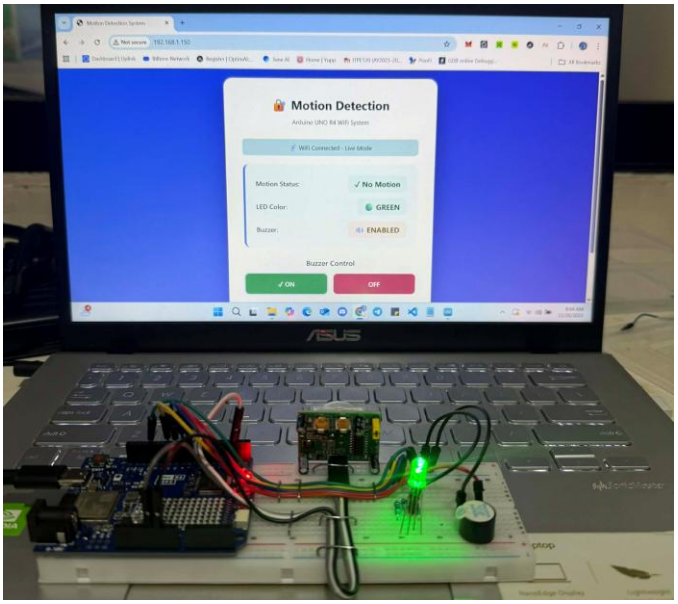
Web-Based Control of RGB LED with Motion Detection and Buzzer on Arduino UNO R4 (Phone/PC Browser Interface).

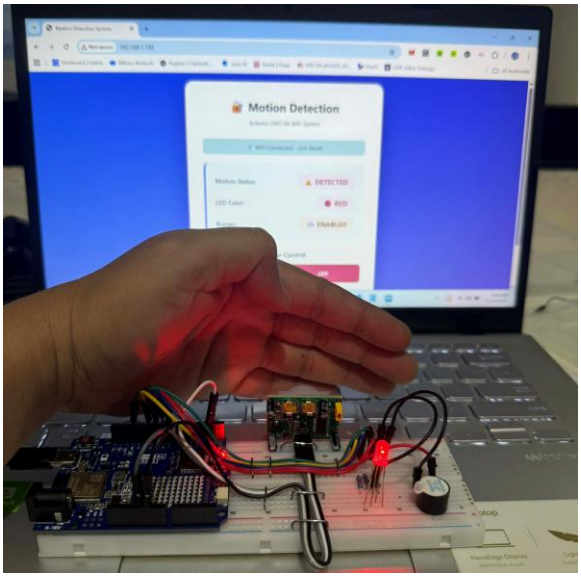
Sabino, Reyjoy P.
Banzali, Eman

Platform Technologies
ITPE121 BSIT – 2A

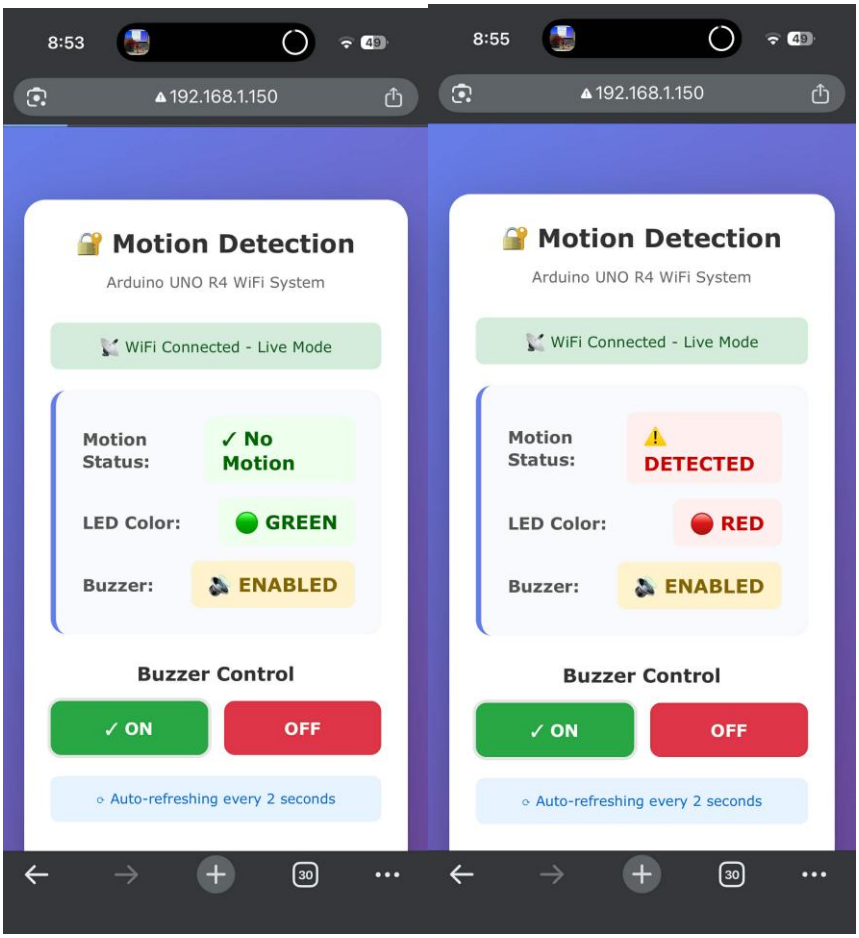
Things needed:
Arduino Uno R4 WiFi
RGB Led light
Active Buzzer
HC-SR501 PIR Motion Sensor
Jumper wires
Resistors



No Motion Detected on Arduino, RGB Green Light, No Motion detected on Web Browser



Motion Detected on Arduino, RGB Red Light and Buzzer toggled ON, Motion Detected on Web Browser



Display Motion Detected Buzzer toggled ON and No Motion Detected on Mobile Browser

Code Snippets

```
#include <WiFiS3.h>

const char* ssid = "Prince Gabriel";          // 2.4GHz WiFi only!
const char* password = "SYFamily4ever#";  // Case-sensitive

IPAddress staticIP(192, 168, 1, 150);      // Arduino's IP address
IPAddress gateway(192, 168, 1, 1);         // Usually your router's IP
IPAddress subnet(255, 255, 255, 0);        // Standard subnet mask
IPAddress dns(8, 8, 8, 8);                 // Google DNS

// Pin definitions
const int PIR_PIN = 2;        // PIR motion sensor
const int RED_PIN = 9;        // RGB LED - Red
const int GREEN_PIN = 10;     // RGB LED - Green
const int BLUE_PIN = 11;      // RGB LED - Blue
const int BUZZER_PIN = 8;     // Buzzer

// State variables
bool motionDetected = false;
bool buzzerEnabled = true;
bool wifiConnected = false;
unsigned long lastMotionTime = 0;
const unsigned long MOTION_TIMEOUT = 2000;

// Create WiFi server on port 80
WiFiServer server(80);

void setup() {
  Serial.begin(115200);
  delay(1000);

  // Initialize pins
  pinMode(PIR_PIN, INPUT);
  pinMode(RED_PIN, OUTPUT);
  pinMode(GREEN_PIN, OUTPUT);
  pinMode(BLUE_PIN, OUTPUT);
  pinMode(BUZZER_PIN, OUTPUT);

  // Start with green LED (no motion)
  setLEDColor(0, 255, 0);
  digitalWrite(BUZZER_PIN, LOW);

  Serial.println("\n╔═══════════════════════════════════════════════════════════╗");
```

```
  Serial.println("╔═══════════════════════════════════════╗");
  Serial.println("║   Motion Detection System v2.0        ║");
  Serial.println("║   Arduino UNO R4 WiFi                 ║");
  Serial.println("╚═══════════════════════════════════════╝\n");

  // Connect to WiFi with diagnostics
  connectToWiFi();
}

void connectToWiFi() {
  Serial.println("▶ WiFi Connection Diagnostics");
  Serial.println("─────────────────────────────────────────");
  Serial.print("SSID: ");
  Serial.println(ssid);
  Serial.print("Password Length: ");
  Serial.println(strlen(password));
  Serial.println();

  // Check if credentials look valid
  if (strlen(ssid) == 0) {
    Serial.println("❌ ERROR: SSID is empty!");
    Serial.println("   Please edit the code and set your WiFi name.");
    flashErrorLED();
    return;
  }

  if (strlen(password) < 8 && strlen(password) != 0) {
    Serial.println("⚠️  WARNING: Password seems short (less than 8 characters)");
    Serial.println("   Make sure this is correct.");
  }

  Serial.println("🔄 Attempting WiFi connection...");
  Serial.println("   (This may take 10-30 seconds)");
  Serial.println();

  WiFi.begin(ssid, password);

  int attempts = 0;
  int maxAttempts = 30; // 30 seconds timeout

  while (WiFi.status() != WL_CONNECTED && attempts < maxAttempts) {
    delay(1000);
    attempts++;
    Serial.print("   Attempt ");
    Serial.print(attempts);
    Serial.print("/");
    Serial.print(maxAttempts);
```

```
  Serial.print(" - Status: ");
  printWiFiStatus();
 }

 Serial.println();

 // Check if connected but no IP assigned (DHCP failure)
 if (WiFi.status() == WL_CONNECTED && WiFi.localIP() == IPAddress(0, 0, 0, 0)) {
  Serial.println("⚠️  Connected but no IP assigned (DHCP failed)");
  Serial.println("   Trying static IP configuration...");
  Serial.println();

  WiFi.disconnect();
  delay(1000);

  // Configure static IP
  WiFi.config(staticIP, dns, gateway, subnet);
  WiFi.begin(ssid, password);

  attempts = 0;
  while (WiFi.status() != WL_CONNECTED && attempts < 20) {
   delay(1000);
   attempts++;
   Serial.print("   Static IP attempt ");
   Serial.print(attempts);
   Serial.print("/20 - Status: ");
   printWiFiStatus();
  }
  Serial.println();
 }

 if (WiFi.status() == WL_CONNECTED) {
  wifiConnected = true;

  // Final check for valid IP
  IPAddress ip = WiFi.localIP();
  if (ip == IPAddress(0, 0, 0, 0)) {
   Serial.println("❌ ERROR: Still no valid IP address!");
   Serial.println();
   Serial.println("🔧 MANUAL CONFIGURATION NEEDED:");
   Serial.println("   1. Check your router's IP range");
   Serial.println("      (Common: 192.168.1.x or 192.168.0.x)");
   Serial.println("   2. Update staticIP in code (lines 16-19)");
   Serial.println("   3. Ensure IP is not already in use");
   Serial.println();
   wifiConnected = false;
```

```
    flashErrorLED();
    return;
  }

  Serial.println("╔═══════════════════════════════════════╗");
  Serial.println("║  ✓ WiFi Connected Successfully!       ║");
  Serial.println("╚═══════════════════════════════════════╝");
  Serial.println();
  Serial.print("📡 IP Address: ");
  Serial.println(WiFi.localIP());
  Serial.print("🌐 Gateway: ");
  Serial.println(WiFi.gatewayIP());
  Serial.print("📶 Signal Strength: ");
  Serial.print(WiFi.RSSI());
  Serial.println(" dBm");
  Serial.println();
  Serial.println("🌐 WEB INTERFACE ACCESS:");
  Serial.print("   http://");
  Serial.println(WiFi.localIP());
  Serial.println();
  Serial.println("   Copy this address to your browser!");
  Serial.println("═══════════════════════════════════════");
  Serial.println("System ready! Monitoring for motion...\n");

  // Start server
  server.begin();

  // Success beep
  tone(BUZZER_PIN, 1000, 100);
  delay(150);
  tone(BUZZER_PIN, 1500, 100);

} else {
  wifiConnected = false;
  Serial.println("╔═══════════════════════════════════════╗");
  Serial.println("║  ❌ WiFi Connection FAILED            ║");
  Serial.println("╚═══════════════════════════════════════╝");
  Serial.println();
  Serial.println("🔧 TROUBLESHOOTING STEPS:");
  Serial.println("   1. Verify SSID is correct (case-sensitive)");
  Serial.println("   2. Verify password is correct");
  Serial.println("   3. Ensure using 2.4GHz WiFi (NOT 5GHz)");
  Serial.println("   4. Check if router has MAC filtering");
  Serial.println("   5. Move Arduino closer to router");
  Serial.println("   6. Restart router if needed");
```

```
    Serial.println();
    Serial.println("⚠️  System will work in LOCAL MODE only");
    Serial.println("   (Motion detection active, no web control)");
    Serial.println();

    flashErrorLED();
  }
}

void printWiFiStatus() {
  switch (WiFi.status()) {
    case WL_IDLE_STATUS:
      Serial.println("Idle");
      break;
    case WL_NO_SSID_AVAIL:
      Serial.println("❌ Network not found!");
      break;
    case WL_CONNECTED:
      Serial.println("✓ Connected");
      break;
    case WL_CONNECT_FAILED:
      Serial.println("❌ Connection failed");
      break;
    case WL_CONNECTION_LOST:
      Serial.println("❌ Connection lost");
      break;
    case WL_DISCONNECTED:
      Serial.println("Disconnected");
      break;
    default:
      Serial.print("Unknown (");
      Serial.print(WiFi.status());
      Serial.println(")");
      break;
  }
}

void flashErrorLED() {
  // Flash red LED to indicate error
  for (int i = 0; i < 5; i++) {
    setLEDColor(255, 0, 0);
    tone(BUZZER_PIN, 500, 100);
    delay(200);
    setLEDColor(0, 0, 0);
    delay(200);
  }
```

```
  setLEDColor(0, 255, 0); // Back to green
}

void loop() {
  // Check motion sensor (works with or without WiFi)
  checkMotion();

  // Handle web clients only if WiFi is connected
  if (wifiConnected) {
    handleWebClient();
  }

  // Update outputs based on state
  updateOutputs();
}

void checkMotion() {
  int pirState = digitalRead(PIR_PIN);

  if (pirState == HIGH) {
    if (!motionDetected || (millis() - lastMotionTime > MOTION_TIMEOUT)) {
      motionDetected = true;
      lastMotionTime = millis();
      Serial.println(">>> ⚠️  MOTION DETECTED!");
    }
  } else {
    if (motionDetected && (millis() - lastMotionTime > MOTION_TIMEOUT)) {
      motionDetected = false;
      Serial.println(">>> ✓ No motion - IDLE");
    }
  }
}

void updateOutputs() {
  if (motionDetected) {
    // Motion detected: RED LED
    setLEDColor(255, 0, 0);

    // Buzzer ON only if enabled
    if (buzzerEnabled) {
      tone(BUZZER_PIN, 2000);
    } else {
      noTone(BUZZER_PIN);
    }
  } else {
    // No motion: GREEN LED
```

```
    setLEDColor(0, 255, 0);
    noTone(BUZZER_PIN);
  }
}

void setLEDColor(int red, int green, int blue) {
  analogWrite(RED_PIN, red);
  analogWrite(GREEN_PIN, green);
  analogWrite(BLUE_PIN, blue);
}

void handleWebClient() {
  WiFiClient client = server.available();

  if (client) {
    Serial.println("🌐 New client connected");
    String request = "";
    String currentLine = "";

    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        request += c;

        if (c == '\n') {
          if (currentLine.length() == 0) {
            // Check for buzzer toggle request
            if (request.indexOf("GET /buzzer/on") >= 0) {
              buzzerEnabled = true;
              Serial.println("   ▶ Buzzer ENABLED via web");
            } else if (request.indexOf("GET /buzzer/off") >= 0) {
              buzzerEnabled = false;
              Serial.println("   ▶ Buzzer DISABLED via web");
            }

            // Send HTTP response
            sendHTMLResponse(client);
            break;
          } else {
            currentLine = "";
          }
        } else if (c != '\r') {
          currentLine += c;
        }
      }
    }
```

```
    delay(10);
    client.stop();
    Serial.println("   ✓ Client disconnected\n");
  }
}

void sendHTMLResponse(WiFiClient &client) {
  // HTTP headers
  client.println("HTTP/1.1 200 OK");
  client.println("Content-type:text/html");
  client.println("Connection: close");
  client.println();

  // HTML content
  client.println("<!DOCTYPE html>");
  client.println("<html lang='en'>");
  client.println("<head>");
  client.println("<meta charset='UTF-8'>");
  client.println("<meta name='viewport' content='width=device-width, initial-scale=1.0'>");
  client.println("<title>Motion Detection System</title>");
  client.println("<style>");
  client.println("* { margin: 0; padding: 0; box-sizing: border-box; }");
  client.println("body { font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
background: linear-gradient(135deg, #667eea 0%, #764ba2 100%); min-height: 100vh;
display: flex; align-items: center; justify-content: center; padding: 20px; }");
  client.println(".container { background: white; border-radius: 20px; box-shadow: 0 20px
60px rgba(0,0,0,0.3); max-width: 500px; width: 100%; padding: 40px; }");
  client.println("h1 { color: #333; text-align: center; margin-bottom: 10px; font-size: 28px;
}");
  client.println(".subtitle { text-align: center; color: #666; margin-bottom: 30px; font-size:
14px; }");
  client.println(".status-card { background: #f8f9fa; border-radius: 15px; padding: 25px;
margin-bottom: 20px; border-left: 5px solid #667eea; }");
  client.println(".status-row { display: flex; justify-content: space-between; align-items:
center; margin-bottom: 15px; }");
  client.println(".status-row:last-child { margin-bottom: 0; }");
  client.println(".label { font-weight: 600; color: #555; font-size: 16px; }");
  client.println(".value { font-weight: bold; font-size: 18px; padding: 8px 16px; border-
radius: 8px; }");
  client.println(".motion-detected { background: #fee; color: #c00; }");
  client.println(".no-motion { background: #efe; color: #060; }");
  client.println(".led-red { background: #fee; color: #c00; }");
  client.println(".led-green { background: #efe; color: #060; }");
  client.println(".control-section { margin-top: 25px; }");
```

```
  client.println(".control-title { font-weight: 600; color: #333; margin-bottom: 15px; font-size: 18px; text-align: center; }");
  client.println(".button-group { display: flex; gap: 15px; }");
  client.println(".btn { flex: 1; padding: 15px; border: none; border-radius: 10px; font-size: 16px; font-weight: 600; cursor: pointer; transition: all 0.3s; text-decoration: none; text-align: center; display: block; }");
  client.println(".btn-on { background: #28a745; color: white; }");
  client.println(".btn-on:hover { background: #218838; transform: translateY(-2px); box-shadow: 0 5px 15px rgba(40,167,69,0.4); }");
  client.println(".btn-off { background: #dc3545; color: white; }");
  client.println(".btn-off:hover { background: #c82333; transform: translateY(-2px); box-shadow: 0 5px 15px rgba(220,53,69,0.4); }");
  client.println(".btn-active { box-shadow: 0 0 0 3px rgba(0,0,0,0.1); }");
  client.println(".footer { text-align: center; margin-top: 25px; color: #999; font-size: 13px; }");
  client.println(".refresh-note { background: #e7f3ff; padding: 12px; border-radius: 8px; text-align: center; color: #0066cc; font-size: 13px; margin-top: 20px; }");
  client.println(".wifi-status { background: #d4edda; color: #155724; padding: 10px; border-radius: 8px; text-align: center; margin-bottom: 20px; font-size: 14px; }");
  client.println("@media (max-width: 480px) { .container { padding: 25px; } h1 { font-size: 24px; } }");
  client.println("</style>");
  client.println("<script>");
  client.println("setTimeout(function(){ location.reload(); }, 2000);");
  client.println("</script>");
  client.println("</head>");
  client.println("<body>");
  client.println("<div class='container'>");

  client.println("<h1>🔐 Motion Detection</h1>");
  client.println("<div class='subtitle'>Arduino UNO R4 WiFi System</div>");

  client.println("<div class='wifi-status'>📡 WiFi Connected - Live Mode</div>");

  client.println("<div class='status-card'>");
  client.println("<div class='status-row'>");
  client.println("<span class='label'>Motion Status:</span>");
  if (motionDetected) {
    client.println("<span class='value motion-detected'>⚠️ DETECTED</span>");
  } else {
    client.println("<span class='value no-motion'>✓ No Motion</span>");
  }
  client.println("</div>");

  client.println("<div class='status-row'>");
  client.println("<span class='label'>LED Color:</span>");
```

```
  if (motionDetected) {
    client.println("<span class='value led-red'>🔴 RED</span>");
  } else {
    client.println("<span class='value led-green'>🟢 GREEN</span>");
  }
  client.println("</div>");

  client.println("<div class='status-row'>");
  client.println("<span class='label'>Buzzer:</span>");
  if (buzzerEnabled) {
    client.println("<span class='value' style='background: #fff3cd; color: #856404;'>🔊
ENABLED</span>");
  } else {
    client.println("<span class='value' style='background: #d1ecf1; color: #0c5460;'>🔕
DISABLED</span>");
  }
  client.println("</div>");
  client.println("</div>");

  client.println("<div class='control-section'>");
  client.println("<div class='control-title'>Buzzer Control</div>");
  client.println("<div class='button-group'>");
  if (buzzerEnabled) {
    client.println("<a href='/buzzer/on' class='btn btn-on btn-active'>✓ ON</a>");
    client.println("<a href='/buzzer/off' class='btn btn-off'>OFF</a>");
  } else {
    client.println("<a href='/buzzer/on' class='btn btn-on'>ON</a>");
    client.println("<a href='/buzzer/off' class='btn btn-off btn-active'>✓ OFF</a>");
  }
  client.println("</div>");
  client.println("</div>");

  client.println("<div class='refresh-note'>🔄 Auto-refreshing every 2 seconds</div>");

  client.println("<div class='footer'>");
  client.print("IP: ");
  client.print(WiFi.localIP());
  client.print(" | Signal: ");
  client.print(WiFi.RSSI());
  client.println(" dBm");
  client.println("</div>");

  client.println("</div>");
  client.println("</body>");
  client.println("</html>");
}
```