Smart Home Monitoring and Automation System

by 24378 - Yin Min Oo

Abstract

This project implements a WiFi-based smart home system using an ESP32 microcontroller. The system integrates motion, temperature, humidity, and light sensors to automate a relay and servo mechanism and sends alerts via Pushover app when motion is detected in Away mode. A web interface allows toggling between Home and Away modes. The system prioritizes real-time motion detection using FreeRTOS task scheduling.

1. Introduction

Smart home systems improve security and energy efficiency by automating device control and providing remote notifications. This work focuses on a compact implementation using ESP32 and commonly available sensors, emphasizing responsiveness and minimal false alerts.

2. System Architecture and Hardware

2.1 System Architecture

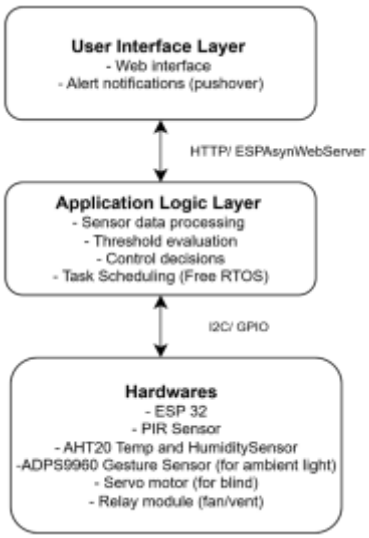

Figure 1: System Architecture of the Home Automation System

The system is organized into three main layers to keep things clear and manageable. The User Interface Layer lets users interact with the system through a simple web page and get motion alerts via Pushover messages. This allows users to check the current status and switch between Home and Away modes remotely. The web interface uses HTTP through the ESPAsyncWebServer library.

The Application Logic Layer is where the main decision-making happens. It reads sensor data, checks if any values cross the set thresholds, and controls the devices based on that. Task scheduling is handled using FreeRTOS, where motion detection is the highest priority, temperature and humidity are in the middle, and light sensing is the lowest. At the base is the Hardware Layer, made up of the ESP32 microcontroller connected to various components: a PIR motion sensor (digital input), AHT20 temperature and humidity sensor (I2C), APDS9960 light sensor (I2C), a relay (digital output), and a servo motor (PWM). These are connected using standard protocols like GPIO and I2C.

The diagram below shows how the main hardware components are connected.

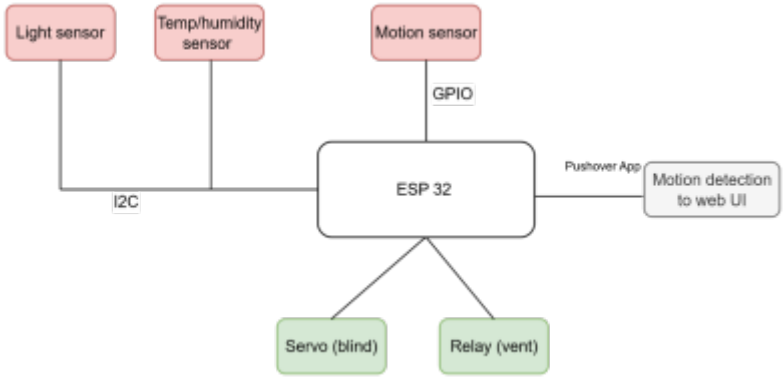

Figure 2: Hardware Assembly of the System

2.1 Hardware Components

1. ESP32: Central microcontroller with WiFi support. 2. PIR Sensor: Detects human movement.

Figure 3: PIR Motion Sensor Used in the Project

3. APDS9960: Measures ambient light.

Figure 4: Gesture Sensor APDS9960 to Measure the Ambient Light Level

4. AHT20: Reads temperature and humidity.

Figure 5: AHT20 Temperature and Humidity Sensor

5. Relay Module: Controls external devices.

Figure 6: Relay Module 3.3V

6. Servo Motor: Adjusts blind position based on ambient light level.

Figure 7: Servo Motor 3.3V

2.2 Software Components

• Arduino IDE with FreeRTOS • ESPAsyncWebServer for web interface • Pushover app for alerting • Servo and sensor libraries (Adafruit, SparkFun)

3. Implementation

3.1 Sensor Integration

Other sensors except motion sensor is activated every 5 seconds. Motion sensor is activated every 5 milliseconds when in “Away” mode. Temperature and humidity activate the relay to switch the ventilator, light intensity controls curtain adjustment via a servo. Motion is processed via interrupt and high-priority task.

3.2 Network and Pushover Setup

ESP32 connects to local WiFi and communicates with push notification with the Pushover APP. Messages are sent only when motion is detected, and the system is in Away mode. The system uses the Pushover service to send motion detection alerts to the user’s mobile device. A user account was created on the Pushover platform, and a new application was registered to obtain the required User Key and API Token. These credentials are used in the ESP32 firmware to authenticate with the Pushover API.

When motion is detected and the system is in “Away” mode as described in the table 1, the ESP32 sends an HTTPS POST request to the Pushover API endpoint with the alert message. This integration allows real-time push notifications, ensuring timely alerts directly to the user’s phone.

3.3 Web Server Interface

Users access a web page hosted on ESP32 to toggle between Home and Away mode. The page also displays current sensor values and relay state.

3.4 FreeRTOS Task Priority and Control Logic

Table 1: Task Priorities of The Seosor Tasks (3 = Highest, 1 = Lowest)

Tasks	Priority	Interval
PIR Motion Task	3	Every 5 ms
Temp/Humidity	2	Every 5 s
Light Sensor	1	Every 5 s

Table 2: Control Logic and Thresholds of Detected Environmental Values

Sensor Values	Threshold	Action
Motion detected	No threshold (digital)	Pushover Notification Alert
Temp and Humidity	t>=25’C or humidity>=60	Relay is on
Temp and Humidity	T<25’C or humidity<60	Relay is off
Light Sensor	T<25’C or humidity<60	Servo is triggered

4. Results

4.1 System Behaviour

The system connects to WiFi successfully and displays its local IP address for user access. Motion alerts are sent via Pushover only when the system is in Away mode and actual human motion is detected. The servo motor responds to ambient light levels by adjusting the blinds—closing when light is too strong and opening when it drops. The relay is activated when either the temperature exceeds a set threshold or humidity rises above the defined limit, helping manage indoor climate conditions effectively.

5. Demonstration

System boots, connects to WiFi, and hosts a web page. Home or away mode can be toggled on the page for the motion sensor detection. The interface also shows the status of light, temperature and humidity.

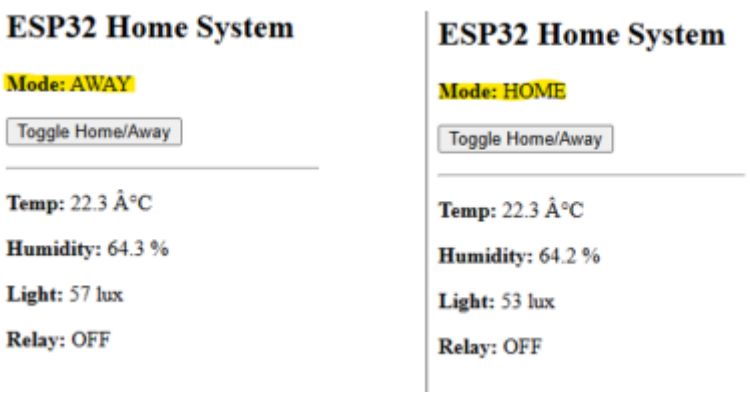

Figure 8: “Home” and “Away” mode in web interface.

Movement in Away mode triggers Pushover notification alert as shown in figure 8.

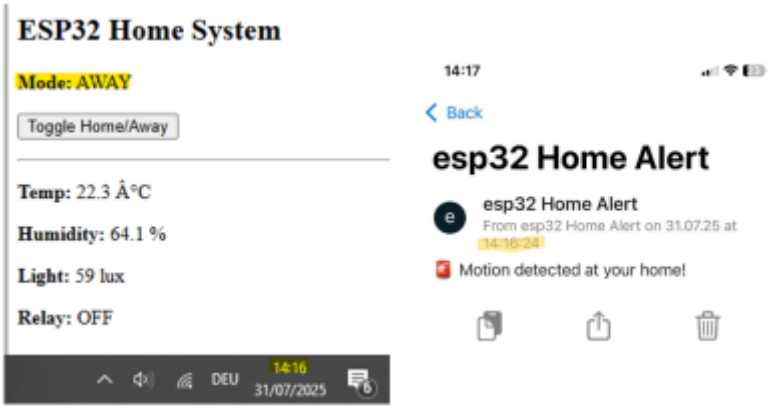

Figure 9: Triggered pushover notification in “away” mode when motion is detected.

The full demonstration of the system can be seen in the video.

yin_min_oo_-_24378_home_automation_.mp4

The video demonstrates the motion sensor detection when the system is set to away mode and how the push notification is triggered. It furthermore shows how the servo is activated when the light level is above or below the threshold and how the relay is activated based on the humidity level. However, the temperature sensor cannot be manipulated above the threshold and as it is dangerous. Nevertheless, it can be clearly seen in the code attached.

6. Limitations and Areas for Improvement

• Sensitivity of Motion Detection: The motion sensor may sometimes fail to trigger alerts. • Reliance on Stable WiFi Connection: The system requires a stable WiFi connection to function properly. Interruptions or changes in IP address may disrupt access to the web interface and push notifications. Implementing fixed IP addressing or network reconnection strategies could enhance reliability. • Humidity Simulation Challenges: It is difficult to simulate and test low humidity conditions in the current setup, limiting the ability to demonstrate relay deactivation under those conditions. Introducing a manual override or test mode could help verify this functionality. • Single User Notification: Currently, push notifications are sent to only one user. Supporting multiple users for alerts would improve the system’s usefulness in shared environments. • Servo Movement Optimization: The servo motor moves to preset positions based on light intensity without checking its current state, which may cause unnecessary movement. Adding state checks before actuation could improve efficiency and reduce wear.

7. Conclusion

This ESP32-based smart home system demonstrates motion-based alerting and light-driven automation. With FreeRTOS prioritization, the system ensures fast response to motion events. The future work includes improving motion detection, enhancing WiFi stability, adding manual test modes, supporting multi-user notifications, and optimizing servo movements based on state checks.

```
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <ESPAsyncWebServer.h>
#include <Wire.h>
#include <SparkFun_APDS9960.h>
#include <Adafruit_AHTX0.h>
#include <ESP32Servo.h>

// WiFi credentials
const char* ssid = "Vodafone-CA64";
const char* password = "2Mm6mN7z6LFCYLNG";

// Pushover credentials
const char* pushoverUserKey = "uphxjot11p6kasrdgmjvkpm6v5hsod";
const char* pushoverAppToken = "am7ebju1mdpp82axev8x9675qai6x3";

// Pin definitions
#define PIR_PIN 13
#define RELAY_PIN 14
#define SERVO_PIN 12

// Hardware objects
SparkFun_APDS9960 apds;
Adafruit_AHTX0 aht;
Servo myservo;
AsyncWebServer server(80);
WiFiClientSecure secureClient;

// States
volatile bool motionDetected = false;
bool isHome = false;
bool canSendAlert = true;

float currentTemp = 0.0, currentHum = 0.0;
uint16_t currentLight = 0;

// Task handles
TaskHandle_t motionTaskHandle;
TaskHandle_t tempHumTaskHandle;
TaskHandle_t lightTaskHandle;

// Motion ISR
void IRAM_ATTR motionISR() {
  if (!isHome) {
    motionDetected = true;
    Serial.println("⚡ Interrupt: Motion flag set");
  }
}

// Send pushover message
void sendPushover(String message) {
```

```
  if (WiFi.status() == WL_CONNECTED) {
    secureClient.setInsecure();

    String postData = "token=" + String(pushoverAppToken) +
                      "&user=" + String(pushoverUserKey) +
                      "&message=" + message;

    if (!secureClient.connect("api.pushover.net", 443)) {
      Serial.println("□ Connection to Pushover failed");
      return;
    }

    secureClient.println("POST /1/messages.json HTTP/1.1");
    secureClient.println("Host: api.pushover.net");
    secureClient.println("Content-Type: application/x-www-form-urlencoded");
    secureClient.print("Content-Length: ");
    secureClient.println(postData.length());
    secureClient.println();
    secureClient.print(postData);

    // Read HTTP response headers
    while (secureClient.connected()) {
      String line = secureClient.readStringUntil('\n');
      Serial.println(line);
      if (line == "\r") break;
    }

    // Read HTTP body
    String response = secureClient.readString();
    Serial.println("□ Pushover Response Body:");
    Serial.println(response);
    Serial.println("□ Message sent");
  } else {
    Serial.println("□ WiFi not connected");
  }
}

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

  pinMode(PIR_PIN, INPUT);
  pinMode(RELAY_PIN, OUTPUT);
  digitalWrite(RELAY_PIN, LOW);
  myservo.attach(SERVO_PIN);
  myservo.write(0);

  Wire.begin(4, 5); // I2C pins

  if (!apds.init()) Serial.println("□ APDS9960 init failed");
  else apds.enableLightSensor(false);
```

```
  if (!aht.begin()) Serial.println("□ AHT20 init failed");

  attachInterrupt(digitalPinToInterrupt(PIR_PIN), motionISR, RISING);

  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\n□ Connected! IP: " + WiFi.localIP().toString());

  // Web UI
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request) {
    String html = "<html><body><h2>ESP32 Home System</h2>";
    html += "<p><b>Mode:</b> " + String(isHome ? "HOME" : "AWAY") + "</p>";
    html += "<form action=\"/toggle\" method=\"POST\">";
    html += "<input type=\"submit\" value=\"Toggle Home/Away\"></form><hr>";
    html += "<p><b>Temp:</b> " + String(currentTemp, 1) + " °C</p>";
    html += "<p><b>Humidity:</b> " + String(currentHum, 1) + " %</p>";
    html += "<p><b>Light:</b> " + String(currentLight) + " lux</p>";
    html += "<p><b>Relay:</b> " + String(digitalRead(RELAY_PIN) ? "ON" :
"OFF") + "</p>";
    html += "</body></html>";
    request->send(200, "text/html", html);
  });

  server.on("/toggle", HTTP_POST, [](AsyncWebServerRequest *request) {
    isHome = !isHome;
    motionDetected = false;
    canSendAlert = true;
    request->redirect("/");
  });

  server.begin();

  // FreeRTOS tasks
  xTaskCreatePinnedToCore(motionTask, "MotionTask", 4096, NULL, 3,
&motionTaskHandle, 0);
  xTaskCreatePinnedToCore(tempHumTask, "TempHumTask", 4096, NULL, 2,
&tempHumTaskHandle, 1);
  xTaskCreatePinnedToCore(lightTask, "LightTask", 4096, NULL, 1,
&lightTaskHandle, 1);
}

void loop() {
  // RTOS handles everything
}

// Motion Task (priority 3)
void motionTask(void *pvParameters) {
```

```
  while (1) {
    if (!isHome && motionDetected) {
      motionDetected = false;
      Serial.println("🚨 Motion Detected! Processing...");

      if (canSendAlert) {
        Serial.println("📤 Attempting to send push message...");
        sendPushover("🚨 Motion detected at your home!");
        canSendAlert = false;

        // Re-enable alert after 5s
        xTimerHandle alertTimer = xTimerCreate("alertTimer",
pdMS_TO_TICKS(5000), pdFALSE, 0, [](TimerHandle_t xTimer) {
          canSendAlert = true;
          Serial.println("✅ Alert re-enabled");
        });
        xTimerStart(alertTimer, 0);
      } else {
        Serial.println("⏳ Alert temporarily blocked");
      }
    }

    vTaskDelay(pdMS_TO_TICKS(5));
  }
}

// Temp/Humidity Task (Priority 2)
void tempHumTask(void *pvParameters) {
  while (1) {
    sensors_event_t humidity, temp;
    aht.getEvent(&humidity, &temp);
    currentTemp = temp.temperature;
    currentHum = humidity.relative_humidity;

    Serial.printf("🌡 Temp: %.1f°C, 💧 Hum: %.1f%%\n", currentTemp,
currentHum);

    if (currentTemp >= 25.0 || currentHum >= 70.0) {
      digitalWrite(RELAY_PIN, HIGH);
      Serial.println("🔌 Relay turned ON");
    } else {
      digitalWrite(RELAY_PIN, LOW);
      Serial.println("🔌 Relay turned OFF");
    }

    vTaskDelay(pdMS_TO_TICKS(5000));
  }
}

// Light Task (priority 1)
void lightTask(void *pvParameters) {
```

```
  bool blindsClosed = false; // Track servo state

  while (1) {
    if (apds.readAmbientLight(currentLight)) {
      Serial.print("□ Ambient Light: ");
      Serial.println(currentLight);

      if (currentLight >= 10000 && !blindsClosed) {
        myservo.write(180);  // Close blinds
        blindsClosed = true;
        Serial.println("□ Blinds CLOSED");
      }
      else if (currentLight < 10000 && blindsClosed) {
        myservo.write(0);    // Open blinds
        blindsClosed = false;
        Serial.println("□ Blinds OPENED");
      }
    }

    vTaskDelay(pdMS_TO_TICKS(5000));
  }
}
```