CSE461

Section : 06

Group : 04

Semester : Spring 2025

Project Report

Member Details:

1. Name: Jannatul Ferdaus ID: 21301062
2. Name: Anonna Dev Nipa ID: 21301191
3. Name: Tangena Islam ID: 21301105
4. Name: Sirajum Munira Lamisa ID: 22101442

Submission Date: 11-May-2025

Instructors: Rafid Ahnaf [RFF] & Md Toki Tahmid [TOKD]

Project Title: Home Security & Automation System

1. Purpose

Objective: The primary goal of this project is to develop a smart home security and automation system that integrates multiple sensors and actuators to enhance security of a residence and automate appliances.

Scope: The system will be designed for residential security and automation, enabling smart access control using RFID authentication, intrusion detection, fire hazard prevention and temperature-based fan control.

Significance: Home security will be enhanced by implementing RFID-based door security and intrusion detection. Fire hazards will be prevented with automatic fire detection and sprinkler activation. Improved convenience and reduction of energy consumption will be created by automating fans based on occupancy and temperature.

2. Components

- Microcontroller: Arduino Uno, ESP32 module
- Sensors:
    - RFID Sensor
    - PIR motion sensor
    - Temperature sensor
    - Flame Sensor
    - Raindrop Module

- Actuators:
    - Servo Motor
    - DC Motor

- Body/Chassis: Enclosure for microcontroller, power management, and sensors.
- Additional Components: Buzzer, jumper wire set, LCD Display, resistors, L298N H-Bridge dual motor driver, breadboard, battery.

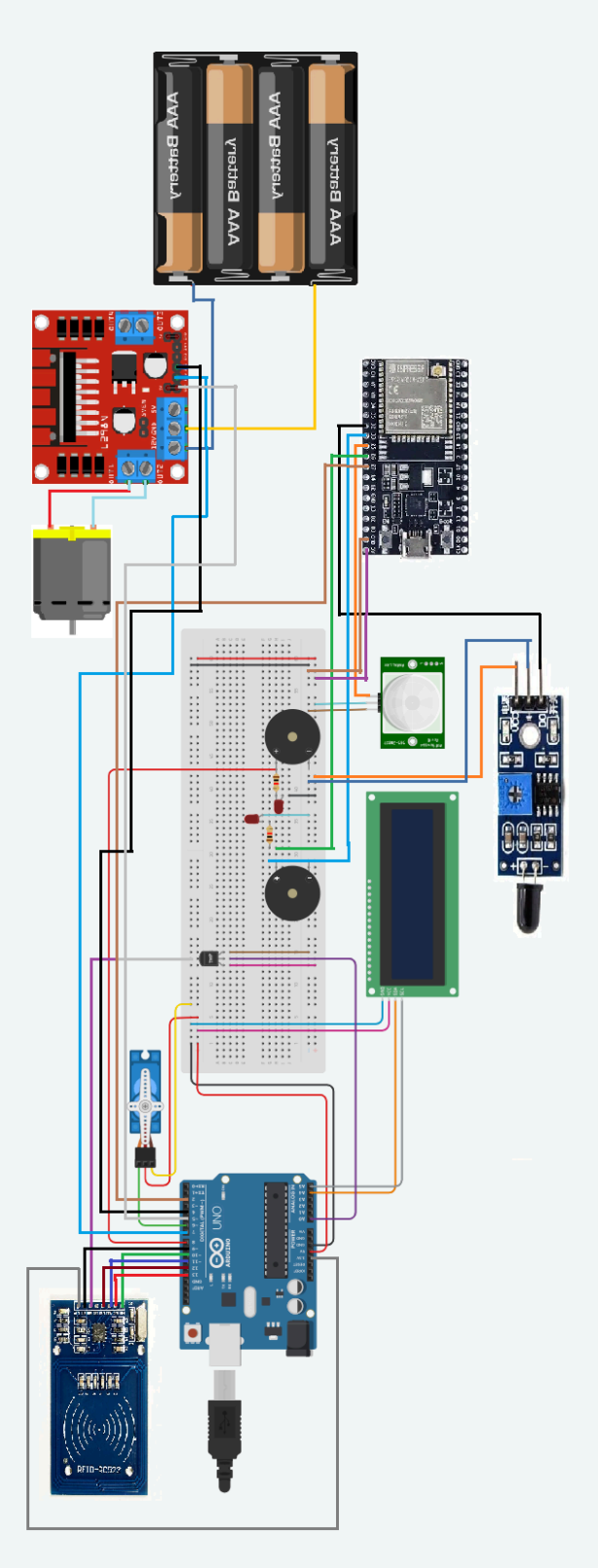

Fig: Circuit Diagram

Arduino Code:

```
#include <SPI.h>
#include <MFRC522.h>
#include <Servo.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

#define SERVO_PIN 6
#define SS_PIN 10
#define RST_PIN 9
#define BUZZER_PIN 8
#define TEMP_PIN A0
#define IN1 7
#define IN2 4
#define ENA 5
#define MOTION_PIN 2
#define RAIN_SENSOR_PIN A1
#define RAIN_SERVO_PIN 3
#define RAIN_THRESHOLD 500

Servo myServo;
Servo rainServo;

MFRC522 mfrc522(SS_PIN, RST_PIN);
LiquidCrystal_I2C lcd(0x27, 16, 2);

byte readCard[4];
String tagID = "";

const int authorizedCount = 3;
String authorizedTags[authorizedCount] = {
  "634C81F7", "33588F7", "A3B5C0A0"
};

unsigned long lastTempPrint = 0;
unsigned long lastMotionTime = 0;
const unsigned long motionHoldDuration = 5000;
unsigned long accessMessageUntil = 0;

float readTemperature();
boolean getID();
boolean isAuthorized(String id);

void setup() {
  Serial.begin(9600);
  SPI.begin();
  mfrc522.PCD_Init();
  delay(4);
  mfrc522.PCD_DumpVersionToSerial();

  pinMode(BUZZER_PIN, OUTPUT);
  digitalWrite(BUZZER_PIN, LOW);

  myServo.attach(SERVO_PIN);
  myServo.write(0);
```

```
  rainServo.attach(RAIN_SERVO_PIN);
  rainServo.write(0);

  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("Access Control");
  lcd.setCursor(0, 1);
  lcd.print("Scan Your Card");

  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(ENA, OUTPUT);
  pinMode(MOTION_PIN, INPUT);

  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  analogWrite(ENA, 0);
}

void loop() {
  bool motionDetected = digitalRead(MOTION_PIN);
  float currentTemp = readTemperature();
  int rainValue = analogRead(RAIN_SENSOR_PIN);  // Read analog value
from AO pin
  //Serial.print(rainValue);
  // Handle rain detection with threshold
  if (rainValue < RAIN_THRESHOLD) {
    rainServo.write(150);  // Close or cover due to rain
  } else {
    rainServo.write(0);    // Open / no rain
  }

  // Update last motion time if motion detected
  if (motionDetected) {
    lastMotionTime = millis();
  }

  bool fanShouldRun = (millis() - lastMotionTime) <= motionHoldDuration;

  // Print temp and rain sensor value every 2 seconds if not showing
access messages
  if (millis() > accessMessageUntil && millis() - lastTempPrint > 2000)
{
    Serial.print("Temperature: ");
    Serial.print(currentTemp);
    Serial.print(" °C | Rain Sensor: ");
    Serial.println(rainValue);

    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Temp: ");
    lcd.print(currentTemp, 1);
    lcd.print((char)223);
    lcd.print("C");
```

```
    // Fan logic based on temp and motion
    if (fanShouldRun) {
      if (currentTemp > 35.0) {
        digitalWrite(IN1, HIGH);
        digitalWrite(IN2, LOW);
        analogWrite(ENA, 255);
        lcd.setCursor(0, 1);
        lcd.print("Fan: HIGH      ");
      } else if (currentTemp >= 33.0) {
        digitalWrite(IN1, HIGH);
        digitalWrite(IN2, LOW);
        analogWrite(ENA, 128);
        lcd.setCursor(0, 1);
        lcd.print("Fan: MEDIUM    ");
      } else {
        digitalWrite(IN1, LOW);
        digitalWrite(IN2, LOW);
        analogWrite(ENA, 0);
        lcd.setCursor(0, 1);
        lcd.print("Fan: OFF       ");
      }
    } else {
      digitalWrite(IN1, LOW);
      digitalWrite(IN2, LOW);
      analogWrite(ENA, 0);
      lcd.setCursor(0, 1);
      lcd.print("Fan: OFF (No M)");
    }

    lastTempPrint = millis();
  }

  // RFID card check
  if (getID()) {
    Serial.print("Card ID: ");
    Serial.println(tagID);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Card ID:");
    lcd.setCursor(0, 1);
    lcd.print(tagID);

    if (isAuthorized(tagID)) {
      Serial.println("Access Granted!");
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Access Granted");
      myServo.write(180);
      delay(3000);
Aa      myServo.write(0);
    } else {
      Serial.println("Access Denied!");
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Access Denied!");
```

```
      digitalWrite(BUZZER_PIN, HIGH);
      delay(1000);
      digitalWrite(BUZZER_PIN, LOW);a
    }

    accessMessageUntil = millis() + 3000;
    delay(2000);
    lcd.clear();1aà
    lcd.setCursor(0, 0);
    lcd.print("Scan Your Card");
  }
}

boolean getID() {
  if (!mfrc522.PICC_IsNewCardPresent()) return false;
  if (!mfrc522.PICC_ReadCardSerial()) return false;

  tagID = "";
  for (uint8_t i = 0; i < 4; i++) {
    tagID.concat(String(mfrc522.uid.uidByte[i], HEX));
  }

  tagID.toUpperCase();
  mfrc522.PICC_HaltA();
  return true;
}

boolean isAuthorized(String id) {
  for (int i = 0; i < authorizedCount; i++) {
    if (id == authorizedTags[i]) return true;
  }
  return false;
}

// float readTemperature() {
//   int rawValue = analogRead(TEMP_PIN);
//   float voltage = rawValue * (5.0 / 1023.0);
//   float temperatureC = voltage * 100.0;
//   return temperatureC;
// }
float readTemperature() {
  long sum = 0;
  const int samples = 10;

  for (int i = 0; i < samples; i++) {
    sum += analogRead(TEMP_PIN);
    delay(5); // Small delay between readings
  }

  float average = sum / (float)samples;
  float voltage = average * (5.0 / 1023.0);
  float temperatureC = voltage * 100.0;
  return temperatureC;
}
```

ESP32 Dev Module Code:

```
#include <WiFi.h>
#include <HTTPClient.h>

#define Buzzer 33
#define Sensor 32  // fire
#define PIR_PIN 25
#define LED_PIN 26
#define MOTION_SIGNAL_PIN 27

//whatsapp message send "I allow callmebot to send me messages"
const char* ssid = "AradAriyaana";        // Replace with your WiFi name
const char* password = "Arad2@1234"; // Replace with your WiFi password

String phoneNumber = "8801820061505"; // Your WhatsApp number
String apiKey = "9816379";             // Your API key from CallMeBot

void setup() {
  Serial.begin(115200);
  pinMode(Buzzer, OUTPUT);
  pinMode(Sensor, INPUT);
  pinMode(PIR_PIN, INPUT);
  pinMode(LED_PIN, OUTPUT);


  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("\nConnected to WiFi with IP: " +
WiFi.localIP().toString());

  sendAlert("Your fire alert system is now active and ready!");
   pinMode(MOTION_SIGNAL_PIN, OUTPUT);
  digitalWrite(MOTION_SIGNAL_PIN, LOW);
}

void loop() {
  bool sensorValue = digitalRead(Sensor);
  if (sensorValue == 0) {
    digitalWrite(Buzzer, HIGH);
    sendAlert("🔥 Warning! Fire Detected!");
    delay(5000); // Delay to prevent message spamming
  } else {
    digitalWrite(Buzzer, LOW);
  }
  static unsigned long lastMotionTime = 0;
bool motionDetected = digitalRead(PIR_PIN);

if (motionDetected) {
  lastMotionTime = millis(); // Save the time motion was detected
}

if (millis() - lastMotionTime < 5000) {
```

```
  digitalWrite(LED_PIN, HIGH);
} else {
  digitalWrite(LED_PIN, LOW);
}

if (motionDetected) {
  digitalWrite(MOTION_SIGNAL_PIN, HIGH);
} else {
  digitalWrite(MOTION_SIGNAL_PIN, LOW);
}

}
void sendAlert(String message) {
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    String encodedMessage = urlEncode(message);
    String url = "http://api.callmebot.com/whatsapp.php?phone=" +
phoneNumber + "&apikey=" + apiKey + "&text=" + encodedMessage;

    http.begin(url);
    http.addHeader("Content-Type", "application/x-www-form-urlencoded");

    int httpResponseCode = http.POST("");
    String response = http.getString();

    if (httpResponseCode == 200) {
      Serial.println("✅ WhatsApp message sent successfully!");
    } else {
      Serial.println("❌ Failed to send WhatsApp message. Code: " +
String(httpResponseCode));
      Serial.println("Server response: " + response);
    }

    http.end();
  } else {
    Serial.println("WiFi not connected. Cannot send alert.");
  }
}


// Manual URL encoder function (replaces the need for UrlEncode.h)
String urlEncode(String str) {
  String encodedString = "";
  char c;
  char code0;
  char code1;
  char code2;
  for (int i = 0; i < str.length(); i++) {
    c = str.charAt(i);
    if (isalnum(c)) {
      encodedString += c;
    } else {
      code1 = (c & 0xf) + '0';
      if ((c & 0xf) > 9) {
        code1 = (c & 0xf) - 10 + 'A';
      }
```

```
      c = (c >> 4) & 0xf;
      code0 = c + '0';
      if (c > 9) {
        code0 = c - 10 + 'A';
      }
      code2 = '\0';
      encodedString += '%';
      encodedString += code0;
      encodedString += code1;
    }
    yield(); // prevent watchdog timer reset
  }
  return encodedString;
}
```

3. Cost Breakdown

No	Components	Quantity	Unit Cost (BDT)	Total Cost (BDT)
1	Arduino Uno	1	710	710
2	RC522 RFID card reader module kit	1	195	195
3	PIR motion sensor	1	110	110
4	Temperature sensor	1	200	200
5	Flame Sensor	1	70	70
6	L298N H- Bridge dual motor driver	1	195	195
7	3V DC Hobby motor with fan	1	120	120
8	Servo Motor	2	150	300
9	Buzzer	2	20	40
10	Jumper wire set	40 piece	100	100
11	Breadboard	2	150	300
12	Battery	4	74	296
13	LCD display	1	250	250
14	ESP32 module	1	550	550
15	Rain drop module	1	141	141
16	Resistors	2	5	5
Total Cost (BDT)				3582

4. Functionality Breakdown

- **Functionality 1:** RFID-Based Door Security System

- **Overview:** Secure authorized access into residence using RFID Authentication
- **Working Procedure:** The RFID sensor will first scan the authorized RFID tag/card. If it is authenticated, the servo motor will open the door. If an unauthorized card is scanned, the buzzer will be activated.

- **Functionality 2:** Automated Fire Detection & Notification System
  - **Overview:** The flame sensor detects the presence of fire and alerts users through a buzzer and a WhatsApp message using the ESP module.
  - **Working Procedure:** When the system is switched on, it immediately sends a message via the ESP module to WhatsApp saying, "Your fire alert system is now active and ready!" After that, the flame sensor begins monitoring the surroundings for any signs of fire. If it detects a flame, the system activates a buzzer to alert the residence. At the same time, a message is sent to WhatsApp that says, "Warning!! Fire detected." This ensures that the user is notified both through sound and remotely via their phone, even if they are not near the system.

- **Functionality 3:** Motion and Temperature-Based Fan Control
  - **Overview:** Controls fan operation based on room occupancy and temperature.
  - **Working Procedure:** The PIR sensor shall detect human presence in the room and if a person is present and the temperature sensor detects high temperature, the fan will be turned on. If no human presence is detected, the fan remains off regardless of temperature.
- **Functionality 4:** Weather-Responsive Clothes Retractor
  - **Overview:** Detects rainfall using a rain sensor and automatically protects clothes that are drying outside. When rain is detected, a servo motor rotates a stick that holds the clothes by 90 or 180 degrees to bring the clothes inside or under a shelter, preventing them from getting wet.
  - **Working Procedure:** The rain sensor continuously monitors the environment for any signs of rain. When raindrops are detected on the sensor, the system activates a servo motor. The motor then rotates the stick carrying the clothes by 90 or 180 degrees moving the clothes to a protected area or inside the house. This automatic movement helps keep the clothes dry without needing manual intervention.

5. Potential Challenges

- **Technical Challenges:**
    1. **DC Motor Battery Challenges:** The DC motor faced inconsistency due to insufficient battery capacity. This led to unreliable performance, especially during extended usage or when power demand spiked.
    2. **ESP32 Dev Module Wi-Fi Installation Issues:** During the initial setup of the ESP32 development board, establishing a stable Wi-Fi connection proved challenging. The module failed to connect to the network reliably, which prevented real-time data transmission. To resolve this, Serial.print() statements were employed to trace connection attempts and pinpoint failures.
- **Design Challenges:** We ensured compact design while integrating multiple components
- **Integration Challenges:**
    1. **Power Failure Impact:** A critical issue occurred when the battery ran out which was powering the DC motor, causing the motor to stop functioning.
    2. **Flame Sensor Logic Error:** An incorrect assumption in the flame sensor code caused a major malfunction. The system was configured to detect fire when the sensor output was 1 but the actual sensor logic was the reverse (0 indicates flame detected, 1 indicates no flame). As a result, the ESP32 misinterpreted normal readings as continuous fire detection and sent 125 fire alert messages to WhatsApp in a short period. This caused the ESP module to overload, leading to a 4-hour downtime in the notification system.
    3. **Temperature Sensor Fluctuation Due to Circuit Noise:** The temperature sensor began showing highly unstable readings, with values rapidly fluctuating up and down. To resolve the issue, a small test script was used to isolate the sensor from the main circuit. The readings were stable in isolation which confirmed that the fluctuations were caused by electrical noise from the larger circuit setup. To solve this, average temperature was used.

- **Budget Constraints**: The project was limited by a tight budget, which led to several compromises. Only one breadboard was used despite the need for more space. Although only two servos were required, four had to be purchased due to hardware failures. Additionally, multiple 9V batteries were needed, as a single battery could not power the full circuit reliably.

- **Risk Mitigation:**
    1. **Breadboard overcrowding:** Increased chances of loose connections, short circuits, and component misplacement, which could cause system instability or failure. To solve that, we carefully managed the layout to optimize space and prevent wiring conflicts.