();
// Use “pastTime” to distinguish time updates
if(currentTime - previousTime > 500){
    previousTime = currentTime;
// Display switch prompt
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print(str3);
    lcd.setCursor(0,1);
    lcd.print(str4);
    laTemp2 = Temp;
    }
}
```