PROJECT REPORT

Smart Home Automation System Using NodeMCU and Sensor Integration

By: ***Amavasya Manoj Kumar Reddy***

ABSTRACT

The Smart Home Automation System designed in this project aims to promote comfort, efficiency, and security through the integration of smart control of important home appliances and systems. Integrated around the NodeMCU ESP8266 microcontroller that has both general input/output function and Wi-Fi, the system manages lighting, motion, temperature sensing, and simple physical actuation. The utilization of low-cost sensors like the Light Dependent Resistor (LDR) and an LM393 comparator, IR motion sensors (TSSP58038), and NTC thermistors enables the NodeMCU to sense conditions real-time and make sound judgments regarding appliance control.

The system automatically responds to external changes. For instance, variation of ambient light intensity can automatically switch on lighting systems, whereas changes in movement sensed by the IR sensor within specified timings can switch on alarm systems or activate servo motors to mimic door movement. NTC-Based thermistor analog data allows the system to react to real-time temperature changes, and actions like starting fans or sending messages in the event of extreme temperatures are possible. In addition to this, manual override and user control through push buttons allow interaction with various outputs such as LEDs, buzzers, and servo motors.

A key aspect is wireless monitoring and access, which allows for real-time control via smartphones or computers through a local network or broader internet based on IoT protocols such as HTTP and MQTT. The project not only demonstrates the feasibility of constructing a self-sustaining automation system with fundamental sensors and controllers but also provides a scalable base for future development such as AI-based control, voice commands, cloud connectivity, and smartphone applications.

Overall, this system shows the capability to convert houses into smart homes through the use of cheap, dependable, and programmable hardware. It fulfills actual-world requirements while being versatile enough for domestic and small-scale business applications.

INTRODUCTION

Advances in embedded systems and Internet of Things (IoT) architectures have radically changed the way humans interact with their homes. Those days when lighting, air-conditioning, and heating units used to function merely with switches and thermostats are over. Nowadays, the term "smart home" describes a smart home environment where home appliances and systems are connected and dynamically react to user command, environmental conditions, scheduling, and remote control. A smart home not only makes life more convenient but also optimizes power utilization and safety through automatic repetition or critical tasks.

The global energy crisis and mounting pressure towards being sustainable ensure that home automation is not just a luxury item — it becomes essential. In such a scenario, the Smart Home Automation System designed through this project is a low-cost, scalable, and dependable prototype through the use of commonly available parts and open-source software. The system is centered on the NodeMCU microcontroller, which uses the ESP8266 chip with the feature of Wi-Fi built in for wireless communication. NodeMCU is renowned for being versatile, low power, having digital and analog interfaces, and easy to program using the Arduino IDE.

Input sensing is performed through different sensors such as the LDR with LM393 comparator module, an NTC temperature sensor, and IR (infrared) motion sensors. They automate operations like turning lights depending on room lighting, marking unauthorized movement for security, or controlling appliances according to thermal information. Outputs like LEDs, buzzers, and servo motors are used as real-time alerts or actuators in the system.

Besides, the user can control the system using push buttons or through a mobile/web interface on Wi-Fi. This provides the user with local and remote access — giving immediate feedback, being customizable, and having override features. Together, the project showcases an implementable perspective toward the application of cost-effective, intuitive, and smart control in residential homes..

SYSTEM DESIGN

The Smart Home Automation System is developed with a modular strategy to split its operations into three major subsystems — sensing, processing, and actuation. The role of each of these phases is significant in obtaining a responsive and reliable result. This section describes in detail each component's functionality, interaction, and control logic utilized in the system.

1. Input Sensing

The system leverages several input sensors to gather environmental data:

- **LDR with LM393 Comparator**: The light sensor functions as a trigger for lighting automation. A potentiometer connected to the comparator allows for adjusting the light sensitivity. When ambient light falls below a threshold, the comparator outputs a LOW signal, prompting the NodeMCU to activate lights.
- **IR Motion Sensor (TSSP58038)**: This sensor is used primarily for security and automation. It emits infrared rays that reflect back when an object moves in its field. When motion is detected, a HIGH signal is sent to the NodeMCU, prompting activation of alerts and potentially moving a servo motor.
- **NTC Thermistor**: Serves as a temperature sensor. The analog value from the thermistor, representing the surrounding temperature, is read via the ADC (analog-digital converter) pin (A0) on the NodeMCU.
- **Push Buttons**: Provide manual interaction, allowing users to override automatic control or trigger devices regardless of environmental conditions.

2. Processing and Control

The NodeMCU microcontroller reads input values, runs logic statements written using Arduino IDE, and accordingly controls the outputs. The logic performs conditional checks on sensor values, activates outputs, and debounces buttons.

3. Output/Actuators

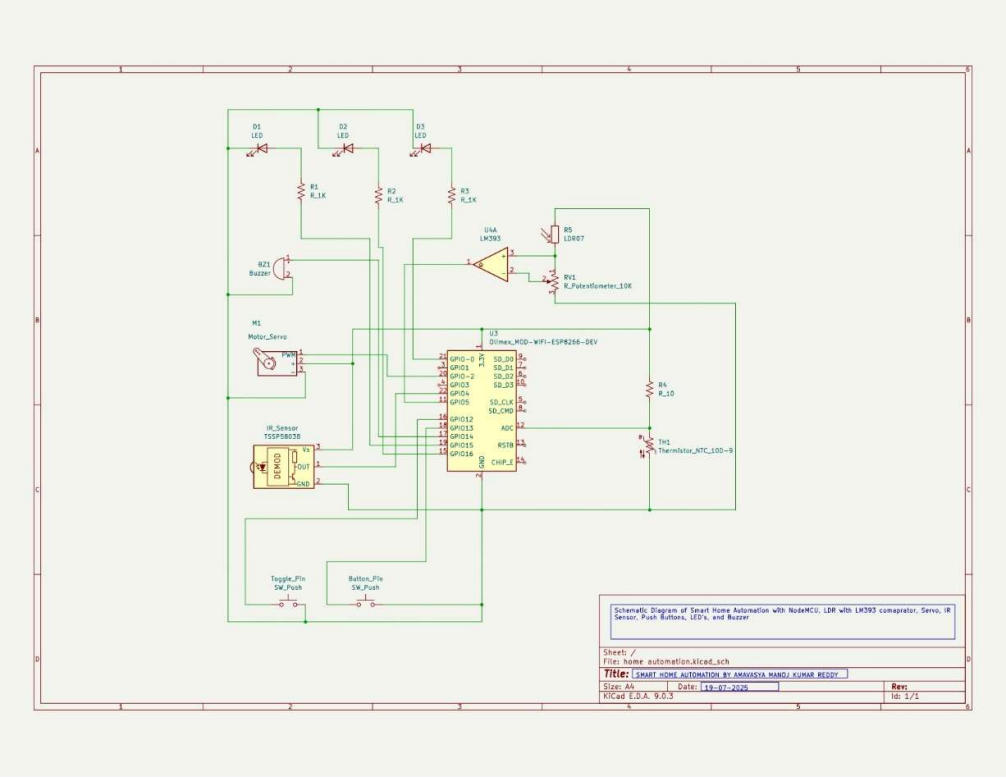
- **LEDs**: Indicate status or serve as automated lights.
- **Buzzer**: Activated during motion detection or abnormal temperature.
- **Servo Motor**: Simulates real-world mechanical action like unlocking doors.
- **Wireless Communication**: Enables control via HTTP, Blynk, or MQTT from a mobile phone or web dashboard.

HARDWARE AND SOFTWARE IMPLEMENTATION

The proper functioning of the system depends on a well-designed implementation of hardware and software components. The system is meant to operate independently and optionally connect via the internet for greater usability.

Hardware Implementation

- **NodeMCU (ESP8266)**: Interfaces to sensors and outputs through GPIO pins. GPIO pins are set for digital input/output (buttons, LEDs, IR sensors), whereas the `A0` pin handles analog signals from the thermistor.
- **Breadboard and Power Supply**: A regulated 5V power supply is utilized for secure supply to NodeMCU and peripheral devices. Power rails are routed through a breadboard for easy prototyping.
- **Circuit Assembly**: Resistors, capacitors, potentiometers, and jumper wires are utilized for voltage division, signal conditioning, and safe connections. The LM393 comparator module generates digital outputs from the LDR, and the potentiometer assists in sensitivity fine-tuning.



Software Implementation

The **Arduino IDE** is used to program the NodeMCU with the required logic using standard C/C++ syntax and ESP8266 libraries.

Key Functions:

- digitalRead() and digitalWrite() – To receive/send HIGH/LOW signals.
- analogRead() – To interpret analog data from the thermistor.
- Servo.attach() and Servo.write() – To control servo motors.
- Wi-Fi libraries including ESP8266WiFi.h – Used to connect with SSID and password.
- Optional extensions using ESP8266WebServer.h or PubSubClient.h – For remote interaction via HTTP or MQTT.

Arduino Code:

```
#include <Servo.h>


// Pin Definitions
#define IR_PIN D2
#define RED_LED D8
#define BUZZ D5
#define GREEN_LED D3
#define SERVO_PIN D4
#define BUTTON_PIN D7
#define TOGGLE_PIN D6
#define BLUE_LED D0
#define LDR_DO D1


Servo myServo;


// Flags & States
bool doorOpened = false;
bool waitingToClose = false;
bool isDark = false;
bool isMotionDetected = false;
bool securityMode = false;
```

```
bool prevTempHigh = false;

unsigned long openTime = 0;
unsigned long lastMotionBuzzTime = 0;

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

  pinMode(IR_PIN, INPUT);
  pinMode(RED_LED, OUTPUT);
  pinMode(BUZZ, OUTPUT);
  pinMode(GREEN_LED, OUTPUT);
  pinMode(BLUE_LED, OUTPUT);
  pinMode(LDR_DO, INPUT);
  pinMode(BUTTON_PIN, INPUT_PULLUP);
  pinMode(TOGGLE_PIN, INPUT_PULLUP);

  myServo.attach(SERVO_PIN);
  myServo.write(90); // Neutral position

  Serial.println("🔧 System Initialized");
  delay(1000);
}

void loop() {
  checkSecurityToggle();   // Check for toggle press

  if (securityMode) {
    // 🔒 Security Mode: Motion detection only
    motion();
    digitalWrite(GREEN_LED, LOW);
    digitalWrite(BLUE_LED, LOW);
    myServo.write(90); // Keep servo neutral
```

```
} else {
  // 🟢 Normal Mode
  static bool lastMotion = false;
  static bool lastLight = false;
  static bool lastTempHigh = false;

  isMotionDetected = false;
  isDark = false;

  temperature(); // Sets prevTempHigh
  motion();
  light();
  door();

  if (isMotionDetected != lastMotion || isDark != lastLight || prevTempHigh != lastTempHigh) {
    Serial.println("--- Measurements ---");
    Serial.print("Motion: ");
    Serial.println(isMotionDetected ? "Detected" : "Not Detected");
    lastMotion = isMotionDetected;

    Serial.print("Light: ");
    Serial.println(isDark ? "Dark" : "Bright");
    lastLight = isDark;

    Serial.print("Temperature: ");
    Serial.println(prevTempHigh ? "HIGH" : "NORMAL");
    lastTempHigh = prevTempHigh;
  }
 }

 delay(100);
}
```

```
// 🔁 Toggle security mode on button press
void checkSecurityToggle() {
  static bool lastState = HIGH;
  static unsigned long lastToggleTime = 0;
  const unsigned long debounceDelay = 200;

  bool currentState = digitalRead(TOGGLE_PIN);
  if (lastState == HIGH && currentState == LOW) {
    if (millis() - lastToggleTime > debounceDelay) {
      securityMode = !securityMode;
      Serial.print("🔐 Security Mode: ");
      Serial.println(securityMode ? "ENABLED" : "DISABLED");
      lastToggleTime = millis();
    }
  }
  lastState = currentState;
}

// 🔥 Raw NTC thermistor via voltage divider on A0
void temperature() {
  int tempValue = analogRead(A0);
  if (tempValue < 400) {
    digitalWrite(BLUE_LED, HIGH);
    tone(BUZZ, 1000, 200); // Buzz 2s
    prevTempHigh = true;
  } else {
    digitalWrite(BLUE_LED, LOW);
    prevTempHigh = false;
  }
}

// 🚨 Motion detection via PIR sensor
void motion() {
```

```
  int motionVal = digitalRead(IR_PIN);
  isMotionDetected = (motionVal == LOW); // Active LOW

  if (isMotionDetected) {
    digitalWrite(RED_LED, HIGH);
    if (millis() - lastMotionBuzzTime > 2000) {
      tone(BUZZ, 4000, 200);
      lastMotionBuzzTime = millis();
    }
    digitalWrite(RED_LED, LOW);
  } else {
    digitalWrite(RED_LED, LOW);
  }
}

// 🌓 LDR-based light detection (digital)
void light() {
  bool lightValue = digitalRead(LDR_DO); // HIGH = dark if module has comparator
  isDark = lightValue;
  digitalWrite(GREEN_LED, isDark ? HIGH : LOW);
}

// 🚪 Servo-controlled door operation with button
void door() {
  if (digitalRead(BUTTON_PIN) == LOW && !doorOpened) {
    myServo.write(0);      // Open
    delay(1000);
    myServo.write(90);     // Stop
    digitalWrite(BLUE_LED, HIGH);
    delay(200);
    digitalWrite(BLUE_LED, LOW);

    openTime = millis();
```

```
  doorOpened = true;
  waitingToClose = true;

  delay(500);
  while (digitalRead(BUTTON_PIN) == LOW) delay(10);
 }

 if (waitingToClose && millis() - openTime >= 10000) {
  myServo.write(180);     // Close
  delay(1000);
  myServo.write(90);      // Stop
  doorOpened = false;
  waitingToClose = false;
 }

 // Failsafe close
 if (doorOpened && !waitingToClose && millis() - openTime >= 20000) {
  myServo.write(180);
  delay(1000);
  myServo.write(90);
  doorOpened = false;
 }
}
```

Once the circuit is energized and code loaded, the system operates in an autonomic mode, responding to inputs and generating corresponding outputs. Logic optimization provides minimum response time delay and optimal use of memory. Interrupt handling and power-saving sleep modes can be included as well to improve performance in long-term applications.

The modular hardware-software architecture makes it simple to debug, upgrade, and incorporate new features later.

RESULTS, DISCUSSIONS AND APPLICATION

Results

The Smart Home Automation System prototype was tested extensively and produced encouraging results. The LDR module operated correctly to turn on LED lights according to the surrounding light intensity. The cut-in and cut-off voltages were well controlled by the potentiometer threshold value. The IR sensor worked successfully to sense motion at 2 meters range, as expected, turning on the buzzer and servo motor.

Thermistor measurements were readily converted from analog units to temperature in degrees using simple calibration curves. Push buttons and LEDs functioned well, yielding manual testing switches and discrete indications of system states. Servo motions were smooth, and positions could be accurately controlled with `Servo.write()`.

Discussion

The system was a responsive and affordable deployment of fundamental smart home capabilities. One of the key elements of its success lies in the effective behavior of sensors in real-world settings, particularly the precise triggering of outputs from environmental threshold conditions. NodeMCU's Wi-Fi capability and processing power enabled stable real-time operation, and web-based monitoring was realized using both a basic HTML page as well as MQTT test brokers such as IoT Eclipse for real-time value streams.

Some of the obstacles were noise in analog signals from the thermistor and modifying comparator thresholds for varying light levels. Debouncing the push buttons and verifying servo calibration were also essential steps to provide smooth operation.

Applications

This system finds a wide range of applicable areas:

- **Smart Residential Homes:** Automated lighting, air conditioning, and security.
- **Offices and Institutions:** Energy-efficient classrooms or conferences.
- **Hospitals or Elder Care:** Fall detection, emergency alerts using motion sensors.
- **Hotels and Guest Houses:** Automated access controls and comfort adjustments.
- **Retail Shops:** Security alerts and temperature-sensitive inventory monitoring.

Real-world scalability and feature expansion can lead to commercial use for customized automation packages.

CONCLUSION AND FUTURE SCOPE

Conclusion

The NodeMCU-based Smart Home Automation System with integrated sensors offers a real-time, scalable, and feasible control mechanism for home appliances. The real-time monitoring of environmental factors such as light, heat, and motion, and automatic response to them, make the project not only an electronics engineering demonstration but a basis for actual IoT usage. Successful interconnection of components, proper software management, and outcome-oriented testing prove the system operational.

Cheap and simple to install, the system provides advantages in accessibility, energy conservation, and safety improvement. More than mere light or alarm tones, the system allows extension to the control of HVAC systems, electric doors, smart monitoring, or cloud connections.

Future Scope

While the basic system is robust, several practical extensions can increase its capabilities:

- **Smartphone App Integration:** Apps made for Android/iOS platforms such as MIT App Inventor or Flutter facilitate more fluid, intuitive operation.
- **Cloud Storage and Analytics:** Utilizing Firebase or ThingSpeak will make possible historical data storage and optimization of energy use.
- **Voice Assistant Integration:** Support for control through Alexa and Google Home by means of APIs or IFTTT improves user experience.
- **Security Cameras:** Camera module integration for monitoring and motion-activated image taking.
- **Advanced Machine Learning Algorithms:** Predictive automation through learning user patterns.
- **Solar-Powered Backup:** Deployment in off-grid rural environments where sustainable power sources are necessary.

These extensions can also add more flexibility, minimize energy wastage, and support smart living solutions that extend beyond home comfort, integrating into smart city and sustainable living paradigms.