ESP32 Control by Web or Python

1. **ESP32** 上的服务器端程序

以下是在 ESP32 上运行的 Web 程序，可以使用 Web 总结控制，也可以使用 Python 通过 post/get 访问从而控制 ESP32上面的传感器。

```
// ESP32_WebServer.ino
#include <WiFi.h>
#include <WebServer.h>
#include <ArduinoJson.h>

// WiFi配置
const char* ssid = "Kabineto";
const char* password = "sxws0630s";

// 创建Web服务器对象，端口80
WebServer server(80);

// 引脚定义
const int ledPin = 27;       // LED
const int relayPin = 4;     // 继电器控制
const int analogPin = 34;   // 模拟输入
const int buttonPin = 35;   // 按钮输入

// 设备状态变量
bool ledState = false;
bool relayState = false;
int analogValue = 0;
bool buttonState = false;

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

  // 初始化GPIO
  pinMode(ledPin, OUTPUT);
  pinMode(relayPin, OUTPUT);
  pinMode(buttonPin, INPUT);
  digitalWrite(ledPin, LOW);
  digitalWrite(relayPin, LOW);

  // 连接WiFi
  connectToWiFi();

  // 设置API路由
  setupRoutes();

  // 启动服务器
  server.begin();
  Serial.println("HTTP服务器已启动");
  printNetworkInfo();
}

void loop() {
  server.handleClient();  // 处理客户端请求
  updateSensorData();     // 更新传感器数据
  delay(10);
}

void connectToWiFi() {
  Serial.println();
  Serial.print("连接WiFi: ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println();
  Serial.println("WiFi连接成功!");
}

String html =
"<html>\n"
"  <head>\n"
"    <title>ESP32控制服务器</title>\n"
"    <meta charset=\"UTF-8\">\n"
"    <style>\n"
"      body { font-family: Arial; margin: 40px; }\n"
"      .endpoint { background: #f5f5f5; padding: 15px; margin: 10px 0; border-radius: 5px; }\n"
"      button { padding: 10px 15px; margin: 5px; cursor: pointer; }\n"
"      .on { background: #4CAF50; color: white; }\n"
"      .off { background: #f44336; color: white; }\n"
"    </style>\n"
"  </head>\n"
"  <body>\n"
"    <h1>ESP32控制服务器</h1>\n"
"    <p><strong>设备IP:</strong> " + WiFi.localIP().toString() + "</p>\n"
"    \n"
"    <div class=\"endpoint\">\n"
"      <h3>📊 获取设备信息</h3>\n"
"      <p><strong>GET</strong> <code>/api/device/info</code></p>\n"
"      <button onclick=\"fetchData('/api/device/info')\">获取信息</button>\n"
"    </div>\n"
"    \n"
"    <div class=\"endpoint\">\n"
"      <h3>💡 LED控制</h3>\n"
"      <p><strong>GET</strong> <code>/api/led/on</code> | <code>/api/led/off</code> |
<code>/api/led/toggle</code></p>\n"
"      <button class=\"on\" onclick=\"fetchData('/api/led/on')\">打开LED</button>\n"
"      <button class=\"off\" onclick=\"fetchData('/api/led/off')\">关闭LED</button>\n"
"      <button onclick=\"fetchData('/api/led/toggle')\">切换LED</button>\n"
"    </div>\n"
"    \n"
"    <div class=\"endpoint\">\n"
"      <h3>🔌 继电器控制</h3>\n"
"      <p><strong>GET</strong> <code>/api/relay/on</code> |
<code>/api/relay/off</code></p>\n"
"      <button class=\"on\" onclick=\"fetchData('/api/relay/on')\">打开继电器</button>\n"
"      <button class=\"off\" onclick=\"fetchData('/api/relay/off')\">关闭继电器</button>\n"
"    </div>\n"
"    \n"
"    <div class=\"endpoint\">\n"
"      <h3>📈 传感器数据</h3>\n"
"      <p><strong>GET</strong> <code>/api/sensor/data</code></p>\n"
"      <button onclick=\"fetchData('/api/sensor/data')\">读取传感器</button>\n"
"    </div>\n"
"    \n"
"    <div id=\"result\" style=\"margin-top: 20px; padding: 15px; background: #e8f4fd;
border-radius: 5px;\"></div>\n"
"    \n"
"    <script>\n"
"      async function fetchData(url) {\n"
"        try {\n"
"          const response = await fetch(url);\n"
"          const data = await response.json();\n"
"          document.getElementById('result').innerHTML = '<pre>' + JSON.stringify(data, null, 2) +
'</pre>';\n"
"        } catch (error) {\n"
"          document.getElementById('result').innerHTML = '错误: ' + error;\n"
"        }\n"
"      }\n"
"    </script>\n"
"  </body>\n"
"</html>";

void setupRoutes() {
  // 根路径 - 显示API文档
  server.on("/", HTTP_GET, []() {
    server.send(200, "text/html", html);
  });

  // 获取设备信息
  server.on("/api/device/info", HTTP_GET, []() {
    DynamicJsonDocument doc(1024);
    doc["device"] = "ESP32";
    doc["ip"] = WiFi.localIP().toString();
    doc["mac"] = WiFi.macAddress();
    doc["free_heap"] = ESP.getFreeHeap();
    doc["chip_id"] = ESP.getEfuseMac();

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

  // LED控制
  server.on("/api/led/on", HTTP_GET, []() {
    digitalWrite(ledPin, HIGH);
    ledState = true;
    sendSuccessResponse("LED已打开");
  });

  server.on("/api/led/off", HTTP_GET, []() {
    digitalWrite(ledPin, LOW);
    ledState = false;
    sendSuccessResponse("LED已关闭");
  });
```

```
server.on("/api/led/toggle", HTTP_GET, []() {
  ledState = !ledState;
  digitalWrite(ledPin, ledState);
  sendSuccessResponse(ledState ? "LED已打开" : "LED已关闭");
});

// 继电器控制
server.on("/api/relay/on", HTTP_GET, []() {
  digitalWrite(relayPin, HIGH);
  relayState = true;
  sendSuccessResponse("继电器已打开");
});

server.on("/api/relay/off", HTTP_GET, []() {
  digitalWrite(relayPin, LOW);
  relayState = false;
  sendSuccessResponse("继电器已关闭");
});

// 传感器数据
server.on("/api/sensor/data", HTTP_GET, []() {
  DynamicJsonDocument doc(512);
  doc["analog_value"] = analogValue;
  doc["voltage"] = (analogValue * 3.3) / 4095.0;
  doc["button_pressed"] = buttonState;
  doc["led_state"] = ledState;
  doc["relay_state"] = relayState;

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

// 未找到的路由
server.onNotFound([]() {
  DynamicJsonDocument doc(256);
  doc["error"] = true;
  doc["message"] = "API端点不存在";

  String response;
  serializeJson(doc, response);
  server.send(404, "application/json", response);
});
```

```
}

void sendSuccessResponse(const String& message) {
  DynamicJsonDocument doc(256);
  doc["success"] = true;
  doc["message"] = message;
  doc["led_state"] = ledState;
  doc["relay_state"] = relayState;

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

void updateSensorData() {
  analogValue = analogRead(analogPin);
  buttonState = digitalRead(buttonPin);
}

void printNetworkInfo() {
  Serial.println("=== 网络信息 ===");
  Serial.print("IP地址: ");
  Serial.println(WiFi.localIP());
  Serial.print("MAC地址: ");
  Serial.println(WiFi.macAddress());
  Serial.print("信号强度: ");
  Serial.print(WiFi.RSSI());
  Serial.println(" dBm");
  Serial.println("================");
}
```

2. PC 的 PYTHON 客户端程序

以下是在 ESP32 上运行的 Web 程序，可以使用 Web 总结控制，也可以使用 Python 通过 post/get访问。

```
# esp32_client.py
import requests
import json
import time
from typing import Dict, Any, Optional
```

```
class ESP32Controller:
    """
    ESP32 Web服务器客户端控制类
    """

    def __init__(self, base_url: str, timeout: int = 5):
        """
        初始化ESP32控制器

        Args:
            base_url: ESP32的IP地址，例如 "http://192.168.0.104"
            timeout: 请求超时时间（秒）
        """
        if not base_url.startswith(('http://', 'https://')):
            base_url = 'http://' + base_url
        self.base_url = base_url.rstrip('/')
        self.timeout = timeout
        self.session = requests.Session()

    def _send_request(self, endpoint: str) -> Dict[str, Any]:
        """发送HTTP请求到ESP32"""
        try:
            url = f"{self.base_url}{endpoint}"
            response = self.session.get(url, timeout=self.timeout)
            response.raise_for_status()
            return response.json()
        except requests.exceptions.RequestException as e:
            return {"error": True, "message": f"请求失败: {e}"}
        except json.JSONDecodeError as e:
            return {"error": True, "message": f"JSON解析失败: {e}"}

    def get_device_info(self) -> Dict[str, Any]:
        """获取设备信息"""
        return self._send_request("/api/device/info")

    def led_control(self, action: str) -> Dict[str, Any]:
        """
        控制LED

        Args:
            action: 'on', 'off', 'toggle'
        """
        valid_actions = ['on', 'off', 'toggle']
        if action not in valid_actions:
            return {"error": True, "message": f"无效的操作，请使用: {valid_actions}"}
        return self._send_request(f"/api/led/{action}")

    def relay_control(self, action: str) -> Dict[str, Any]:
        """
        控制继电器

        Args:
            action: 'on', 'off'
        """
        valid_actions = ['on', 'off']
        if action not in valid_actions:
            return {"error": True, "message": f"无效的操作，请使用: {valid_actions}"}
        return self._send_request(f"/api/relay/{action}")

    def get_sensor_data(self) -> Dict[str, Any]:
        """获取传感器数据"""
        return self._send_request("/api/sensor/data")

    def get_status(self) -> Dict[str, Any]:
        """获取完整状态（设备信息 + 传感器数据）"""
        device_info = self.get_device_info()
        sensor_data = self.get_sensor_data()

        if "error" in device_info or "error" in sensor_data:
            return {"error": True, "message": "获取状态失败"}

        return {
            "device_info": device_info,
            "sensor_data": sensor_data
        }

def print_response(response: Dict[str, Any], title: str = ""):
    """美化打印响应结果"""
    if title:
        print(f"\n{'='*50}")
        print(f"📋 {title}")
        print(f"{'='*50}")

    if "error" in response and response["error"]:
        print(f"❌ 错误: {response.get('message', '未知错误')}")
    else:
        print(json.dumps(response, indent=2, ensure_ascii=False))
```

```
def demo_automated_test(controller: ESP32Controller):
    """自动化演示测试"""
    print("🚀 开始自动化演示测试")

    # 1. 获取设备信息
    print_response(controller.get_device_info(), "设备信息")
    time.sleep(1)

    # 2. LED控制演示
    print_response(controller.led_control("on"), "打开LED")
    time.sleep(1)

    print_response(controller.led_control("off"), "关闭LED")
    time.sleep(1)

    print_response(controller.led_control("toggle"), "切换LED")
    time.sleep(1)

    # 3. 继电器控制演示
    print_response(controller.relay_control("on"), "打开继电器")
    time.sleep(1)

    print_response(controller.relay_control("off"), "关闭继电器")
    time.sleep(1)

    # 4. 传感器数据读取
    print_response(controller.get_sensor_data(), "传感器数据")
    time.sleep(1)

    # 5. 最终状态
    print_response(controller.get_status(), "完整状态")

    print("\n✅ 自动化测试完成!")

def interactive_control(controller: ESP32Controller):
    """交互式控制界面"""
    commands = {
        "1": {"name": "📊 获取设备信息", "func": controller.get_device_info},
        "2": {"name": "💡 LED控制", "submenu": {
            "1": {"name": "打开LED", "func": lambda: controller.led_control("on")},
            "2": {"name": "关闭LED", "func": lambda: controller.led_control("off")},
            "3": {"name": "切换LED", "func": lambda: controller.led_control("toggle")}
        }},
        "3": {"name": "🔌 继电器控制", "submenu": {
```

```
        "1": {"name": "打开继电器", "func": lambda: controller.relay_control("on")},
        "2": {"name": "关闭继电器", "func": lambda: controller.relay_control("off")}
    }},
    "4": {"name": "📈 读取传感器数据", "func": controller.get_sensor_data},
    "5": {"name": "🔄 获取完整状态", "func": controller.get_status},
    "6": {"name": "🎬 自动化演示测试", "func": lambda: demo_automated_test(controller)},
    "0": {"name": "❌ 退出程序", "func": None}
}

while True:
    print("\n" + "="*60)
    print("🎛️ ESP32 智能控制器")
    print("="*60)

    # 显示主菜单
    for key, value in commands.items():
        print(f"{key}. {value['name']}")

    try:
        choice = input("\n请输入选项编号: ").strip()

        if choice == "0":
            print("👋 再见！")
            break
        elif choice in commands:
            if "submenu" in commands[choice]:
                # 显示子菜单
                submenu = commands[choice]["submenu"]
                print(f"\n--- {commands[choice]['name']} ---")
                for sub_key, sub_value in submenu.items():
                    print(f"  {sub_key}. {sub_value['name']}")

                sub_choice = input("请选择操作: ").strip()
                if sub_choice in submenu:
                    result = submenu[sub_choice]["func"]()
                    print_response(result, submenu[sub_choice]["name"])
                else:
                    print("❌ 无效的选择")
            else:
                # 执行直接功能
                result = commands[choice]["func"]()
                if choice != "6":  # 自动化测试自己会打印结果
                    print_response(result, commands[choice]["name"])
        else:
                print("❌ 无效的选择，请重新输入")

        except KeyboardInterrupt:
            print("\n\n👋 程序被用户中断，再见！")
            break
        except Exception as e:
            print(f"❌ 发生错误: {e}")

def monitor_sensor_data(controller: ESP32Controller, interval: int = 2):
    """实时监控传感器数据"""
    print(f"\n📊 开始实时监控传感器数据 (每{interval}秒更新)")
    print("按 Ctrl+C 停止监控")

    try:
        while True:
            sensor_data = controller.get_sensor_data()
            if "error" not in sensor_data:
                print(f"\r🕒 {time.strftime('%H:%M:%S')} - " +
                      f"模拟值: {sensor_data.get('analog_value', 'N/A'):4d} | " +
                      f"电压: {sensor_data.get('voltage', 0):.2f}V | " +
                      f"按钮: {'按下' if sensor_data.get('button_pressed') else '释放':3s} | " +
                      f"LED: {'开启' if sensor_data.get('led_state') else '关闭':3s} | " +
                      f"继电器: {'开启' if sensor_data.get('relay_state') else '关闭':3s}",
                      end="", flush=True)
            else:
                print(f"\r❌ 读取失败: {sensor_data.get('message', '未知错误')}", end="")

            time.sleep(interval)

    except KeyboardInterrupt:
        print("\n\n🛑 监控已停止")

def main():
    """主函数"""
    print("🌐 ESP32 Web客户端控制器")
    print("="*50)

    # 配置ESP32的IP地址
    esp32_ip = input("请输入ESP32的IP地址 (例如: 192.168.1.100): ").strip()

    if not esp32_ip:
        print("❌ IP地址不能为空")
        return
    # 创建控制器实例
    controller = ESP32Controller(esp32_ip)

    # 测试连接
    print("\n🔗 测试连接中...")
    try:
        info = controller.get_device_info()
        if "error" not in info:
            print("✅ 连接成功!")
            print(f"📱 设备: {info.get('device', 'Unknown')}")
            print(f"🌐 IP: {info.get('ip', 'Unknown')}")
            print(f"📶 信号强度: {info.get('rssi', 'N/A')} dBm")

            # 进入交互式控制
            interactive_control(controller)
        else:
            print("❌ 连接失败，请检查:")
            print("   - IP地址是否正确")
            print("   - ESP32和电脑是否在同一网络")
            print("   - ESP32服务器是否正在运行")

    except Exception as e:
        print(f"❌ 连接测试失败: {e}")

if __name__ == "__main__":
    # 安装依赖: pip install requests

    try:
        main()
    except KeyboardInterrupt:
        print("\n\n👋 程序退出")
    except Exception as e:
        print(f"💥 程序异常: {e}")
```