Taj Al-Ma'rifa Journal مجلة تاج المعرفة
ISSN : 3104-8846, ISSN-L: 3104-8838

IoT Smart Home System Using NodeMCU (ESP8266) for Remote and Energy-Efficient Automation

*** Hisham Aboulghasem Ali Esherwi**

Departmen of Automatic Control, College of Computer Technology-Tripoli

hisham.esherwi@gmail.com

*Corresponding Author: * Hisham Aboulghasem Ali Esherwi

Keyword

Internet of Things (IoT), Smart Home, NodeMCU, ESP8266, Blynk, Sensors, Wireless Control, Actuators.

Abstract

Rapid advances in the Internet of Things (IoT) have transformed automation, especially in smart home applications. This study presents the design and implementation of a smart home system built on the NodeMCU development board (ESP8266) and integrated with a set of environmental and security sensors. The proposed system allows users to monitor conditions and control household devices remotely via the Blynk IoT platform, providing a flexible, low-cost, and energy-efficient solution. The hardware configuration includes sensors for temperature, humidity, air quality, motion, light intensity, and obstacle detection, in addition to actuators such as servo and DC motors for automating a garage door and curtains. An OLED display provides on-site environmental readings, while the Blynk mobile application enables real-time control over Wi-Fi. Experimental testing confirmed stable connectivity, responsive control, and reliable continuous operation. The modular architecture also supports easy expansion and customization to meet different residential needs. Overall, the results demonstrate that open-source IoT platforms can deliver scalable, cost-effective, and sustainable smart home automation.

Received : 02/03/2026 Accepted : 10/03/2026

DOI: https://doi.org/10.64943/jkc.2026.040118

نظام منزل ذكي قائم على إنترنت الأشياء باستخدام NodeMCU (ESP8266)للتحكم عن بُعد والأتمتة ذات الكفاءة في استهلاك الطاقة

* هشام أبوالقاسم علي الشروي

محاضر - قسم التحكم الالي، كلية تقنية الحاسوب – طرابلس

hisham.esherwi@gmail.com

*الباحث المرسل: * هشام أبوالقاسم علي الشروي

الكلمة المفتاحية: إنترنت الأشياء(IoT)، المنزل الذكي، NodeMC، ESP8266؛ Blynk، المستشعرات، التحكم اللاسلكي، المشغلات.

الملخص

شهد مجال إنترنت الأشياء (IoT) تطورًا متسارعًا أسهم في إحداث تحول ملحوظ في أنظمة الأتمتة، لا سيّما في تطبيقات المنازل الذكية. يقدّم هذا البحث تصميمًا وتنفيذًا لنظام منزل ذكي يعتمد على لوحة التطوير NodeMCU المزودة بالمتحكم ESP8266، ومتكاملًا مع مجموعة من الحساسات البيئية والأمنية. يتيح النظام للمستخدم مراقبة الأجهزة المنزلية والتحكم بها عن بُعد عبر منصة Blynk IoT، بما يوفّر حلًا مرنًا ومنخفض التكلفة وفعّالًا من حيث استهلاك الطاقة.يتضمن النظام حساسات لقياس درجة الحرارة والرطوبة وجودة الهواء، إلى جانب حساسات الحركة وشدة الإضاءة واكتشاف العوائق، كما يضم مشغلات مثل محركات السيرفو ومحركات التيار المستمر للتحكم في بوابة المرآب وستائر النوافذ. وتُستخدم شاشة OLED لعرض القراءات البيئية محليًا، بينما يوفّر تطبيق Blynk على الهواتف الذكية مراقبة لحظية وتحكمًا فوريًا عبر الاتصال اللاسلكي.Wi-Fi أظهرت نتائج التطبيق العملي استقرارًا ملحوظًا في الاتصال، واستجابة سريعة ودقيقة في التحكم بالمشغلات، وتشغيلًا موثوقًا تحت ظروف اختبار مستمرة. كما تُتيح البنية المعمارية المعيارية للنظام قابلية التوسّع وإضافة وحدات جديدة بما يتناسب مع احتياجات سكنية مختلفة. وتؤكد الدراسة أن الاعتماد على منصات إنترنت الأشياء مفتوحة المصدر يمثّل خيارًا عمليًا لتطوير أنظمة منازل ذكية قابلة للتطوير ومستدامة من حيث التكلفة وكفاءة الطاقة.

تاريخ الإستقبال: 2026/03/02 تاريخ القبول: 2026/03/10

DOI: https://doi.org/10.64943/jkc.2026.040118

1. Introduction

The rapid evolution of digital technologies and the widespread adoption of the Internet of Things (IoT) have reshaped modern lifestyles and residential environments (Al-Fuqaha et al., 2015; Kassab & Darabkh, 2020) . Among the most influential IoT applications, smart homes aim to enhance comfort, safety, and energy efficiency by connecting everyday devices to the internet and enabling real-time monitoring and remote control (Kumar & Patel, 2014). By integrating sensors, actuators, and cloud-based platforms, smart home systems can support automated actions and basic decision-making that

improve the user's daily experience (Al-Fuqaha et al., 2015; Kumar & Patel, 2014).

In recent years, various smart home architectures have been proposed; however, many existing solutions remain expensive, complex to deploy, or limited in terms of integration between sensing and actuation modules (Kassab & Darabkh, 2020). In addition, traditional home automation often relies on wired infrastructures, which reduces flexibility and makes expansion more difficult. With the availability of low-cost, Wi-Fi-enabled microcontrollers—such as the ESP8266 and NodeMCU—together with user-friendly IoT platforms like Blynk, it has become feasible to develop affordable, wireless, and highly customizable smart home systems (Mahindrakar et al., 2024; Singh et al., 2021).

This paper presents the design and implementation of an IoT-based smart home system that combines environmental monitoring, security detection, and motor-driven automation for household devices. NodeMCU serves as the main controller, while the Blynk IoT cloud platform enables real-time data exchange and remote interaction. Through the Blynk mobile application, users can monitor environmental conditions, control appliances from anywhere with internet access, and receive alerts when specific events are detected.

The main objectives of this research are to:

- Develop a reliable and low-cost smart home automation system using IoT technologies.
- Integrate multiple sensors and actuators into a unified hardware–software framework.
- Enable remote monitoring and control using the Blynk mobile application.

  • Evaluate and compare the system's performance against recent smart home solutions (2023–2025).

By achieving these objectives, the study contributes a practical and scalable smart home model that can be adapted to different residential needs,

especially in developing regions where affordability, simplicity, and energy efficiency are critical requirements.

2. Literature Review

Recent advances in Internet of Things (IoT) technologies have significantly improved the integration of sensors, actuators, and wireless microcontrollers, enabling more practical and scalable smart home automation systems. Several studies have explored low-cost IoT architectures; however, limitations remain in terms of comprehensive functionality, offline resilience, and energy-aware automation.

Ogenyi (2023) proposed an IoT-based smart home automation system using ESP8266, highlighting affordability and simplicity through the NodeMCU platform and Wi-Fi connectivity. The implementation successfully supported temperature and motion sensing, yet it lacked a local feedback mechanism (e.g., a display), which reduced usability during offline conditions and limited direct user interaction at the device level.

Similarly, Singh et al. (2021) developed a wireless home automation system employing NodeMCU/ESP8266 modules and relay-based switching for controlling household appliances. Their work demonstrated the practicality of ESP8266 for real-time control and cloud-based interaction via mobile applications. However, the study mainly addressed basic on/off operations and did not investigate multi-sensor integration, data fusion, or strategies for improving energy efficiency.

Pancane et al. (2024), published in the Formosa Journal of Computer and Information Science, extended the concept toward energy efficiency by introducing an adaptive control approach that adjusts device operation based on sensor readings to reduce unnecessary power consumption. Despite these improvements, their system was largely limited to environmental monitoring and did not include broader security and automation functions such as motion-triggered control or more diverse actuation tasks.

More recently, Ayeni and Adesoba (2025) presented an IoT-based home control system using NodeMCU integrated with Firebase, achieving real-

time synchronization and secure authentication. While the system demonstrated high scalability, its heavy dependence on continuous cloud connectivity introduces vulnerability to network outages and may increase latency under congested conditions.

Overall, prior studies reflect substantial progress in IoT-enabled smart home automation, yet persistent gaps remain. Many solutions emphasize isolated functionalities—such as monitoring, simple switching, or cloud synchronization—without combining sensing and actuation within a unified, modular, and energy-optimized framework. This paper addresses these limitations by proposing a comprehensive smart home system that integrates environmental sensing, real-time actuation, energy-aware control logic, local display feedback, and Blynk-based remote monitoring, delivering a more holistic and resilient architecture suitable for practical residential deployment.

3. System Architecture and Methodology

The proposed smart home system is designed around the NodeMCU (ESP8266) microcontroller as the central processing and communication unit. The architecture integrates multiple environmental and security sensors, motor-driven actuators, and a cloud-based IoT platform (Blynk) to support real-time monitoring and remote control (Al-Fuqaha et al., 2015).

3.1 System Overview

The system continuously monitors key environmental parameters, including temperature, humidity, air quality, and light intensity, in addition to motion and obstacle detection (Pancane et al., 2025). Based on sensor data and predefined threshold values, the system can automatically operate connected devices such as curtains, a garage door, and a water pump. Sensor readings are acquired by the NodeMCU, processed locally, and transmitted via Wi-Fi to the Blynk Cloud Platform. Users can then observe real-time system status and issue manual commands through the Blynk mobile application interface.

3.2 Hardware Components

The system hardware includes the following major modules (Mahindrakar et al., 2024):

- **NodeMCU (ESP8266):** Wi-Fi-enabled microcontroller board serving as the main controller.
- **DHT11 Sensor:** Measures ambient temperature and humidity.
- **MQ-135 Sensor:** Detects air quality by sensing gases (e.g., CO_2, NH_3) and related pollutants.
- **PIR Sensor:** Detects human motion to trigger alerts or automation events.
- **LDR Module:** Measures light intensity to automate lighting and/or curtain control.
- **IR Sensor:** Provides obstacle detection, especially during garage door operation.
- **Servo Motor:** Enables precise angular control for the garage door mechanism.
- **DC Motor:** Drives the curtain automation system.
- **Mini 5V Water Pump:** Supports automated plant watering.
- **L293D H-Bridge Driver:** Controls DC motor direction and power handling.
- **OLED Display (0.96″ 128×64):** Displays local readings such as temperature, humidity, and system status.
- **Power Supply (5V):** Provides regulated power for stable operation of the full system.

3.3 Software Design

The software implementation is developed using the Arduino IDE and integrates Blynk cloud services for real-time communication (Singh et al., 2021). The firmware is organized into key functional stages:

- **Sensor Initialization:** Sensors are configured during startup and verified for stable readings.
- **Data Acquisition and Cloud Update:** The NodeMCU reads sensor values (analog/digital), processes them, and uploads data to Blynk using virtual pins at defined time intervals.
- **Decision Logic and Automation:** If sensor values exceed specified thresholds, the system triggers actions such as activating a motor, switching a device, or starting the water pump.
- **Mobile Interface Control:** The Blynk dashboard displays live sensor data and provides control widgets (buttons/switches/sliders) for manual override and remote operation.

3.4 Block Diagram

Figure 1 illustrates the overall block diagram of the proposed system, showing the flow of information from sensors to the NodeMCU, then to the Blynk cloud platform, and finally to the actuators based on user commands or automated decision logic.

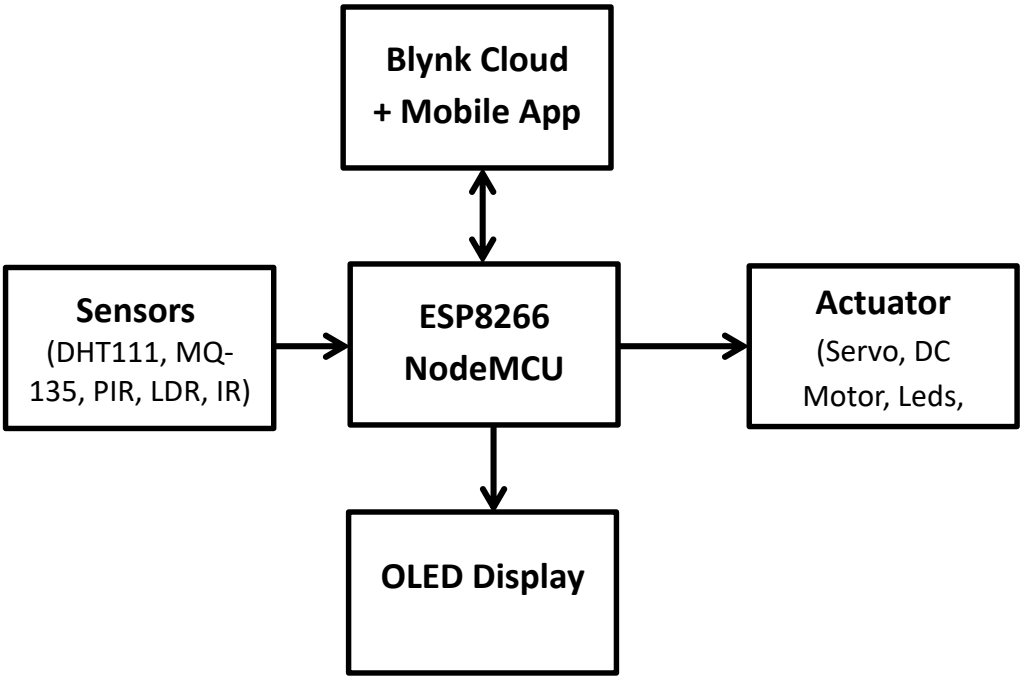

Figure 1. Block Diagram of the Smart Home System

The diagram presents the overall workflow of the proposed system, showing sensor inputs (DHT11, MQ-135, PIR, LDR, and IR), processing and decision-making on the NodeMCU (ESP8266), bidirectional communication with the Blynk Cloud platform, and output actions executed through actuators (lighting, motors, LEDs, water pump) and local feedback via the OLED display.

4. Implementation

The proposed IoT-based smart home system was implemented using real hardware modules integrated with the NodeMCU controller and evaluated through practical testing (Mahindrakar et al., 2024; Ogenyi, 2023). System integration was achieved via physical circuit wiring and cloud-based interaction using the Blynk IoT platform. This section describes the hardware construction, software development, and prototype deployment, followed by the experimental validation outcomes.

4.1 Hardware Implementation

The complete circuit was assembled on a breadboard and interfaced with the NodeMCU, as illustrated in Figure 2. Sensors were positioned to capture environmental and motion-related data, while actuators were connected through the L293D motor driver to operate devices such as the curtain mechanism and garage door model. A regulated 5 V DC power supply was used to ensure stable operation across all components.

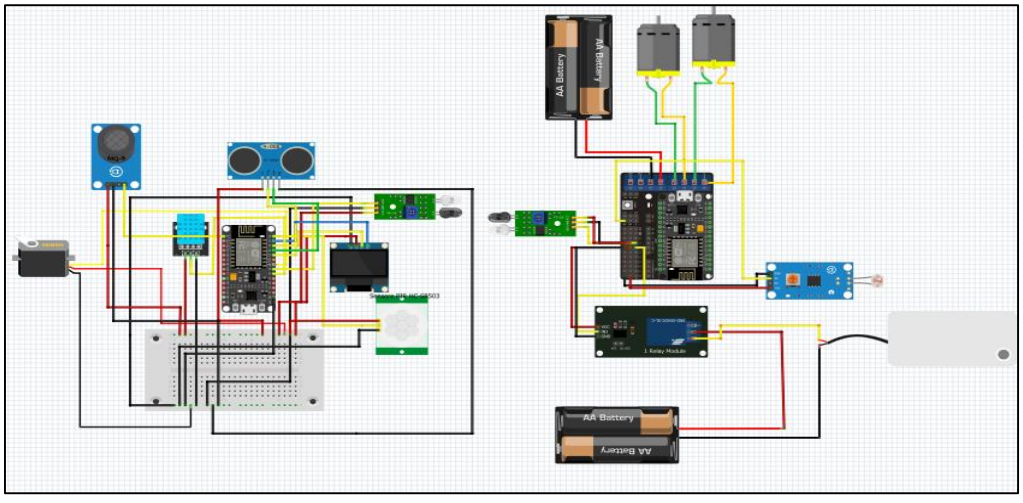

Figure 2. Circuit Connection Diagram of the Smart Home System

This figure illustrates the wiring connections among the NodeMCU, sensors, motor drivers, actuators, and the power supply, representing the actual implementation layout used during system testing.

4.2 Software Implementation

Firmware development was conducted using the Arduino IDE in C/C++, with Blynk libraries integrated to support cloud connectivity. The program initializes all sensors, periodically acquires readings, and transmits sensor data to the Blynk cloud using virtual pins. User commands received from the Blynk mobile application are processed by the NodeMCU to control connected devices such as motors and switching modules.

A snapshot of the Blynk dashboard is shown in Figure 3, illustrating real-time temperature, humidity, and air-quality indicators, along with manual control widgets for user interaction.

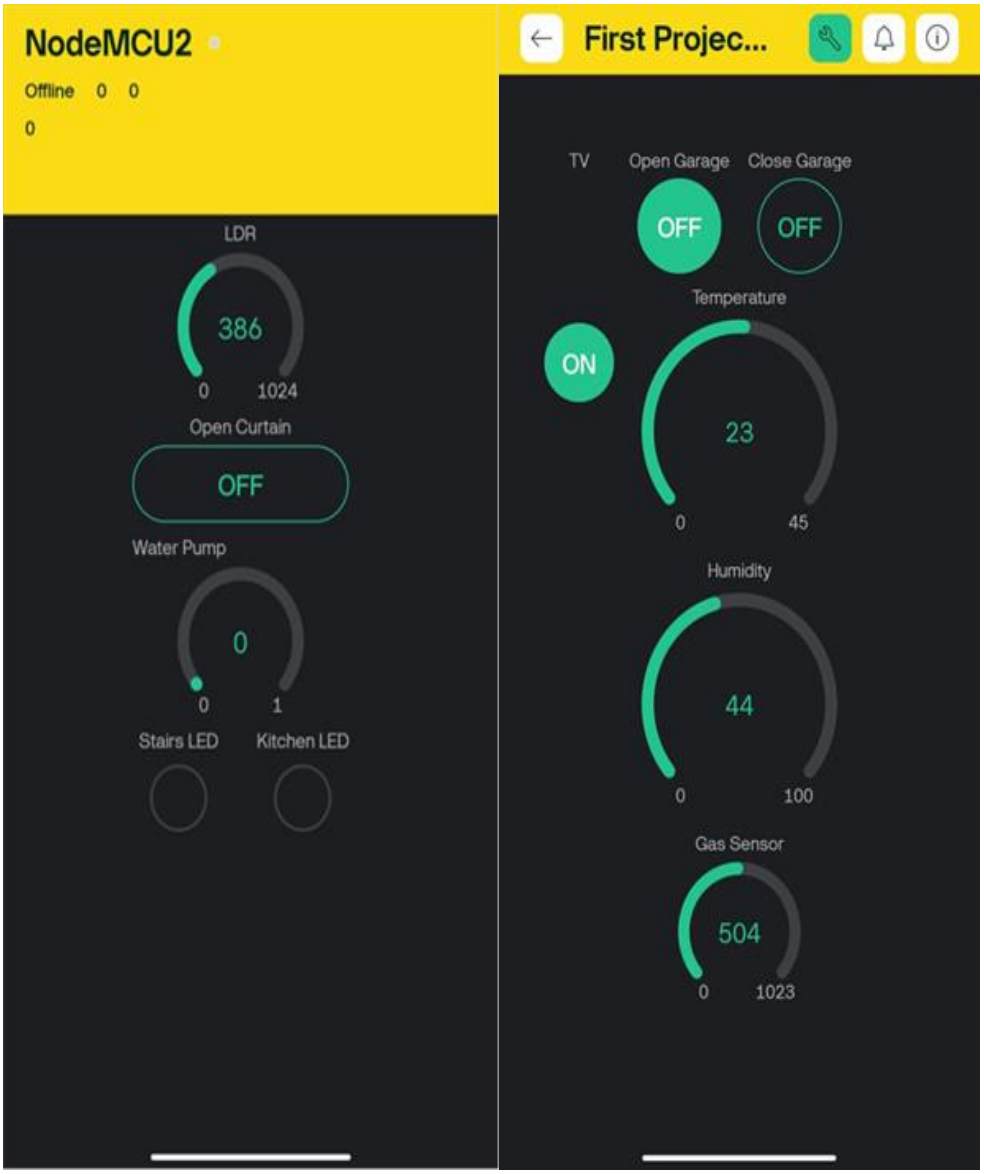

Figure 3. Blynk Mobile Interface

The mobile dashboard enables users to view real-time sensor data and control appliances remotely through virtual buttons.

4.3 Experimental Setup and Prototype

A functional physical prototype was constructed to demonstrate end-to-end system operation. As shown in Figure 4, the prototype integrates sensing and

actuation modules within a miniature smart home model. Experimental demonstrations confirmed remote lighting control, curtain automation, and live environmental monitoring through IoT connectivity.

Figure 4. Smart Home Physical Prototype Model

The prototype demonstrates integrated monitoring and automated control of multiple household functions using the proposed IoT architecture.

5. Performance Results and Discussion

5.1 Performance Results

The proposed smart home system was implemented and tested successfully in a controlled indoor environment. All subsystems—including environmental sensors (DHT11, MQ-135, and LDR), security modules (PIR motion detection and IR obstacle sensing), and actuation components (servo motor and DC motor)—operated reliably and according to design specifications.

The Blynk IoT platform enabled stable wireless communication between the NodeMCU and the mobile interface, supporting real-time monitoring and remote control of connected devices. During testing, sensor readings were

transmitted consistently with minimal delay, and actuator response was fast and predictable. The OLED display provided immediate local feedback for key environmental values, while the Blynk dashboard accurately reflected updates over Wi-Fi.

In addition, the implemented automation logic performed consistently under different test conditions. For example, curtain movement was triggered appropriately based on ambient light intensity, and ventilation-related actions responded correctly when air-quality thresholds were exceeded. The system also maintained stable operation during extended testing periods, with no significant firmware crashes or frequent network interruptions, indicating that the NodeMCU–Blynk combination is suitable for continuous home automation tasks.

5.2 Discussion

The obtained results confirm that open-source IoT tools can deliver reliable, low-cost smart home automation suitable for domestic use, which is consistent with findings reported in previous IoT home automation studies (Singh et al., 2021; Pancane et al., 2025). The modular hardware–software design facilitated straightforward integration of multiple sensors and actuators without requiring specialized equipment or complex configuration steps. Moreover, Blynk contributed to usability by offering a simple and intuitive mobile interface for monitoring and control.

From an engineering perspective, the system benefits from a hybrid monitoring approach that combines local feedback (OLED display) with cloud-based supervision (Blynk). This design supports better user experience and improves operational resilience. Specifically, local display feedback allows users to observe system status even when network connectivity is temporarily degraded, while cloud connectivity enables remote access and future scalability.

Finally, the study indicates that firmware optimization—particularly conditional actuation and efficient event-based data updates—can improve responsiveness and enhance system stability, which is essential for practical IoT control systems deployed in real homes.

6. Conclusion

This study presented the design and implementation of an IoT-based smart home system using the NodeMCU (ESP8266) controller and the Blynk IoT platform. The proposed solution successfully integrated multiple sensing and actuation modules to support environmental monitoring, basic security detection, and remote appliance control. Experimental testing validated stable system performance, low-latency response, and reliable wireless communication. Overall, the project demonstrates that accessible open-source IoT technologies can be combined to build flexible, energy-aware, and user-friendly smart home automation systems, which is supporting conclusions reported in recent IoT automation research (Mahindrakar et al., 2024; Ogenyi, 2023). The developed prototype also provides a practical model for educational purposes and real-world deployment, particularly in developing regions where affordability and simplicity are key requirements.

7. Future Work

Future enhancements to the proposed system may include the following directions:

1. **Integration of Machine Learning (ML) Techniques:** Implement predictive models to learn user patterns and automatically optimize lighting, ventilation, and other environmental parameters.
2. **Voice-Controlled Automation:** Add compatibility with voice assistants (e.g., Google Assistant and Amazon Alexa) to enable hands-free interaction and voice-based control.
3. **Edge Computing Deployment:** Improve real-time performance by shifting more processing and decision-making tasks to the microcontroller (edge), reducing dependence on continuous cloud connectivity.
4. **Enhanced Security Mechanisms:** Incorporate stronger security measures such as AES encryption and two-factor authentication (2FA) to secure device-to-cloud communication.

5. **Scalability and Multi-Node Networking:** Expand the architecture to support multiple nodes for larger environments using mesh or short-range networking solutions such as ESP-NOW or ZigBee.
6. **Comprehensive Data Analytics:** Implement long-term data logging and visualization dashboards to analyze environmental trends, user behavior, and energy consumption patterns.

8. References

1. Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015). Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials, 17*(4), 2347–2376. https://doi.org/10.1109/COMST.2015.2444095
2. Ayeni, P. O., & Adesoba, O. C. (2025). IoT-based home control system using NodeMCU and Firebase. *Journal of Edge Computing, 4*(1), 17–34. https://doi.org/10.55056/jec.814
3. Kassab, W., & Darabkh, K. A. (2020). A–Z survey of internet of things: Architectures, protocols, applications, recent advances, future directions and recommendations. *Journal of Network and Computer Applications, 163(1)*, Article 102663. https://doi.org/10.1016/j.jnca.2020.102663
4. Kumar, J. S., & Patel, D. R. (2014). A survey on Internet of Things: Security and privacy issues. *International Journal of Computer Applications, 90*(11), 20–26. https://doi.org/10.5120/15764-4454
5. Mahindrakar, A., Kalal, R., Ravdive, A., Chougule, A., & Jagdale, V. (2024). IoT-based smart home automation system using NodeMCU. *International Journal for Multidisciplinary Research, 6*(6), 30217. https://www.ijfmr.com/papers/2024/6/30217.pdf
6. Ogenyi, O. H. (2023). Design and implementation of IoT-based smart home automation system using [Zenodo]. https://doi.org/10.5281/zenodo.15869261

7. Pancane, I. W. D., Hermawan, Y., & Kumara, I. N. I. (2025). Design and implementation of IoT-based smart home system with ESP8266 for energy efficiency. *Formosa Journal of Computer and Information Science, 4*(1), 71–82. https://doi.org/10.55927/fjcis.v4i1.14084

8. Singh, P., Bathla, G., Singh, R. K., & Aggarwal, A. (2021). A novel approach for wireless home automation system using IoT. In *Proceedings of the International Conference on Emerging Technologies: AI, IoT and CPS for Science & Technology Applications (ICET 2021)* (Vol. 3058). https://ceur-ws.org/Vol-3058/Paper-089.pdf

Appendix

Appendix A. Source code for the first NodeMCU unit demonstrating the configuration and operation of the employed sensors.

```
#define BLYNK_TEMPLATE_ID "TMPL4xdNB8ba5"
#define BLYNK_TEMPLATE_NAME "first project"
#define BLYNK_AUTH_TOKEN "9Od-KrY-wdm65cGFXPQPNTG3pGEWaog0"
// ------------------- Includes -------------------
#include <Servo.h>
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <DHT.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
// ------------------- OLED Settings -------------------
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);
// ------------------- Pin Assignments -------------------
const int trigPin = D4;
const int echoPin = D3;
const int ledPin = D8;
const int servoPin = D7;
const int irPin = D5;
const int dhtPin = D6;
const int mq135Pin = A0;
// ------------------- WiFi Credentials -------------------
char ssid[] = "Trackgator alligator";
char pass[] = "SpaceCowboy66";
// ------------------- DHT Settings -------------------
#define DHTTYPE DHT11
DHT dht(dhtPin, DHTTYPE);
```

```
// ------------------- Globals -------------------
Servo garageServo;
BlynkTimer timer;
bool garageOpen = false;
bool blynkManualControl = false;
bool oledOn = true;
bool ultrasonicCooldown = false;
bool fireAlertSent = false;
bool gasAlertSent = false;
float temperatureC = 0;
float humidity = 0;
unsigned long irStartTime = 0;
unsigned long ultrasonicCooldownStart = 0;
unsigned long blynkControlStart = 0;
unsigned long lastUltrasonicPrint = 0;
unsigned long lastDHTPrint = 0;
// ------------------- Setup -------------------
void setup() {
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(ledPin, OUTPUT);
  pinMode(irPin, INPUT);
  digitalWrite(ledPin, LOW);
  garageServo.attach(servoPin);
  garageServo.write(180);  // start closed
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    while (true);  // OLED init fail loop
  }
  display.clearDisplay();
  display.setTextSize(2);
  display.setTextColor(WHITE);
  display.setCursor(10, 20);
  display.println("Welcome");
  display.display();
  delay(2000);
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
  dht.begin();
  timer.setInterval(2000L, readDHTSensor);
  timer.setInterval(1500L, updateOLED);
  timer.setInterval(3000L, readGasSensor);
}
// ------------------- Loop -------------------
void loop() {
  Blynk.run();
  timer.run();
  checkBlynkPause();
```

```
  handleUltrasonic();
  handleIRSensor();
}
// ------------------- BLYNK Buttons -------------------
BLYNK_WRITE(V0) {
  if (param.asInt() == 1) {
    Serial.println("Blynk Open - Pausing Sensors");
    blynkManualControl = true;
    blynkControlStart = millis();
    openGarage();
  }
}
BLYNK_WRITE(V4) {
  if (param.asInt() == 1) {
    Serial.println("Blynk Close - Pausing Sensors");
    blynkManualControl = true;
    blynkControlStart = millis();
    closeGarage();
  }
}
BLYNK_WRITE(V3) {
  oledOn = (param.asInt() == 1);
  if (!oledOn) {
    display.clearDisplay();
    display.display();
  }
}
// ------------------- Functions -------------------
void openGarage() {
  for (int pos = garageServo.read(); pos >= 5; pos--) {
    garageServo.write(pos);
    delay(15);
  }
  digitalWrite(ledPin, HIGH);
  garageOpen = true;
}
void closeGarage() {
  for (int pos = garageServo.read(); pos <= 180; pos++) {
    garageServo.write(pos);
    delay(15);
  }
  digitalWrite(ledPin, LOW);
  garageOpen = false;
}
void readDHTSensor() {
  float h = dht.readHumidity();
  float t = dht.readTemperature();
```

```
  if (!isnan(h) && !isnan(t)) {
    humidity = h;
    temperatureC = t;
    if (millis() - lastDHTPrint > 2000) {
      Serial.print("Temp: ");
      Serial.print(t);
      Serial.print(" °C | Hum: ");
      Serial.println(h);
      lastDHTPrint = millis();
    }
    Blynk.virtualWrite(V1, t);
    Blynk.virtualWrite(V2, h);
    if (t > 25 && !fireAlertSent) {
        Blynk.logEvent("fire_detected", "□ WARNING: Temperature > 25°C, starting 5-second warning..");
      fireAlertSent = true;
    }
    if (t <= 25) {
      fireAlertSent = false;
    }
  }
}
void readGasSensor() {
  int gasVal = analogRead(mq135Pin);
  Serial.print("Gas Value: ");
  Serial.println(gasVal);
  Blynk.virtualWrite(V11, gasVal);
  if (gasVal > 400 && !gasAlertSent) {
    Blynk.logEvent("gas_leak", "Gas Leak Detected! Take Action!");
    gasAlertSent = true;
  }
  if (gasVal <= 400) {
    gasAlertSent = false;
  }
}
void updateOLED() {
  if (!oledOn) return;
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0, 0);
  display.println("Garage Status:");
  display.setCursor(0, 15);
  display.println(garageOpen ? "Garage: OPEN" : "Garage: CLOSED");
  display.setCursor(0, 30);
  display.print("Temp: ");
  display.print(temperatureC);
```

```
  display.println(" C");
  display.setCursor(0, 45);
  display.print("Hum: ");
  display.print(humidity);
  display.println(" %");
  display.display();
}
void handleUltrasonic() {
  static bool timerRunning = false;
  static unsigned long detectionStart = 0;
  if (blynkManualControl || ultrasonicCooldown) return;
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  long duration = pulseIn(echoPin, HIGH, 30000);
  if (duration == 0) return;
  int distance = duration * 0.034 / 2;
  if (millis() - lastUltrasonicPrint >= 5000) {
    Serial.print("Ultrasonic Distance: ");
    Serial.print(distance);
    Serial.println(" cm");
    lastUltrasonicPrint = millis();
  }
  if (distance > 2 && distance < 10) {
    if (!timerRunning) {
      detectionStart = millis();
      timerRunning = true;
    } else if (millis() - detectionStart >= 3000 && !garageOpen) {
      Serial.println("Ultrasonic detected <10cm for 3s — Opening Garage");
      openGarage();
      ultrasonicCooldown = true;
      ultrasonicCooldownStart = millis();
      timerRunning = false;
    }
  } else {
    timerRunning = false;
  }
}
void handleIRSensor() {
  if (blynkManualControl) return;
  if (garageOpen) {
    if (digitalRead(irPin) == LOW) {
      if (irStartTime == 0) {
        irStartTime = millis();
        Serial.println("IR Detected — Starting 5s timer");
```

```
      }
      if (millis() - irStartTime >= 5000) {
        Serial.println("IR Confirmed — Closing Garage");
        closeGarage();
        irStartTime = 0;
      }
    } else {
      irStartTime = 0;
    }
  }
}
void checkBlynkPause() {
  if (blynkManualControl && millis() - blynkControlStart >= 4000) {
    blynkManualControl = false;
    Serial.println("Blynk Control timeout — sensors re-enabled");
  }
  if (ultrasonicCooldown && millis() - ultrasonicCooldownStart >= 5000) {
    ultrasonicCooldown = false;
    Serial.println("Ultrasonic cooldown complete");
  }
}
```

Appendix B:

The Code Below will demonstrate the Sensors Used in (Second NodeMCU)

```
#define BLYNK_TEMPLATE_ID "TMPL4Vf4_2LAh"
#define BLYNK_TEMPLATE_NAME "first projectCopy"
#define BLYNK_AUTH_TOKEN "cgjal5RxeGFaHTlX6NL9H1CFkCRNdJMZ"
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
char ssid[] = "Trackgator alligator";
char pass[] = "SpaceCowboy66";
#define CURTAIN_OPEN_PIN D1
#define CURTAIN_CLOSE_PIN D2
#define PUMP_PIN D3
#define LDR_PIN A0
bool curtainOpen = false;
bool curtainMoving = false;
unsigned long curtainStartTime = 0;
bool pumpRunning = false;
unsigned long pumpStartTime = 0;
bool manualOverride = false;
unsigned long manualOverrideTime = 0;
unsigned long lastLdrPrintTime = 0;
BlynkTimer timer;
// -------------------- SETUP --------------------
void setup() {
  Serial.begin(9600);
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
  pinMode(CURTAIN_OPEN_PIN, OUTPUT);
  pinMode(CURTAIN_CLOSE_PIN, OUTPUT);
```

```
  pinMode(PUMP_PIN, OUTPUT);
  digitalWrite(CURTAIN_OPEN_PIN, LOW);
  digitalWrite(CURTAIN_CLOSE_PIN, LOW);
  digitalWrite(PUMP_PIN, HIGH); // Pump OFF
  timer.setInterval(2500L, sendSensorData);  // LDR print every 2.5 seconds
}
// -------------------- MAIN LOOP --------------------
void loop() {
  Blynk.run();
  timer.run();
  int ldrVal = analogRead(LDR_PIN);
  // AUTO MODE - CURTAIN ONLY
  if (!manualOverride && !curtainMoving) {
    if (ldrVal > 800 && !curtainOpen) {
      digitalWrite(CURTAIN_OPEN_PIN, HIGH);
      curtainStartTime = millis();
      curtainMoving = true;
      curtainOpen = true;
      Serial.println("Auto: Opening curtain");
      Blynk.virtualWrite(V9, "Opening");
    } else if (ldrVal <= 800 && curtainOpen) {
      digitalWrite(CURTAIN_CLOSE_PIN, HIGH);
      curtainStartTime = millis();
      curtainMoving = true;
      curtainOpen = false;
      Serial.println("Auto: Closing curtain");
      Blynk.virtualWrite(V9, "Closing");
    }
  }
  // STOP CURTAIN after 2.5s and trigger pump simulation
  if (curtainMoving && millis() - curtainStartTime >= 2500) {
    digitalWrite(CURTAIN_OPEN_PIN, LOW);
    digitalWrite(CURTAIN_CLOSE_PIN, LOW);
    curtainMoving = false;
    Serial.println("Curtain motor stopped");
    // Start pump logic
    digitalWrite(PUMP_PIN, LOW); // ON
    pumpStartTime = millis();
    pumpRunning = true;
    Serial.println("Pump worked automatically for 3 seconds");
    Blynk.virtualWrite(V10, "Pump ON");
    Blynk.virtualWrite(V13, 1); // Send 1 to gauge
  }
  // STOP PUMP after 3s
  if (pumpRunning && millis() - pumpStartTime >= 3000) {
    digitalWrite(PUMP_PIN, HIGH); // OFF
    pumpRunning = false;
    Serial.println("Pump stopped after automatic watering");
    Blynk.virtualWrite(V10, "Pump OFF");
    Blynk.virtualWrite(V13, 0); // Send 0 to gauge
  }
  // Exit manual override mode after 10s
  if (manualOverride && millis() - manualOverrideTime >= 10000) {
    manualOverride = false;
```

```
    Serial.println("Manual override expired, back to Auto");
    Blynk.virtualWrite(V9, "Auto Mode");
  }
  delay(50);
}
// -------------------- SENSOR DATA EVERY 2.5s --------------------
void sendSensorData() {
  int ldrVal = analogRead(LDR_PIN);
  Serial.print("LDR: ");
  Serial.println(ldrVal);
  Blynk.virtualWrite(V8, ldrVal);
}
// -------------------- MANUAL CONTROL: OPEN CURTAIN --------------------
BLYNK_WRITE(V10) {
  int state = param.asInt();
  if (state == 1 && !curtainMoving) {
    digitalWrite(CURTAIN_OPEN_PIN, HIGH);
    curtainStartTime = millis();
    curtainMoving = true;
    curtainOpen = true;
    manualOverride = true;
    manualOverrideTime = millis();
    Serial.println("Manual: Open curtain");
    Blynk.virtualWrite(V9, "Manual Open");
  }
}
// -------------------- MANUAL CONTROL: CLOSE CURTAIN --------------------
BLYNK_WRITE(V12) {
  int state = param.asInt();
  if (state == 1 && !curtainMoving) {
    digitalWrite(CURTAIN_CLOSE_PIN, HIGH);
    curtainStartTime = millis();
    curtainMoving = true;
    curtainOpen = false;
    manualOverride = true;
    manualOverrideTime = millis();
    Serial.println("Manual: Close curtain");
    Blynk.virtualWrite(V9, "Manual Close");
  }
}
```