Setting up a home automation system with an ESP8266 that allows for control via both a web page and a WhatsApp bot is a great way to manage devices like an RC fan, buzzer, water pump, and LED. Here's a brief overview of how to set up such a system:

### **1. Components Needed**

* **ESP8266 Module**: For Wi-Fi connectivity and control.
* **Relays**: To switch high-voltage devices on and off. Use a relay module that can handle the current and voltage requirements of your devices.
* **Devices**:
  + **RC Fan**
  + **Buzzer**
  + **Water Pump**
  + **LED**
* **Power Supply**: Suitable for both the ESP8266 and the relay module.
* **Breadboard and Jumper Wires**: For prototyping connections.
* **Smartphone**: For WhatsApp control.
* **Arduino IDE**: For programming the ESP8266.

### **2. Hardware Setup**

1. **Connect the ESP8266 to the Relay Module**:
   * Connect the relay module to the ESP8266. Each relay will need a control pin connected to the ESP8266 GPIO.
   * Relay connections:
     + **VCC**: To 3.3V (or 5V depending on relay module specification).
     + **GND**: To GND on the ESP8266.
     + **IN1, IN2, IN3, IN4**: Connect these to available GPIO pins on the ESP8266 for controlling each relay.
     + **COM** and **NO** or **NC**: Connect these to the devices you want to control.
2. **Wire the Devices**:
   * **RC Fan**: Connect to one of the relay’s **COM** and **NO** terminals.
   * **Buzzer**: Connect similarly to another relay.
   * **Water Pump**: Connect to a third relay.
   * **LED**: Connect to a fourth relay.

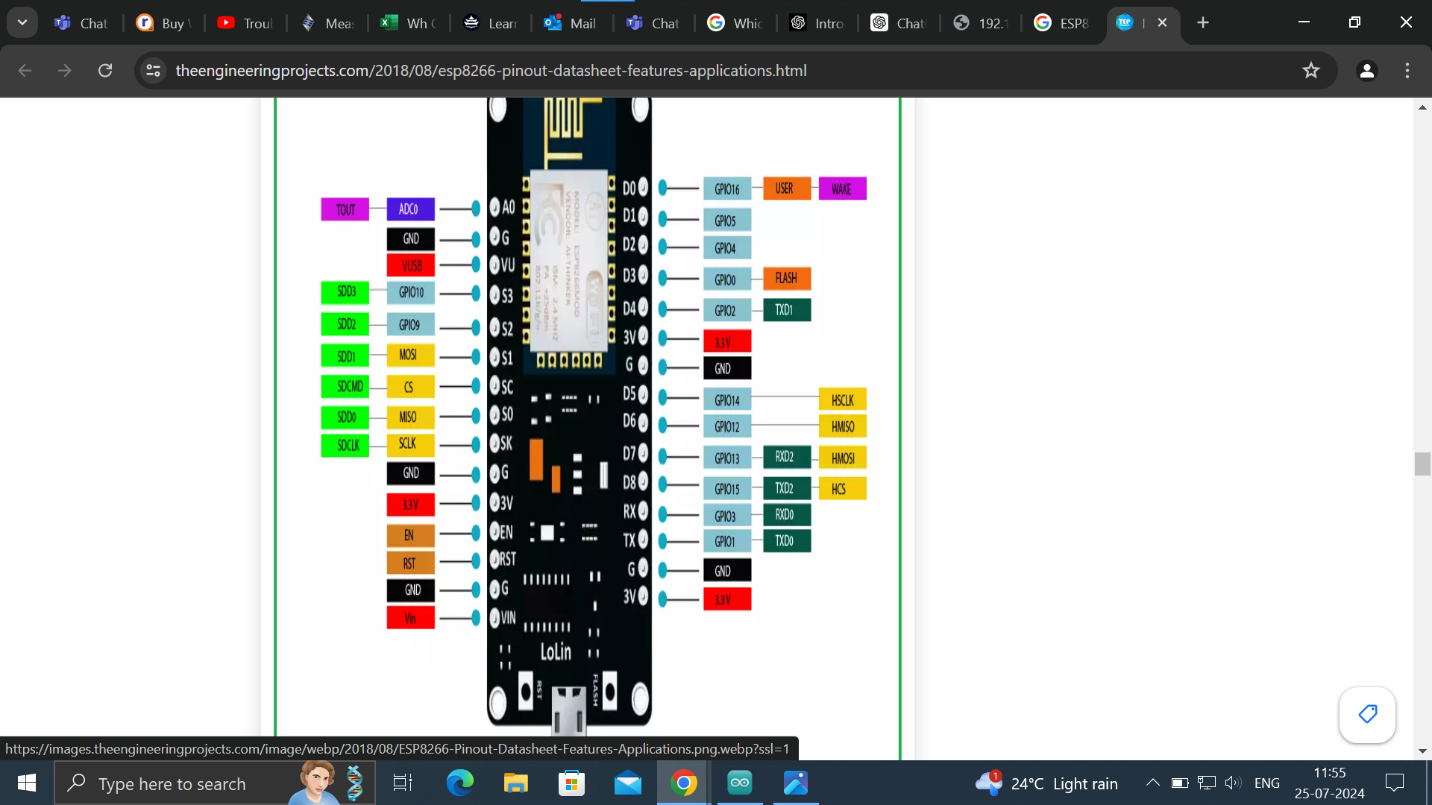
### **3. Programming the ESP8266**

1. **Set Up the Arduino IDE**:
   * Install the ESP8266 board package in the Arduino IDE as mentioned in the previous response.
   * Add the ESP8266 board to the IDE by navigating to File > Preferences and adding the following URL to the "Additional Board Manager URLs": <http://arduino.esp8266.com/stable/package_esp8266com_index.json.>

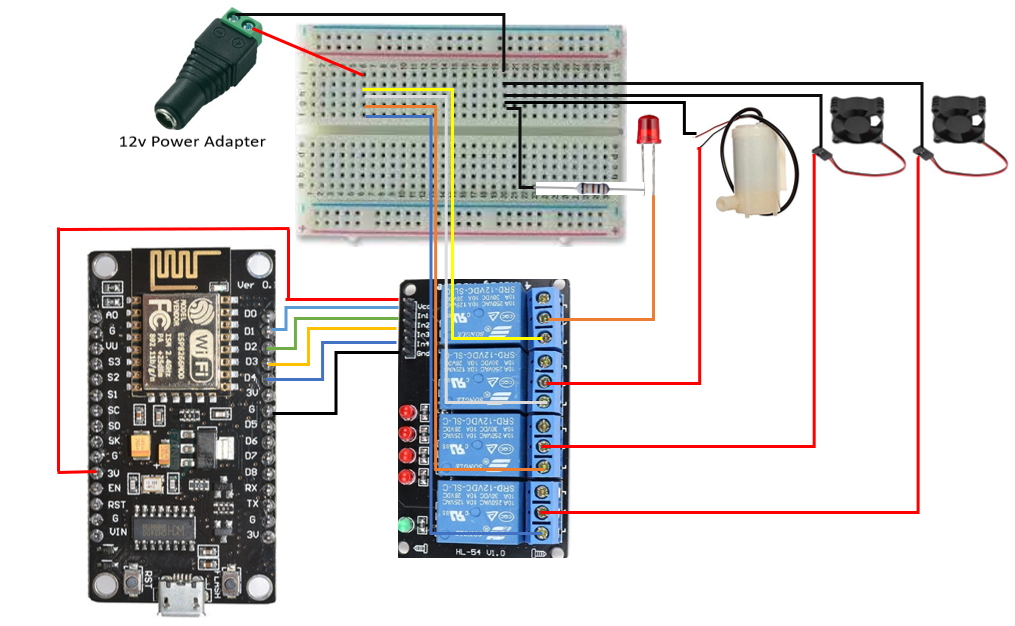
**Component:-**

|  |  |  |
| --- | --- | --- |
| **Component** | **Specification** | **Quantity** |
| ESP8266 | Microcontroller Board | 1 |
| Relay Module | 4 relays, 5V control | 1 |
| DC Power Jack Adapter Connector | For connecting to 9V battery | 1 |
| Battery | Standard 9V battery | 1 |
| Mini Breadboard | Small prototype board | 1 |
| DC 3-6V Mini Micro Submersible Water Pump | 3-6V Submersible Pump | 1 |
| Cooling Fan | 12V Cooling Fan, 30mm x 10mm | 2 |
| LED | 5mm Red LED | 1 |
| Male to Female Jumper Wires | 20cm long, male to female | 20(PCS) 1 PACK |
| Male to Male Jumper Wires | 20cm long, male to male | 9(PCS) 1PACK |
| DC Power Jack Adapter | For connecting to external power supply | 1 |
| Resistor | 1k Ohm Resistor | 1 |

**Pin0ut Of ESP8266:-**

****

**Connection Diagram:-**

****

**Connection ESP8266 With Relay module:-**

|  |  |
| --- | --- |
| **ESP8266** | **Relay Module** |
| 3V | VCC |
| GND | GND |
| D1 | INT1 |
| D2 | INT2 |
| D3 | INT2 |
| D4 | INT3 |

**Code:-**

#include <ESP8266WiFi.h>

#include <ThingESP.h>

// ThingESP setup

ThingESP8266 thing("Vaibhavsul", "HOMEAUTOMATION", "VAIBHAVSUL");

// Device pins

const int WATERPUMP = D1; // Water Pump

const int BEDROOMLIGHT = D2; // Bedroom Light

const int KITCHENFAN = D3; // Buzzer

const int BUZZER = D4; // Kitchen Fan

// HTTP server setup

WiFiServer server(80);

// HTTP request handling

String header;

String outputWaterpumpState = "off";

String outputBedroomLightState = "off";

String outputBuzzerState = "off";

String outputKitchenFanState = "off";

// Timing

unsigned long currentTime = millis();

unsigned long previousTime = 0;

const long timeoutTime = 2000;

void setup() {

Serial.begin(115200);

// Initialize device pins

pinMode(WATERPUMP, OUTPUT);

pinMode(BEDROOMLIGHT, OUTPUT);

pinMode(BUZZER, OUTPUT);

pinMode(KITCHENFAN, OUTPUT);

// Set initial state of devices

digitalWrite(WATERPUMP, HIGH);

digitalWrite(BEDROOMLIGHT, HIGH);

digitalWrite(BUZZER, HIGH);

digitalWrite(KITCHENFAN, HIGH);

// Connect to WiFi

WiFi.begin("IOT", "123456789");

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.print("WiFi connected. IP address: ");

Serial.println(WiFi.localIP());

// Start HTTP server

server.begin();

// Initialize ThingESP

thing.SetWiFi("IOT", "123456789");

thing.initDevice();

}

void loop() {

// Handle HTTP requests

WiFiClient client = server.available();

if (client) {

Serial.println("New Client.");

String currentLine = "";

currentTime = millis();

previousTime = currentTime;

while (client.connected() && currentTime - previousTime <= timeoutTime) {

currentTime = millis();

if (client.available()) {

char c = client.read();

Serial.write(c);

header += c;

if (c == '\n') {

if (currentLine.length() == 0) {

sendHeader(client);

// Handle HTTP device control

handleDevice(client, WATERPUMP, outputWaterpumpState, 1, "Water Pump", "#FF5733");

handleDevice(client, BEDROOMLIGHT, outputBedroomLightState, 2, "Bedroom Light", "#45B39D");

handleDevice(client, BUZZER, outputBuzzerState, 3, "Buzzer", "#5499C7");

handleDevice(client, KITCHENFAN, outputKitchenFanState, 4, "Kitchen Fan", "#F4D03F");

sendFooter(client);

break;

} else {

currentLine = "";

}

} else if (c != '\r') {

currentLine += c;

}

}

}

header = "";

client.stop();

Serial.println("Client disconnected.");

Serial.println("");

}

// Handle ThingESP commands

thing.Handle();

}

void sendHeader(WiFiClient& client) {

client.println("HTTP/1.1 200 OK");

client.println("Content-type:text/html");

client.println("Connection: close");

client.println();

client.println("<!DOCTYPE html>");

client.println("<html><head>");

client.println("<meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">");

client.println("<title>ESP8266 IoT Control</title>");

client.println("<style>");

client.println("body { font-family: Arial, Helvetica, sans-serif; background-color: #f0f0f0; text-align: center; margin: 0; padding: 0;}");

client.println(".container { max-width: 600px; margin: auto; padding: 20px;}");

client.println(".device-box { background-color: #ffffff; padding: 20px; margin-bottom: 20px; border-radius: 10px; box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1); transition: transform 0.3s;}");

client.println(".device-box:hover { transform: scale(1.05); }");

client.println(".device-title { font-size: 24px; color: #333; margin-bottom: 10px; }");

client.println(".device-state { font-size: 18px; color: #666; margin-bottom: 15px; }");

client.println(".button { display: inline-block; padding: 10px 20px; font-size: 16px; cursor: pointer; border: none; border-radius: 5px; margin-right: 10px; transition: background-color 0.3s, color 0.3s; text-decoration: none;}");

client.println(".button:hover { filter: brightness(90%); }");

client.println(".buttonOn { background-color: #2ecc71; color: #fff; }");

client.println(".buttonOff { background-color: #e74c3c; color: #fff; }");

client.println("</style></head>");

client.println("<body>");

client.println("<div class=\"container\">");

}

void sendFooter(WiFiClient& client) {

client.println("</div>");

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

}

void handleDevice(WiFiClient& client, int devicePin, String& outputState, int deviceNumber, const char\* deviceName, const char\* color) {

client.println("<div class=\"device-box\" style=\"border-left: 6px solid " + String(color) + "\">");

client.println("<div class=\"device-title\">" + String(deviceName) + "</div>");

client.println("<div class=\"device-state\">State: " + outputState + "</div>");

if (header.indexOf("/" + String(deviceNumber) + "/on") >= 0) {

if (outputState == "off") {

Serial.println(String(deviceName) + " is turned on");

outputState = "on";

digitalWrite(devicePin, LOW); // Turn the device on

}

} else if (header.indexOf("/" + String(deviceNumber) + "/off") >= 0) {

if (outputState == "on") {

Serial.println(String(deviceName) + " is turned off");

outputState = "off";

digitalWrite(devicePin, HIGH); // Turn the device off

}

}

client.println("<a class=\"button buttonOn\" href=\"/" + String(deviceNumber) + "/on\">Turn On</a>");

client.println("<a class=\"button buttonOff\" href=\"/" + String(deviceNumber) + "/off\">Turn Off</a>");

client.println("</div>");

}

// Handle ThingESP commands

String HandleResponse(String query) {

if (query == "waterpump off") {

digitalWrite(WATERPUMP, HIGH);

return "Done: Waterpump Turned OFF";

}

else if (query == "waterpump on") {

digitalWrite(WATERPUMP, LOW);

return "Done: Waterpump Turned ON";

}

if (query == "bedroomlight off") {

digitalWrite(BEDROOMLIGHT, HIGH);

return "Done: Bedroom Light Turned OFF";

}

else if (query == "bedroomlight on") {

digitalWrite(BEDROOMLIGHT, LOW);

return "Done: Bedroom Light Turned ON";

}

if (query == "buzzer off") {

digitalWrite(BUZZER, HIGH);

return "Done: Buzzer Turned OFF";

}

else if (query == "buzzer on") {

digitalWrite(BUZZER, LOW);

return "Done: Buzzer Turned ON";

}

if (query == "kitchenfan off") {

digitalWrite(KITCHENFAN, HIGH);

return "Done: Kitchen Fan Turned OFF";

}

else if (query == "kitchenfan on") {

digitalWrite(KITCHENFAN, LOW);

return "Done: Kitchen Fan Turned ON";

}

if (query == "all on") {

digitalWrite(WATERPUMP, LOW);

digitalWrite(BEDROOMLIGHT, LOW);

digitalWrite(BUZZER, LOW);

digitalWrite(KITCHENFAN, LOW);

return "Done: All Lights, Buzzer, Kitchen Fan, and Waterpump Turned ON";

}

else if (query == "all off") {

digitalWrite(WATERPUMP, HIGH);

digitalWrite(BEDROOMLIGHT, HIGH);

digitalWrite(BUZZER, HIGH);

digitalWrite(KITCHENFAN, HIGH);

return "Done: All Lights, Buzzer, Kitchen Fan, and Waterpump Turned OFF";

}

return "Your query was invalid..";

}

**4. Implement WhatsApp Control**:

* + **Set Up Twilio**: Sign up for a Twilio account and set up a WhatsApp sandbox or number.
  + **Create a Webhook**: Use Twilio’s API to send received WhatsApp messages to a webhook.
  + **Handle WhatsApp Messages**:
    - Write a server-side script to process incoming messages from Twilio, parse the commands, and send HTTP requests to the ESP8266 to control the relays.
  + **Example**:
    - Use a service like Node.js or Python with Flask/Django to handle incoming HTTP POST requests from Twilio, parse the message content, and send corresponding HTTP commands to the ESP8266.

### 5. **Deploy and Test**

1. **Upload Code to ESP8266**:
   * Connect the ESP8266 to your computer and upload the code using the Arduino IDE.
2. **Test the Web Interface**:
   * Open a web browser and navigate to the ESP8266’s IP address to access the control page and test device control.
3. **Test WhatsApp Control**:
   * Send commands to your WhatsApp bot and verify that they control the devices as expected.
4. **Secure Your System**:
   * Implement basic security measures like password protection for the web interface and secure the WhatsApp bot.

### 6. **Expand and Automate**

* **Add More Devices**: Integrate additional relays and devices as needed.
* **Create Automation Rules**: Develop automation scripts to control devices based on conditions or schedules.

This setup provides both a user-friendly web interface and the flexibility of WhatsApp control, making it a versatile solution for home automation.