A PROJECT REPORT

ON

“Home Automation System”

SUBMITED TO

Computer Programming (1st Sem)

Internet of Things – Arduino (COMP 1045)

120 Bloor St E, Toronto, ON M4W 1B7

Academic Year: 2024 –2025

GUIDED BY:

Prof. Gurleen Kaur

PROJECT BY:

Utsav Patel
Aditi Trivedi
Yash Patel
Smit Patel

ABSTRACT

Home automation systems have revolutionized the way individuals interact with their living spaces, offering convenience, energy efficiency, and enhanced security. Among the various platforms available for developing such systems, Arduino stands out for its versatility, affordability, and ease of use. This paper presents a comprehensive review of home automation systems implemented using Arduino technology, focusing on recent advancements and innovative applications.

The review begins by outlining the fundamental components of home automation systems, including sensors, actuators, and communication protocols. It then delves into the capabilities of Arduino microcontrollers and their suitability for powering smart home devices. Various hardware modules compatible with Arduino, such as relay boards, motion sensors, and temperature sensors, are discussed in detail, highlighting their functionalities and integration possibilities.

Moreover, the paper addresses challenges and limitations associated with Arduino-based home automation systems, such as limited computational power, compatibility issues, and reliability concerns. Strategies for mitigating these challenges, such as optimizing code efficiency and selecting appropriate hardware components, are discussed to aid developers in creating robust and reliable solutions.

Finally, the review discusses future directions and opportunities for research and development in Arduino-based home automation systems. Topics such as machine learning integration, decentralized control algorithms, and adaptive user interfaces are identified as potential areas for innovation and advancement.

Project Index

UNIT – 1 Introduction & Objective of Project

1.1 INTRODUCTION OF HOME AUTOMATION SYSTEM

In today's fast-paced world, the concept of home automation has gained significant traction as people seek to enhance the comfort, convenience, and security of their living spaces. Home automation systems enable users to control various household appliances and devices remotely, automate routine tasks, and monitor their homes' status in real-time. Among the plethora of platforms available for implementing such systems, Arduino has emerged as a popular choice due to its flexibility, affordability, and robust community support.

1.2 OBJECTIVE OF PROJECT

The objective of the project on Home Automation System using Arduino is to design and develop a cost-effective and efficient system that enhances convenience, security, and energy efficiency within residential environments. Leveraging the capabilities of Arduino microcontrollers, the project aims to automate various household tasks and functions, such as lighting, temperature control, security monitoring, and appliance management, through an integrated and customizable platform.

Key Components of the Objective:

- Smart door automation
- Smart door opening sensor.
- Room temperature controller
- Gas detection!

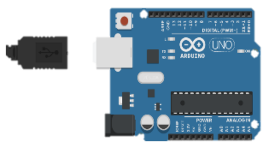
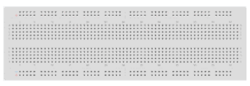
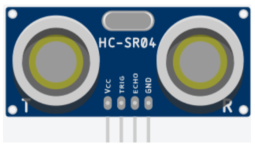
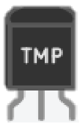
UNIT – 2 Tools, Platform and Languages

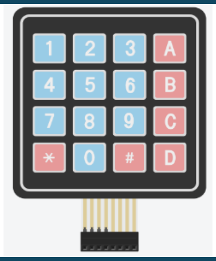
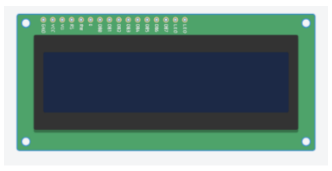
2.1 SOFTWARE REQUIREMENTS

Software Requirement at Development Side:

Tinker cad	TIN KER CAD		C++	C++

2.2 DESCRIPTION OF COMPONENTS USED

Tinker cad			Breadboard	
Led			Ultrasonic Distance Sensor	
Resistors			Gas Sensor	
Micro Servo			Temperature Sensor	

Keypad			LCD 16*2	

2.3 WORK RESPONSIBILITY OF TEAM MEMBERS

- **Utsav Patel:**
  - Responsible for programming of circuit using Arduino (work on sensor integration e.g., motion sensors, temperature sensors, gas sensor) to gather environment data for automated decision-making, testing and debugging of program and responsible for making *"Project Report"* and "Presentation".

- **Aditi Trivedi**
  - Responsible for designing circuits using Arduino (work on selection of appropriate electronic components e.g., sensors, microcontrollers etc.) based on requirements. Build prototype circuits and design a performance of components.

- **Yash Patel**
  - Responsible for maintaining detailed documentation of system architecture, design specifications and software implementation. Conduct testing procedures to validate the functionality, performance, and reliability of home automation system.

- **Smit Patel**
  - Maintain the detail documentation of circuit designs, specifications, test results, and design iterations throughout the development process, experimentation to explore novel circuit designs, sensor technologies

UNIT – 3 Complete Project Structure

3 DETAIL DESCRIPTION OF FEATURES

➢ **Smart door automation:**

- This feature allows users to enter password using keypad to enter the home. The password which is used by users will be manually programmed inside the code of the circuit. It increases the security of the house.

Functions:

(1) EnterPassword()
(2) CheckPassword()

➢ **Room temperature controller:**

- Control fan based on temperature and PIR sensor input.
- Reads the value from temperature sensor connected to pin A4 and calculates the temperature.
- If the motion is detected or the temperature exceeds 30’C, turn on a fan connected to pin 10.

Functions:

(1) CheckTemperatureSensor()
(2) readTemperature()

➢ **Smart door sensor:**

- This feature allows users to open the door automatically using *ultrasonic distance sensor*. The door will open automatically when the user comes into range of *100cm,* and the door closes immediately after the object goes out above 100*cm*.

Functions:

(1) CheckUltrasonicSensor()
(2) readUltrasonicDistance()
(3) ServoOpen() *and* ServoClose()
(4) OpenDoor() *and* CloseDoor()

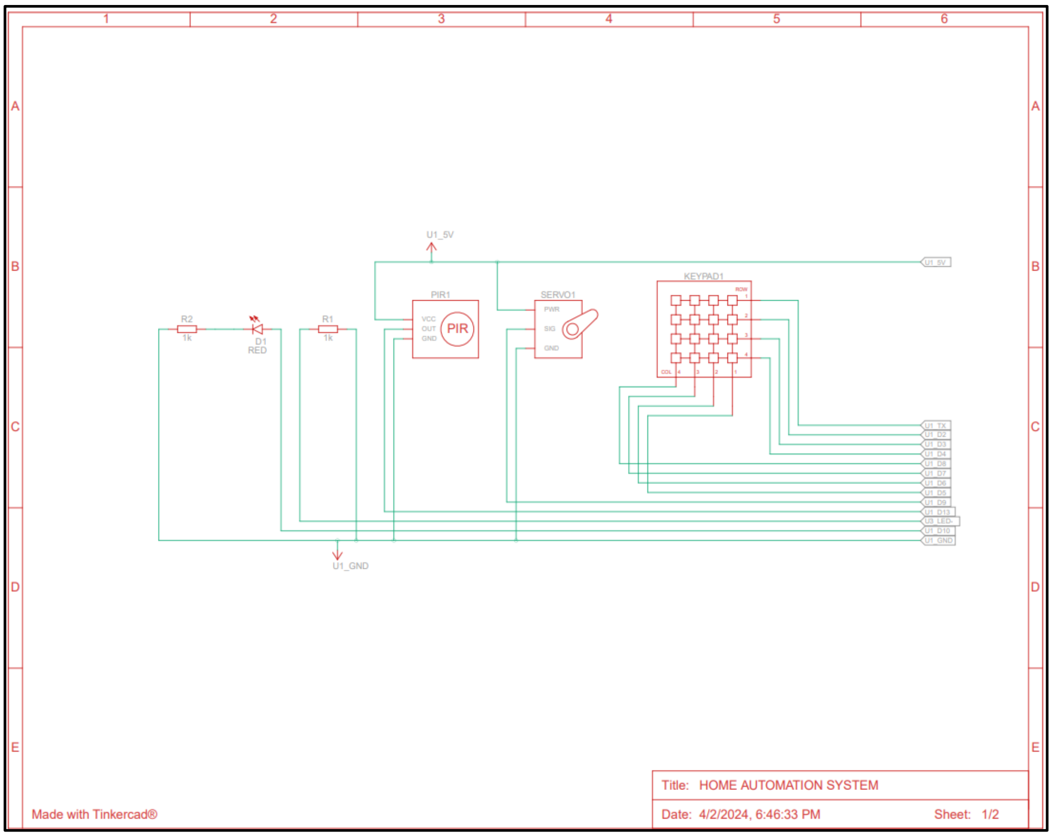
- **Gas detection sensor:**
  - This feature allows users to enter password using keypad to enter the home. The password which is used by users will be manually programmed inside the code of the circuit.
  - Reads the value from the sensor and checks if it exceeds a predefined limit (600).
  - If the gas level is high, triggers a buzzer connected to pin 8.
  - Displays gas level and status on the LCD.

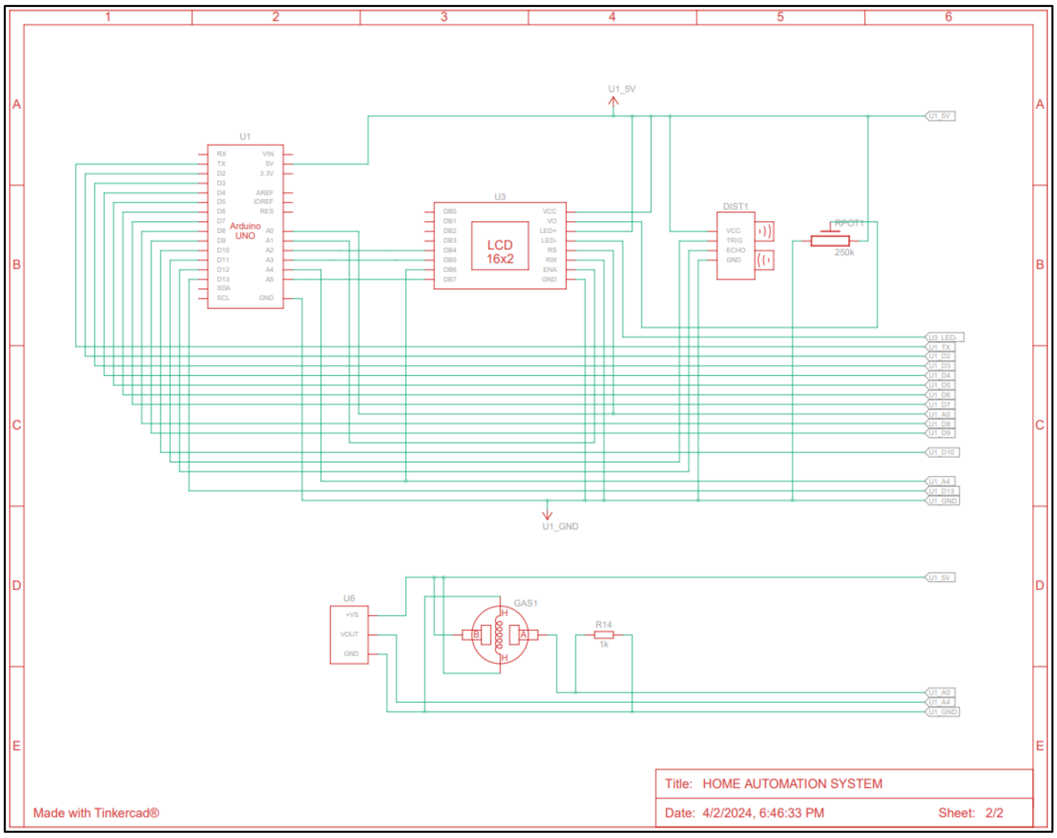
  Functions:

  (1) CheckGasSensor()

- **Overall:** The code demonstrates various functionalities such as sensor reading, actuator control, and I2C communication. It provides a comprehensive example of how to integrate different components in an Arduino project.

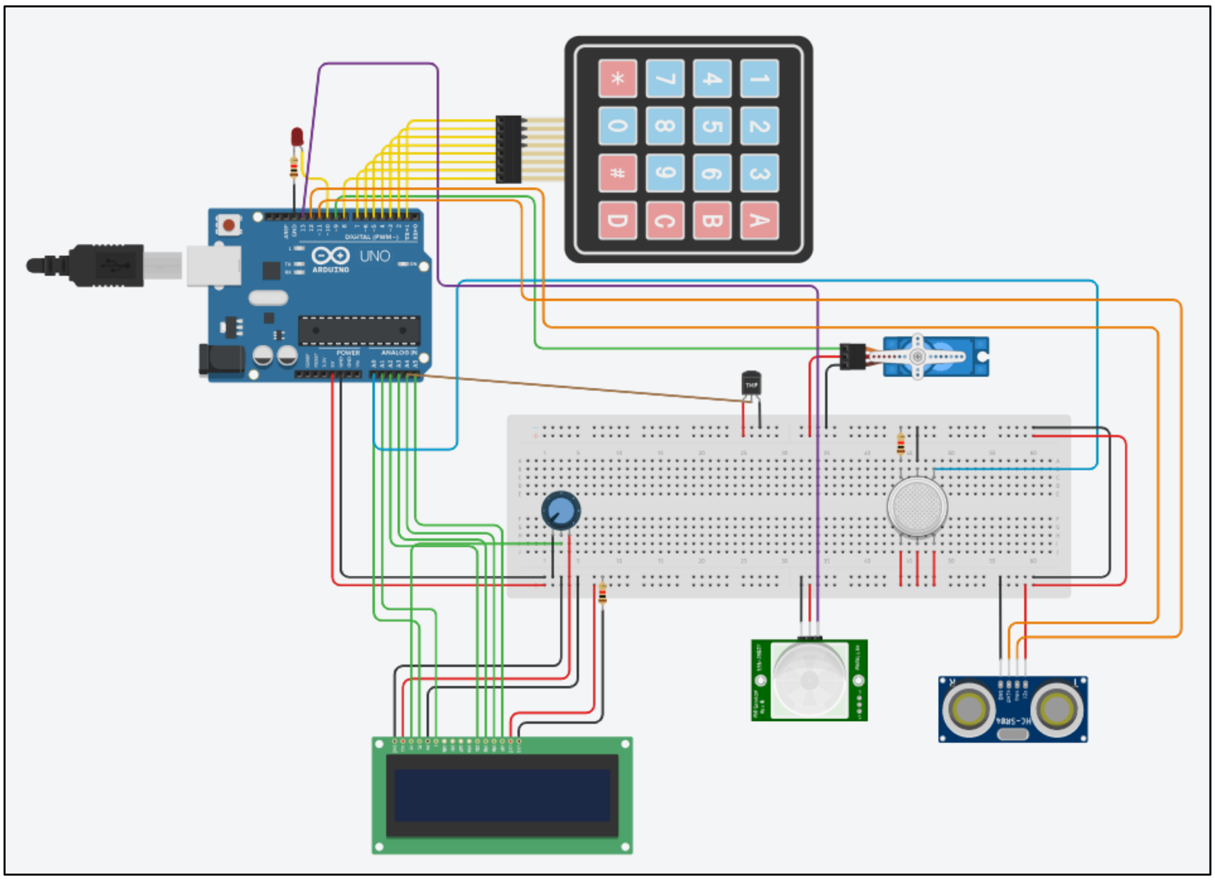
3.1 BLUE-PRINT OF CIRCUIT





UNIT – 4 SNAPSHOTS

4.1 CIRCUIT:



4.2 CIRCUIT CODE:

```
// PROJECT : SMART HOME AUTOMATION
// PRESENTED BY : ADITI TRIVEDI, UTSAV PATEL, SMIT PATEL, YASH PATEL
// GUIDED BY : GURLEEN KAUR

#include <Servo.h>
#include <LiquidCrystal.h>
#include <Keypad.h>

Servo myservo;

LiquidCrystal lcd(A0, A1, A2, A3, A4, A5);
```

```
// Define sensor pin

const int gas_sensor = A0;
const int temperature_sensor = A4;
const int pir_sensor = 13;
const int ultrasonic_trigger_pin = 11;
const int ultrasonic_echo_pin = 12;
int sensorState = LOW;
int sensorValue = 0;

#define Password_Length 5 // Give enough room for six chars + NULL char

int pos = 0;    // variable to store the servo position

char Data[Password_Length]; // 6 is the number of chars it can hold + the null
char = 7
char Master[Password_Length] = "1234";
byte data_count = 0, master_count = 0;
bool Pass_is_good;
char customKey;

const byte ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] = {
  {'1', '2', '3'},
  {'4', '5', '6'},
  {'7', '8', '9'},
  {'*', '0', '#'}
};
bool door = true;
bool passwordEntered = false;

byte rowPins[ROWS] = {1, 2, 3, 4}; //connect to the row pinouts of the keypad
byte colPins[COLS] = {5, 6, 7, 8}; //connect to the column pinouts of the
keypad

Keypad customKeypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS);
//initialize an instance of class NewKeypad

enum State {
  ENTER_PASSWORD,
  CHECK_PASSWORD,
  CHECK_GAS_SENSOR,
  CHECK_TEMPERATURE_SENSOR,
  CHECK_ULTRASONIC_SENSOR,
  OPEN_DOOR,
  CLOSE_DOOR
};
```

```
State currentState = ENTER_PASSWORD;

void setup() {
  pinMode(10, OUTPUT);
  pinMode(gas_sensor, INPUT);
  pinMode(temperature_sensor, INPUT);
  pinMode(pir_sensor, INPUT);
  pinMode(ultrasonic_trigger_pin, OUTPUT);
  pinMode(ultrasonic_echo_pin, OUTPUT);
  myservo.attach(9);
  ServoClose();
  lcd.begin(16, 2);
  lcd.print(" Arduino Door");
  lcd.setCursor(0, 1);
  lcd.print("--Look project--");
  delay(3000);
  lcd.clear();
}

void loop() {
  switch (currentState) {
    case ENTER_PASSWORD:
      EnterPassword();
      break;

    case CHECK_PASSWORD:
      CheckPassword();
      break;

    case CHECK_GAS_SENSOR:
      CheckGasSensor();
      break;

    case CHECK_TEMPERATURE_SENSOR:
      CheckTemperatureSensor();
      break;

    case CHECK_ULTRASONIC_SENSOR:
      CheckUltrasonicSensor();
      break;

    case OPEN_DOOR:
      OpenDoor();
      break;

    case CLOSE_DOOR:
      CloseDoor();
      break;
  }
}

void EnterPassword() {
  lcd.setCursor(0, 0);
  lcd.print(" Enter Password");

  customKey = customKeypad.getKey();
  if (customKey) {
    Data[data_count] = customKey;
    lcd.setCursor(data_count, 1);
    lcd.print(Data[data_count]);
    data_count++;
  }

  if (data_count == Password_Length - 1) {
    if (!strcmp(Data, Master)) {
      lcd.clear();
      digitalWrite(10, HIGH);
      ServoOpen();
      door = 0;
      passwordEntered = true;
      currentState = CHECK_GAS_SENSOR;
    } else {
      lcd.clear();
      lcd.print("  Wrong Password");
      delay(1000);
      clearData();
      lcd.clear();
    }
  }
}

void CheckPassword() {
  // Placeholder for future functionality if needed
  currentState = CHECK_GAS_SENSOR;
}

void CheckGasSensor() {
  int gasLevel = analogRead(gas_sensor);
  if (gasLevel > 600) {
    lcd.setCursor(0, 0);
    lcd.print("Gas level: ");
    lcd.print(gasLevel);
    lcd.setCursor(0, 1); // Fixed duplicate setCursor
    lcd.print("Fan ON");
    delay(3000);
  } else {
    lcd.clear();
    lcd.print("Gas level: ");
    lcd.print(gasLevel);
    lcd.setCursor(0, 1); // Fixed duplicate setCursor
    lcd.print("Fan OFF");
    delay(500);
  }



 currentState = CHECK_TEMPERATURE_SENSOR;
}

void CheckTemperatureSensor() {
  int temperature = 30; // Fixing the temperature to 30 degrees Celsius
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Temperature: ");
  lcd.print(temperature);
  lcd.setCursor(0, 1);
  lcd.print("degree Celsuis");
  delay(3000);


  currentState = CHECK_ULTRASONIC_SENSOR;
}

void CheckUltrasonicSensor() {
  int distance = 0.01723 *readUltrasonicDistance(ultrasonic_trigger_pin,
ultrasonic_echo_pin);
  int pirValue = digitalRead(pir_sensor); // Read PIR sensor value
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Distance: ");
  lcd.print(distance);
  lcd.print(" cm");

  delay(3000);

  if (distance <= 60) {
    currentState = OPEN_DOOR;
    digitalWrite(10, HIGH);
    delay(1000);
  }
  else {
    currentState = CLOSE_DOOR;
    digitalWrite(10, LOW);
  }
```

```
}

void OpenDoor() {
  lcd.clear();
  lcd.print("Door Opening");
  ServoOpen();
  delay(2000);
  currentState = CHECK_GAS_SENSOR;
}

void CloseDoor() {
  lcd.clear();
  lcd.print("Door Closing");
  ServoClose();
  delay(2000);
  currentState = CHECK_GAS_SENSOR;
}

void clearData() {
  while (data_count != 0) {
    Data[data_count--] = 0;
  }
}

void ServoOpen() {
  for (pos = 180; pos >= 0; pos -= 5) {
    myservo.write(pos);
    delay(15);
  }
}

void ServoClose() {
  for (pos = 0; pos <= 180; pos += 5) {
    myservo.write(pos);
    delay(15);
  }
}


long readUltrasonicDistance(int triggerPin, int echoPin) {
  pinMode(triggerPin, OUTPUT);
  digitalWrite(triggerPin, LOW);
  delayMicroseconds(2);
  digitalWrite(triggerPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerPin, LOW);
  pinMode(echoPin, INPUT);
  return pulseIn(echoPin, HIGH);
```

UNIT – 5 Future Enhancement

In the "Future Enhancement" of *Home Automation System* using Arduino, explore integrating sensors for detecting specific events or hazards, such as smoke detectors, water leakage sensors, and motion detector sensors for security purposes. Implement features for scheduling automated tasks, setting preferences, and receiving notifications or alerts. Implement algorithms and logic to optimize energy usage based on real-time data and user preferences. Design the home automation system with scalability in mind, allowing for easy expansion and addition of new devices or functionalities. Adopt a modular architecture that facilitates integration with third-party plugins, extensions, or custom components developed by the community. Implement a process for continuous improvement, releasing regular updates and patches to address bugs, add new features, and enhance overall system performance.