HOME AUTOMATION SYSTEM

GITHUB LINK :https://github.com/Tejaswini72/HOME-AUTOMATION

BY-

IOTB

VIDHARSHANA 22011102120

S TEJASWINI 22011102085

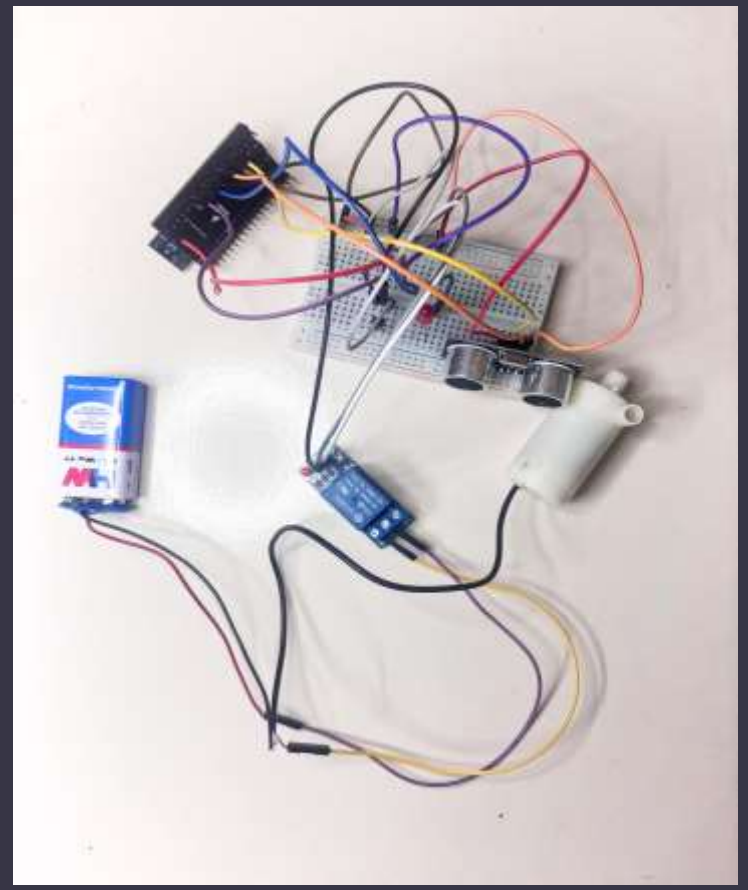
AADHAVAN 22011102080

INDEX

- PROJECT OVERVIEW
- KEY FUNCTIONALITIES
- BENEFITS OF THE SYSTEM
- PRACTICAL APPLICATION OF THE SYSTEM
- LITERATURE SURVEY
- COMPARITIVE ANANLYSIS OF TRADITIONAL AND OUR AUTOMATION SYSTEM
- COMPONENTS
- CODE
- OUTPUT WITH WORKING VIDEO
- FUTURE SCOPE
- REFERENCES

PROJECT OVERVIEW

- This **IoT-Based Home Automation System** project integrates the ESP32 microcontroller with various sensors and components to enable remote control and monitoring of essential household functions. Designed for user convenience, safety, and energy efficiency, this system connects to a Wi-Fi network, allowing users to interact with their home devices from any location through a simple web interface.



KEY FUNCTIONALITIES

- **Remote Control of Devices** : The system allows users to toggle a LED light and a motor (which could represent appliances like fans, pumps, or even window blinds) on or off via a web-based interface. By connecting these components to the ESP32 microcontroller, users can control them in real time, regardless of their physical proximity to the devices. This functionality is particularly useful for managing energy consumption, as it enables users to turn off devices that may have been accidentally left on.
- **Real-Time Temperature Monitoring** : A temperature sensor continuously collects data on the room's ambient temperature. The ESP32 microcontroller processes this data and relays it to the user's web interface, where it is displayed in real time. Monitoring the temperature remotely can help users maintain optimal conditions at home, with the added benefit of setting up alerts for extreme temperatures that may indicate a hazard, such as overheating or potential fire risks. This feature is valuable for protecting household items and electronics that are sensitive to temperature changes, such as plants, pets, and even appliances.

KEY FUNCTIONALITIES

- Proximity Detection with Ultrasonic Sensor : The ultrasonic sensor adds a layer of security by detecting the presence of objects within a specified range. If the sensor registers movement, this information is sent to the user's interface, alerting them to potential nearby activity. The application of this functionality is particularly useful for sensitive areas in the home, such as lockers or safes, where proximity detection acts as a deterrent or alert system when the homeowner is away. The sensor can help enhance security by notifying users of any unusual activity around these critical zones .
- Centralized Control through Wi-Fi : The ESP32's built-in Wi-Fi capability enables it to connect to the internet and transmit data between the system's components and the user's device. This connectivity is crucial for enabling remote control and real-time data access, which makes the system truly automated and accessible. Users can view all sensor data, control devices, and receive alerts on any device connected to the internet, including smartphones, laptops, and tablets.

BENEFITS OF THE SYSTEM

- **Convenience**: Users can remotely monitor and control lights, temperature, and security features in their home, making it easier to maintain a comfortable and safe environment.
- **Energy Efficiency**: The system's real-time control helps users reduce energy consumption by ensuring that lights and devices are only on when needed.
- **Enhanced Security**: The ultrasonic sensor acts as a proactive security measure, alerting users to potential unauthorized access.
- **User-Friendly Interface**: The web interface offers a simple and intuitive control hub where users can view data and manage device settings at a glance.

PRACTICAL APPLICATION

1. **Home Security**:

   The ultrasonic sensor's motion detection capability makes it ideal for monitoring sensitive areas. Placing the sensor near valuable assets like lockers or entryways can provide alerts to users when someone approaches, thus enhancing security when they are away.

2. **Environmental Monitoring**:

   With temperature monitoring, users can track room conditions and ensure a comfortable environment, which is particularly useful in rooms with delicate items, such as plants or electronics. Alerts for unusual temperature changes help users take quick preventive measures against potential hazards.

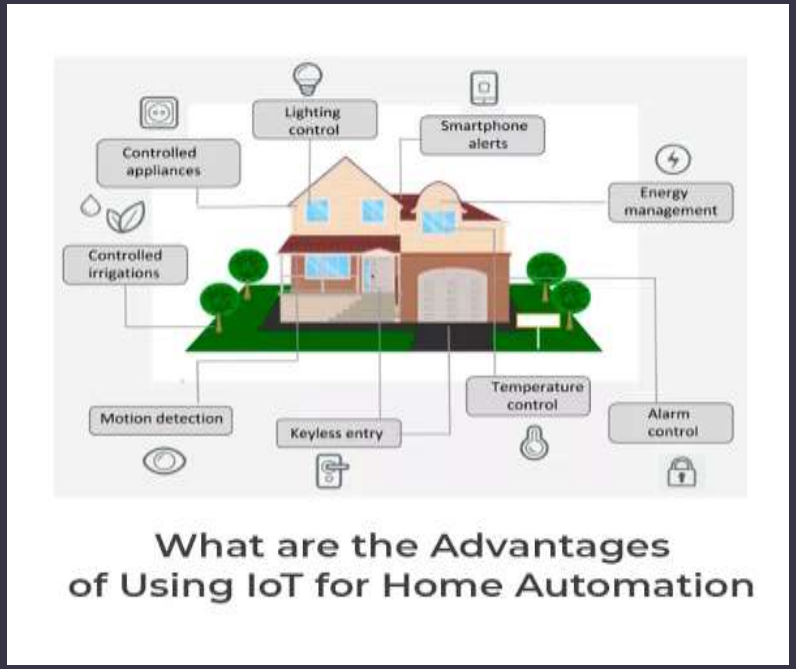
3. **Appliance Automation**:

   By automating routine tasks, such as turning lights on and off or managing fans or pumps, users save time and ensure devices are used only when necessary, leading to significant energy savings over time.

LITERATURE SURVEY

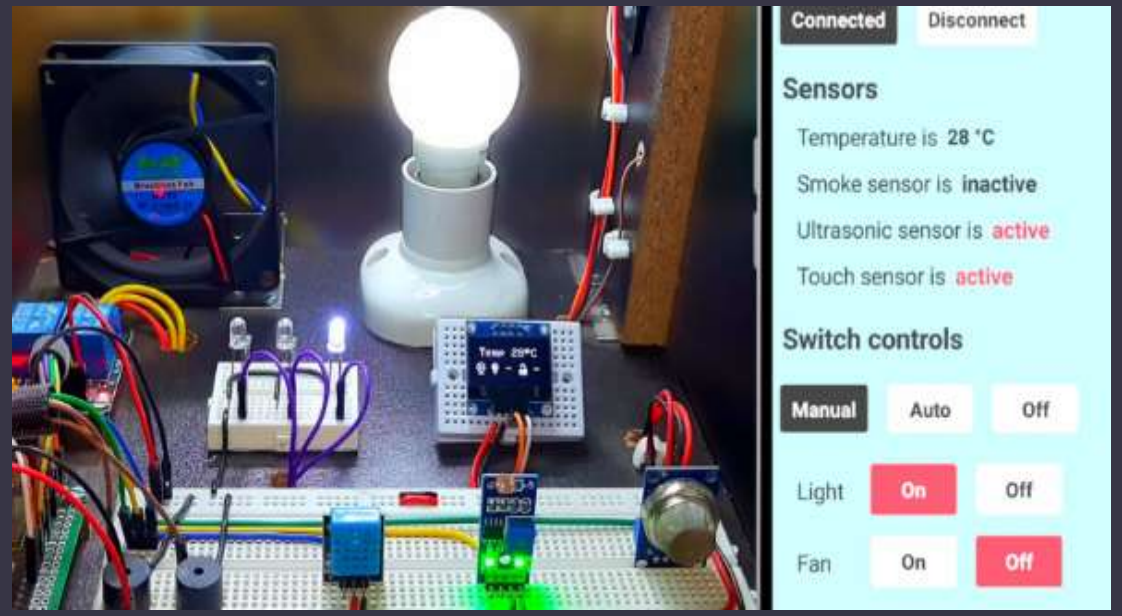
HOME AUTOMATION

Home automation systems have gained significant attention over the past decade, driven by advancements in IoT, wireless technologies, and microcontroller capabilities. A smart home automates control of various appliances and systems, enhancing convenience, energy efficiency, and security. The ESP32 microcontroller, known for its built-in Wi-Fi and Bluetooth, is increasingly used for automation due to its low cost, reliability, and versatility in connecting multiple sensors and devices.



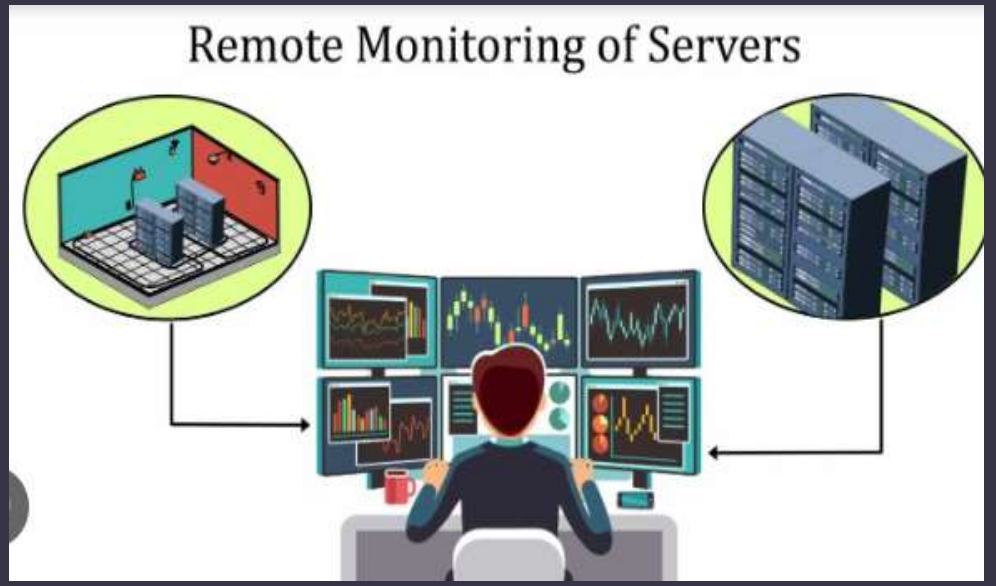
ESP32 IN HOME AUTOMATION

IoT has become the backbone of modern home automation, enabling devices to communicate over the internet for remote control and monitoring. Studies have shown that IoT-enabled automation systems contribute significantly to energy efficiency by allowing dynamic control over lighting, heating, and other appliances based on environmental data (Dewi et al., 2022). IoT technology has also facilitated mobile applications that enable users to monitor and control devices from anywhere, enhancing user experience and control.



OCCUPANCY DETECTION AND ROOM MONITORING

The ESP32 microcontroller has become a popular choice for IoT-based automation projects because it combines wireless connectivity with adequate processing power and low energy consumption. Research highlights ESP32's utility in real-time applications, such as remote lighting control, climate management, and occupancy detection (Ghosh & Pal, 2021). ESP32 is also compatible with a wide variety of sensors (temperature, humidity, motion), making it suitable for developing multi-functional, scalable automation systems.



COMPARITIVE ANALYSIS OF TRADITIONAL AND OUR HOME AUTOMATION SYSTEM

Feature	Traditional Non-Automated Household	Partially Automated Household	Our IoT-Based Home Automation System
Control Accessibility	Manual operation of lights, fans, and appliances on-site	Limited remote control; some devices may be controlled via remote apps	Complete remote control via Wi-Fi; accessible through a single interface on mobile or web platforms
Energy Efficiency	High energy usage; devices often left on unintentionally	Some energy savings through time-based schedules	Optimized energy usage with occupancy-based controls and remote monitoring, allowing users to switch devices on/off as needed
Temperature Monitoring	Manual adjustment of fans or heating, no temperature tracking	Basic thermostat control	Real-time temperature monitoring with data accessible remotely; allows proactive climate control adjustments
Security & Surveillance	Limited security; manual locks, no occupancy detection	Basic security, possibly with CCTV	Enhanced security with ultrasonic sensor for occupancy detection, sending alerts if someone approaches sensitive areas

Feature	Traditional Non-Automated Household	Partially Automated Household	Our IoT-Based Home Automation System
Convenience	Requires physical presence for all tasks	Some convenience features (e.g., automatic lights)	Full convenience with remote control of devices, real-time updates, and multi-functional controls in a single platform
Scalability	Not easily scalable; each new device requires manual setup	Limited scalability; adding new automation requires additional setup	Highly scalable; ESP32 supports multiple sensors and can integrate additional devices with minimal setup
Cost-Effectiveness	No upfront cost, but high energy bills over time	Medium cost; automation equipment is usually proprietary	Cost-effective with low-cost ESP32 and sensors, and offers significant savings on energy and maintenance over time

COMPONENTS

ESP32 MICROCONTROLLER

- **Description**: The ESP32 is a powerful microcontroller with built-in Wi-Fi and Bluetooth capabilities, ideal for IoT applications. It has dual-core processors, low power consumption, and an extensive I/O system that makes it highly adaptable for connecting various sensors and modules.

- **Role in the Project**: Serving as the central processing unit, the ESP32 enables communication between the user interface and the hardware components. Its Wi-Fi functionality allows remote access, letting users control devices and receive sensor data in real time. The microcontroller also manages power consumption, which is crucial for long-term efficiency in IoT systems.

LED LIGHT

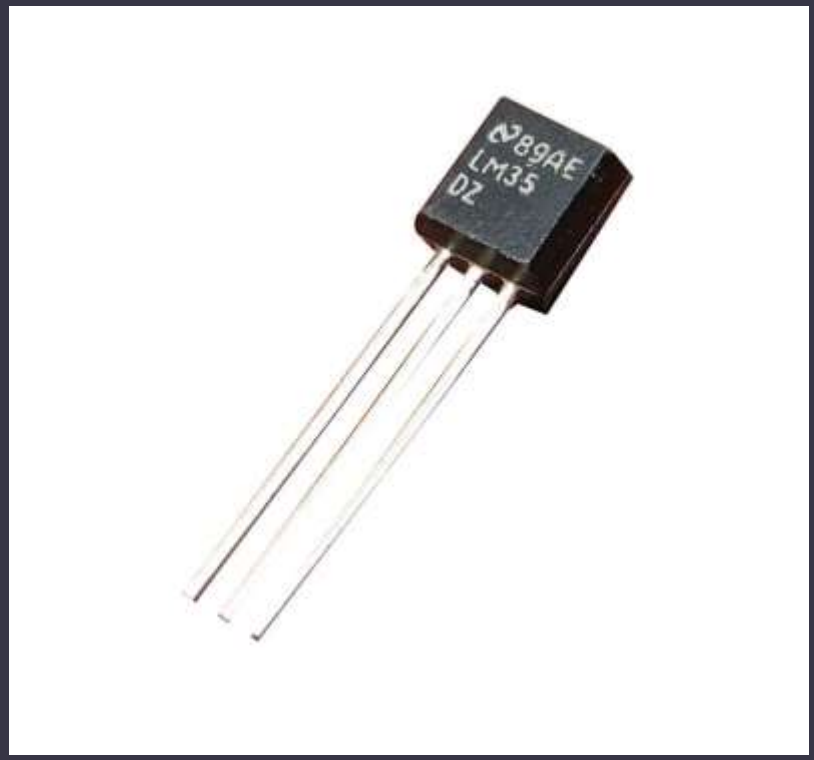
- Description: A simple, low-power light-emitting diode (LED) is connected to the ESP32, enabling it to be controlled remotely. LEDs are energy-efficient and have long operational lives, making them ideal for IoT applications where power conservation is a priority.
- Role in the Project: The LED light showcases the system's capability for remote control. It acts as an example of lighting automation, which is commonly used in smart homes for both convenience and security.

MOTOR

- **Description**: The motor is connected to the ESP32, allowing remote control of its on/off states. Although a simple motor is used here, it could represent various types of motorized devices in a home, such as a fan, water pump, or automatic curtain system.
- **Role in the Project**: The motor exemplifies the ability to control larger home devices that require power regulation. Motorized devices can be particularly useful for automating tasks such as watering plants, managing ventilation, or other daily routines.

TEMPERATURE SENSOR

- **Description**: A temperature sensor (commonly a DHT11 or DS18B20 sensor) continuously measures the room's ambient temperature. These sensors are reliable, compact, and offer precise temperature readings.
- **Role in the Project**: The temperature sensor monitors the room's temperature, providing the user with real-time data. This component is essential for environmental monitoring, enabling users to keep track of room conditions, which can be helpful in temperature-sensitive rooms.



ULTRASONIC SENSOR

- **Description**: The ultrasonic sensor (such as the HC-SR04 model) emits ultrasonic sound waves to measure the distance of objects within its range. It calculates distance by sending out a sound wave and measuring the time it takes for the echo to return.
- **Role in the Project**: The ultrasonic sensor acts as a basic security measure by detecting nearby movement. It's typically positioned near valuable or sensitive areas, such as lockers or entryways, where it can alert users of any nearby activity.

CODE

sketch_nov10a.ino

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

// Network credentials for Access Point
const char* ssid = "ESP32-Control-Panel";
const char* password = "12345678";

// Pin definitions
const int LED_PIN = 25;        // Built-in LED on most ESP32 boards
const int PUMP_PIN = 26;       // Relay control pin for water pump
const int LM35_PIN = 32;       // Analog pin for LM35 temperature sensor
const int TRIG_PIN = 13;       // Ultrasonic sensor trigger pin
const int ECHO_PIN = 14;       // Ultrasonic sensor echo pin

// Constants for stranger detection
const int DISTANCE_THRESHOLD = 5;  // Distance threshold in cm
bool strangerDetected = false;

// Web server on port 80
WebServer server(80);

void setup() {
  Serial.begin(115200);

  // Initialize pins
  pinMode(LED_PIN, OUTPUT);
  pinMode(PUMP_PIN, OUTPUT);
  pinMode(LM35_PIN, INPUT);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);

  // Initially turn off LED and pump
  digitalWrite(LED_PIN, LOW);
  digitalWrite(PUMP_PIN, HIGH);  // Initialize pump relay as OFF (HIGH)

```

```
  // Configure ESP32 as Access Point
  WiFi.softAP(ssid, password);

  Serial.println("Access Point Started");
  Serial.print("IP Address: ");
  Serial.println(WiFi.softAPIP());

  // Define web server routes
  server.on("/", handleRoot);
  server.on("/led", handleLED);
  server.on("/pump", handlePump);
  server.on("/temperature", handleTemperature);
  server.on("/stranger", handleStranger);

  // Start web server
  server.begin();
}

void loop() {
  server.handleClient();
  checkForStranger();
  delay(100);  // Small delay to prevent watchdog timer issues
}

// Read temperature from LM35
float readTemperature() {
  int rawValue = analogRead(LM35_PIN);
  float voltage = (rawValue / 4095.0) * 3.3;
  float temperatureC = (voltage * 1000.0) / 10.0;
  return temperatureC;
}

```

```
68  // Measure distance using ultrasonic sensor
69  int measureDistance() {
70    digitalWrite(TRIG_PIN, LOW);
71    delayMicroseconds(2);
72    digitalWrite(TRIG_PIN, HIGH);
73    delayMicroseconds(10);
74    digitalWrite(TRIG_PIN, LOW);
75
76    long duration = pulseIn(ECHO_PIN, HIGH);
77    int distance = duration * 0.034 / 2;  // Calculate distance in cm
78
79    return distance;
80  }
81
82  void checkForStranger() {
83    int distance = measureDistance();
84    Serial.println(distance);
85    strangerDetected = (distance < DISTANCE_THRESHOLD);
86  }
87
```

```
void handleRoot() {
  String html = R"(
<!DOCTYPE html>
<html>
<head>
    <meta name='viewport' content='width=device-width, initial-scale=1.0'>
    <title>ESP32 Smart Control Panel</title>
    <style>
        :root {
            --primary-color: #4a90e2;
            --danger-color: #e74c3c;
            --success-color: #2ecc71;
            --warning-color: #f1c40f;
            --text-color: #2c3e50;
            --bg-color: #f5f7fa;
        }

        * {
            box-sizing: border-box;
            margin: 0;
            padding: 0;
            font-family: 'Inter', -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, sans-serif;
        }

        body {
            background: var(--bg-color);
            min-height: 100vh;
            padding: 2rem;
            color: var(--text-color);
        }

        .dashboard {
            max-width: 1200px;
            margin: 0 auto;
        }

```

```
.header {
    text-align: center;
    margin-bottom: 2rem;
}

.header h1 {
    font-size: 2.5rem;
    color: var(--primary-color);
    margin-bottom: 0.5rem;
}

.grid {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));
    gap: 1.5rem;
    margin-bottom: 2rem;
}

.card {
    background: white;
    border-radius: 1rem;
    padding: 1.5rem;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
    transition: transform 0.3s ease;
}

.card:hover {
    transform: translateY(-5px);
}

.card h2 {
    font-size: 1.25rem;
    margin-bottom: 1rem;
    color: var(--primary-color);
}

```

```
.metric {
    font-size: 3rem;
    font-weight: bold;
    color: var(--text-color);
    margin: 1rem 0;
}

.controls {
    display: flex;
    gap: 1rem;
    flex-wrap: wrap;
}

.button {
    flex: 1;
    min-width: 120px;
    padding: 1rem;
    border: none;
    border-radius: 0.5rem;
    font-size: 1rem;
    font-weight: 600;
    cursor: pointer;
    transition: all 0.3s ease;
    text-decoration: none;
    text-align: center;
    color: white;
    background: var(--primary-color);
}

.button:hover {
    opacity: 0.9;
    transform: translateY(-2px);
}

.alert {
    padding: 1rem;
```

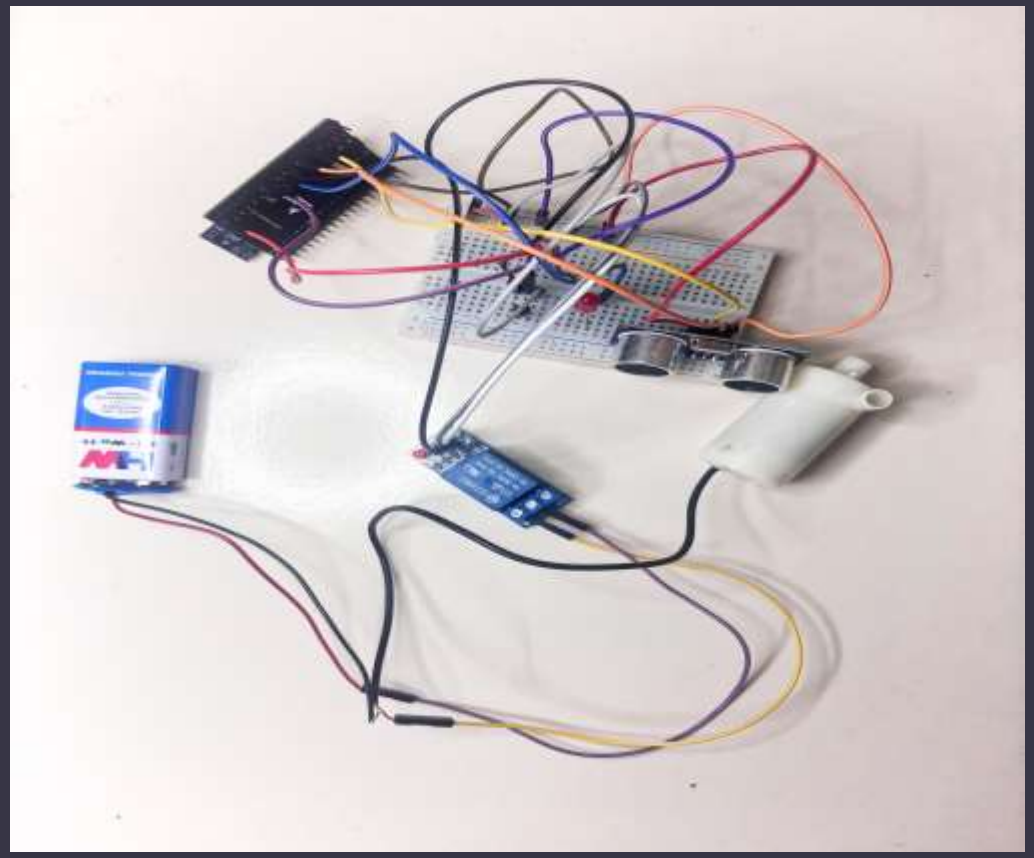
```
        .alert {
            padding: 1rem;
            border-radius: 0.5rem;
            background: var(--danger-color);
            color: white;
            margin-bottom: 1rem;
            display: none;
        }

        .alert.show {
            display: block;
            animation: slideIn 0.3s ease;
        }

        @keyframes slideIn {
            from {
                transform: translateY(-20px);
                opacity: 0;
            }
            to {
                transform: translateY(0);
                opacity: 1;
            }
        }

        @media (max-width: 768px) {
            .grid {
                grid-template-columns: 1fr;
            }

            .button {
                width: 100%;
            }
        }
    </style>
    <script>
```

```
    <script>
        function updateMetrics() {
            // Update temperature
            fetch('/temperature')
                .then(response => response.text())
                .then(data => {
                    document.getElementById('temperature').innerHTML = parseFloat(data).toFixed(1);
                });

            // Check for stranger detection
            fetch('/stranger')
                .then(response => response.text())
                .then(data => {
                    const alert = document.getElementById('strangerAlert');
                    if (data === '1') {
                        alert.classList.add('show');
                    } else {
                        alert.classList.remove('show');
                    }
                });
        }

        // Update metrics every 2 seconds
        setInterval(updateMetrics, 2000);
    </script>
</head>
<body onload='updateMetrics()'>
    <div class="dashboard">
        <div class="header">
            <h1>ESP32 Smart Control Panel</h1>
        </div>

        <div id="strangerAlert" class="alert">
            Motion Detected! Someone might be nearby.
        </div>

```

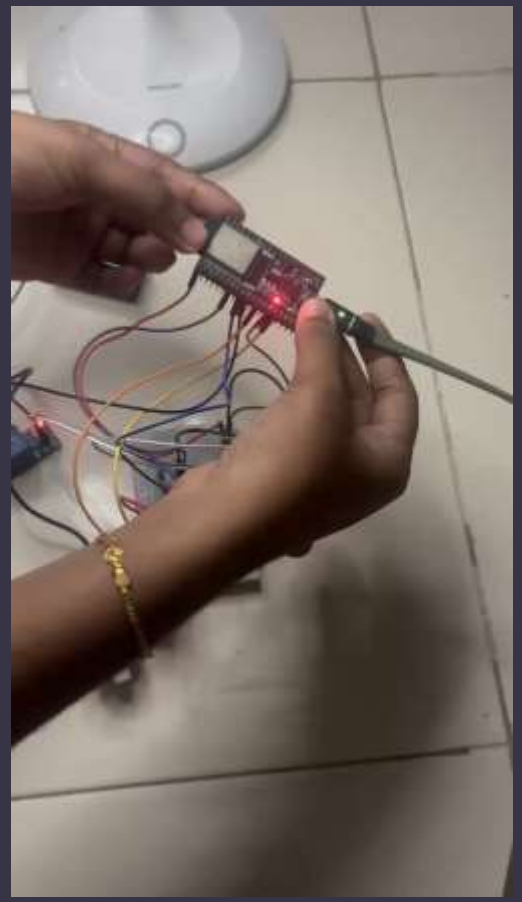
```
<body onload='updateMetrics()'>
    <div class="dashboard">
        <div class="header">
            <h1>ESP32 Smart Control Panel</h1>
        </div>

        <div id="strangerAlert" class="alert">
            Motion Detected! Someone might be nearby.
        </div>

        <div class="grid">
            <div class="card">
                <h2>Temperature Monitor</h2>
                <div class="metric">
                    <span id="temperature">--</span>
                </div>
            </div>

            <div class="card">
                <h2>System Controls</h2>
                <div class="controls">
                    <a href="/led" class="button">Toggle LED</a>
                    <a href="/pump" class="button">Toggle Pump</a>
                </div>
            </div>
        </div>
    </div>
</body>
</html>
)";
  server.send(200, "text/html", html);
}
```

```
287
288 void handleLED() {
289   static int ledState = LOW;
290   ledState = (ledState == LOW) ? HIGH : LOW;
291   digitalWrite(LED_PIN, ledState);
292   server.sendHeader("Location", "/");
293   server.send(303);
294 }
295
296 void handlePump() {
297   static int pumpState = HIGH;
298   pumpState = (pumpState == HIGH) ? LOW : HIGH;
299   digitalWrite(PUMP_PIN, pumpState);
300   server.sendHeader("Location", "/");
301   server.send(303);
302 }
303
304 void handleTemperature() {
305   float temp = readTemperature();
306   server.send(200, "text/plain", String(temp, 1));
307 }
308
309 void handleStranger() {
310   server.send(200, "text/plain", strangerDetected ? "1" : "0");
311 }
```

OUTPUT:




ESP32 Smart Control Panel
Temperature Monitor
39.8
System Controls
Toggle LED
Toggle Pump



WORKING VIDEO

DRIVE LINK:
https://drive.google.com/drive/folders/1tGRKxlc2iBa1xT0Z_-PB-nIUZeapnka1?usp=sharing

FUTURE SCOPE OF PROJECT:

Several enhancements could expand the functionality and robustness of this home automation system:

1. **Expanded Sensor Network**: Additional sensors, such as gas leak detectors, humidity sensors, and smoke detectors, could increase safety by providing early warnings for various household dangers.
2. **Advanced Security Features**: Incorporating cameras or biometric systems would enable more comprehensive home security, especially for monitoring entry points.
3. **Machine Learning for Predictive Analysis**: By adding predictive analytics, the system could learn user patterns and optimize settings, such as lighting or temperature, based on historical data, enhancing both comfort and energy savings.
4. **Voice Control Integration**: Integrating voice assistants, like Alexa or Google Assistant, would make the system more user-friendly and accessible, enabling hands-free operation.
5. **Mobile App Development**: Creating a dedicated mobile app would centralize control and monitoring of the system, providing an intuitive interface for users to receive alerts, view data, and control devices seamlessly.

REFERENCES

- Dewi, R., et al. (2022). "Impact of IoT on Energy Efficiency in Home Automation Systems." Journal of IoT Applications.

- Ghosh, S., & Pal, R. (2021). "ESP32 in IoT-Based Smart Home Applications." International Journal of Electronics and Communication.

- Patel, M., et al. (2021). "Room Occupancy Detection using Ultrasonic Sensors in Smart Homes." Journal of Automation and Control Engineering.

- Johnson, D., & Singh, A. (2020). "Temperature Control Systems in Home Automation." Environmental Engineering and Management Journal.

- Lee, J., & Kim, H. (2020). "Mobile Application Development for IoT-Based Home Automation." International Journal of Smart Home Technology.

- Singh, P., & Gupta, R. (2021). "Machine Learning and AI in Future Home Automation Systems." Journal of Artificial Intelligence and Robotics.

THANK YOU