超智能小风扇设计报告

一、概要：

该小风扇采用乐鑫公司生产的 ESP32 作为主控芯片，基于 arduino 平台开发，采用 FreeRTOS 实时操作系统作为底层开发框架，利用嘉立创 EDA 进行原理图、PCB 设计。具备多功能的一款智能小风扇。

二、设计方案：

软件的框架是采用 FreeRTOS(Free Real Time Operating System)进行开发，FreeRTOS 有助于提高系统性能和管理模块的资源。FreeRTOS 允许用户处理多项任务，如处理传感器返回值，处理网络交互请求，控制电机速度等，所有这些任务都可以同时独立运行。

1. 软件部分框架如下图：

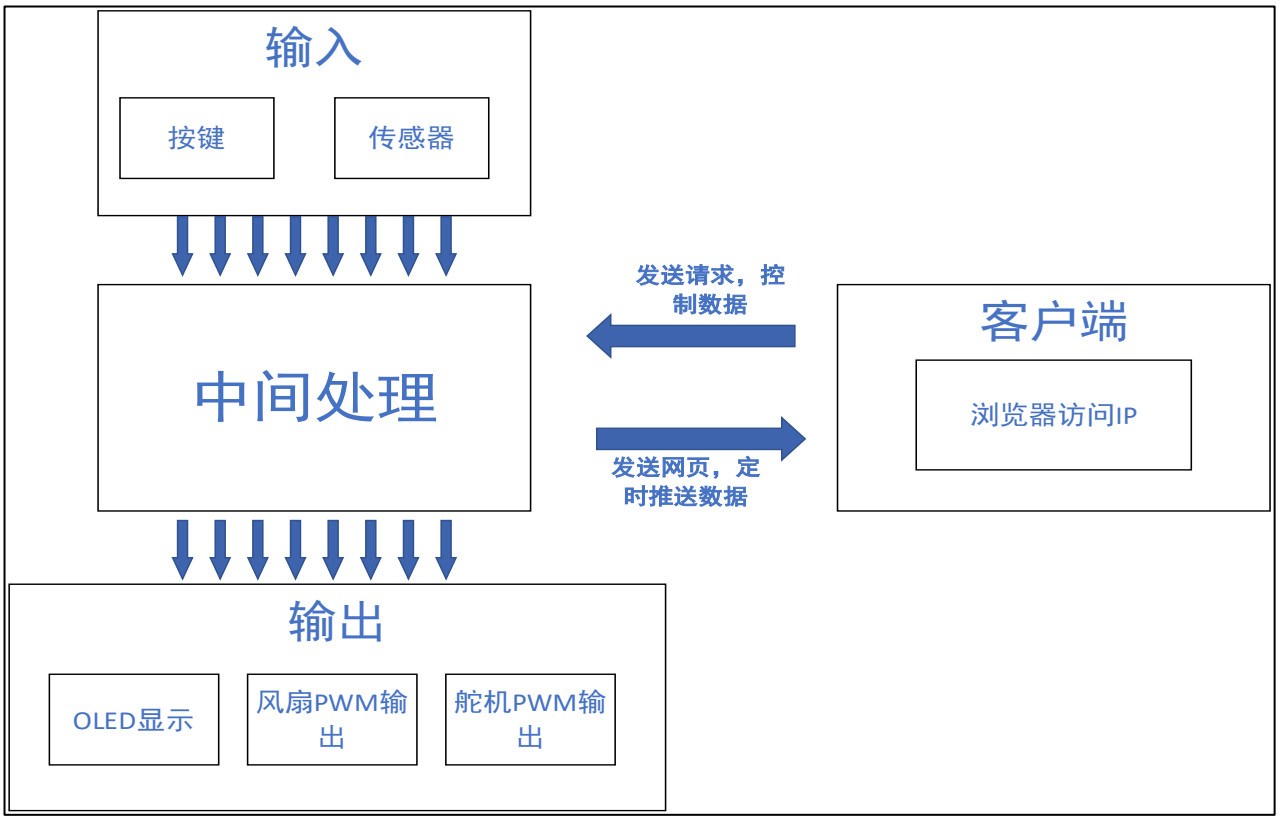

图 1 程序输入输出概览

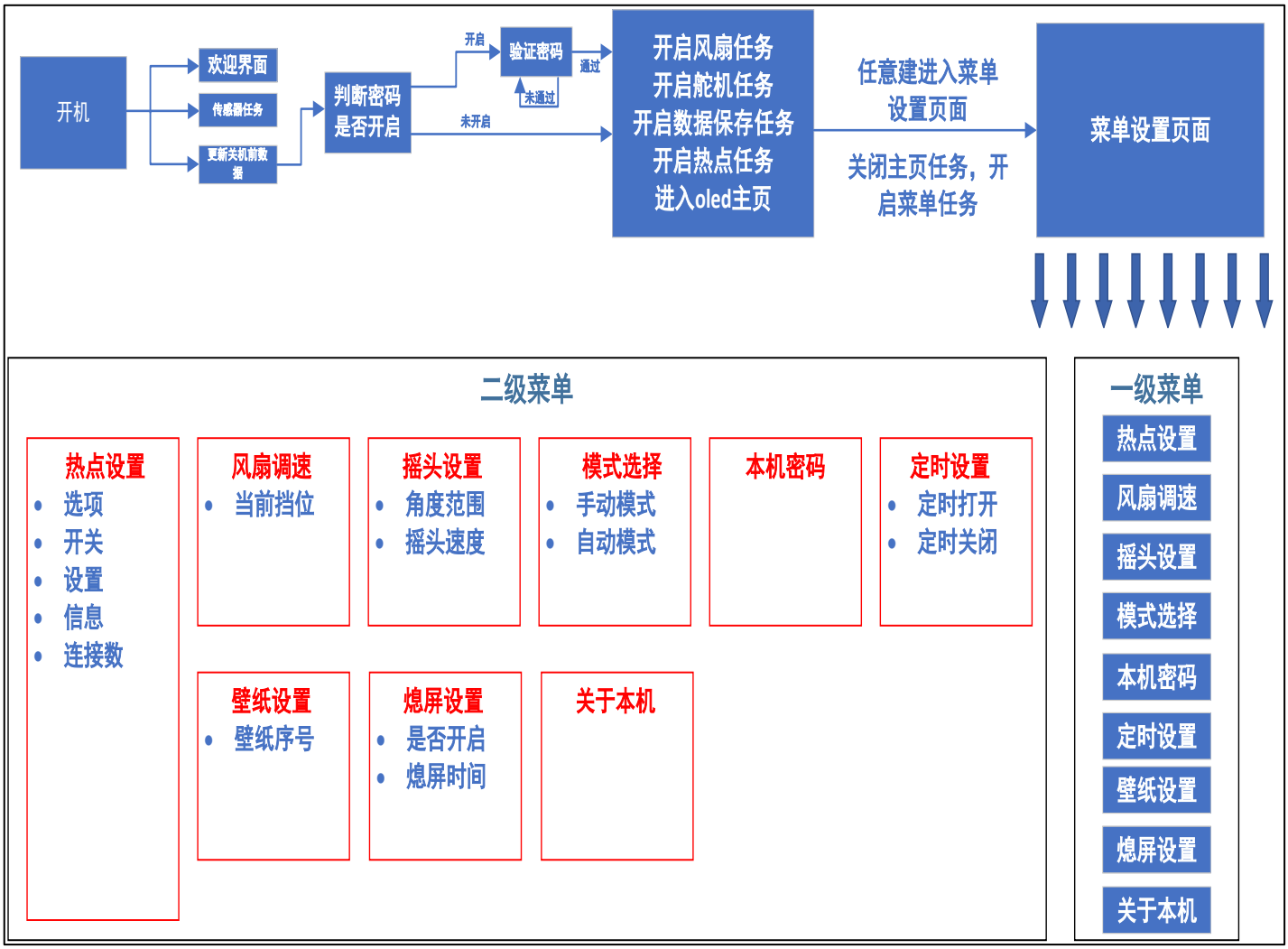

图 1 程序流程图

2.硬件设计方案如下

硬件的最终方案采用模块为主，其中通过现在流行接口 type-c 进行充电，通过开发板上自带的 micro-B 进行烧录，同时内置可充电锂电池和电源管理芯片 ETA6093，方便携带。同时，通过 LM2596S 模块进行稳压降压 5V 供电，拥有四个操作按键，分别为上、下、确定、返回。拥有一块 0.96 英寸的 OLED 显示屏，用来更加直观展现信息。此外，还有一个 DHT11 温度传感器，用来获取环境温度和湿度。最后，还有充电指示灯、放电指示灯、电源指示灯。

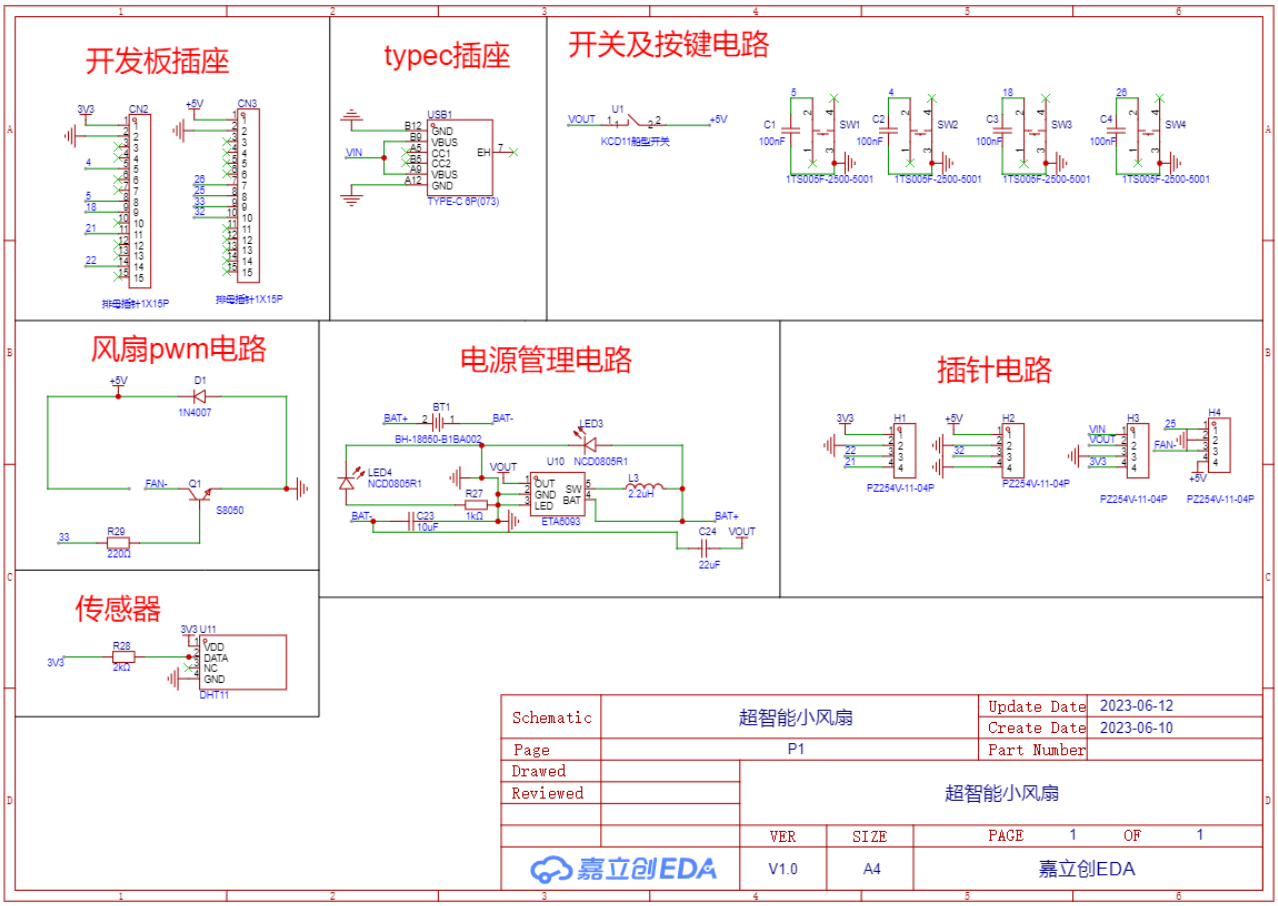

图 3 硬件原理图

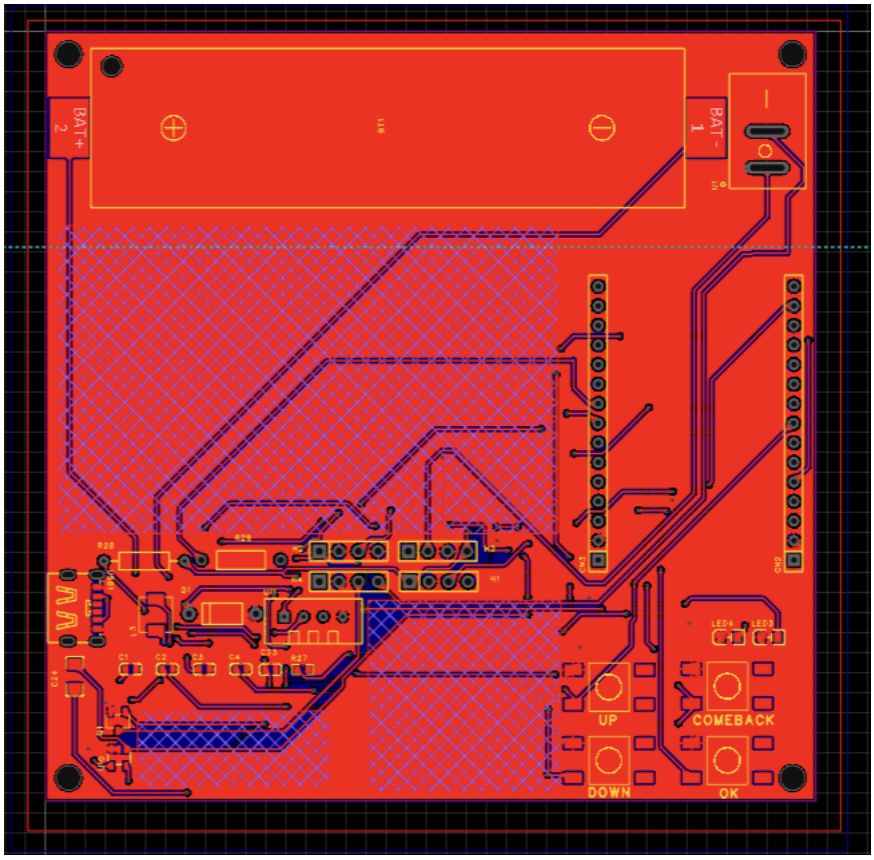

图 4.1 硬件 PCB 图（已覆铜，正面）

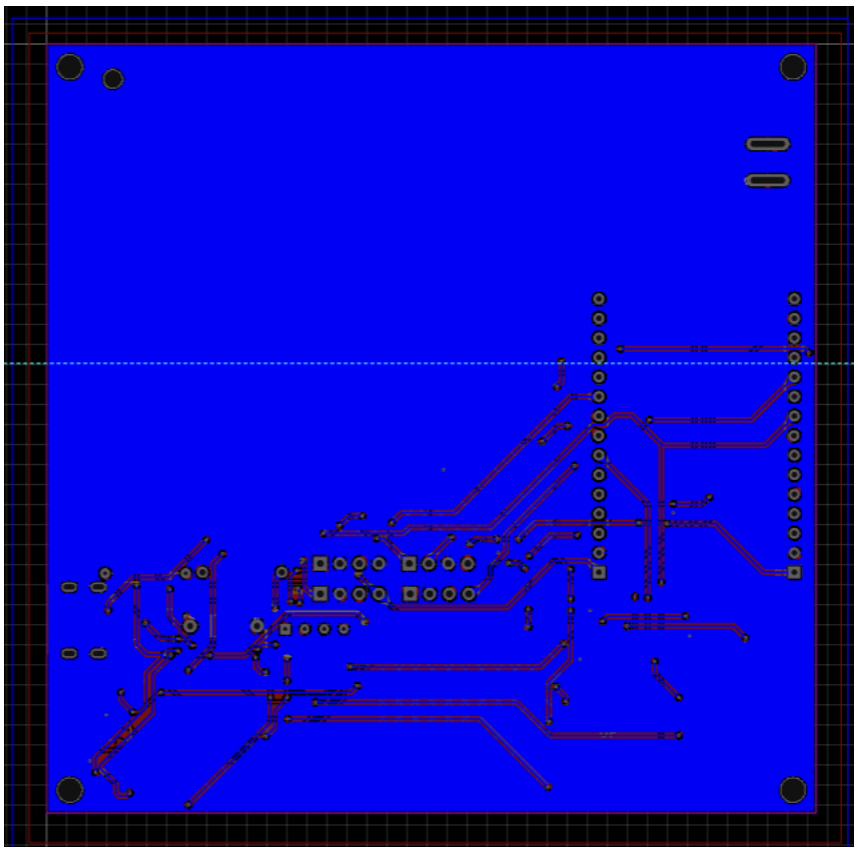

图 4.2 硬件 PCB 图
（已覆铜，反面）

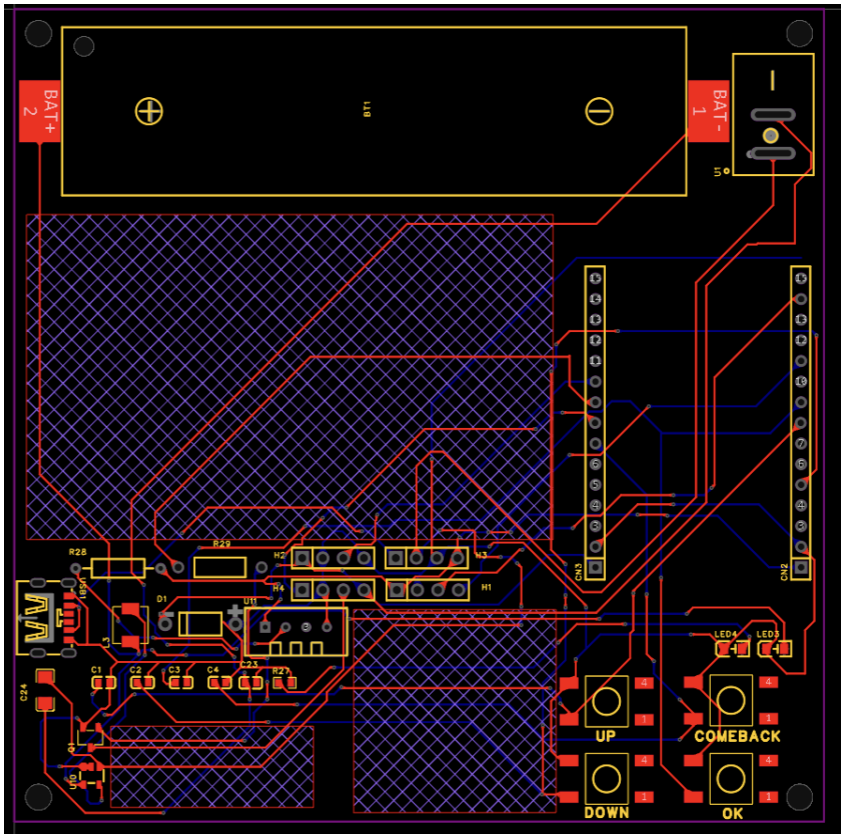

图 4.3 硬件 PCB 图
（未覆铜，正面）

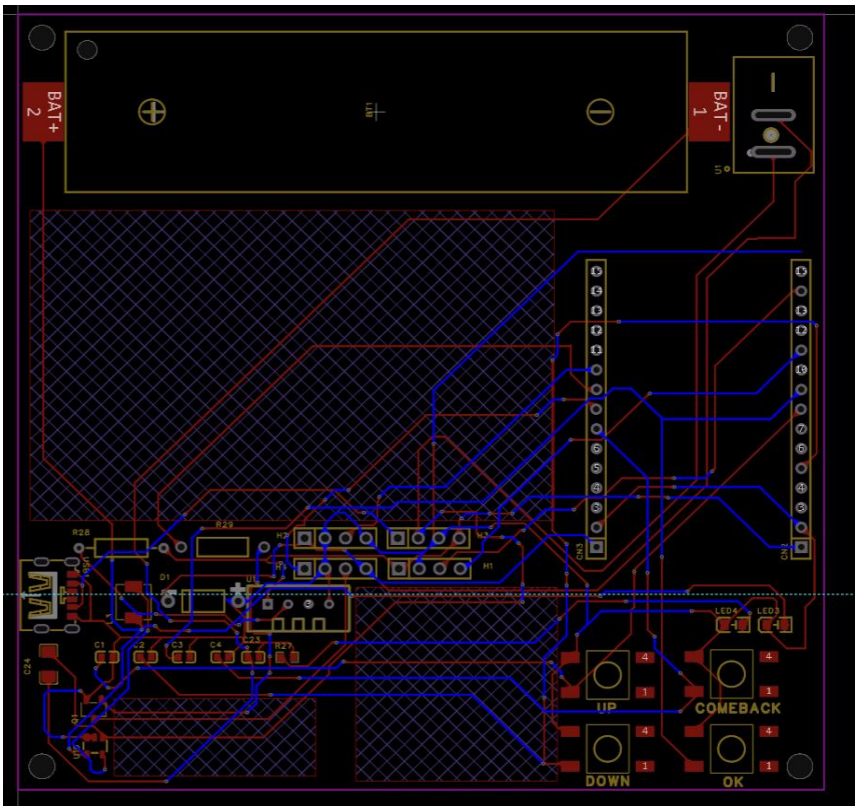

图 4.4 硬件 PCB 图
（未覆铜，反面）

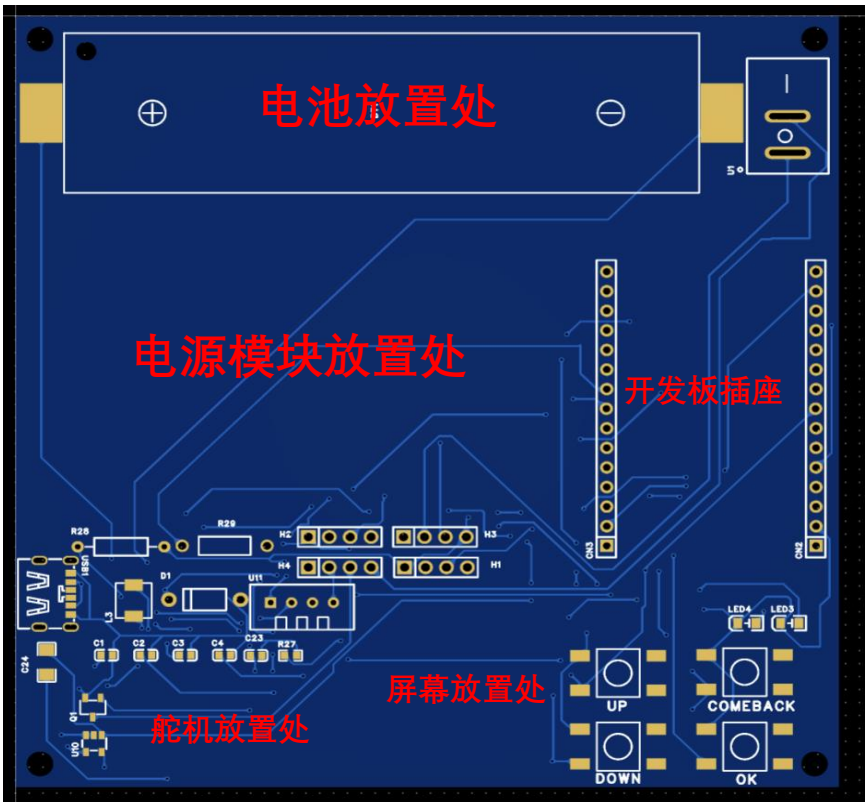

图 4.5 电路板 2D 预览图

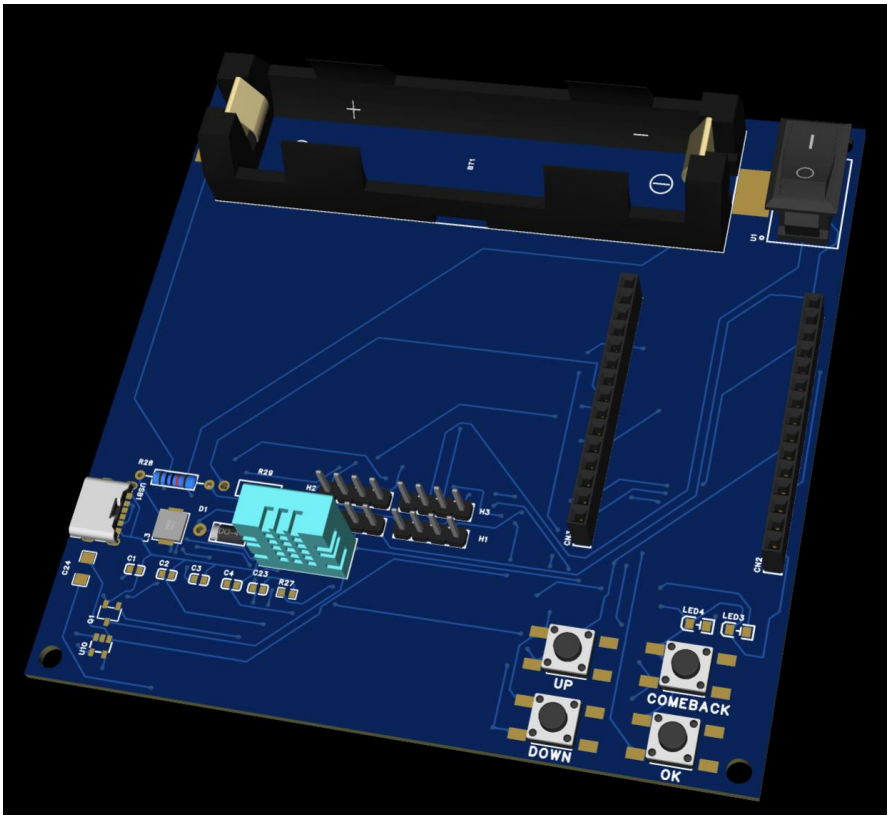

图 4.5 电路板 3D 预览图

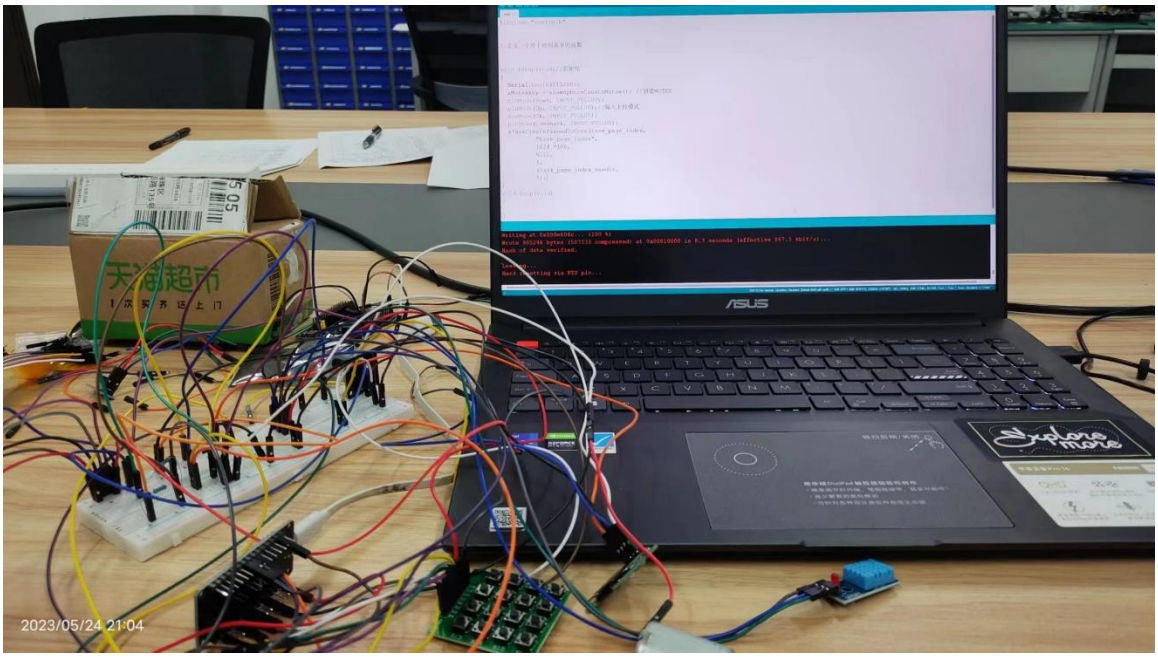

图 4.6 开发板实拍

三、测试调试

1. 电源测试

由于直流稳压电源是由交流电源经整流稳压等环节而形成的，因而不可避免地在直流稳定量中带有交流成分，也就是纹波，电源的纹波大小会影响芯片性能，甚至会影响电路的工作，因此对电源电压大小测量以及纹波的测量十分重要。

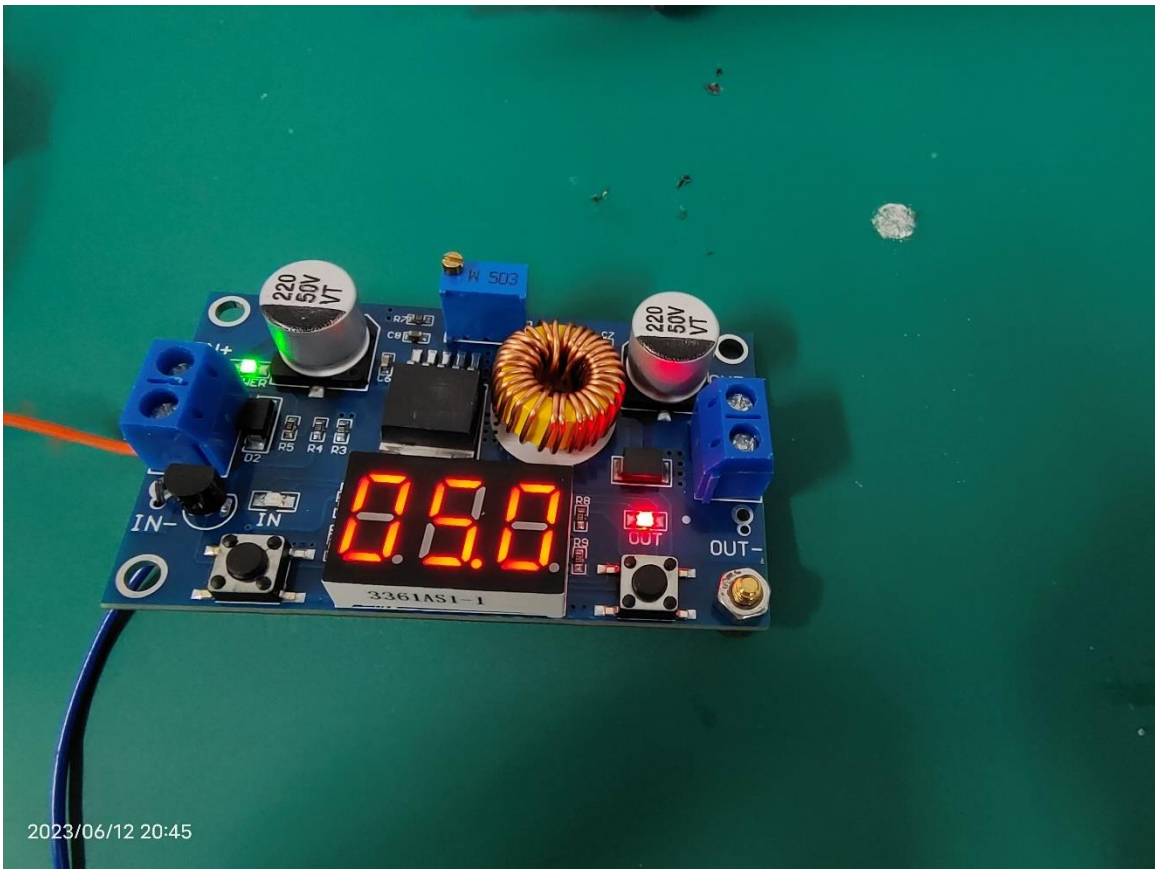

图 5 电源模板显示的电压输出

图 6 万用表测量得到的实际电压

图 7 电源纹波的测量

（上方的是输入电源信号，下方的是输出电源信号，因电源输入信号来源于

面包板电源模块，故纹波不明显，但对比后输出信号基本无纹波，可见 LM2596S 模块滤波稳压作用明显。

2. ESP32 脉宽调制信号（PWM）的测量。

LEDC 通道共有两组，分别为 8 路高速通道和 8 路低速通道。高速通道模式在硬件中实现，可以自动且无干扰地改变 PWM 占空比。低速通道模式下，PWM 占空比需要由软件中的驱动器改变。每组通道都可以使用不同的时钟源。

本项目中相关代码如下：

```
/*定义一个风扇任务，根据速度调整输出脉宽*/
void task_fan(void *pt)
{
  float count = pow(2, 12);
  ledcSetup(pwm_fan, 1000, 12);
  ledcAttachPin(Fan, pwm_fan);
  for (;;)
  {
    if (fan_mode == 0)
    {
      if (fan_speed == 0)
      {
        fan_duty = 0;
      }
      if (fan_speed == 1)
      {
        fan_duty = (int)(0.4 * count);
      }
      else if (fan_speed == 2)
      {
        fan_duty = (int)(0.6 * count);
      }
      else if (fan_speed == 3)
      {
        fan_duty = (int)(0.8 * count);
      }
      else if (fan_speed == 4)
      {
        fan_duty = (int)count;
```

```
30.       }
31.     }
32.     else if (fan_mode == 1)
33.     {
34.       fan_duty = map(temperature, fan_min, fan_max, (int)(0.4 * count), (int)c
   ount);
35.     }
36.     ledcWrite(pwm_fan, fan_duty);
37.     vTaskDelay(200);
38.   }
39.}
40.
41./*定义一个舵机任务，根据角度和速度来调整控制脉宽输出*/
42.void task_turn(void *pd )
43.{
44.   float count = pow(2, 14);
45.   int direction = 0;
46.   ledcSetup(pwm_turn, 50, 14);
47.   ledcAttachPin(Turn, pwm_turn);
48.   for (;;)
49.   {
50.     if (turn_angle == 0)
51.     {
52.       turn_min = 0;
53.       turn_max = 0;
54.     }
55.     else if (turn_angle == 4)
56.     {
57.       turn_min = (int)(0.5 / (20 / count));
58.       turn_max = (int)(2.5 / (20 / count));
59.     }
60.     else if (turn_angle == 3)
61.     {
62.       turn_min = (int)(2 / 3 / (20 / count));
63.       turn_max = (int)(7 / 3 / (20 / count));
64.     }
65.     else if (turn_angle == 2)
66.     {
67.       turn_min = (int)(5 / 6 / (20 / count));
68.       turn_max = (int)(13 / 6 / (20 / count));
69.     }
70.     else if (turn_angle == 1)
71.     {
72.       turn_min = (int)(1 / (20 / count));
```

```
73.       turn_max = (int)(2 / (20 / count));
74.     }
75.     if (turn_speed == 0)
76.     {
77.       turn_step = 0;
78.     }
79.     else if (turn_speed == 1)
80.     {
81.       turn_step = 2;
82.     }
83.     else if (turn_speed == 2)
84.     {
85.       turn_step = 10;
86.     }
87.     else if (turn_speed == 3)
88.     {
89.       turn_step = 15;
90.     }
91.     else if (turn_speed == 4)
92.     {
93.       turn_step = 20;
94.     }
95.
96.     if (direction == 0)
97.     {
98.       turn_duty += turn_step;
99.     }
100.     else if (direction == 1)
101.     {
102.       turn_duty -= turn_step;
103.     }
104.     if (turn_duty >= turn_max)
105.     {
106.       turn_duty = turn_max;
107.       direction = 1;
108.     }
109.     else if (turn_duty <= turn_min)
110.     {
111.       turn_duty = turn_min;
112.       direction = 0;
113.     }
114.     ledcWrite(pwm_turn, turn_duty);
115.     vTaskDelay(20);
116.   }
```

```
117.}
```

下面是对风扇电机脉宽的测量：

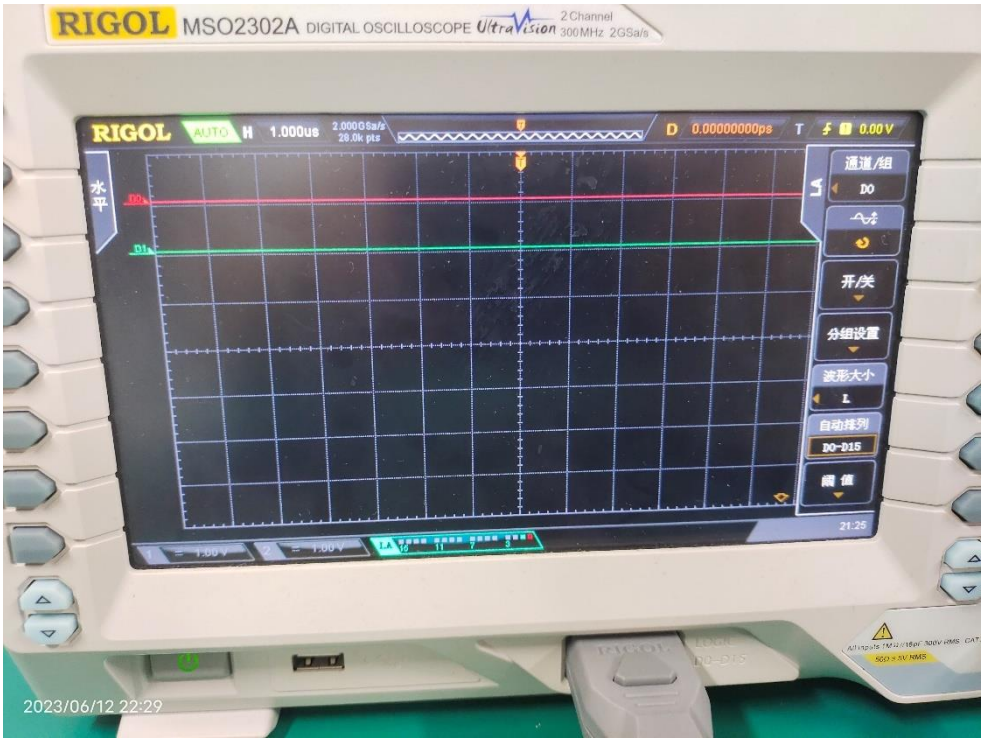

图 8.1 零档 PWM 信号

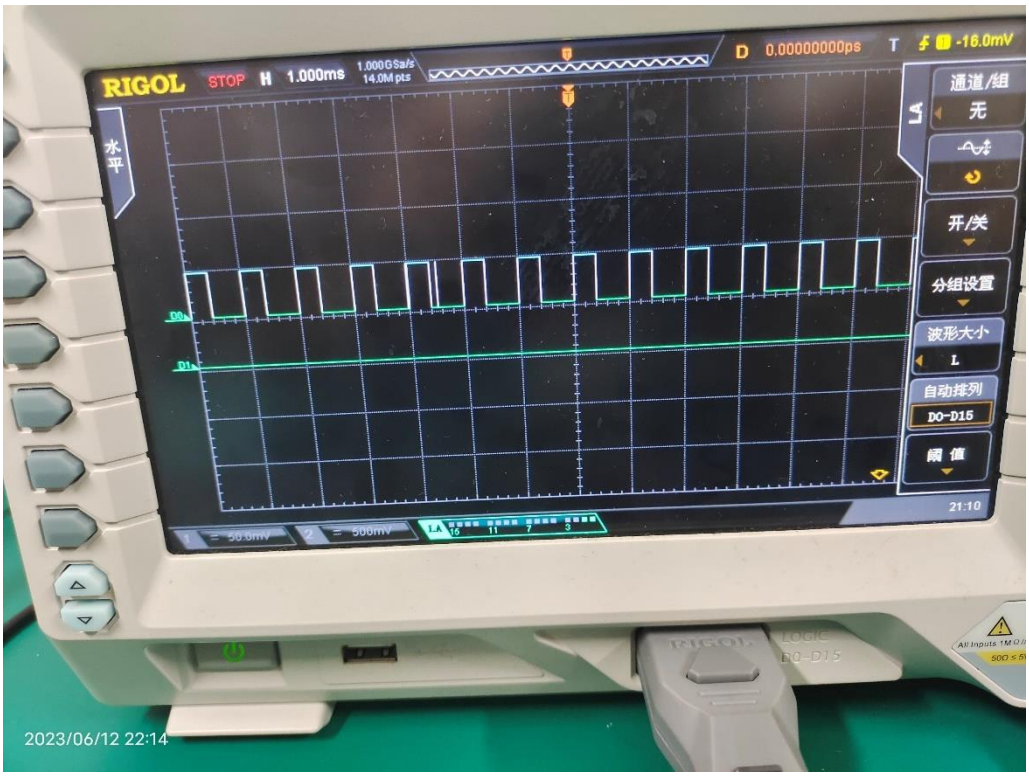

图 8.2 一档 PWM 信号

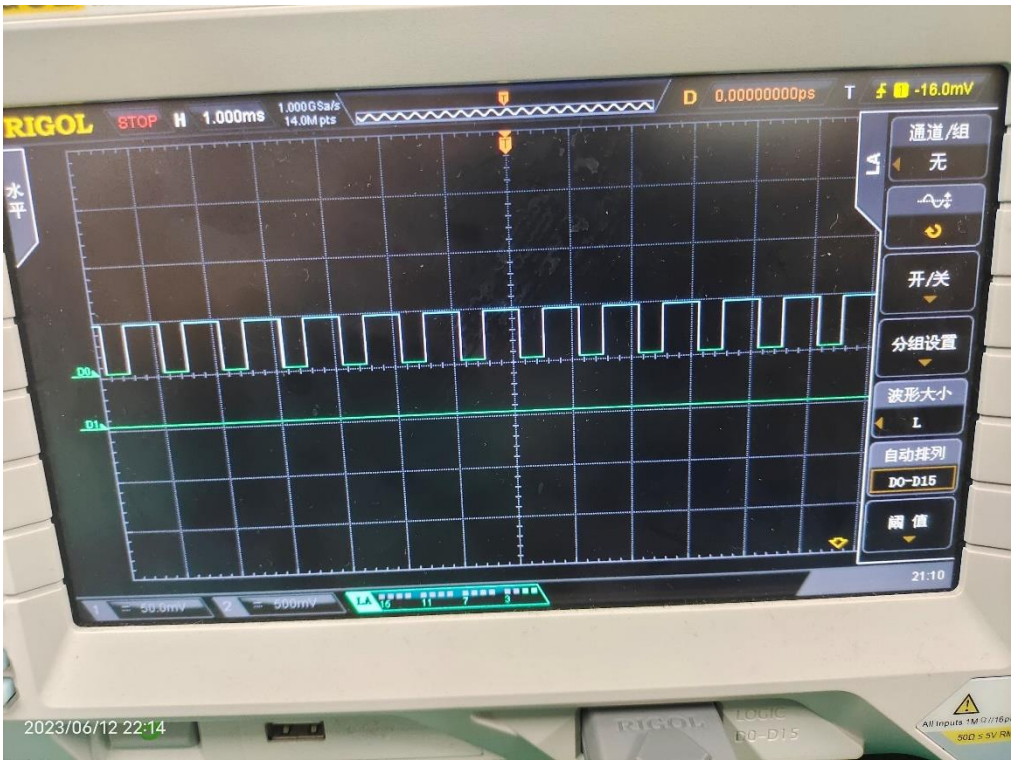

图 8.3 二档 PWM 信号

图 8.4 三档 PWM 信号

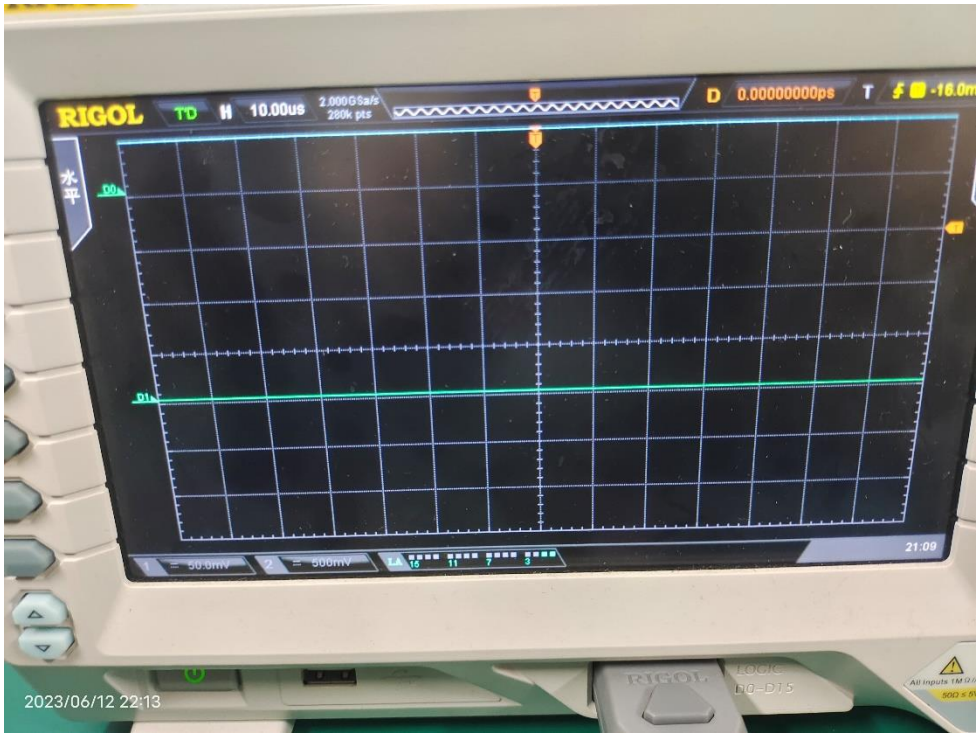

图 8.5 四档 PWM 信号

3. ESP32 网络服务及远程操控测试。

本项目采用 arduino 的 ESPAsyncWebServer 和 WiFi 库使用 ESP32 建立 AP，手机连入 AP 后通过浏览器访问 ip 地址即可访问储存在 ESP32 的网页，并进行数据交互和远程遥控。

项目相关代码如下

```
AsyncWebServer server(80);
void task_wifi(void *pt)
{
  WiFi.mode(WIFI_AP); //设置工作在 AP 模式
  WiFi.softAP(ssid, password);
  server.on("/", HTTP_GET, handleRootPage);
  server.on("/data", HTTP_GET, handleDataRequest);
  server.on("/command", HTTP_GET, handleCommandRequest);
  // 开启服务器
  server.begin();

```

```
12.   for (;;)
13.   {
14.     if (fan_mode == 0)
15.     {
16.       fanMode = "手动模式";
17.       fanSpeed = String(fan_speed) + "档";
18.     }
19.     else if (fan_mode == 1)
20.     {
21.       fanMode = "智能模式";
22.       fanSpeed = String(100 * fan_duty / pow(2, 12)) + "%";
23.     }
24.     if (turn_angle == 0)
25.     {
26.       turnAngle = "未开启";
27.     }
28.     else if (turn_angle == 1)
29.     {
30.       turnAngle = "90℃";
31.     }
32.     else if (turn_angle == 2)
33.     {
34.       turnAngle = "120℃";
35.     }
36.     else if (turn_angle == 3)
37.     {
38.       turnAngle = "150℃";
39.     }
40.     else if (turn_angle == 4)
41.     {
42.       turnAngle = "180℃";
43.     }
44.     vTaskDelay(200);
45.   }
46.}
47.
48.void handleRootPage(AsyncWebServerRequest *request) {
49.   // 构建网页内容
50.   String htmlContent = "<html><head>";
51.   htmlContent += "<title>超智能小风扇</title>";
52.   htmlContent += "<style>";
53.   htmlContent += "body { background-color: lightblue; font-
    family: Arial, sans-serif; margin: 0; padding: 20px; }";
54.   htmlContent += "h1 { text-align: center; }";
```

```
55.  htmlContent += ".data-container { display: flex; justify-content: space-between; align-items: center; margin-bottom: 20px; }";
56.  htmlContent += ".data-label { font-weight: bold; }";
57.  htmlContent += ".data-value { font-size: 24px; }";
58.  htmlContent += ".btn-container { display: flex; justify-content: space-between; margin-top: 10px; }";
59.  htmlContent += ".btn { padding: 10px; font-size: 16px; }";
60.  htmlContent += "</style>";
61.  htmlContent += "<meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0\">";
62.  htmlContent += "<script>";
63.  htmlContent += "function sendCommand(command) { var xhr = new XMLHttpRequest(); xhr.open('GET', '/command?cmd=' + command, true); xhr.send(); }";
64.  htmlContent += "function updateData() { var xhr = new XMLHttpRequest(); xhr.onreadystatechange = function() { if (xhr.readyState === 4 && xhr.status === 200) { var data = JSON.parse(xhr.responseText); document.getElementById('temperature').innerText = data.temperature; document.getElementById('humidity').innerText = data.humidity; document.getElementById('fanMode').innerText = data.fanMode;document.getElementById('fan-speed').innerText = data.fanSpeed; document.getElementById('servo-level').innerText = data.turn_speed; document.getElementById('servo-angle').innerText = data.turnAngle; } }; xhr.open('GET', '/data', true); xhr.send(); }";
65.  htmlContent += "setInterval(updateData, 2000);";
66.  htmlContent += "</script>";
67.  htmlContent += "</head><body>";
68.  htmlContent += "<h1>超智能小风扇</h1>";
69.  htmlContent += "<div class='data-container'><span class='data-label'>温度:</span><span class='data-value' id='temperature'>--</span></div>";
70.  htmlContent += "<div class='data-container'><span class='data-label'>湿度:</span><span class='data-value' id='humidity'>--</span></div>";
71.  htmlContent += "<div class='data-container'><span class='data-label'>模式:</span><span class='data-value' id='fanMode'>--</span><div class='btn-container'><button class='btn' onclick=\"sendCommand('change_mode')\">切换</button></div></div>";
72.  htmlContent += "<div class='data-container'><span class='data-label'>风扇档位:</span><span class='data-value' id='fan-speed'>--</span><div class='btn-container'><button class='btn' onclick=\"sendCommand('increase_fan_speed')\">+</button><button class='btn' onclick=\"sendCommand('decrease_fan_speed')\">-</button></div></div>";
73.  htmlContent += "<div class='data-container'><span class='data-label'>舵机档位:</span><span class='data-value' id='servo-level'>--</span><div class='btn-container'><button class='btn' onclick=\"sendCommand('increase_servo_level')\"
```

```
    >+</button><button class='btn' onclick=\"sendCommand('decrease_servo_level')\"
    >-</button></div></div>";
  htmlContent += "<div class='data-container'><span class='data-label'>舵机角
    度:</span><span class='data-value' id='servo-angle'>--</span><div class='btn-
    container'><button class='btn' onclick=\"sendCommand('increase_servo_angle')\"
    >+</button><button class='btn' onclick=\"sendCommand('decrease_servo_angle')\"
    >-</button></div></div>";
  htmlContent += "</body></html>";

  // 发送网页内容
  request->send(200, "text/html", htmlContent);
}

// 处理数据请求的函数
void handleDataRequest(AsyncWebServerRequest *request) {
  // 构建 JSON 格式的数据
  String data = "{";
  data += "\"temperature\": \"" + String(temperature) + " °C\",";
  data += "\"fanMode\": \"" + fanMode + "\",";
  data += "\"humidity\": \"" + String(humidity) + "%\",";
  data += "\"fanSpeed\": \"" + fanSpeed + "\",";
  data += "\"turn_speed\": \"" + String(turn_speed) + "档\",";
  data += "\"turnAngle\": \"" + turnAngle + "°\"";
  data += "}";

  // 发送数据
  request->send(200, "application/json", data);
}

// 处理命令请求的函数
void handleCommandRequest(AsyncWebServerRequest *request) {
  if (request->hasParam("cmd")) {
    // 获取命令参数值
    String command = request->getParam("cmd")->value();

    // 根据命令执行相应操作
    if (command == "increase_fan_speed" && fan_speed <= 4)
    {
      fan_speed += 1;
    }
    else if (command == "decrease_fan_speed" && fan_speed >= 0)
    {
      fan_speed -= 1;
    }
```

```
    else if (command == "increase_servo_level" && turn_speed <= 4)
    {
      turn_speed += 1;
    }
    else if (command == "decrease_servo_level" && turn_speed >= 0)
    {
      turn_speed -= 1;
    }
    else if (command == "increase_servo_angle" && turn_angle <= 3)
    {
      turn_angle += 1;
    }
    else if (command == "decrease_servo_angle" && turn_angle >= 0)
    {
      turn_angle -= 1;
    }
    else if (command == "change_mode" && fan_mode == 0)
    {
      fan_mode = 1;
    }
    else if (command == "change_mode" && fan_mode == 1)
    {
      fan_mode = 0;
    }

    // 发送成功响应
    request->send(200, "text/plain", "Command received: " + command);
  }
}
```

界面如下：

图 9 远程控制界面

经检验，网页能够更新传感器抓取的温湿度数据，并且能够通过按钮向主控芯片发送命令，实现遥控功能。

四、最终实现的功能

- 支持 5 档 PWM 调速
- 支持自定义摇头角度范围，摇头速度
- 支持开启开机密码，并支持自定义开机密码
- 支持开启热点进行远程操控，网页端支持查看实时温度和湿度，支持摇头，调速，风扇模式等基本设置
- 支持自定义热点密码，暂不支持热点名称更改
- 支持断电记忆，重启后可以自动恢复关闭前的状态
- 支持智能模式和手动模式之间切换，智能模式可根据温度自动调整风速
- 支持定时开启和定时关闭操作
- 支持主页更换，内置 10 多种不同风格主页，可自定义更换
- 支持设置熄屏时间，支持一键熄屏操作
- 内置电源管理芯片，可通过 type-c 充电，也可以通过内置锂电池供电
- 支持 12v-24v 电压输入

五、元件来源

电路板是通过嘉立创 EDA 设计制造，嘉立创 PCB 下单打样，电源模块取自于实验室，ESP32 开发板、0.96 英寸 OLED、按键、发光二极管、锂电池在淘宝采购，贴片电阻和电容、电池盒、DHT11 传感器在立创商城采购。直插元件来源于实验室。

六、参考文献

1. 《arduino 程序设计基础》;
2. 《ESP-WOROOM-32 技术参考规格书》;
3. 《ESP32 技术参考手册》
4. 开发板开源网址;
5. 开源项目 esp8266 迷你像素灯;
6. CSDN 文章 ESP32 网页控制显示数据原来如此简单