NEXT LEVEL PROJECT TUTORIAL: CONTROL YOUR ESP8266 WITH NATURAL LANGUAGE COMMANDS

Project: Smart Home Automation with ESP8266 using LIMMA
Tool: Python Package - Language Interface Model for Machine Automation

---

🚀 Overview

Ye project aapko allow karega ki aap apne **ESP8266 devices** ko **natural language commands** ke through control karein.
Aap simply bolenge ya type karenge commands jaise "*Turn on the living room light*" aur aapka device respond karega instantly.

🛠️ Prerequisites

- ESP8266 board (NodeMCU ya similar)
- VS Code installed
- Python 3 installed
- Internet connection
- Basic understanding of Python

Step 1: Installation

Install required Python packages:

```
pip install limma pyvoicekit
```

Ye packages aapko voice commands aur LIMMA API integration ke liye chahiye.

Step 2: Get API Key

1. Open LIMMA website
2. Sign up or log in
3. Generate a new **API Key**
4. Copy the key for use in your Python code

> ⚠ Keep your API Key safe, do not share publicly.

Step 3: Upload Code to ESP8266

1. Connect your ESP8266 to your computer

2. Open Arduino IDE or PlatformIO
3. Upload your **Home Automation System code** to the ESP8266

> Ensure your ESP is connected to the same Wi-Fi network as your computer for testing.

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <ArduinoJson.h>
#include <EEPROM.h>
#include <ESP8266mDNS.h>

// ==========================================
// CONFIGURATION
// ==========================================

// WiFi Configuration
const char* ssid = "YOUR_WIFI_NAME";         // Change to your WiFi SSID
const char* password = "YOUR_WIFI_PASSWORD";  // Change to your WiFi password

// Device Configuration
const String device_name = "LIMMA-ESP8266";
const String device_version = "0.1.0";

// Pin Definitions
const int CH01_PIN = D1;
const int CH02_PIN = D2;
const int CH03_PIN = D3;
const int CH04_PIN = D4;
const int CH05_PIN = D5;
const int CH06_PIN = D6;

// Status LED
const int STATUS_LED = LED_BUILTIN;

// Web Server
ESP8266WebServer server(80);

// Device States
struct DeviceState {
  bool ch01 = false;
  bool ch02 = false;
  bool ch03 = false;
  bool ch04 = false;
  bool ch05 = false;
  bool ch06 = false;
  String last_command = "";
  unsigned long uptime = 0;
  int request_count = 0;
};
DeviceState device_state;
```

```
// ==========================================
// SETUP
// ==========================================
void setup() {
  Serial.begin(115200);
  Serial.println();
  Serial.println("================================");
  Serial.println("LIMMA ESP8266 Home Automation (STA MODE)");
  Serial.println("Version: " + device_version);
  Serial.println("================================");

  EEPROM.begin(512);
  initializePins();
  connectToWiFi();
  setupMDNS();
  setupWebServer();

  server.begin();
  Serial.println("✅ HTTP server started");
  Serial.print("Device IP: ");
  Serial.println(WiFi.localIP());

  startupSequence();
}

// ==========================================
// LOOP
// ==========================================
void loop() {
  server.handleClient();
  MDNS.update();
  device_state.uptime = millis();

  static unsigned long last_blink = 0;
  if (millis() - last_blink > 2000) {
    digitalWrite(STATUS_LED, !digitalRead(STATUS_LED));
    last_blink = millis();
  }
}

// ==========================================
// FUNCTIONS
// ==========================================
void initializePins() {
  pinMode(CH01_PIN, OUTPUT);
  pinMode(CH02_PIN, OUTPUT);
  pinMode(CH03_PIN, OUTPUT);
  pinMode(CH04_PIN, OUTPUT);
  pinMode(CH05_PIN, OUTPUT);
  pinMode(CH06_PIN, OUTPUT);
  pinMode(STATUS_LED, OUTPUT);

  digitalWrite(CH01_PIN, LOW);
  digitalWrite(CH02_PIN, LOW);
```

```
  digitalWrite(CH03_PIN, LOW);
  digitalWrite(CH04_PIN, LOW);
  digitalWrite(CH05_PIN, LOW);
  digitalWrite(CH06_PIN, LOW);
  digitalWrite(STATUS_LED, HIGH);
}

void connectToWiFi() {
  Serial.println("Connecting to WiFi...");
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
    digitalWrite(STATUS_LED, !digitalRead(STATUS_LED));
  }

  Serial.println();
  Serial.println("✅ Connected to WiFi");
  Serial.print("IP Address: ");
  Serial.println(WiFi.localIP());
}

void setupMDNS() {
  if (MDNS.begin("limma")) {
    Serial.println("✅ mDNS started (http://limma.local)");
    MDNS.addService("http", "tcp", 80);
  }
}

void startupSequence() {
  for (int i = 0; i < 3; i++) {
    digitalWrite(CH01_PIN, HIGH);
    digitalWrite(CH02_PIN, HIGH);
    delay(200);
    digitalWrite(CH01_PIN, LOW);
    digitalWrite(CH02_PIN, LOW);
    delay(200);
  }
  Serial.println("✅ Startup complete");
}

// ========================================
// WEB SERVER ROUTES
// ========================================
void setupWebServer() {
  server.on("/", HTTP_GET, []() { handleRoot(); });
  server.on("/status", HTTP_GET, []() { handleStatus(); });
  server.on("/ping", HTTP_GET, []() { handlePing(); });
  server.onNotFound(handleNotFound);

  // Channel control
  server.on("/ch01on", HTTP_GET, []() { handleChannel(1, true); });
```

```
  server.on("/ch01off", HTTP_GET, []() { handleChannel(1, false); });
  server.on("/ch02on", HTTP_GET, []() { handleChannel(2, true); });
  server.on("/ch02off", HTTP_GET, []() { handleChannel(2, false); });
  server.on("/ch03on", HTTP_GET, []() { handleChannel(3, true); });
  server.on("/ch03off", HTTP_GET, []() { handleChannel(3, false); });
  server.on("/ch04on", HTTP_GET, []() { handleChannel(4, true); });
  server.on("/ch04off", HTTP_GET, []() { handleChannel(4, false); });
  server.on("/ch05on", HTTP_GET, []() { handleChannel(5, true); });
  server.on("/ch05off", HTTP_GET, []() { handleChannel(5, false); });
  server.on("/ch06on", HTTP_GET, []() { handleChannel(6, true); });
  server.on("/ch06off", HTTP_GET, []() { handleChannel(6, false); });
}

// ========================================
// HANDLERS
// ========================================
void handlePing() {
  device_state.request_count++;
  DynamicJsonDocument doc(200);
  doc["status"] = "pong";
  doc["device"] = device_name;
  doc["version"] = device_version;

  String json;
  serializeJson(doc, json);
  server.send(200, "application/json", json);
}

void handleStatus() {
  device_state.request_count++;
  DynamicJsonDocument doc(1024);

  doc["device"] = device_name;
  doc["version"] = device_version;
  doc["uptime"] = device_state.uptime;
  doc["requests"] = device_state.request_count;

  JsonObject wifi = doc.createNestedObject("wifi");
  wifi["ssid"] = WiFi.SSID();
  wifi["ip"] = WiFi.localIP().toString();
  wifi["rssi"] = WiFi.RSSI();

  String json;
  serializeJson(doc, json);
  server.send(200, "application/json", json);
}

void handleChannel(int ch, bool state) {
  int pin;
  bool* st;

  switch (ch) {
    case 1: pin = CH01_PIN; st = &device_state.ch01; break;
    case 2: pin = CH02_PIN; st = &device_state.ch02; break;
```

```
    case 3: pin = CH03_PIN; st = &device_state.ch03; break;
    case 4: pin = CH04_PIN; st = &device_state.ch04; break;
    case 5: pin = CH05_PIN; st = &device_state.ch05; break;
    case 6: pin = CH06_PIN; st = &device_state.ch06; break;
    default: server.send(400, "text/plain", "Invalid channel"); return;
  }

  digitalWrite(pin, state ? HIGH : LOW);
  *st = state;
  server.send(200, "text/plain", "CH" + String(ch) + (state ? " ON" : " OFF"));
}

void handleRoot() {
  String html = R"rawliteral(
  <!DOCTYPE html>
  <html>
  <head>
    <title>LIMMA ESP8266 Control</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <style>
      body { font-family: Arial, sans-serif; background: #1a1a1a; color: #fff;
text-align: center; }
      h1 { color: #4CAF50; }
      .device-info { margin: 20px; padding: 10px; background: #2d2d2d; border-
radius: 8px; }
      .relay { margin: 10px; padding: 10px; border-radius: 6px; display: inline-
block; width: 150px; }
      .on { background: #4CAF50; }
      .off { background: #f44336; }
      a { text-decoration: none; color: white; display: block; padding: 8px;
margin-top: 5px; border-radius: 4px; }
      .btn-on { background: #4CAF50; }
      .btn-off { background: #f44336; }
    </style>
  </head>
  <body>
  )rawliteral";

  html += "<h1>🏠 LIMMA ESP8266</h1>";
  html += "<div class='device-info'>";
  html += "Device: " + device_name + "<br>";
  html += "Version: " + device_version + "<br>";
  html += "IP: " + WiFi.localIP().toString() + "<br>";
  html += "Uptime: " + String(device_state.uptime / 1000) + " sec<br>";
  html += "</div>";

  // Function to add relay cards dynamically
  auto addRelayCard = [&](int ch, bool state) {
    html += "<div class='relay " + String(state ? "on" : "off") + "'>";
    html += "<h3>Channel " + String(ch) + "</h3>";
    html += "<p>Status: " + String(state ? "ON" : "OFF") + "</p>";
    html += "<a class='btn-on' href='/ch0" + String(ch) + "on'>Turn ON</a>";
    html += "<a class='btn-off' href='/ch0" + String(ch) + "off'>Turn OFF</a>";
    html += "</div>";
```

```
  };

  addRelayCard(1, device_state.ch01);
  addRelayCard(2, device_state.ch02);
  addRelayCard(3, device_state.ch03);
  addRelayCard(4, device_state.ch04);
  addRelayCard(5, device_state.ch05);
  addRelayCard(6, device_state.ch06);

  html += "</body></html>";

  server.send(200, "text/html", html);
}


void handleNotFound() {
  server.send(404, "text/plain", "Not found");
}
```

Step 4: Prepare Python Code

1. Open VS Code
2. Create a new file named `app.py`
3. Paste the following starter code (replace `YOUR_API_KEY` later):

```
from limma import Limma, LimmaConfig, NetworkUtils
from pyvoicekit import listen, speak
import time

apiKey = 'YOUR_API_KEY'
espIP = 'ESP_STA_IP_ADDRESS' # COPY FORM YOUR ARDUINO IDE SERIAL MONITOR

def main():
    print("🚀 LIMMA Client Starting...")

    # Configuration
    config = LimmaConfig(
        esp_ip=espIP,  # Will be auto-discovered if needed
        application_type="home",
        device_map={
            "light": "ch01",
            "fan": "ch02",
            "ac": "ch03",
            "tv": "ch04",
            "bedroom light": "ch05",
            "kitchen light": "ch06"
        },
        api_key=apiKey,  # Get from the website
        reply=True
```

```
)

# Initialize LIMMA
limma = Limma(config)

# Auto-discover ESP if current IP doesn't work
if not limma.esp_manager.check_connection():
    print("❌ Cannot connect to configured ESP IP")
    print("🔍 Attempting auto-discovery...")
    discovered_ip = limma.auto_discover_esp()
    if discovered_ip:
        config.esp_ip = discovered_ip
        limma.config.esp_ip = discovered_ip
        print(f"☑️ Using discovered ESP: {discovered_ip}")
    else:
        print("❌ No ESP devices found. Please check your network.")
        return

# Setup ESP (optional)
print("\n🔧 Setting up ESP...")
if limma.setup_esp():
    print("☑️ ESP setup completed")
else:
    print("⚠️ ESP setup had issues, continuing anyway...")

print("\n" + "="*50)
print("🎙️ LIMMA Voice Control Ready!")
print("Commands you can try:")
print("- 'turn on the light'")
print("- 'turn off all devices'")
print("- 'turn on fan and ac'")
print("- 'what's the status?' (for info)")
print("- 'clear context' (to reset memory)")
print("- 'quit' or 'exit' (to stop)")
print("="*50 + "\n")

try:
    while True:
        print("🎤 Listening for command...")

        # Get voice input
        command = listen()

        if command is None:
            continue

        print(f"👤 You said: {command}")

        # Handle special commands
        if command.lower() in ['quit', 'exit', 'stop']:
            speak("Goodbye!")
            break
        elif command.lower() in ['clear context', 'reset memory']:
            limma.clear_context()
```

```
                speak("Context cleared")
                continue
            elif command.lower() in ['status', 'context info']:
                info = limma.get_context_info()
                speak(f"I remember {info['context_count']} previous commands")
                continue

            # Process the command
            success, reply = limma.execute_command(command)

            if success and reply:
                print(f"🤖 Reply: {reply}")
                speak(reply)
            elif success:
                speak("Command executed successfully")
            else:
                speak("Sorry, there was an issue executing that command.")

            print("\n" + "-"*30 + "\n")

    except KeyboardInterrupt:
        print("\n👋 Shutting down LIMMA...")
        speak("Goodbye!")

    except Exception as e:
        print(f"❌ Unexpected error: {e}")
        speak("An error occurred, shutting down")

    finally:
        limma.close()
        print("🔘 LIMMA shut down complete")


if __name__ == "__main__":
    main()
```

Step 5: Add Your API Key

Replace `"YOUR_API_KEY"` with the key you generated in Step 2.

Step 6: Connect to ESP and Run

1. Connect your ESP8266 to your computer via USB
2. Ensure the device is powered and on the same network
3. Run your Python script:

```
python app.py
```

4. Try speaking or typing commands, e.g.:
    - `"Abe light jala be"`
    - `"garmi lag rahi hai pankha chala"`

You should see your ESP device respond accordingly.

☑ Testing & Feedback

- Test **multiple commands**
- Note any **bugs** or **unexpected behavior**
- Fill the feedback form (if provided by mentor)

💡 Tips

- Make sure Wi-Fi credentials are correctly set in ESP code
- Speak clearly for voice commands
- Use logs/prints in Python to debug

Have fun controlling your home devices with just your voice! 🎉