



SMART HOME REPORT

CSE252 Control Systems



Submitted in May, 2024



Faculty of Engineering Mansoura University

Mechatronics Engineering Department

CSE252 Control Systems

Smart Home Report

Submitted in May, 2024











Under the supervision of:

Dr. Amira Yasin

TA. Menna Sedik

TA. Dalia Tarek

Submitted by:

<i>Student Name</i>	<i>Student ID</i>
 Hossam Medhat Shokry Almoursy	806092221
 Mohaned Mohammed Elsayed	805493864
 Sherif Ibrahim Mohammed Mansour	806095628
 Mahmoud Mohammed Ibrahim	806143446
 Sohip Ibrahim Abd Elmoaty	806096245
 Ahmed Mohammed Mohamed Hammad	<u>805494115</u>
 Mohammed Ibrahim Mohammed	<u>805494083</u>
 Mohammed Rabia Ahmed Ibrahim	<u>808607448</u>
 Abdelrahman Ahmed Mohammed	<u>803054561</u>
 Ahmed Yasser Ahmed Omar	<u>808976446</u>

ABSTRACT

This report aims to display the implementation of a simple smart home, that's a simulation of what a real smart home could be.

It also provides a snippet on the importance of smart homes, their advantages and possible disadvantages.

The implementation has been discussed in terms of software (using Arduino software) and hardware.

The implementation focuses on designing a smart home, that starts with a garden gate which is controlled by an IR sensor, followed by a keypad to insert a password.

The password is provided through an LCD. In case of inserting a wrong password, a loud noise would be triggered by a buzzer, and an error message “*wrong password*” is displayed on the LCD. However, in case of inserting the correct password, a faint sound would be triggered by the buzzer, and a welcome message “*Welcome Home*” is displayed on the LCD.

A few more sensors are implemented as well such as: (LM35, LDR)

The LM35: measures the temperature, where if the temperature is more than 25°C, a fan will work, and if the temperature is more than 35°C, the door will automatically open, and buzzer will work loudly. This can be stopped using a push button.

LDR: measures light intensity, where if the light intensity is low, LEDs will turn on automatically.

TABLE OF CONTENTS

Table of Contents

Introduction.....	1
The Software Behind A Smart Home.....	2
<i>The Algorithm of The Smart Home.....</i>	<i>2</i>
<i>Libraries Used</i>	<i>3</i>
<i>Library for Servo motor</i>	<i>3</i>
<i>Library for Keypad</i>	<i>4</i>
<i>Library for LCD_I2C.....</i>	<i>5</i>
Defining Variables	6
<i>Defining Constant Pins.....</i>	<i>6</i>
<i>Defining Variable Pins</i>	<i>7</i>
Setup Pins.....	8
Loop Part.....	8
<i>Explanation in of The Code in C Programming Language</i>	<i>8</i>
<i>Explanation of The Used Functions.....</i>	<i>12</i>
The Hardware Implementation of A Smart Home	14
The Used Hardware Components.....	14
<i>LEDs</i>	<i>15</i>
<i>Servo Motor</i>	<i>16</i>
<i>Keypad</i>	<i>16</i>
<i>LCD_I2C.....</i>	<i>16</i>
<i>LM35.....</i>	<i>17</i>
<i>LDR.....</i>	<i>17</i>
<i>Buzzer</i>	<i>17</i>
<i>Push Button.....</i>	<i>18</i>
Designing A Smart Home in Solid.....	19
Introduction on The Designed Parts	19
<i>Front</i>	<i>20</i>
<i>Side</i>	<i>20</i>
<i>Top.....</i>	<i>20</i>
<i>Garage</i>	<i>21</i>
<i>After finished</i>	<i>21</i>
Discussion & Conclusion	22
References	24

INTRODUCTION

Nowadays, it is difficult to find a house that does not have a home automation system in effect. Smart home systems have gained popularity in recent decades because they improve comfort and quality of life (S & B, n.d.).

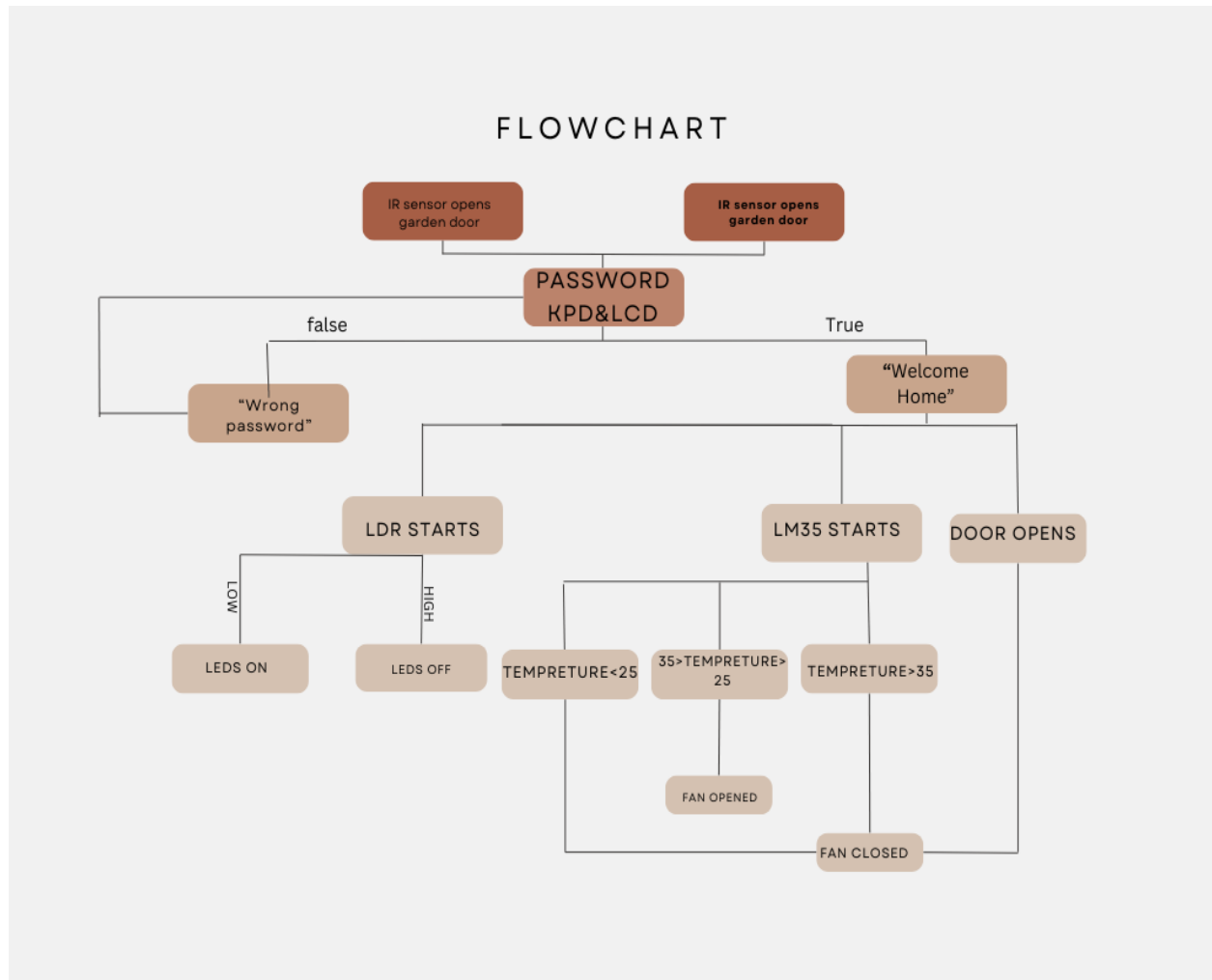
the concept of smart homes has gained significant traction, driven by advancements in technology and the growing demand for convenience, efficiency, and security in residential living. A smart home leverages interconnected devices, sensors, and actuators to automate tasks, monitor environmental conditions, and enhance overall quality of life for occupants. With the advent of platforms like Arduino, creating DIY smart home systems has become more accessible and affordable, empowering individuals to customize and tailor solutions to their specific needs and preferences.

This report explores the design, development, and implementation of a smart home system using Arduino microcontrollers. The project aims to showcase the capabilities of Arduino technology in enabling home automation and IoT (Internet of Things) applications, while also addressing key considerations such as sensor integration, actuator control, user interface design, and security measures.

The introduction of this report provides an overview of the motivation behind the project, highlighting the increasing interest in smart home technology and the potential benefits it offers to homeowners. It sets the stage for the subsequent sections, which delve into the details of the smart home system design, hardware components, software implementation, and practical considerations for deployment and operation.

Throughout the report, emphasis is placed on practical experimentation, hands-on learning, and the iterative development process involved in creating a functional smart home prototype. By documenting the project's progress, challenges encountered, and lessons learned, this report aims to serve as a valuable resource for individuals interested in exploring Arduino-based smart home projects and advancing their skills in electronics, programming, and home automation technology.

The Algorithm of the Smart Home:




Libraries Used:



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

i. Library for Servo Motor:



```
#include <Servo.h>
```

The **<Servo.h>** library is a commonly used library in Arduino programming for controlling servo motors. Servo motors are widely used in various projects, such as robotics, remote-controlled vehicles, animatronics, and more, because of their precise control over angular position.

This library provides a simple and convenient way to interface with servo motors using Arduino boards. It abstracts away the complexity of generating the precise PWM (Pulse Width Modulation) signals required to control servo motors, making it easier for developers to focus on their project logic rather than low-level details.

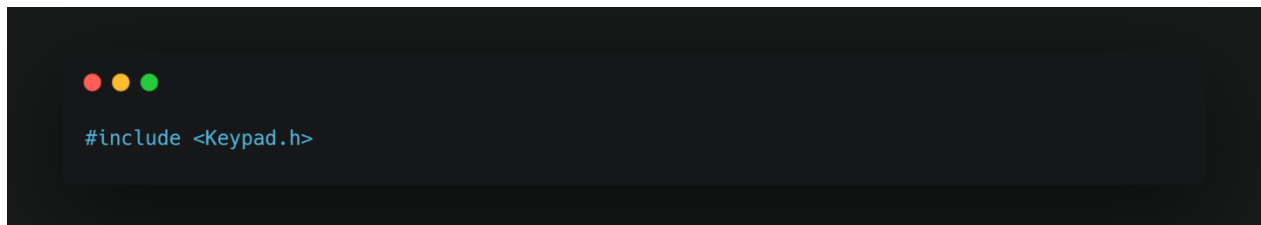
Key features of the **Servo.h** library include:

1. **Simple API:** The library provides easy-to-use functions for attaching a servo motor to a specific pin, setting its angle, and detaching it when not in use.
2. **Precise Control:** Servo motors require precise control over their position, which is achieved by sending PWM signals with specific pulse widths. The

library handles the generation of these signals, ensuring accurate positioning of the servo motor.

3. **Multiple Servos:** It supports control of multiple servo motors simultaneously, allowing developers to create complex motion systems with multiple degrees of freedom.
4. **Compatibility:** The **Servo.h** library is compatible with various Arduino boards, making it versatile and widely applicable across different projects

ii. Library for Keypad:



The **Keypad.h** library is a popular library used in Arduino projects for interfacing with matrix keypads. Matrix keypads are commonly found in various electronic devices such as calculators, digital locks, and security systems. They allow users to input data or commands by pressing combinations of keys arranged in rows and columns.

The **Keypad.h** library simplifies the process of reading keypresses from a matrix keypad, providing an intuitive interface for Arduino programmers. It abstracts away the complexity of scanning the keypad matrix and debouncing keypresses, allowing developers to focus on implementing the desired functionality of their projects.

Key features of the **Keypad.h** library include:

1. **Support for Matrix Keyboards:** The library supports matrix keypads of various sizes, from small 2x2 keypads to larger ones with more rows and columns.

2. **Flexible Configuration:** Developers can easily configure the library to match the specific wiring and layout of their keypad, including the number of rows and columns, key mappings, and debounce settings.
3. **Event-Driven Programming:** The library supports event-driven programming, allowing developers to define callback functions that are executed when specific keys are pressed or released.
4. **Efficient Key Scanning:** The library implements efficient algorithms for scanning the keypad matrix, minimizing processing overhead and ensuring responsive user interaction.

iii. Library for LCD_I2C:



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

The **LiquidCrystal_I2C.h** library is an extension of the standard LiquidCrystal library for Arduino that adds support for LCD displays using the I2C communication protocol. LCD displays are commonly used in Arduino projects for displaying text and graphical information.

The I2C protocol, also known as TWI (Two-Wire Interface), allows for communication between microcontrollers and peripheral devices using only two wires (SDA for data and SCL for clock). Using an I2C interface reduces the number of required pins on the Arduino board, which is beneficial especially when working with projects that have limited available pins.

The **LiquidCrystal_I2C.h** library simplifies the process of interfacing with I2C-enabled LCD displays, providing functions for initializing the display, writing text to specific positions, controlling the cursor, and more.

Key features of the **LiquidCrystal_I2C.h** library include:

1. **Easy Initialization:** The library makes it easy to initialize the LCD display with the appropriate I2C address and dimensions.
2. **Flexible Text Display:** Developers can easily write text to the LCD display at specific positions, allowing for the creation of custom user interfaces and menus.
3. **Cursor Control:** The library provides functions for controlling the position and visibility of the cursor on the LCD display.
4. **Backlight Control:** If the LCD display has a backlight, the library typically includes functions for controlling its brightness or turning it on/off.

Defining Variables:

i. Defining Constant Pins:

```
// Constants
const int thresholdTemp = 25; // Temperature threshold in Celsius
const int highTemp = 35; // High temperature threshold in Celsius
const int servoPin = 9; // Servo motor pin
const int servoPin_2 = 6; // Servo motor pin
int irSensor1 = 2; // IR sensor 1 pin
int irSensor2 = 3; // IR sensor 2 pin
const int buzzerPin = 4; // Buzzer pin
const int greenLEDPin = 5; // Green LED pin
const int tempSensorPin = A1; // LM35 temperature sensor pin
const int fanPin = 8; // DC motor (fan) pin
const int ldrPin = A0; // LDR sensor pin
const int led1Pin = 10; // LED 1 pin
const int led2Pin = 11; // LED 2 pin
const int buttonPin = 7; // Push button pin
```

Avoiding Errors: Using constants helps prevent accidental changes to pin assignments during the development process. If a pin assignment needs to be changed, developers can update the constant value once, rather than having to find and update every occurrence of the pin number in the code, which reduces the risk of errors.

ii. Defining Variable Pins:

```
// Variables
bool isDoorOpen = 0;
bool isAlarmOn = 0;

int temp=0;

// Initialize the servo
Servo doorServo;
Servo doorServo_2;

// Initialize the LCD
LiquidCrystal_I2C lcd(0x27, 16, 2);

// Keypad setup
const byte ROWS = 2; // Two rows
const byte COLS = 2; // Two columns
char keys[ROWS][COLS] = {
  {'1','2'},
  {'4','5'}
};
byte rowPins[ROWS] = {12, 13}; // Connect to the row pinouts of the keypad
byte colPins[COLS] = {A2, A3}; // Connect to the column pinouts of the keypad
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

char Password[]="2141";
char passwordIndex = sizeof(Password)-1;
char EPassword[sizeof(Password)-1];
char counter =0;
char Flag =0;
```

We defined these variables to control hardware parts we used in project as door in garden (**doorServo**) and another one for Home's door (**doorServo_2**), also we use variables to check if door is open or not.

Then we make some variables for Keypad as rows and columns we use ,and array which have the correct password .

Setup Pins:

```
void setup() {
  pinMode(irSensor1, INPUT);
  pinMode(irSensor2, INPUT);
  pinMode(buzzerPin, OUTPUT);
  pinMode(greenLEDPin, OUTPUT);
  pinMode(fanPin, OUTPUT);
  pinMode(led1Pin, OUTPUT);
  pinMode(led2Pin, OUTPUT);
  pinMode(buttonPin, INPUT_PULLUP);
  doorServo.attach(servoPin);
  doorServo_2.attach(servoPin_2);
  lcd.init(); // Initialize the LCD
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("Enter Password");
  lcd.setCursor(0,1);
}
```

We defined the variables we use if it gives output signals as (LEDs , buzzer,fan) and we defined some as an input pins like Sensors(IR,LM35,LDR) and Pushbutton.

Loop Part:

i. Explanation in of The Code in C Programming Language:

```
void loop() {

  int value1 = digitalRead(irSensor1) ;
  int value2 = digitalRead(irSensor2);
  // Door control logic
  if ((value1 == HIGH )&&(value2==LOW)) {
    doorServo.write(180); // Open the door

  } else if ((value1== LOW) && (value2 == HIGH )) {
    doorServo.write(0); // Close the door

  }
}
```

We start loop with garden task as we check if person outdoor or indoor , that we use 2 IR Sensors , first one in garden door check if value is HIGH and another LOW , and if you enter the door value out will be LOW and the second one will be HIGH.

Then we will go to function **Accesshome()**;, we will talk about this after that.

```
// Temperature control logic
if(temp==1){
  int tempReading = analogRead(tempSensorPin);
  float voltage = tempReading * 5.0 / 1024.0;
  float temperature = voltage * 100; // Convert voltage to temperature
```

This check temp if we access home successfully, we start by function that convert analog – From LM35 - to digital by this equation.

```
if (temperature > thresholdTemp&&temperature<highTemp) {
  lcd.setCursor(0,0);
  digitalWrite(fanPin, HIGH); // Turn on the fan
  lcd.print("temp: ");
  lcd.print(temperature);
  lcd.print(" C");
}
```

We make that condition that temperature between 25 and 35 , the fan starts .

```
if (temperature > highTemp ) {  
  lcd.setCursor(0,0);  
  digitalWrite(buzzerPin, HIGH);  
  // Turn on the high temperature alarm  
  digitalWrite(fanPin, LOW); // Turn on the fan  
  lcd.print("temp: ");  
  lcd.print(temperature);  
  lcd.print(" C");  
  digitalWrite(4,HIGH);  
}
```

We make that condition that temperature more than 35 , door open automatically and buzzer .

```
if (digitalRead(buttonPin) == HIGH) {  
  lcd.setCursor(0,0);  
  digitalWrite(4,LOW);  
  digitalWrite(buzzerPin, LOW); // Turn off the high temperature alarm  
  digitalWrite(fanPin, LOW); // Turn on the fan  
  lcd.print("temp: ");  
  lcd.print(temperature);  
  lcd.print(" C");  
}
```

We make that condition that if we press push button , buzzer stop and fan.



```
// LDR and LEDs brightness control logic
int ldrValue = analogRead(ldrPin);
int ledBrightness = ldrValue;
lcd.setCursor(0, 1);
if(ledBrightness<20)
{
  digitalWrite(led2Pin, HIGH);
  digitalWrite(led1Pin,LOW);
  lcd.print("LEDs: ");
  lcd.print(ledBrightness);
  delay(2000);
}
else {
  digitalWrite(led1Pin, HIGH);
  digitalWrite(led2Pin,LOW);
  lcd.print("LEDs: ");
  lcd.print(ledBrightness);
  delay(2000);
}
```

This part we make to use LDR and measure Light intensity, and if the intensity is low , LEDs will open automatically .

ii. Explanation of The Used Functions:

```
void AccessHome ()
{
    char customKey = keypad.getKey(); // Check and Store Pressed Keys

    if ((customKey>'0')&&(customKey != '5')) // Condition To Enter Password
    {
        lcd.print('*'); // Print Star When We Enter Our Password
        EPassword[counter] = customKey; // Store Our Password
        counter++;
    }
    else if (customKey == '5') // When We press D We make Sure We Enter A password
    {
        if(counter == passwordIndex) // if We Enter The Number Of Password Numbers is true
        {
            for(char counter1 = 0;counter1<passwordIndex;counter1++)
            {
                if(EPassword[counter1] == Password[counter1]) // Check if The Enter Password Numbers
is True
                {
                    Flag++; // Count Number of True Condition
                }
            }
            if(Flag == passwordIndex)
            {
                PrintRightPassword();

            }
            else
            {
                PrintWrongPassword();
            }
        }
        else
        {
            PrintWrongPassword();
        }
    }
}
```

This function that check password if correct or not , and we make submit the password by pressing number “5” .

```

void PrintRightPassword()
{
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Welcome Home  ");
    digitalWrite(5,HIGH);
    digitalWrite(4,LOW);
    temp=1;
    doorServo_2.write(180);
    delay(5000);
    doorServo_2.write(0);
    counter = 0;
    Flag = 0;
}

```

This function in else if condition if the password is correct will print in LCD “welcome Home” , door open , green Led opens and Red Led closed .

```



void PrintWrongPassword()
{
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Wrong Password");
    digitalWrite(4,HIGH);
    delay(2000);
    counter = 0;
    Flag = 0;
    lcd.setCursor(0,0);
    lcd.print("Enter Password");
    lcd.setCursor(0,1);
}






```

This function in else if condition if the password is correct will print in LCD “wrong password ” , green Led closed and Red Led opened .

The Used Hardware Components:

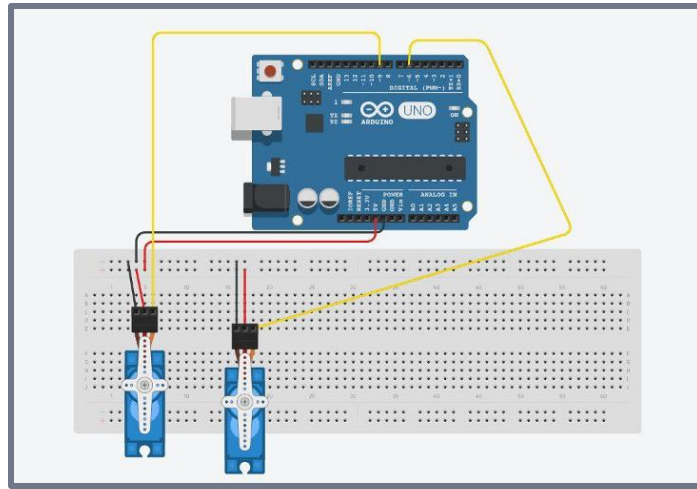
i. Components List:

Component	Description	Function
	<i>Servo Motor:</i> An electric motor that converts electrical power into mechanical power.	Opens and closes garden door and main home door
	<i>Keypad:</i>	We use it to write password
	<i>LCD_12C:</i> A display module that consists of a classic parallel LCD and an I2C LCD adapter.	We use it to provide password as '*' and provide words with user

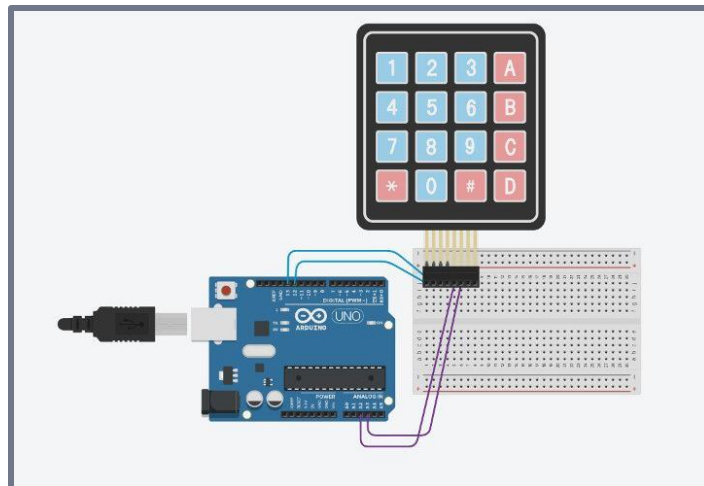
	<p>LEDs:</p> <p>DC forward current: 25 mA (max). Forward voltage (typical): 1.95 V.</p>	<p>We use them in status as closed or opened with green and red . Also, when there is no intensity in LDR home LEDs opened.</p>
	<p>LM35:</p> <p>A temperature sensor that outputs an analog signal which is proportional to the instantaneous temperature.</p>	<p>We use it to measure temperature</p>
	<p>LDR (Light Dependent Resistor):</p> <p>Its resistance decreases according to an increase in the intensity of light.</p>	<p>We use it to measure Light intensity</p>
	<p>Buzzer:</p> <p>Creates an audible tone under the influence of an applied external voltage.</p>	<p>Makes sounds in different status</p>
	<p>Push Button:</p> <p>Mechanical device to control an electrical circuit (close or open an electrical circuit by pressing the switch).</p>	<p>We use it to stop service condition as closing buzzer</p>

ii. Connections:

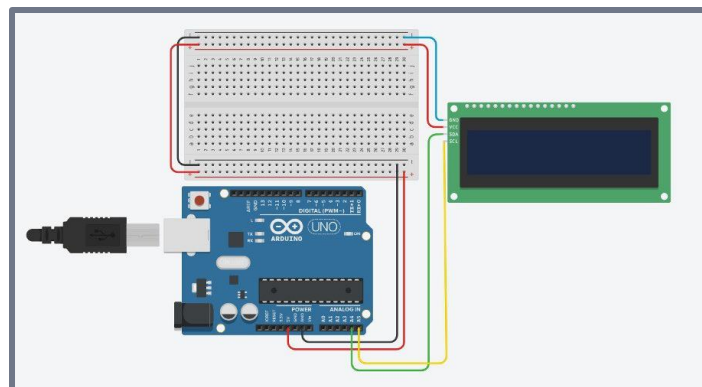
- **Connecting the Servo Motor:**



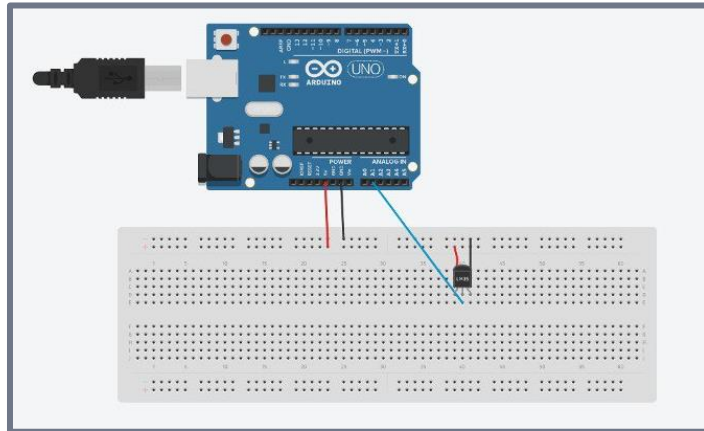
- **Connecting the Keypad:**



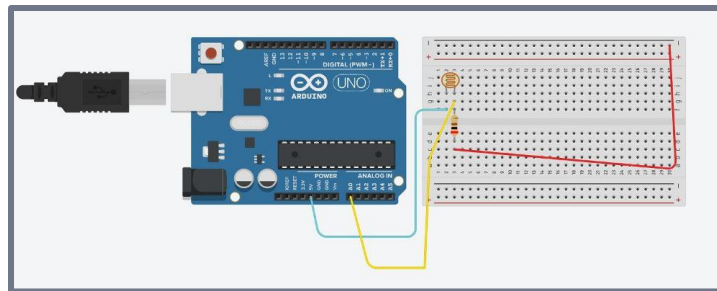
- **Connecting the LCD_12C:**



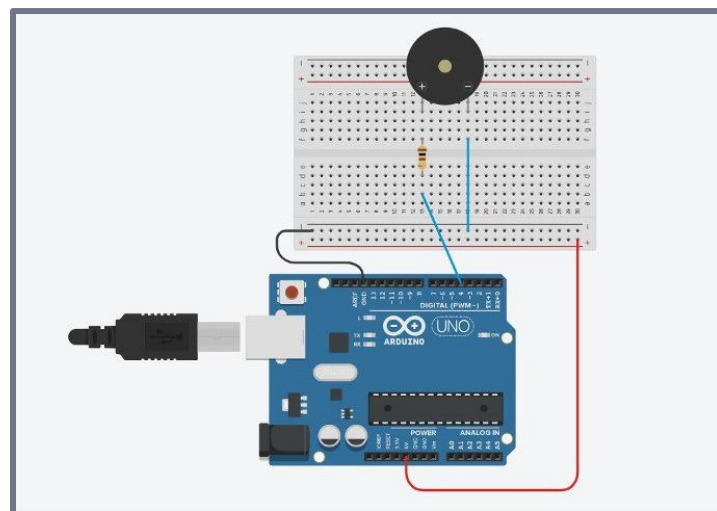
- **Connecting LM35:**



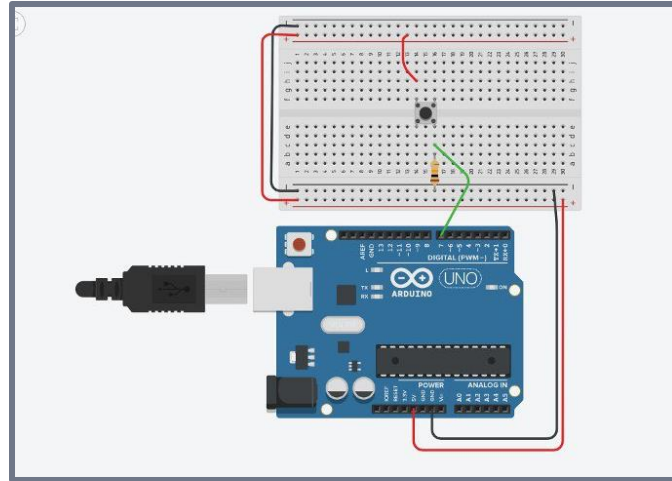
- **Connecting the LDR:**



- **Connecting the Buzzer:**



- **Connecting the Push Button:**



Introduction on The Designed Parts:

We started with a design in a paper and we started to make this design in SOLID WORKS program .

After we finished this design we went to workshop which prints designs in CNC machine in thin wood and cut it into parts we did it.

It maquette is a scaled-down model that showcases the various mechanical components of the smart home system. It provides a tangible representation of the physical infrastructure and demonstrates how the different elements interact with each other.

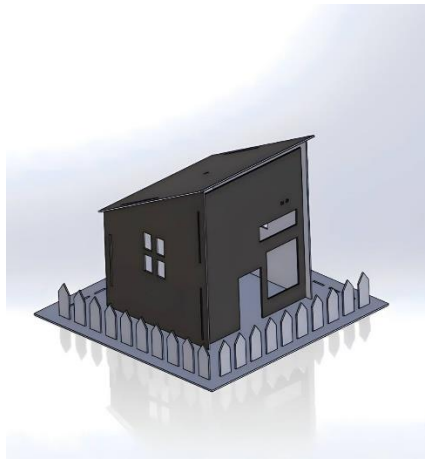
During the designing progress some considerations were taken into account as :

1-Component Placement: Careful attention was given to the dimensions and positioning of the components within the maquette. Specific provisions were made, such as creating a designated hole for the LDR sensor to showcase its capability of detecting environmental changes and triggering appropriate responses. Additionally, space was allocated for an LCD display to provide users with relevant messages, a keypad for entering the home password, and a designated location for a fan that activates when the temperature exceeds 35 degrees.

2- Structural Framework: A 35cm*47cm framework was created using wooden sheets, providing stability and support for the various components. The framework was designed to mimic the layout and structure of the real smart home system.

3- Doors and Windows: The maquette featured working doors and windows that could be opened and closed manually. These elements were designed to replicate the functionality of the real doors and windows found in a smart home, showcasing their integration with the overall system.

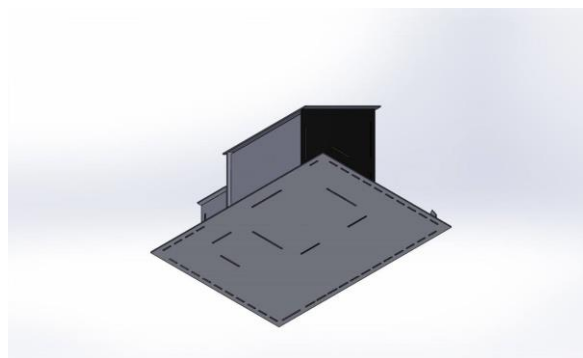
i. Front:



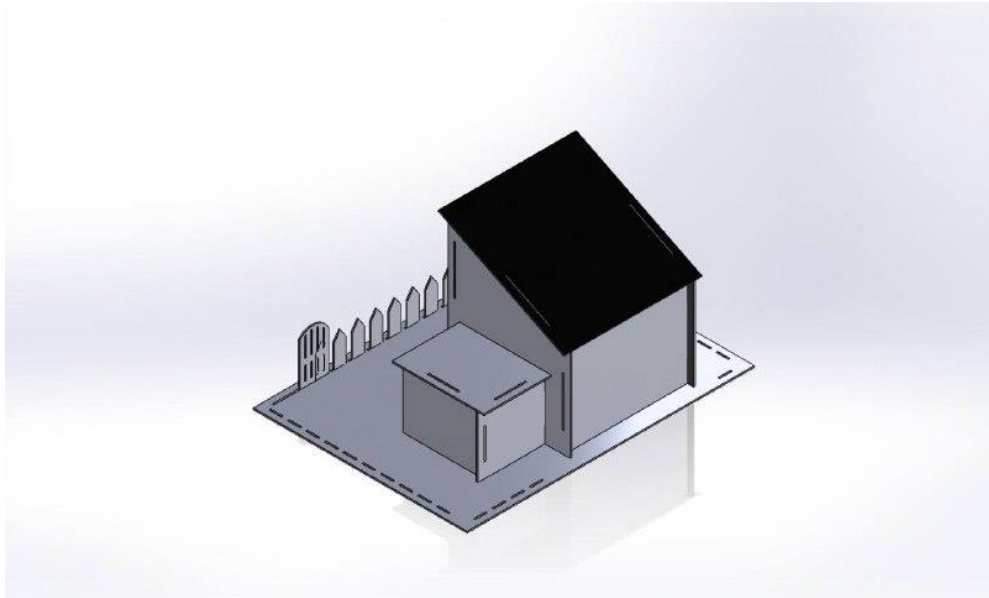
ii. Side:



iii. Top:



iv. Garage:



v. After Finished:



Discussion:

1. **Functionality:** Determine the core functionalities of the smart home system using Arduino. This could include automated lighting, climate control, security features, and energy management.
2. **Arduino Board Selection:** Choose an appropriate Arduino board based on the project requirements. For example, Arduino Uno for simpler projects or Arduino Mega for more complex systems requiring multiple sensor and actuator connections.
3. **Sensor Integration:** Integrate sensors compatible with Arduino to collect environmental data. Use sensors like DHT11/DHT22 for temperature and humidity, PIR motion sensors for security, LDR for light sensing, and LM35 for temperature sensing.
4. **Actuator Control:** Utilize Arduino to control actuators such as servos, motors, and relays for performing actions based on sensor data or user input. For instance, controlling lights, fans, door locks, or window blinds.
5. **User Interface:** Develop a user interface for controlling and monitoring the smart home system using Arduino. This could involve creating a simple LCD menu interface, implementing button controls, or integrating with a smartphone app via Bluetooth or Wi-Fi.
6. **Automation Logic:** Implement automation logic using Arduino programming to make the system smart. Use conditional statements and timers to automate tasks based on sensor readings, schedules, or user-defined preferences.
7. **Security:** Implement security measures in Arduino-based smart home projects to safeguard against unauthorized access. This may involve using secure communication protocols, encryption techniques, and user authentication methods.

Conclusion:

In conclusion, an Arduino-based smart home project offers the flexibility and affordability to create a customizable and DIY solution for home automation. By leveraging Arduino microcontrollers, sensors, and actuators, it's possible to design a smart home system tailored to specific needs and preferences.

DISCUSSION & CONCLUSION

However, successful implementation of an Arduino-based smart home project requires careful planning, experimentation, and programming skills. It's essential to consider factors like hardware compatibility, power consumption, and scalability to create a robust and efficient system.

In summary, Arduino-based smart home projects empower individuals to explore the exciting possibilities of home automation while gaining valuable experience in electronics, programming, and IoT (Internet of Things) technology. With creativity and innovation, Arduino enthusiasts can create smart homes that enhance comfort, convenience, and security for themselves and their families.

REFERENCES

S, M., & B, M. K. (n.d.). Smart Home Automation System. *International Journal on Integrated Education*, 2(5), 222–229.

1. **Official Arduino Website:** <https://www.arduino.cc/>

- The official website provides a wealth of information, including documentation, tutorials, project ideas, and forums.

2. **Arduino Forum:** <https://forum.arduino.cc/>

- A community-driven forum where you can ask questions, share projects, and interact with other Arduino enthusiasts.

3. **Arduino Datasheets:**

- ATmega328P Datasheet: <https://www.digipart.com/>
- Arduino Uno Datasheet: [UNO R3 | Arduino Documentation](#)

Datasheet for the microcontroller used in Arduino Uno and many other Arduino boards, providing comprehensive information on features, registers, and electrical characteristics.

Sensor Resources:

LM35 Temperature Sensor Datasheet: <https://www.alldatasheet.com/datasheet-pdf/pdf/517588/TI1/LM35.html>

Datasheet for the LM35 precision temperature sensor, providing details on specifications, electrical characteristics, and application circuit examples.

LDR (Light Dependent Resistor)

Guide<https://www.alldatasheet.com/datasheet-pdf/pdf/640530/ETC2/LDR-03612.html>

A guide explaining how LDRs work, how to use them in circuits, and example projects.

IR Sensor (Infrared Sensor) Guide: <https://www.alldatasheet.com/datasheet-pdf/pdf/202817/KODENSHI/KME-M002C.html>

A tutorial on IR sensors, including how they work, interfacing with Arduino, and example projects like obstacle detection.