Project Report: ESP32 Motion-Activated Lighting Control System

Bibek Nandan Naik

With Proper License(s)

1. Introduction

This project involves creating a smart lighting control system that responds to motion using PIR (Passive Infrared) sensors and ESP32 microcontroller. The system is designed to control three lights, each activated by one of three consecutive PIR sensors. Two lights are connected through relays, while the third light is directly controlled via an ESP32 GPIO pin. An additional feature allows remote override control via a mobile-friendly web server hosted on the ESP32, enabling users to toggle between automatic and manual light control modes.

2. Objectives

- To create a motion-sensitive lighting system with three PIR sensors, each controlling a corresponding light.
- To implement a system where two lights are controlled via relays and one through a GPIO pin of the ESP32.
- To provide a web interface allowing remote control of lighting modes (Auto, On, Off) using a smartphone or computer.

3. Components

1. **ESP32 Microcontroller**: Serves as the central processing unit, handling PIR sensor data, controlling relays, and hosting the web server.
2. **PIR Sensors (3)**: Detect motion in three different zones, each triggering a respective light.
3. **Relays (2)**: Act as switches to control two of the three lights.
4. **LED/Light Bulbs (3)**: Each light corresponds to one PIR sensor, illuminating when motion is detected.
5. **Wi-Fi Network**: Provides connectivity for the ESP32, allowing access to the control server

4. System Design and Implementation

4.1 Circuit Design

- **PIR Sensors**: Each PIR sensor has an output pin connected to a designated GPIO pin on the ESP32.
- **Relays**: Two relays are used to switch lights on and off. The relays are connected to separate GPIO pins on the ESP32, with external power supplies for the lights.
- **Direct GPIO Control**: The third light is controlled directly through a GPIO pin on the ESP32 without a relay, providing simpler control without PWM.

.

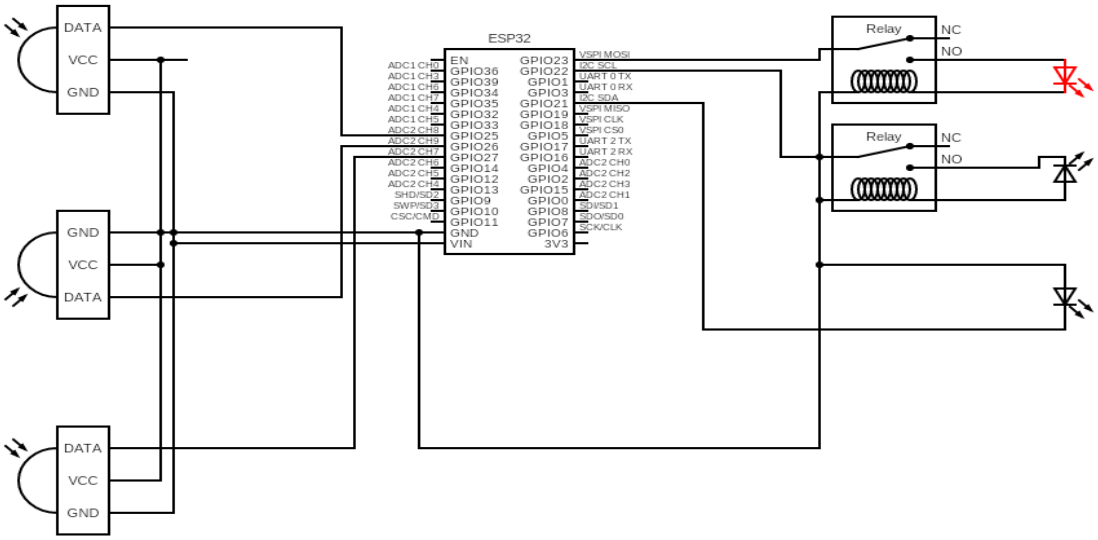

Figure I: Circuit Diagram for our Project

4.2 Software Design

The project code is implemented in Arduino IDE, combining Wi-Fi functionality, web server handling, and GPIO control. Key parts of the code include:

- **Wi-Fi Setup**: ESP32 connects to a local Wi-Fi network using SSID and password for web server access.
- **Web Server**: The ESP32 hosts a web server that provides a user interface for selecting lighting modes (Auto, On, Off). In Auto mode, lights respond to PIR sensors, while On and Off modes override automatic control.

- **Motion Detection**: Each PIR sensor is checked periodically in the main loop. When motion is detected, the corresponding light is activated for a set duration (e.g., 5 seconds).
- **Mode Control**: A state variable `currentMode` manages light behavior, switching between Auto (sensor-based), On (all lights on), and Off (all lights off).

4.3 Flow of Operation

1. **Initial Setup**: The ESP32 initializes Wi-Fi, connects to the network, sets up GPIO pins, and starts the web server.
2. **Motion Detection**: In Auto mode, PIR sensors detect motion. When a sensor triggers, the associated light turns on via the relay or GPIO pin for a brief duration.
3. **Remote Control**: Users can access the web server from a smartphone or PC to override automatic mode. The web interface displays the current mode and provides options to switch to Auto, On, or Off.
4. **Mode Switching**:

- Auto Mode: Lights respond to detected motion.
- On Mode: All lights stay on continuously.
- Off Mode: All lights remain off regardless of sensor input.

5. Results

The system successfully fulfils its objectives:

- **Motion-Activated Lighting**: Each light turns on briefly when motion is detected by its corresponding PIR sensor, simulating zone-based lighting.
- **Remote Control**: The web server interface allows users to toggle lighting modes remotely, adding flexibility.
- **Reliability**: The ESP32's Wi-Fi connectivity is stable, enabling remote access without frequent disconnections.

Figure I: Architectural Model for our Project

6. Testing and Troubleshooting

Challenges and Solutions

1. **Wi-Fi Connection Issues**: Initial connection issues were resolved by ensuring correct SSID and password, maintaining a stable power supply, and updating the ESP32 board firmware.
2. **Relay Control**: Occasional issues with relay response were mitigated by using high-quality relays with appropriate power ratings.
3. **Web Server Access**: Some difficulties accessing the server were addressed by verifying network configurations, including ensuring the client device and ESP32 were on the same Wi-Fi network.

7. Future Improvements

- **PWM Dimming Control**: Adding PWM for the third light could enable dimming, providing smoother lighting transitions.
- **Cloud Integration**: Integration with cloud services like Blynk or MQTT for remote control outside the local network.
- **Notification Feature**: Send alerts when motion is detected, notifying users via a mobile app or email.

8. Conclusion

This project successfully demonstrates a smart, motion-sensitive lighting system that can be remotely controlled using an ESP32 and Wi-Fi. By leveraging PIR sensors and relays, the system provides efficient and responsive lighting. The addition of a web server enhances user experience, making it a versatile solution for automated lighting control in home or industrial applications.

Code with PIR:

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

// PIR sensor pins
#define PIR1_PIN 12
#define PIR2_PIN 13
#define PIR3_PIN 14

// Relay pins for two lights
#define RELAY1_PIN 27
#define RELAY2_PIN 26

// Direct GPIO pin for the third light
#define LIGHT3_PIN 15

// Wi-Fi credentials
const char* ssid = "LNT";
const char* password = "123456789";

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

// Light control states
enum LightMode { AUTO, ON, OFF };
LightMode currentMode = AUTO;
```

```
// Motion detection and light timers
bool motionDetected1 = false;
bool motionDetected2 = false;
bool motionDetected3 = false;

unsigned long lastMotionTime1 = 0;
unsigned long lastMotionTime2 = 0;
unsigned long lastMotionTime3 = 0;

const unsigned long LIGHT_ON_TIME = 5000; // Duration to keep lights on after motion

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

  // PIR sensors
  pinMode(PIR1_PIN, INPUT);
  pinMode(PIR2_PIN, INPUT);
  pinMode(PIR3_PIN, INPUT);

  // Relay pins
  pinMode(RELAY1_PIN, OUTPUT);
  pinMode(RELAY2_PIN, OUTPUT);
  digitalWrite(RELAY1_PIN, LOW);
  digitalWrite(RELAY2_PIN, LOW);

  // Direct pin for the third light
  pinMode(LIGHT3_PIN, OUTPUT);
  digitalWrite(LIGHT3_PIN, LOW);

  // Wi-Fi connection
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
```

```
  }
  Serial.println("\nConnected to Wi-Fi");
  Serial.println(WiFi.localIP());

  // Web server endpoints
  server.on("/", handleRoot);
  server.on("/mode/auto", setAutoMode);
  server.on("/mode/on", setOnMode);
  server.on("/mode/off", setOffMode);

  server.begin();
  Serial.println("Web server started");
}

void loop() {
  server.handleClient();

  if (currentMode == AUTO) {
    checkForMotion();
    manageLights();
  } else if (currentMode == ON) {
    activateLights(true); // All lights ON
  } else if (currentMode == OFF) {
    activateLights(false); // All lights OFF
  }
}

// Check for motion for each PIR sensor
void checkForMotion() {
  if (digitalRead(PIR1_PIN) == HIGH) {
    motionDetected1 = true;
    lastMotionTime1 = millis();
    Serial.println("Motion detected on PIR1");
  }
  if (digitalRead(PIR2_PIN) == HIGH) {
```

```
    motionDetected2 = true;
    lastMotionTime2 = millis();
    Serial.println("Motion detected on PIR2");
  }
  if (digitalRead(PIR3_PIN) == HIGH) {
    motionDetected3 = true;
    lastMotionTime3 = millis();
    Serial.println("Motion detected on PIR3");
  }
}

// Manage each light independently based on motion timers
void manageLights() {
  if (motionDetected1 && millis() - lastMotionTime1 <= LIGHT_ON_TIME) {
    digitalWrite(RELAY1_PIN, HIGH);  // Turn on light 1
  } else {
    digitalWrite(RELAY1_PIN, LOW);   // Turn off light 1
    motionDetected1 = false;
  }

  if (motionDetected2 && millis() - lastMotionTime2 <= LIGHT_ON_TIME) {
    digitalWrite(RELAY2_PIN, HIGH);  // Turn on light 2
  } else {
    digitalWrite(RELAY2_PIN, LOW);   // Turn off light 2
    motionDetected2 = false;
  }

  if (motionDetected3 && millis() - lastMotionTime3 <= LIGHT_ON_TIME) {
    digitalWrite(LIGHT3_PIN, HIGH);  // Turn on light 3
  } else {
    digitalWrite(LIGHT3_PIN, LOW);   // Turn off light 3
    motionDetected3 = false;
  }
}
```

```
// Activate or deactivate all lights based on mode
void activateLights(bool state) {
  digitalWrite(RELAY1_PIN, state ? HIGH : LOW);
  digitalWrite(RELAY2_PIN, state ? HIGH : LOW);
  digitalWrite(LIGHT3_PIN, state ? HIGH : LOW);  // Set third light state
}

// Web server root - shows current mode
void handleRoot() {
  String html = "<h1>Light Control</h1>";
  html += "<p>Current Mode: " + getModeName() + "</p>";
  html += "<a href=\"/mode/auto\">Auto Mode</a><br>";
  html += "<a href=\"/mode/on\">Turn On</a><br>";
  html += "<a href=\"/mode/off\">Turn Off</a><br>";
  server.send(200, "text/html", html);
}

// Set mode to AUTO
void setAutoMode() {
  currentMode = AUTO;
  server.send(200, "text/plain", "Mode set to AUTO");
}

// Set mode to ON
void setOnMode() {
  currentMode = ON;
  server.send(200, "text/plain", "Mode set to ON");
}

// Set mode to OFF
void setOffMode() {
  currentMode = OFF;
  server.send(200, "text/plain", "Mode set to OFF");
}
// Helper to get the mode name
```

```
String getModeName() {
  switch (currentMode) {
    case AUTO: return "AUTO";
    case ON: return "ON";
    case OFF: return "OFF";
    default: return "UNKNOWN";
  }
}
```

Code for IR:

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

// Wi-Fi credentials
const char* ssid = "WPA";
const char* password = "12345678";

// IR sensor pins
const int sensor1 = 27;
const int sensor2 = 26;
const int sensor3 = 25;

// Light control pins
const int relay1 = 23;  // Controls Light1
const int relay2 = 22;  // Controls Light2
const int led3 = 21;    // Light3 controlled directly by GPIO

// States and modes
bool motion1 = false, motion2 = false, motion3 = false;
bool light1 = false, light2 = false, light3 = false; // Light states for manual
mode
bool isAutomatic = true; // Mode: true = Auto, false = Manual
```

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

// Generate HTML for the web page
String generateHTML() {
  String html = "<!DOCTYPE html><html><head><title>Light Control</title>";
  html += "<style>body {font-family: Arial; text-align: center;}";
  html += ".btn {padding: 10px 20px; margin: 5px; font-size: 16px;}";
  html += ".on {background: yellow;} .off {background: gray;}";
  html += ".light {width: 50px; height: 50px; display: inline-block; margin: 10px;}";
  html += "</style></head><body>";
  html += "<h1>Motion-Activated Lights</h1>";
  html += "<h2>Mode: " + String(isAutomatic ? "Automatic" : "Manual") + "</h2>";

  html += "<button class='btn' onclick=\"location.href='/toggleMode'\">Switch to " +
          String(isAutomatic ? "Manual" : "Automatic") + " Mode</button>";

  html += "<div class='light " + String(light1 ? "on" : "off") + "'></div>";
  html += "<div class='light " + String(light2 ? "on" : "off") + "'></div>";
  html += "<div class='light " + String(light3 ? "on" : "off") + "'></div>";

  if (!isAutomatic) {
    html += "<br><button class='btn' onclick=\"location.href='/toggleLight?light=1'\">Toggle Light 1</button>";
    html += "<button class='btn' onclick=\"location.href='/toggleLight?light=2'\">Toggle Light 2</button>";
    html += "<button class='btn' onclick=\"location.href='/toggleLight?light=3'\">Toggle Light 3</button>";
  }

  html += "</body></html>";
  return html;
}
```

```
// Handle root request
void handleRoot() {
  server.send(200, "text/html", generateHTML());
}

// Toggle between Manual and Automatic modes
void handleToggleMode() {
  isAutomatic = !isAutomatic;
  if (isAutomatic) {
    // Turn off all lights when switching back to Auto
    light1 = light2 = light3 = false;
    digitalWrite(relay1, LOW);
    digitalWrite(relay2, LOW);
    digitalWrite(led3, LOW);
  }
  handleRoot();
}

// Toggle individual light in Manual mode
void handleToggleLight() {
  if (!isAutomatic) {
    String light = server.arg("light");
    if (light == "1") light1 = !light1;
    else if (light == "2") light2 = !light2;
    else if (light == "3") light3 = !light3;

    // Update light states
    digitalWrite(relay1, light1 ? HIGH : LOW);
    digitalWrite(relay2, light2 ? HIGH : LOW);
    digitalWrite(led3, light3 ? HIGH : LOW);
  }
  handleRoot();
}

void setup() {
```

```
  // Initialize serial communication
  Serial.begin(115200);

  // Set up Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nWiFi connected. IP address: ");
  Serial.println(WiFi.localIP());

  // Set up web server routes
  server.on("/", handleRoot);
  server.on("/toggleMode", handleToggleMode);
  server.on("/toggleLight", handleToggleLight);
  server.begin();
  Serial.println("Web server started.");

  // Initialize pins
  pinMode(sensor1, INPUT);
  pinMode(sensor2, INPUT);
  pinMode(sensor3, INPUT);

  pinMode(relay1, OUTPUT);
  pinMode(relay2, OUTPUT);
  pinMode(led3, OUTPUT);

  // Turn off all lights initially
  digitalWrite(relay1, LOW);
  digitalWrite(relay2, LOW);
  digitalWrite(led3, LOW);
}

void loop() {
```

```
  // Handle motion in Automatic mode
  if (isAutomatic) {
    motion1 = digitalRead(sensor1);
    motion2 = digitalRead(sensor2);
    motion3 = digitalRead(sensor3);

    digitalWrite(relay1, motion1 ? LOW : HIGH);
    digitalWrite(relay2, motion2 ? HIGH : LOW);
    digitalWrite(led3, motion3 ? HIGH : LOW);
  }

  // Handle web requests
  server.handleClient();
}
```