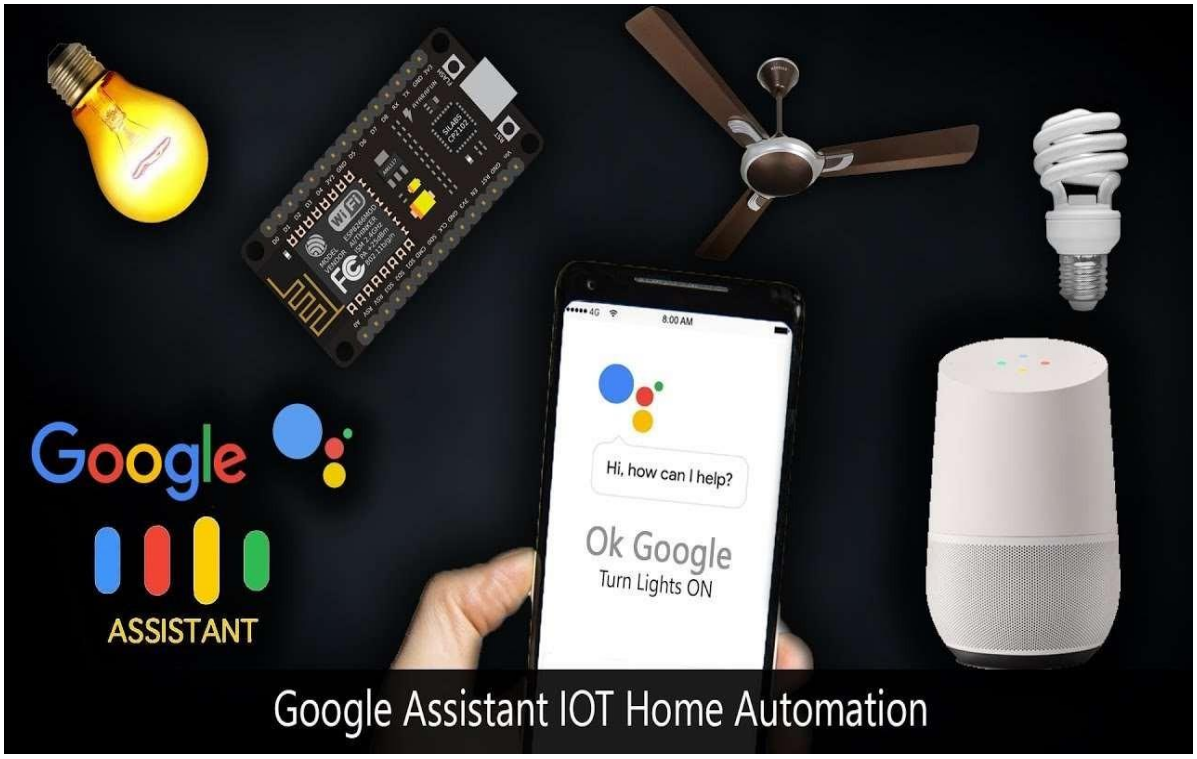
Chapter1 - Introduction

1.1 Background

Home automation is essential for enhancing convenience, efficiency, and safety in modern living. IoT-based systems address these needs by integrating sensors, microcontrollers, cloud platforms, and voice-controlled assistants for seamless control and automation of household devices.

The **Smart IoT System for Home Automation using Voice Control** leverages components like the NodeMCU ESP8266 microcontroller, various sensors (temperature, humidity, motion, and light), and the Alexa voice assistant. These enable real-time monitoring and control of lighting, appliances, and environmental settings, ensuring optimal energy usage and convenience.

Enhanced by machine learning, the system predicts usage patterns and automates operations based on user behavior, delivering a personalized and efficient experience. With real-time notifications, the system also addresses safety concerns by providing alerts for anomalies like fire, intrusion, or device malfunctions. Scalable and cost-effective, this IoT-based solution is a benchmark in intelligent and sustainable home automation.



1.1 Problem Statement

Design and implement a **Smart IoT System for Home Automation using Voice Control** that automates the control of household devices and appliances, providing enhanced convenience, efficiency, and safety. The system integrates IoT technologies to monitor and control lighting, temperature, and appliances in real-time, while also responding to voice commands through Alexa.

It enables users to control and monitor their home environment remotely via a mobile application or cloud platform. Sensors track parameters like motion, temperature, and humidity, triggering automated responses or sending alerts for anomalies such as fire, intrusion, or device malfunctions. The system enhances energy efficiency by analyzing usage patterns and automating operations based on user behavior.

This scalable and cost-effective solution ensures seamless interaction with home devices, offering a smarter, safer, and more connected living experience for users.

1.1.1 Objectives

- **Automation of Home Devices**

Enable seamless control of household appliances such as lighting, fans, and climate control systems through voice commands and mobile applications.

- **Integration with Voice Assistants**

Leverage voice-controlled systems like Alexa to facilitate hands-free operation and enhance user convenience.

- **Real-time Monitoring and Alerts**

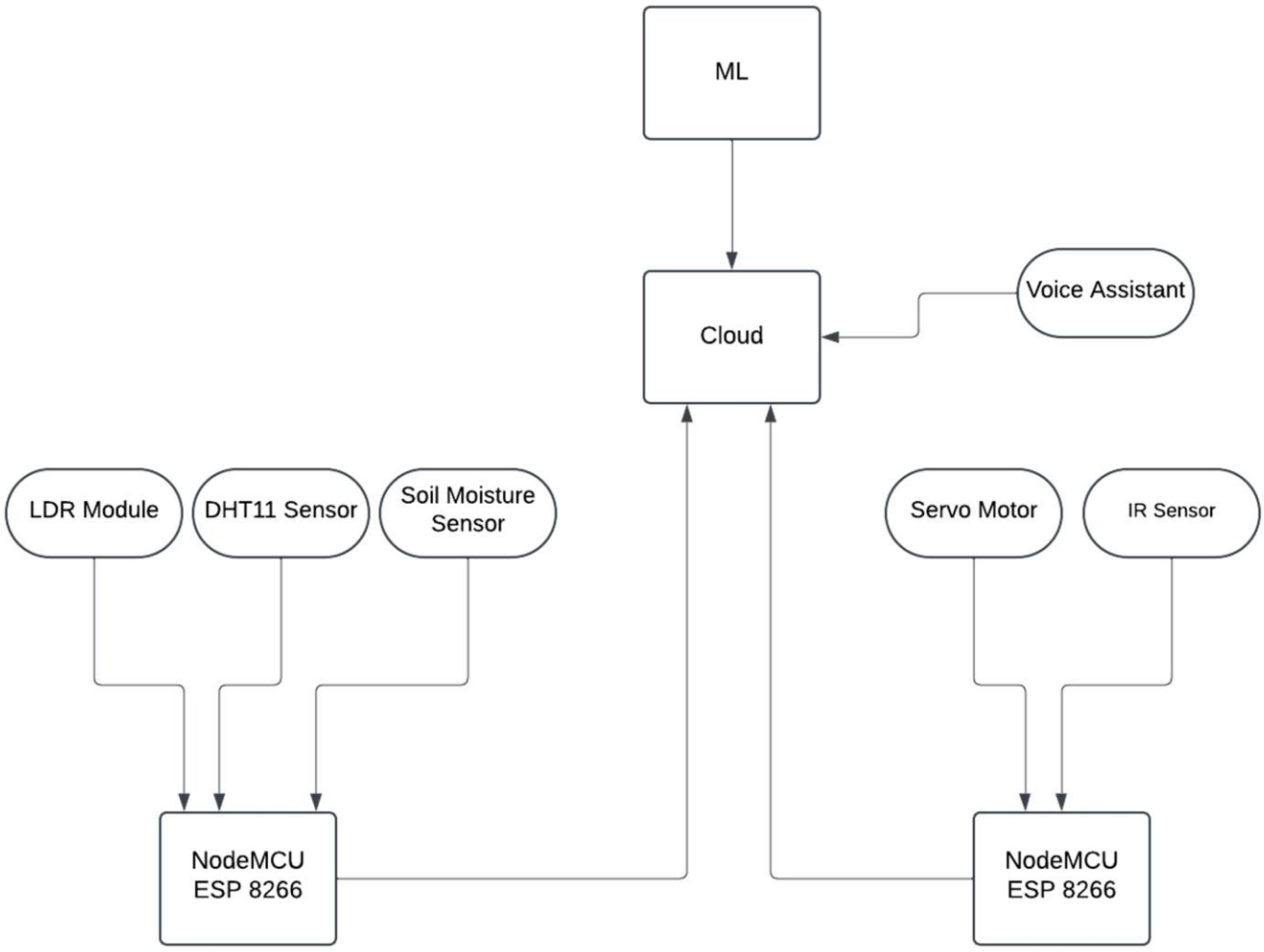
Utilize IoT sensors to monitor parameters like temperature, humidity, and motion in real time, and send alerts for anomalies such as fire, intrusion, or equipment malfunction.

- **Energy Efficiency**

Optimize energy consumption by analyzing usage patterns and automating device operations to minimize wastage and enhance sustainability.

Chapter 2

Implemented Framework



Project Research

- **Gather Information**

Collect data on home automation systems, focusing on IoT-based solutions that integrate voice control, real-time monitoring, and energy efficiency. Explore technologies like NodeMCU, Alexa, and smart sensors for automation and safety.

- **Review Existing Solutions**

Examine current home automation technologies, emphasizing integration with voice-controlled systems such as Alexa and mobile interfaces. Identify gaps in existing solutions and areas for improvement, such as personalized automation and predictive analytics.

- **Define Key Objectives**

Outline objectives such as automated control of lighting, appliances, and climate; real-time monitoring of parameters like temperature and motion; safety alerts for anomalies; and user-friendly interfaces for seamless operation.

- **Analyze Sensor Data**

Study data collected from IoT sensors (e.g., motion detectors, temperature and humidity sensors) to understand their role in triggering automated actions and enhancing system responsiveness.

- **Data Analysis**

Examine usage patterns and energy consumption trends to develop insights for predictive automation, ensuring system efficiency and personalization.

- **Statistical Analysis**

Apply statistical methods to identify correlations between environmental conditions, device usage, and user behavior. This will help refine automation rules and optimize system performance.

- **Validate Assumptions**

Test assumptions regarding user behavior, energy usage, and safety triggers (e.g., fire or intrusion alerts) to ensure the accuracy and reliability of automation and alert mechanisms.

Data Preprocessing

- **Normalization:**

Standardize data from the MQ-2 gas sensor (leakage detection) and HX711 load cell (gas cylinder weight) to ensure consistency across readings. Address issues like outliers and inconsistent data to improve reliability.

- **Scaling:**

Apply scaling techniques to enhance the performance and accuracy of the machine learning models during analysis and prediction.

Model Selection

- **Criteria Definition:**

Establish selection criteria such as accuracy, interpretability, and computational efficiency to evaluate models effectively.

- **Algorithm Testing:**

Experiment with various algorithms, including Gradient Boosting, Decision Trees, and Support Vector Machines (SVM).

- **Model Choice:**

Select the Random Forest Classifier for its robustness, ability to handle sensor data, and high accuracy in classification tasks such as gas leakage detection and refill predictions.

Performance Analysis

- **Model Training and Validation:**

Train and validate the Random Forest Classifier using the preprocessed sensor dataset.

- **Evaluation Metrics:**

Assess model performance with metrics like accuracy, precision, recall, F1 score, and confusion matrix.

- **Optimization:**

Perform hyperparameter tuning to refine the model's predictions and enhance its reliability in real-world scenarios.

Chapter 3

3.1 Implementation Phase

3.1 Implementation Phase

A. Workflow of the System

The Smart Home Automation System using Voice Control integrates IoT devices and voice assistants to provide seamless control of household functions. The system begins with sensors monitoring various environmental parameters, such as temperature, humidity, and motion. These sensors send data to the ESP8266 microcontroller, which processes the inputs and communicates with the cloud using MQTT or HTTP protocols.

The centralized gateway aggregates sensor data, enabling real-time control and monitoring of devices such as lights, fans, and appliances. Users can interact with the system via Alexa voice commands or a mobile/web interface powered by the Blynk IoT platform. Automated routines are triggered based on sensor readings, such as adjusting lighting when motion is detected or optimizing temperature settings for energy efficiency. Safety alerts, such as fire or intrusion detection, are immediately communicated to users through notifications, ensuring prompt action. The system's machine learning algorithms analyze user behavior to recommend personalized automation settings, enhancing the overall experience.

B. Methods and Materials

The Smart Home Automation System employs a combination of hardware and software for efficient and interactive operation:

Hardware Components:

- NodeMCU ESP8266: Acts as the processing hub, collecting sensor data and enabling cloud communication.
- Sensors:
  - DHT11: Monitors temperature and humidity.
  - PIR Motion Sensor: Detects movement for automated lighting or security.
  - LDR (Light Dependent Resistor): Adjusts lighting based on ambient light levels.
- Smart Relays: Enables control of appliances like fans and lights.

Software Components:

- Arduino IDE: Used for programming the ESP8266 microcontroller.
- ESP8266 WiFi Library: Facilitates internet connectivity.

• **Google Home Integration**: Uses the Google Home API for voice commands and seamless interaction with smart devices.Predictive Automation:
Machine learning models analyze sensor data and user patterns to optimize energy usage and automate routines. For instance, lights are turned off when no motion is detected, or temperature is adjusted based on user preferences and environmental conditions.

This integration of IoT and voice control ensures a smarter, safer, and more efficient home environment, setting a new benchmark in home automation technology.

3.2 Material Specifications:

1. NodeMCU ESP8266 : The **ESP8266** is a low-cost, power-efficient, and highly versatile Wi-Fi-enabled microcontroller developed by Espressif Systems. It is widely used in IoT (Internet of Things) projects due to its compact size, robust features, and ease of integration with various sensors and devices.

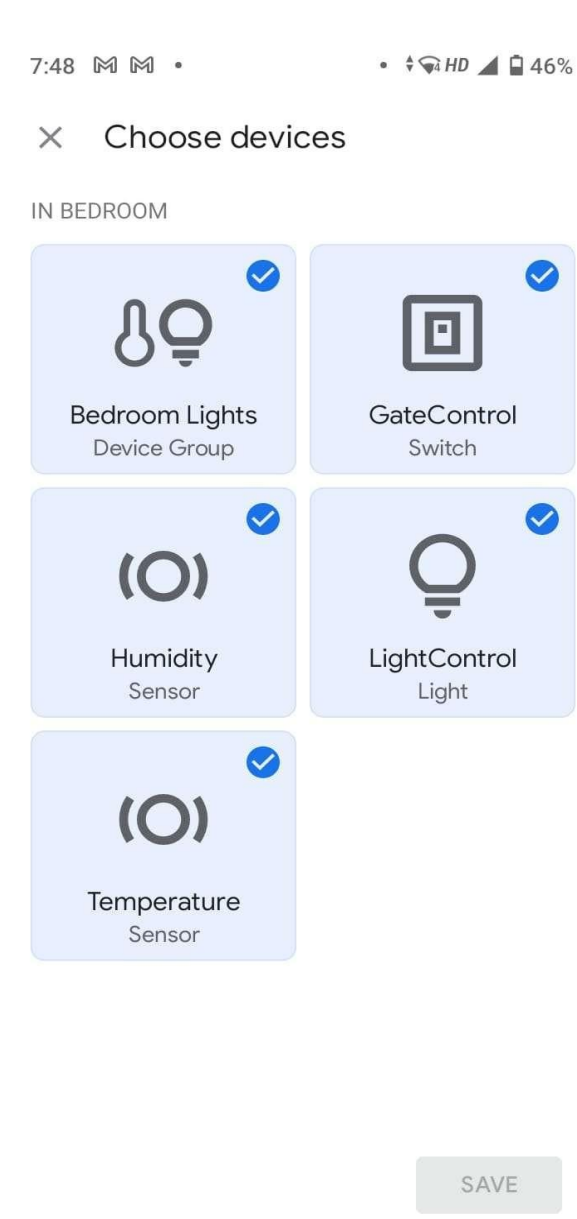
2. DHT11: The **DHT11** is a low-cost, digital temperature and humidity sensor widely used in IoT and automation projects. It integrates a capacitive humidity sensor and a thermistor to measure the surrounding air and outputs a calibrated digital signal via its data pin.

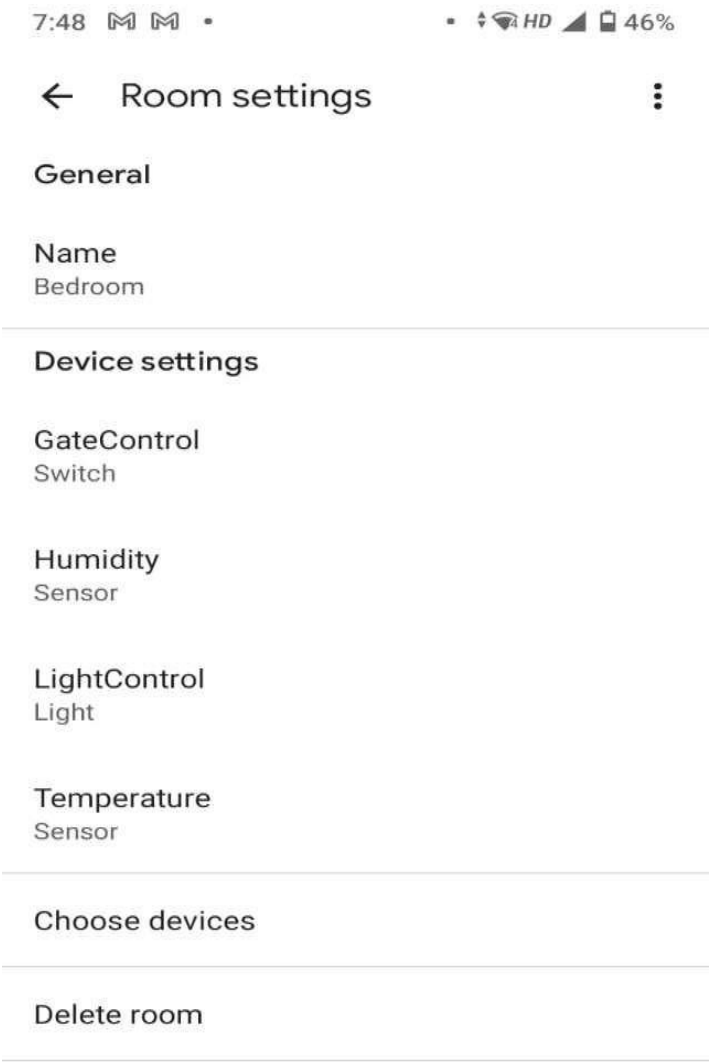
3. LDR Sensor: An **LDR (Light Dependent Resistor)**, also known as a photoresistor, is a light-sensitive electronic component that alters its resistance based on the intensity of light falling on it. The resistance of an LDR decreases with increasing light intensity and increases as the light intensity decreases, making it an effective tool for detecting and measuring light levels.

4. Smart Relay: A **Smart Relay** is an advanced IoT-enabled device designed to control and automate electrical appliances and circuits. It acts as a switch that can be operated remotely through a smartphone app, voice commands, or automated triggers based on pre-set conditions.

5. Servo Motor : A **servo motor** is a rotary actuator or linear actuator that allows precise control of angular or linear position, velocity, and acceleration. It consists of a motor coupled with a sensor for position feedback and a control circuit to manage its motion.

3.2 Visually Engaging User Interface





3.4 Code : Voice Controlled Light, Fan , Gate and Reading Temperature , Humidity

```
#include <ESP8266WiFi.h>
#include <ESP8266WiFiMulti.h>
#include <Servo.h>
#include <DHT.h>
#include "thingProperties.h"

const int led_pin = D1; // Define the relay pin
const int RELAY_PIN = D2;
Servo myServo; // Create servo object to control a servo
const int servoPin = D5; // D5 connected to the servo motor
// Define the DHT sensor type and pin
#define DHTTYPE DHT11
#define DHTPIN D8 // GPIO4 on NodeMCU

// Initialize the DHT sensor
DHT dht(DHTPIN, DHTTYPE);

ESP8266WiFiMulti WiFiMulti; // Object to manage multiple Wi-Fi networks

void setup() {
  // Initialize serial and wait for port to open:
  Serial.begin(9600);
  delay(1500);

    myServo.attach(servoPin); // Attaches the servo on pin 5 to the servo object
  Serial.println("Servo control ready");

  // Initialize pin modes
  pinMode(led_pin, OUTPUT);
  pinMode(RELAY_PIN, OUTPUT);

  // Initialize DHT sensor
  dht.begin();

  // Set Wi-Fi credentials
  WiFiMulti.addAP("Sadalgekar House", "8080808080");

  // Attempt to connect to Wi-Fi
  Serial.println("Connecting to Wi-Fi...");
  while (WiFiMulti.run() != WL_CONNECTED) {
```

```
  delay(500);
  Serial.print(".");
 }
 Serial.println("\nConnected to Wi-Fi");




 // Defined in thingProperties.h
 initProperties();

 // Connect to Arduino IoT Cloud
 ArduinoCloud.begin(ArduinoIoTPreferredConnection);

 /*
    The following function allows you to obtain more information
    related to the state of network and IoT Cloud connection and errors
    the higher number the more granular information you'll get.
    The default is 0 (only errors).
    Maximum is 4
 */
 setDebugMessageLevel(2);
 ArduinoCloud.printDebugInfo();
}
void loop() {
 // Reconnect to Wi-Fi if disconnected
 if (WiFiMulti.run() != WL_CONNECTED) {
  Serial.println("Reconnecting to Wi-Fi...");
  while (WiFiMulti.run() != WL_CONNECTED) {
   delay(500);
   Serial.print(".");
  }
  Serial.println("\nReconnected to Wi-Fi");
 }

 // Update IoT Cloud
 ArduinoCloud.update();

 // Perform sensor readings
 float temp = dht.readTemperature(); // Temperature in Celsius
 float hum = dht.readHumidity();     // Humidity in percentage

 // Check if the readings are valid
 if (isnan(temp) || isnan(hum)) {
  Serial.println("Failed to read from DHT sensor!");
  return;
 }

 // Print the readings to the Serial Monitor
```

```
  Serial.print("Temperature: ");
  Serial.print(temp);
  Serial.println(" °C");
  Serial.print("Humidity: ");
  Serial.println(hum);

  // Update the IoT Cloud variables
  temperature = temp;
  humidity = hum;




  // Control lights and fans
  onLightChange();
  onFanChange();
  onFanChange();
}

void onLightChange() {
  if (light) {
    digitalWrite(led_pin, HIGH); // Turn ON relay
  } else {
    digitalWrite(led_pin, LOW); // Turn OFF relay
  }
}

void onFanChange() {
  digitalWrite(RELAY_PIN, fan ? HIGH : LOW);
}

void onGateControlChange()  {
  if(gateControl){
    // Open the gate
  Serial.println("Opening gate...");
  myServo.write(160); // Rotate the servo to 90 degrees (or another angle to open the gate)
  }else{
    Serial.println("Closing  gate...");
  myServo.write(0); // Rotate the servo back to 0 degrees (or another angle to close the gate)
  }
}
```

CODE for soil moisture detection on Blynk IoT , Theft detection , LDR based light

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClientSecure.h>


#define BLYNK_TEMPLATE_ID "TMPL3-N9iDRMw"
```

```
#define BLYNK_TEMPLATE_NAME "soilmoisture"
#define BLYNK_AUTH_TOKEN "blXmUiIi2kYYJDmpOUFQIlhjWHunCuUr"

#include <BlynkSimpleEsp8266.h>

//ldr pins configs
const int ledPin = D1;
const int ldrPin = D2;

#define SOIL_SENSOR_PIN D0
const int irPin = D6;     // IR sensor OUT pin connected to GPIO12 (D6)
const int buzzerPin = D7; // Buzzer pin connected to GPIO13 (D7)

// WiFi and Telegram credentials
const char* ssid = "Sadalgekar House";         // Replace with your WiFi SSID
const char* password = "8080808080"; // Replace with your WiFi password




const String botToken = "7815883849:AAEYZ-b0IhkY7lXo3V1-TgrA02pKA8Le8Us";  // Replace
with your Telegram Bot Token
const String chatID = "1288673446";        // Replace with your Chat ID

// Create a secure WiFi client
WiFiClientSecure client;

void setup() {
  // Initialize serial communication
  Serial.begin(115200);

  // Initialize Blynk
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, password);

  pinMode(ledPin, OUTPUT);
  pinMode(ldrPin, INPUT);
    // Configure IR sensor and buzzer pins
  pinMode(irPin, INPUT);
  pinMode(buzzerPin, OUTPUT);
  digitalWrite(buzzerPin, LOW); // Ensure buzzer is OFF initially

  // Connect to WiFi
//  connectToWiFi();

  // Enable insecure connection (for simplicity; can be improved with certificates)
  client.setInsecure();

  // Configure the moisture sensor pin
  pinMode(SOIL_SENSOR_PIN, INPUT);
```

```
  Serial.println("Soil Moisture Sensor with Pump Control (Blynk)");
}

void loop() {
  Blynk.run(); // Keep Blynk connected

  // Read moisture level
  int moistureValue = digitalRead(SOIL_SENSOR_PIN);

  // Send moisture level to Blynk
  Blynk.virtualWrite(V1, moistureValue); // Assume V1 is for moisture level display on Blynk

  // Interpret the signal (1: Dry, 0: Wet)
  if (moistureValue == 1) {
    Blynk.virtualWrite(V2, "Soil is dry."); // Assume V2 is for status display
    Serial.println("Soil is dry. Turning ON the pump.");
  } else {
    Blynk.virtualWrite(V2, "Soil has enough moisture."); // Assume V2 is for status display
    Serial.println("Soil has enough moisture. Turning OFF the pump.");
  }
  checkIntrusion();




  int value = digitalRead(ldrPin);
  if(value == 1){
  digitalWrite(ledPin, HIGH);
  Serial.println("Dark mode");
  }
  else{
    digitalWrite(ledPin, LOW);
  }

  delay(500); // Wait for 2 seconds before the next reading
}

void connectToWiFi() {

  Serial.println("Connecting to WiFi...");
  WiFi.mode(WIFI_STA); // Set WiFi mode to Station (client)
  WiFi.begin(ssid, password);


  unsigned long startAttemptTime = millis();
  const unsigned long timeout = 10000; // 10 seconds timeout

  while (WiFi.status() != WL_CONNECTED && (millis() - startAttemptTime) < timeout) {
    delay(100); // Check every 100 ms
    Serial.print(".");
```

```
  }

  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("\nWiFi connected.");
    Serial.print("IP Address: ");
    Serial.println(WiFi.localIP());
  } else {
    Serial.println("\nFailed to connect to WiFi. Retrying...");
  }


}

void checkIntrusion() {
  int intruderDetected = digitalRead(irPin);

  if (intruderDetected == LOW) { // Intruder detected
    digitalWrite(buzzerPin, HIGH); // Turn ON buzzer
    Serial.println("Intruder Alert! Buzzer ON");
    sendTelegramMessage("Intrusion detected at your home! Please check immediately.");
    delay(1000); // Alert duration
  } else {



    digitalWrite(buzzerPin, LOW); // Turn OFF buzzer
    Serial.println("Area Clear. Buzzer OFF");
  }



  delay(50); // Small delay for stability
}

void sendTelegramMessage(String message) {
  if (WiFi.status() == WL_CONNECTED) { // Ensure WiFi is connected
    HTTPClient http;

    // Construct Telegram API URL
    String url = "https://api.telegram.org/bot" + botToken + "/sendMessage?chat_id=" + chatID +
"&text=" + message;
    Serial.println("Request URL: " + url); // Debug the URL

    // Use the secure client with HTTPClient
    http.begin(client, url);
    int httpCode = http.GET(); // Send the GET request

    if (httpCode > 0) {
      Serial.println("Message sent to Telegram.");
```

```
  } else {

    Serial.println("Failed to send Telegram message. HTTP Code: " + String(httpCode));
  }

  http.end(); // Close the connection
 } else {
  Serial.println("WiFi not connected.");
 }
```

Chapter 4 - Results and Outcomes

Under all test conditions, the AGaRMS system exhibited reliable performance in real-time monitoring, gas leakage detection, and automated booking. The MQ-2 gas sensor efficiently detected gas leaks with high sensitivity, and the HX711 load cell offered accurate cylinder weight measurements for accurate estimation of the gas level. It also integrated with the ESP8266 microcontroller to smoothly process data and transmit it to cloud platforms. The Blynk IoT platform allowed for intuitive and easy visualization of gas levels, leakage alerts, and refill status in mobile applications and web dashboards. Users received timely notification, thus responding quickly to potential hazards.

Predictive analytics accurately forecasted gas consumption patterns based on historical usage, environmental factors, and user behavior, thus allowing booking of refills before the gas went dry. This minimized disruptions in supply and ensured service continuity. Automated communication with supplier APIs streamlined the refilling procedure, resulting in significantly fewer manual interventions and related delays. Optimization recommendations on energy use facilitated changes in user patterns to improve efficiency and reduce waste. Tests run in the real world did not reveal significant delays or false alarms. AGaRMS scaled well in various deployment sizes, from small household units to large industrial setups. Minor issues that were found include network latency at some times when transferring data and less compatibility with some of the supplier API formats. Such problems can be solved through advanced network protocols and standardized supplier integrations. Overall, the AGaRMS system successfully integrates IoT technologies, predictive analytics, and user-centric interfaces to provide a robust, efficient, and automated solution for gas level monitoring and management.

4.1 Conclusions

The **Smart Home Automation System using Voice Control** integrates IoT technologies, machine learning, and user-friendly platforms to address the critical needs of convenience, safety, and energy efficiency in modern households. With sensors for motion, temperature, humidity, and light, alongside the NodeMCU ESP8266 microcontroller, the system ensures real-time monitoring and seamless automation of household functions. The cloud-based Blynk IoT platform provides a visual interface for users, enabling intuitive control and monitoring via mobile or web platforms.

The integration of Alexa voice assistant allows for hands-free operation, making the system accessible and easy to use. Predictive analytics enhances the system's intelligence by learning user behavior patterns to optimize energy consumption and automate daily routines proactively. Safety features, such as real-time alerts for fire, intrusion, or device malfunctions, ensure a secure environment, while energy efficiency recommendations contribute to cost savings and environmental benefits.

The **Smart Home Automation System** demonstrated reliability, scalability, and ease of use, making it suitable for residential and small-scale commercial applications. While minor challenges such as network latency and compatibility with specific voice assistant APIs may arise, these can be addressed in future enhancements. The system represents a significant advancement in IoT-based home automation, offering a smarter, safer, and more efficient living experience.

Future work on the system will include expanding predictive analytics to further personalize automation, improving Wi-Fi protocols for better connectivity, and integrating advanced sensors for more comprehensive monitoring. Adding features such as renewable energy integration, AI-based energy optimization, and enhanced user data security with encryption will further improve the system's utility and sustainability. Additionally, incorporating more voice assistant platforms and refining the user interface will enhance the overall experience, making this solution a benchmark in intelligent home automation.