吉林农业大学
JILIN AGRICULTURAL UNIVERSITY

物联网控制技术实验报告

Mamdani 模糊控制下的 ESP32C6 温控风扇

学　　院：信 息 技 术 学 院
班　　级：物 联 网 工 程 二 班
姓　　名：张 景
学　　号：2212100416
指导教师：王 光 耀

2025 年 6 月 6 日

Mamdani 模糊控制下的 ESP32C6 温控风扇

1 实验目的

本次物联网控制实验通过 ESP32 作为主控芯片，实现一个基于模糊控制算法 Mamdani 的温湿度控制系统。实验的具体目标如下：

1. 掌握 ESP32 开发环境的搭建及 FreeRTOS 多任务编程；
2. 学习使用 DHT11 温湿度传感器进行数据采集；
3. 熟悉 OLED 屏幕的初始化与数据显示；
4. 实现模糊控制算法，并根据实际温度调节输出控制信号（如电机转速）；
5. 综合应用硬件驱动、传感器采集、控制算法和人机交互，提升嵌入式系统开发能力。

2 功能模块设计

本系统基于 ESP32 嵌入式平台开发，结合 DHT11 温湿度传感器、OLED 显示屏和 L9110H 电机驱动模块，构建了一个具备环境感知、数据可视化与自动控制能力的模糊控制系统。整个系统以 FreeRTOS 实时操作系统为基础，采用多任务并发机制，实现对温湿度信息的采集、显示以及基于模糊逻辑的智能调节。

2.1 主控模块

本次系统设计的微处理芯片采用 FireBeetle 2 ESP32-C6 芯片，支持 2.4 GHz Wi-Fi 6、Bluetooth 5、Zigbee 3.0 及 Thread 1.3 的系统级芯片 (SoC)，集成了高性能 RISC-V 32 位处理器和低功耗 RISC-V 32 位处理器、Wi-Fi、Bluetooth LE、802.15.4 基带和 MAC、RF 模块及外设等。

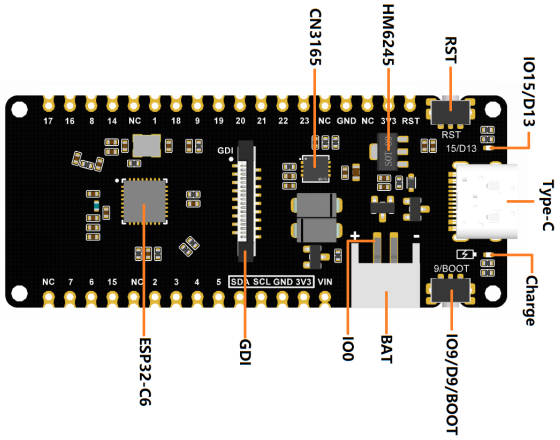


图 1: ESP32C6 芯片

- 硬件引脚接线：

  1. DHT11 传感器：
     - 数据引脚：连接到 GPIO1
     - 电源引脚：连接到 3.3V
     - 地引脚：连接到 GND
  2. OLED 显示屏：
     - SCL 引脚：连接到 GPIO4
     - SDA 引脚：连接到 GPIO3
     - 电源引脚：连接到 3.3V
     - 地引脚：连接到 GND
  3. 电机驱动模块：
     - INA 引脚：连接到 GPIO7（PWM 输出）。
     - INB 引脚：连接到 GPIO6（方向控制）。
     - 电源引脚：连接到外部电源。
     - 地引脚：连接到 GND。

- 模块实现的功能

  1. 数据采集：通过连接 DHT11 温湿度传感器，采集环境的温度和湿度数据。
  2. 数据处理：使用模糊控制算法，根据采集的温湿度数据计算控制输出，用于调节电机速度。
  3. 通信与显示：通过 I2C 接口与 OLED 显示屏通信，实时显示温湿度数据和控制输出。
  4. 任务调度：使用 FreeRTOS 实现多任务并行运行，包括传感器数据采集任务、模糊控制任务和电机控制任务。

2.2 数据采集模块

本模块基于 DHT11 温湿度传感器进行环境数据采集。DHT11 是一款数字温湿度传感器，通过单总线协议与主控芯片通信，具有较高的稳定性和较低的成本。

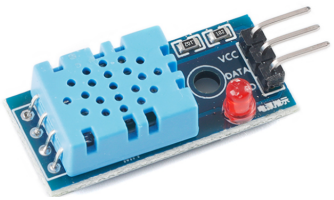


图 2: DHT11 温湿度传感器

- 模块实现的功能：

  1. 实现对当前环境温度和湿度的周期性采集；
  2. 支持浮点数格式输出，提高精度；

3. 通过 GPIO 引脚与 ESP32 进行双向通信；
4. 提供错误检测机制，确保数据完整性。

- DHT11 的工作原理：DHT11 通过单总线协议与主机通信，传输温度数据。

1. 初始化阶段
   - 主机（MCU）通过拉低信号线向 DH11 发送启动信号。
   - DHT11 接收到启动信号后，拉低信号线 80 s，然后拉高信号线 80 s，表示准备好发送数据。
2. 数据传输阶段
   - DH11 通过单总线协议发送 40 位数据（5 字节），包括湿度和温度信息。
   - 数据格式：前 8 位：湿度整数部分。中间 8 位：湿度小数部分。接着 8 位：温度整数部分。再接 8 位：温度小数部分。最后 8 位：校验和（前 4 字节的和）。
3. 数据位的编码

   每一位数据通过高电平的持续时间来表示：
   - 低电平 50 s + 高电平 26-28 s：表示逻辑 0。
   - 低电平 50 s + 高电平 70 s：表示逻辑 1。
4. 校验阶段
   - 主机接收完 40 位数据后，计算前 4 字节的和，与校验和进行比较。
   - 如果校验通过，则数据有效；否则数据无效。

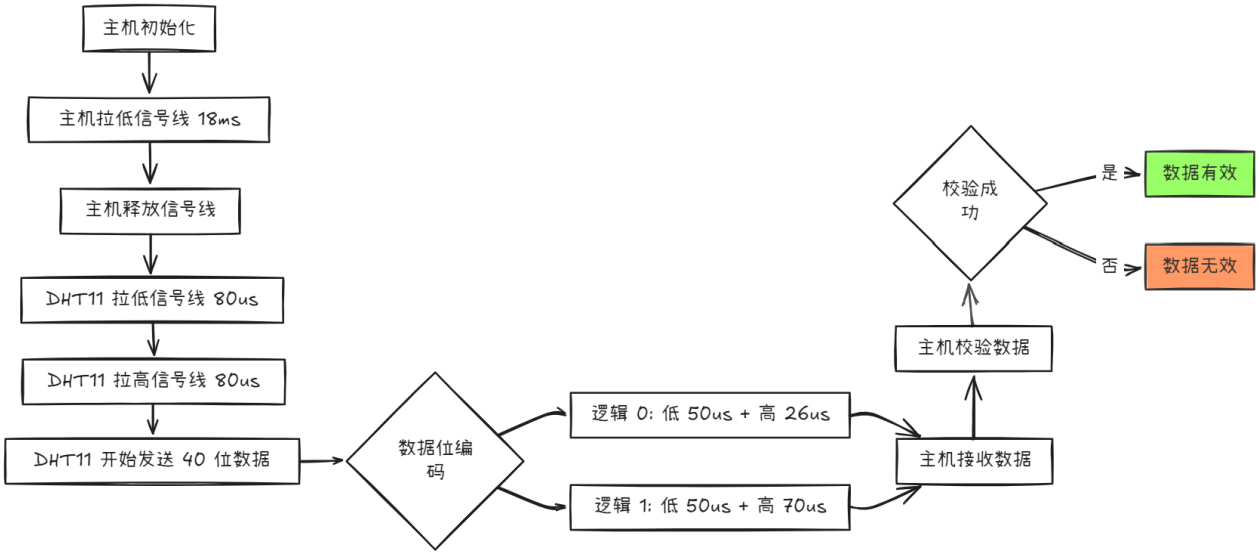


图 3: DHT11 工作流程图

2.3 显示模块

关于显示模块，本次实验我采用了 ssd1306 模块，它是一种自发光显示技术，利用有机材料在电流作用下发光。它具有高对比度、低功耗、广视角等优点。

图 4: ssd1306 显示屏

- 模块实现的功能：对系统检测到的温湿度进行实时显示，并当 DHT11 传感器读取失败时，提示错误信息。
- ssd1306 的工作原理：
  - 像素发光：OLED 显示屏由多个像素组成，每个像素包含红、绿、蓝（RGB）子像素。当电流通过有机材料时，材料发光，形成图像。
  - 驱动方式：OLED 显示屏通常采用矩阵驱动方式，通过行列扫描控制像素点亮。
  - 通信协议：OLED 显示屏通常通过 I2C 或 SPI 接口与主机通信。主机通过发送命令和数据控制 OLED 的显示内容。
  - 显示内容更新：显存数组存储当前显示内容。更新显示时，显存数据通过通信接口发送到 OLED 显示屏。

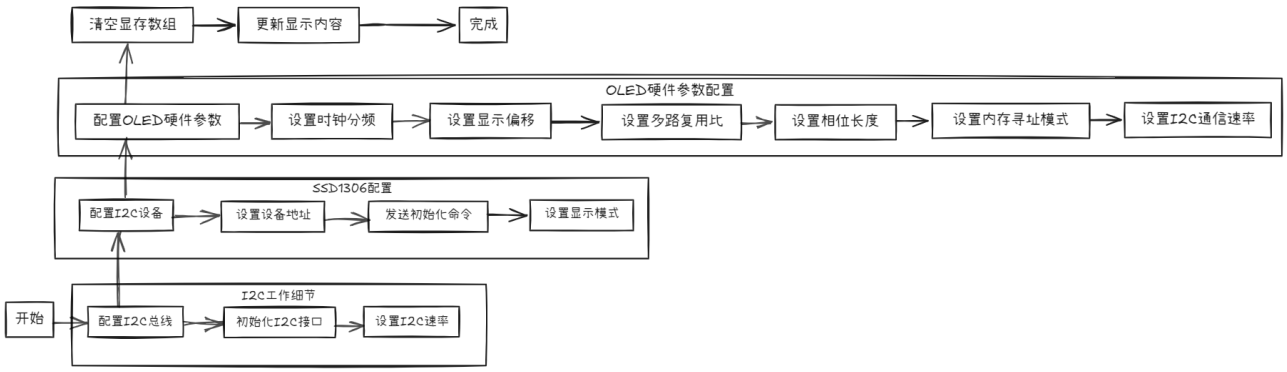


图 5: ssd1306 显示工作原理

2.4 电机控制模块

本次实验采用 L9110H 作为被控对象，它是一种双通道推挽式功率放大器集成电路，广泛应用于电机驱动、继电器控制以及小型电磁阀驱动等场景。它能够提供较高的输出电流，并且具备良好的热稳定性和可靠性。该芯片内部集成了两个 NPN 和 PNP 晶体管对，可以实现对负载的正向和反向驱动控制。

- 模块实现的功能：
  - PWM 输出：通过 PWM 信号控制电机的速度。
  - 方向控制：通过 GPIO 引脚连接 INA 与 INB，控制电机的转速和旋转方向。

图 6: L9110H 电机模块

  - 实时调节：根据模糊控制器的输出实时调整电机速度。

- L9110H 电机工作原理：
  - 初始化：配置 GPIO 引脚，连接电机驱动模块的 INA、INB。
  - 速度控制：根据模糊控制器的输出设置 PWM 占空比。
  - 方向控制：通过控制电机两端的电压来实现对电机旋转方向的控制。

2.5 模糊控制模块

在模糊控制模块中，我采用 Mamdani 型模糊控制算法，构建了一个基于模糊逻辑的闭环温度调控系统。系统通过传感器实时采集当前环境温度，并与设定的目标温度进行比较，计算出温度误差及其变化率，作为模糊控制器的两个输入变量。利用三角隶属度函数对误差和误差变化率进行模糊化处理，将其映射到三个模糊集合：Low、Medium 和 High，从而获得对应的隶属度值。然后，根据预先设定的模糊规则库，系统执行模糊推理，将输入的模糊集合与规则库匹配，生成输出模糊集合，并通过最大值法对所有规则的输出进行合成，形成统一的模糊输出区域。最终，为将模糊结果转化为实际可执行的控制信号，用重心法（COG）进行去模糊化，计算出一个精确的控制量。该控制量被用于调节电机转速，从而实现对温度的动态调节。整个模糊控制系统具有较强的非线性适应能力和良好的可解释性，适用于本实验中对温控系统的柔性调节需求。具体流程如下图所示：

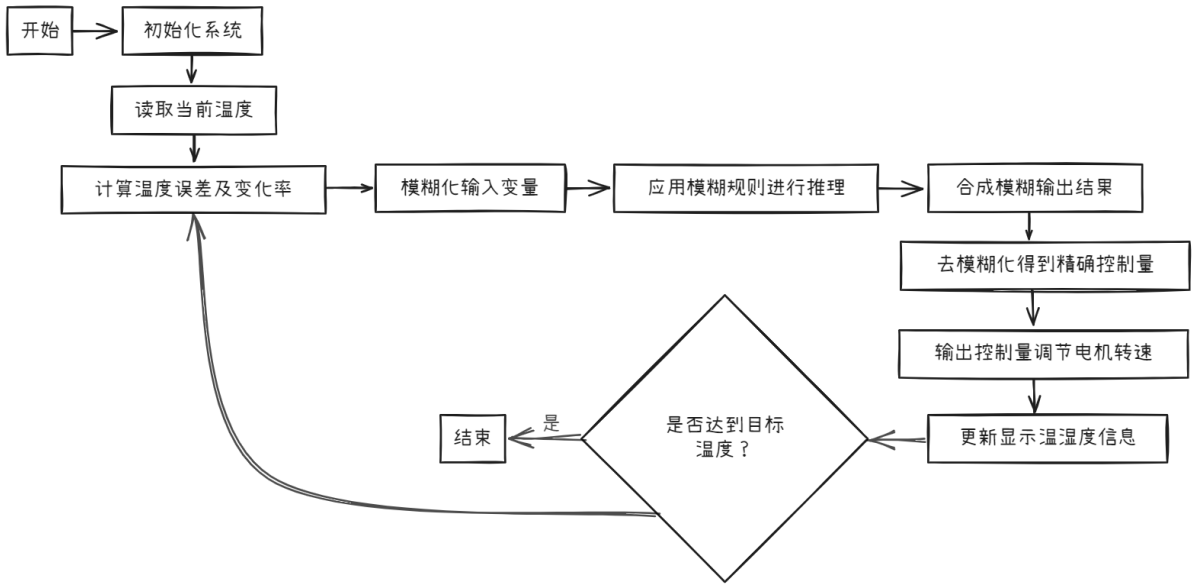


图 7: Mamdani 模糊控制

3 软件设计

在本次实验的软件设计中，系统采用了模块化的设计思路，以实现对温控系统的模糊控制。主程序通过调用定义好的传感器驱动和模糊控制算法接口函数，完成温度数据的采集、处理及控制输出。

1. 初始化与主循环逻辑：在 main.c 中完成了 OLED 显示屏、DHT 温湿度传感器以及 PWM 电机驱动的初始化，并创建了一个 FreeRTOS 任务用于周期性执行模糊控制

```
void fuzzy_control_task(void *pvParameter)
{
  float current_temperature = 0.0;
  float humidity = 0.0;
  static float last_error = 0.0;

  while (1){
    // 读取温湿度数据
    esp_err_t result = dht_read_float_data(DHT_SENSOR_TYPE,
    DHT_SENSOR_PIN, &humidity, &current_temperature);
    if (result == ESP_OK){
      // 计算误差和误差变化
      float error = target_temperature - current_temperature;
      float delta_error = error - last_error;

      // 模糊化
      FuzzyMembership error_membership = fuzzify_error(error);
      FuzzyMembership delta_error_membership = fuzzify_delta_error(delta_error);

      // 模糊推理
      FuzzyMembership control_membership = infer_control(error_membership,
      delta_error_membership);

      // 去模糊化
      float control_output = defuzzify(control_membership);

      // 更新上一次误差
      last_error = error;

      // 显示温湿度和控制输出
      display_temp_hum(current_temperature,humidity);
      ESP_LOGI(TAG,"Temp:%.1f,Hum:%.1f,Error: %.1f,Delta Error:%.1f,Control Output
          :%.1f",current_temperature,humidity,error,delta_error,control_output);

      set_motor_speed((int)control_output);
    }
    else{
      ESP_LOGE(TAG, "Failed to read data from DHT sensor: %s", esp_err_to_name(
          result));
      OLED_Clear();
```

```
39            OLED_ShowString(0, 0, "Sensor Error", OLED_8X16);
40            OLED_Update();
41          }
42          vTaskDelay(pdMS_TO_TICKS(1000));
43        }
44      }
```

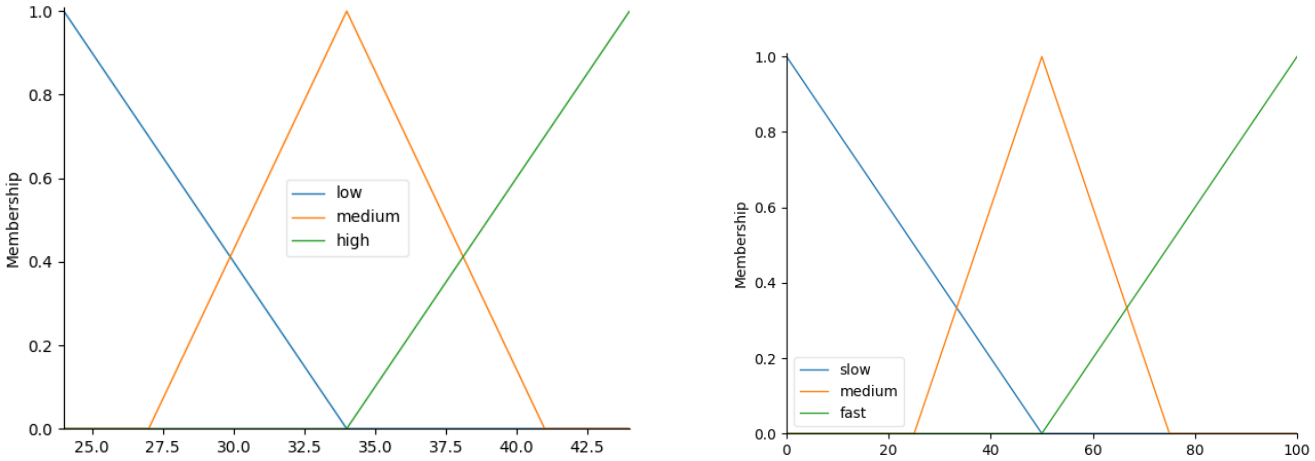
2. 模糊控制器接口定义：TMF.h 定义了模糊控制所需的数据结构和函数接口

```
1     // 定义模糊集合
2     typedef struct {
3       float low;
4       float medium;
5       float high;
6     } FuzzyMembership;
7
8     FuzzyMembership fuzzify_error(float error);
9     FuzzyMembership fuzzify_delta_error(float delta_error);
10    FuzzyMembership infer_control(FuzzyMembership error_membership, FuzzyMembership
          delta_error_membership);
11    float defuzzify(FuzzyMembership control_membership);
12    float triangle(float x, float a, float b, float c);
```

3. 模糊化函数：在 TMF.c 中实现了模糊化的具体逻辑，将误差和误差变化率映射到模糊集合

```
1     FuzzyMembership fuzzify_error(float error) {
2       FuzzyMembership membership = {0.0, 0.0, 0.0};
3
4       membership.high = triangle(error, -7.0, -7.0, 0.0);
5       membership.medium = triangle(error, -3.0, 0.0, 3.0);
6       membership.low = triangle(error, 0.0, 7.0, 7.0);
7
8       return membership;
9     }
10
11    FuzzyMembership fuzzify_delta_error(float delta_error) {
12      FuzzyMembership membership = {0.0, 0.0, 0.0};
13
14      membership.high = triangle(delta_error, -2.0, -2, 0.0);
15      membership.medium = triangle(delta_error, -1.0, 0.0, 1.0);
16      membership.low = triangle(delta_error, 0.0, 2, 2.0);
17
18      return membership;
19    }
```

本次设计的温度转换和电机转速的隶属度函数，如下图所示：



(a) Temp (b) Speed

图 8: 温度与电机转速的隶属度函数

4. 模糊推理规则实现：模糊推理部分根据预设规则进行判断并合成输出模糊集合

```
FuzzyMembership infer_control(FuzzyMembership error_membership, FuzzyMembership
    delta_error_membership) {
  FuzzyMembership control_membership = {0.0, 0.0, 0.0};

  control_membership.low = fmax(control_membership.low, fmax(error_membership.low,
      delta_error_membership.low));
  control_membership.medium = fmax(control_membership.medium, fmax(error_membership
      .medium, delta_error_membership.medium));
  control_membership.high = fmax(control_membership.high, fmax(error_membership.
      high, delta_error_membership.high));

  control_membership.high = fmax(control_membership.high, fmax(error_membership.
      high, delta_error_membership.low));
  control_membership.low = fmax(control_membership.low, fmax(error_membership.low,
      delta_error_membership.high));

  return control_membership;
}
```

5. 去模糊化函数：采用重心法对模糊输出进行去模糊化，得到一个精确值用于控制

```
float defuzzify(FuzzyMembership control_membership) {
  float numerator = 0.0;
  float denominator = 0.0;

  for (float x = 0.0; x <= 33.3; x += 1.0) {
    float membership = fmin(control_membership.low, triangle(x, 0.0, 16.65, 33.3));
    numerator += x * membership;
    denominator += membership;
  }
```

```
10
11      for (float x = 33.3; x <= 66.6; x += 1.0) {
12        float membership = fmax(control_membership.medium, triangle(x, 33.3, 50.0,
              66.6));
13        numerator += x * membership;
14        denominator += membership;
15      }
16
17      for (float x = 66.6; x <= 100.0; x += 1.0) {
18        float membership = fmax(control_membership.high, triangle(x, 66.6, 83.3, 100.0)
              );
19        numerator += x * membership;
20        denominator += membership;
21      }
22
23      float output = (denominator == 0.0) ? 0.0 : (numerator / denominator);
24      return output * 255.0 / 100.0;  // 映射到 [0,255]
25    }
```

整个软件设计通过调用 fuzzify 对输入变量进行模糊化，使用 infer_control 进行模糊推理，最后通过 defuzzify 将模糊结果转化为实际可用的控制信号，最终用于调节电机转速，实现对温度的闭环控制。同时，OLED 屏幕实时显示当前温度、湿度以及控制状态，便于观察和调试。

4 系统测试

验证系统各模块的功能是否正常，包括温湿度数据采集、模糊控制输出计算、电机速度调节以及 OLED 显示功能。

- 硬件环境：
  1. 主控芯片：ESP32-C6
  2. 温湿度传感器：DHT11
  3. 电机驱动模块：L9110H
  4. 显示模块：OLED
- 软件环境:
  1. 开发工具：Visual Studio Code
  2. 编程语言：C
  3. FreeRTOS 任务调度系统
- 硬件接线，如下图所示：

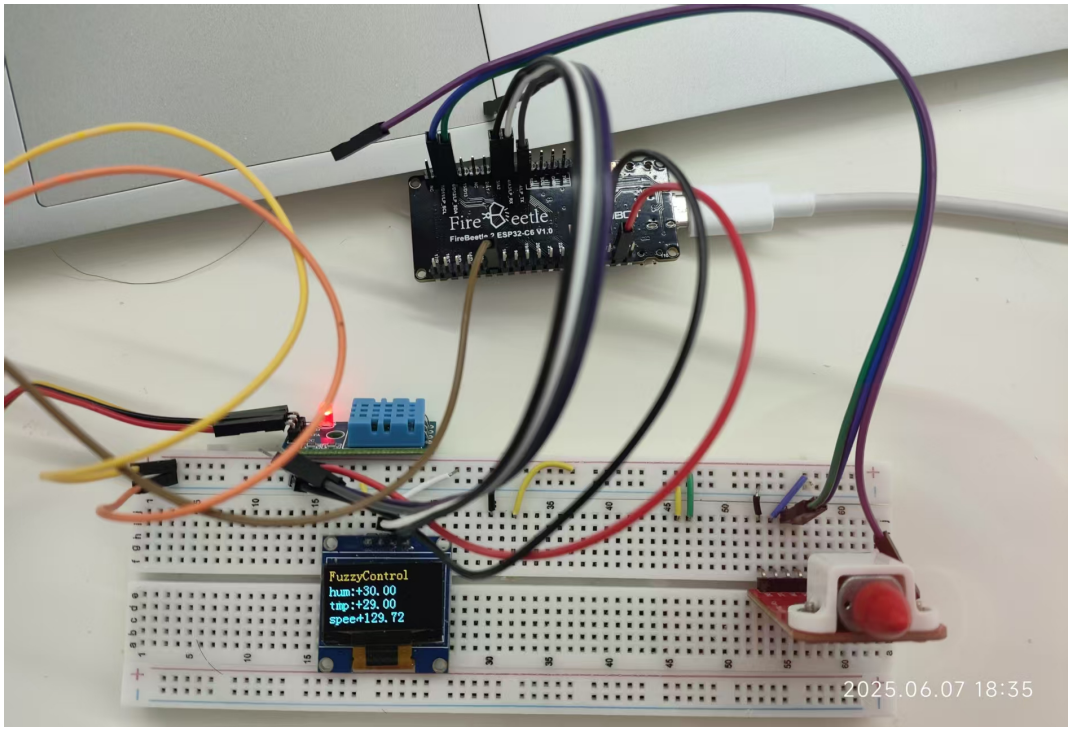


图 9: 硬件接线实物图

- 测试效果图

1. 低温范围：29℃，风扇相对速度：129.72

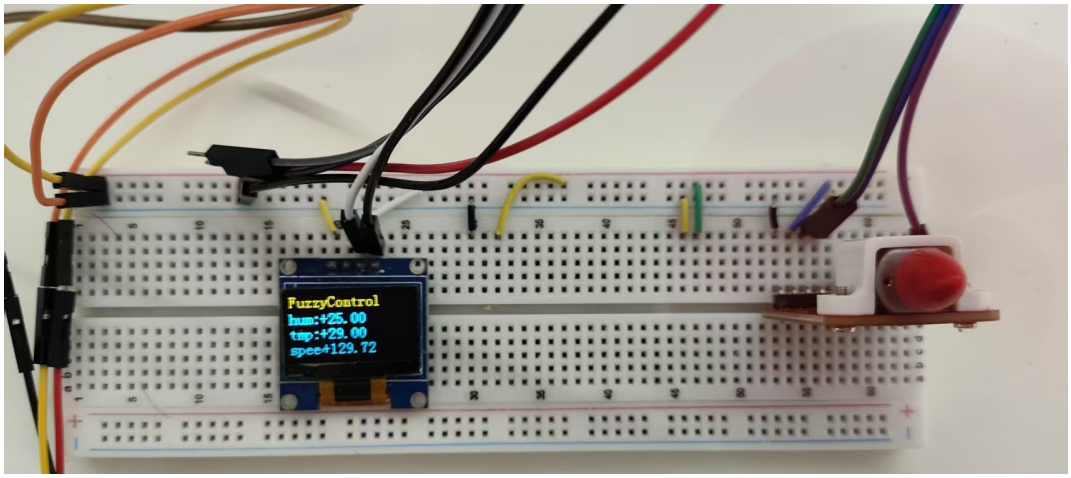


图 10: 相对低温效果图

2. 中温范围：32℃，风扇相对速度：141.13

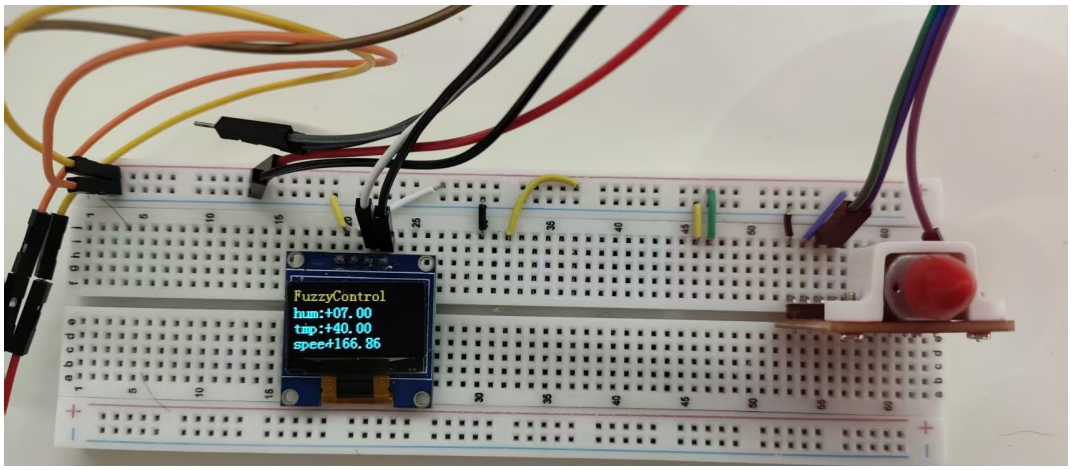


图 11: 相对中等温度效果图

3. 高温范围：40℃，风扇相对速度：166.86

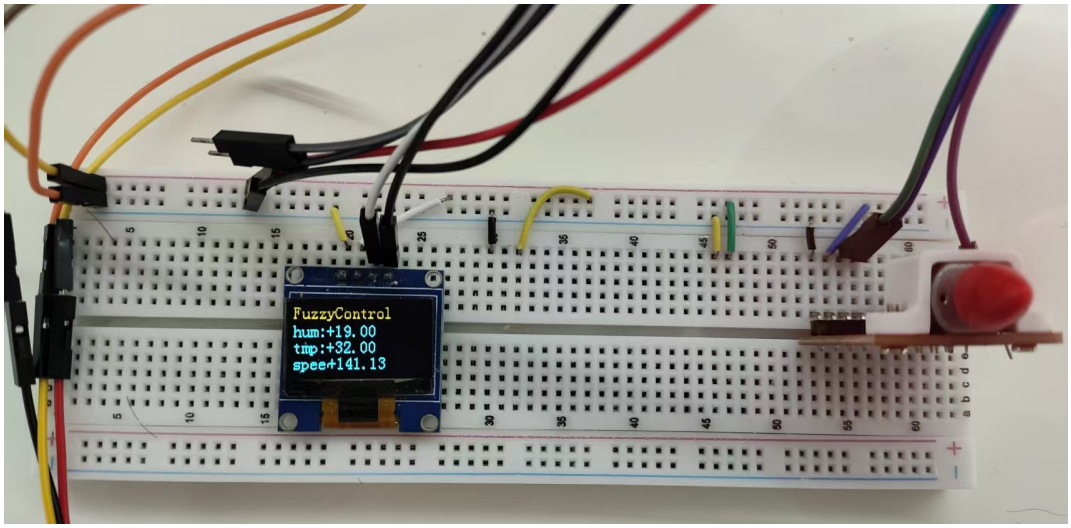


图 12: 相对高温效果图

5 实验思考与总结

通过本次物联网控制技术实验，我深入理解并实践了基于 Mamdani 模糊控制算法的温湿度控制系统设计。实验不仅巩固了我对 ESP32 开发环境、FreeRTOS 多任务编程、DHT11 温湿度传感器数据采集、OLED 屏幕数据显示以及 L9110H 电机驱动模块应用的理论知识，而且提升了我的实际操作能力和问题解决能力。

- 实验收获

1. 理论知识与实践结合：通过亲自搭建和编程 ESP32 开发环境，我对嵌入式系统开发有了更深刻的理解。实验过程中，我将理论知识应用于实际问题的解决，加深了对物联网控制技术的认识。

2. 多任务编程技能提升：在 FreeRTOS 环境下进行多任务编程，让我学会了如何在实时操作系统中有效地管理任务和资源，这对于未来从事嵌入式系统开发工作具有重要意义。
3. 传感器数据处理能力增强：通过使用 DHT11 温湿度传感器进行数据采集，我学会了如何处理和分析传感器数据，这对于进行环境监测和智能控制应用至关重要。
4. 模糊控制算法应用：实验中实现的 Mamdani 模糊控制算法，让我掌握了如何利用模糊逻辑进行复杂系统的控制，这对于提高系统的鲁棒性和适应性具有重要作用。

- 实验反思

1. 系统稳定性问题：在实验过程中，我遇到了系统稳定性问题，这提示我在系统设计时需要更加关注硬件选择和软件优化，以确保系统的稳定运行。
2. 算法优化空间：虽然 Mamdani 模糊控制算法在本次实验中取得了良好的效果，但我也意识到算法还有进一步优化的空间，例如通过调整隶属度函数和规则库来提高控制精度。
3. 系统集成挑战：在将各个功能模块集成到一个系统中时，我遇到了一些挑战，这让我认识到在复杂系统设计中，模块化设计和接口标准化的重要性。

附录 A　主程序

Listing 1: main.c

```
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "OLED.h"
#include "dht.h"
#include "esp_log.h"
#include <driver/gpio.h>
#include <driver/ledc.h>
#include "TMF.h"
#include "mydht11.h"

#define OLED_I2C I2C_NUM_0

#define OLED_SCL 4
#define OLED_SDA 3
#define OLED_ADD 0x78
#define OLED_SPEED 400000
#define DHT_SENSOR_PIN 1
#define DHT_SENSOR_TYPE DHT_TYPE_DHT11
#define DELAY_TIME 3000

static const char *TAG = "OLED_DHT";
static const char *TAG1 = "L9110H";

float target_temperature = 34.0; // 目标温度

// 显示温湿度数据
void display_temp_hum(float temperature, float humidity)
{
  OLED_Clear();
  OLED_ShowString(0, 0, "hum:", OLED_8X16);
  OLED_ShowString(0, 16, "tmp:", OLED_8X16);
  OLED_ShowFloatNum(32, 0, humidity, 2, 2, OLED_8X16);
  OLED_ShowFloatNum(32, 16, temperature, 2, 2, OLED_8X16);
  OLED_Update();
}

// 模糊控制任务
void fuzzy_control_task(void *pvParameter)
{
  float current_temperature = 0.0;
  float humidity = 0.0;
  static float last_error = 0.0;
```

```
44
45      while (1){
46        // 读取温湿度数据
47        esp_err_t result = dht_read_float_data(DHT_SENSOR_TYPE,
48        DHT_SENSOR_PIN, &humidity, &current_temperature);
49        if (result == ESP_OK){
50          // 计算误差和误差变化
51          float error = target_temperature - current_temperature;
52          float delta_error = error - last_error;
53
54          // 模糊化
55          FuzzyMembership error_membership = fuzzify_error(error);
56          FuzzyMembership delta_error_membership = fuzzify_delta_error(delta_error);
57
58          // 模糊推理
59          FuzzyMembership control_membership = infer_control(error_membership,
60          delta_error_membership);
61
62          // 去模糊化
63          float control_output = defuzzify(control_membership);
64
65          // 更新上一次误差
66          last_error = error;
67
68          // 显示温湿度和控制输出
69          display_temp_hum(current_temperature, humidity);
70          ESP_LOGI(TAG,"Temp: %.1f, Hum: %.1f, Error: %.1f,Delta Error: %.1f, Control Output:
                %.1f",
71          current_temperature, humidity, error, delta_error, control_output);
72
73          set_motor_speed((int)control_output);
74        }
75        else{
76          ESP_LOGE(TAG, "Failed to read data from DHT sensor: %s", esp_err_to_name(result));
77          OLED_Clear();
78          OLED_ShowString(0, 0, "Sensor Error", OLED_8X16);
79          OLED_Update();
80        }
81
82        vTaskDelay(pdMS_TO_TICKS(1000));
83      }
84    }
85    void dht_task(void *pvParameter)
86    {
87      gpio_set_pull_mode(DHT_SENSOR_PIN, GPIO_PULLUP_ONLY);
88      float temperature = 0.0;
89      float humidity = 0.0;
```

```
90
91      while (1)
92      {
93        esp_err_t result = dht_read_float_data(DHT_SENSOR_TYPE,
94        DHT_SENSOR_PIN, &humidity, &temperature);
95        if (result == ESP_OK)
96        {
97          ESP_LOGI(TAG, "Temperature: %.1f C, Humidity: %.1f %%",
98          temperature, humidity);
99          display_temp_hum(temperature, humidity);
100       }
101       else
102       {
103         ESP_LOGE(TAG, "Failed to read data from DHT sensor: %s", esp_err_to_name(result));
104         OLED_Clear();
105         OLED_ShowString(0, 0, "Sensor Error", OLED_8X16);
106         OLED_Update();
107       }
108       vTaskDelay(pdMS_TO_TICKS(1000));
109     }
110   }
111
112   void app_main(void)
113   {
114     l9110_pwm_init();
115     OLED_Init(OLED_I2C, OLED_ADD, OLED_SCL, OLED_SDA, OLED_SPEED);
116     // 设置 DHT11 引脚为上拉模式
117     // gpio_set_pull_mode(DHT_SENSOR_PIN, GPIO_PULLUP_ONLY);
118
119     xTaskCreate(fuzzy_control_task,"FuzzyControlTask", 4096, NULL, 5, NULL);
120   }
```

附录 B　OLED 显示代码

Listing 2: OLED.c

```c
#include "OLED.h"
uint8_t OLED_DisplayBuf[8][128];

//I2C总线句柄
i2c_master_bus_handle_t oled_bus_handle;

//OLED设备句柄
i2c_master_dev_handle_t oled_dev_handle;

/**
* 函    数：OLED写命令
* 参    数：Command 要写入的命令值，范围：0x00-0xFF
* 返 回 值：无
*/
void OLED_WriteCommand(uint8_t Command)
{
  uint8_t writebuffer[2];

  writebuffer[0] = 0x00;
  writebuffer[1] = Command;
  ESP_ERROR_CHECK(i2c_master_transmit(oled_dev_handle, writebuffer, 2, -1));
}

/**
* 函    数：OLED写数据
* 参    数：Data 要写入数据的起始地址
* 参    数：Count 要写入数据的数量
* 返 回 值：无
*/
void OLED_WriteData(uint8_t *Data, uint8_t Count)
{
  uint8_t i;
  uint8_t writebuffer[Count+1];

  writebuffer[0] = 0x40;

  for(i = 0; i<Count; i ++) writebuffer[i+1] = Data[i];
  ESP_ERROR_CHECK(i2c_master_transmit(oled_dev_handle,writebuffer,Count+1,-1));
}

/**
* 函    数：OLED初始化
* 参    数：port  I2C端口号
```

```
44 * 参     数: add   oled地址
45 * 参     数: scl   SCL引脚
46 * 参     数: sda   SDA引脚
47 * 参     数: speed SCL时钟频率(Hz), 不应大于400k
48 * 返 回 值: 无
49 * 说     明: 使用前, 需要调用此初始化函数
50 */
51 void OLED_Init(int port, uint8_t add, int scl, int sda, int speed)
52 {
53   //配置I2C总线
54   i2c_master_bus_config_t oled_i2c_mst_cfg =
55   {
56     .clk_source = I2C_CLK_SRC_DEFAULT,      //使用默认时钟源
57     .i2c_port = port,                       //指定I2C端口号
58     .scl_io_num = scl,                      //指定SCL引脚号
59     .sda_io_num = sda,                      //指定SDA引脚号
60     .glitch_ignore_cnt = 7,                 //设置毛刺忽略计数
61     .flags.enable_internal_pullup = false,  //禁用内部上拉电阻
62   };
63
64   //创建I2C总线并获取句柄
65   ESP_ERROR_CHECK(i2c_new_master_bus(&oled_i2c_mst_cfg, &oled_bus_handle));
66
67   //配置I2C从机设备
68   i2c_device_config_t oled_dev_cfg =
69   {
70     .dev_addr_length = I2C_ADDR_BIT_LEN_7,  //设置设备地址长度为7位
71     .device_address = add >> 1,             //指定设备地址
72     .scl_speed_hz = speed,                  //设置I2C时钟速度
73     .flags.disable_ack_check = false,       //启用ACK检查
74   };
75
76   //将设备添加到I2C总线并获取设备句柄
77   ESP_ERROR_CHECK(i2c_master_bus_add_device(oled_bus_handle, &oled_dev_cfg, &oled_dev_handle
       ));
78
79   /*写入一系列的命令, 对OLED进行初始化配置*/
80   OLED_WriteCommand(0xAE);  //设置显示开启/关闭, 0xAE关闭, 0xAF开启
81
82   OLED_WriteCommand(0xD5);  //设置显示时钟分频比/振荡器频率
83   OLED_WriteCommand(0x80);  //0x00-0xFF
84
85   OLED_WriteCommand(0xA8);  //设置多路复用率
86   OLED_WriteCommand(0x3F);  //0x0E-0x3F
87
88   OLED_WriteCommand(0xD3);  //设置显示偏移
89   OLED_WriteCommand(0x00);  //0x00-0x7F
```

```
90
91    OLED_WriteCommand(0x40);  //设置显示开始行，0x40-0x7F
92
93    OLED_WriteCommand(0xA1);  //设置左右方向，0xA1正常，0xA0左右反置
94
95    OLED_WriteCommand(0xC8);  //设置上下方向，0xC8正常，0xC0上下反置
96
97    OLED_WriteCommand(0xDA);  //设置COM引脚硬件配置
98    OLED_WriteCommand(0x12);
99
100   OLED_WriteCommand(0x81);  //设置对比度
101   OLED_WriteCommand(0xCF);  //0x00-0xFF
102
103   OLED_WriteCommand(0xD9);  //设置预充电周期
104   OLED_WriteCommand(0xF1);
105
106   OLED_WriteCommand(0xDB);  //设置VCOMH取消选择级别
107   OLED_WriteCommand(0x30);
108
109   OLED_WriteCommand(0xA4);  //设置整个显示打开/关闭
110
111   OLED_WriteCommand(0xA6);  //设置正常/反色显示，0xA6正常，0xA7反色
112
113   OLED_WriteCommand(0x8D);  //设置充电泵
114   OLED_WriteCommand(0x14);
115
116   OLED_WriteCommand(0xAF);  //开启显示
117
118   OLED_Clear();        //清空显存数组
119   OLED_Update();         //更新显示，清屏，防止初始化后未显示内容时花屏
120 }
121
122 /**
123 * 函    数：OLED设置显示光标位置
124 * 参    数：Page 指定光标所在的页，范围：0-7
125 * 参    数：X 指定光标所在的X轴坐标，范围：0-127
126 * 返 回 值：无
127 * 说    明：OLED默认的Y轴，只能8个Bit为一组写入，即1页等于8个Y轴坐标
128 */
129 void OLED_SetCursor(uint8_t Page, uint8_t X)
130 {
131   /*如果使用此程序驱动1.3寸的OLED显示屏，则需要解除此注释*/
132   /*因为1.3寸的OLED驱动芯片（SH1106）有132列*/
133   /*屏幕的起始列接在了第2列，而不是第0列*/
134   /*所以需要将X加2，才能正常显示*/
135   //  X += 2;
136
```

```
137   /*通过指令设置页地址和列地址*/
138   OLED_WriteCommand(0xB0 | Page);          //设置页位置
139   OLED_WriteCommand(0x10 | ((X & 0xF0) >> 4));  //设置X位置高4位
140   OLED_WriteCommand(0x00 | (X & 0x0F));      //设置X位置低4位
141 }
142
143 /**
144 * 函    数：次方函数
145 * 参    数：X 底数
146 * 参    数：Y 指数
147 * 返 回 值：等于X的Y次方
148 */
149 uint32_t OLED_Pow(uint32_t X, uint32_t Y)
150 {
151   uint32_t Result = 1;  //结果默认为1
152   while (Y --)       //累乘Y次
153   {
154     Result *= X;     //每次把X累乘到结果上
155   }
156   return Result;
157 }
158
159 /**
160 * 函    数：判断指定点是否在指定多边形内部
161 * 参    数：nvert 多边形的顶点数
162 * 参    数：vertx verty 包含多边形顶点的x和y坐标的数组
163 * 参    数：testx testy 测试点的X和y坐标
164 * 返 回 值：指定点是否在指定多边形内部，1：在内部，0：不在内部
165 */
166 uint8_t OLED_pnpoly(uint8_t nvert, int16_t *vertx, int16_t *verty, int16_t testx, int16_t
        testy)
167 {
168   int16_t i, j, c = 0;
169
170   for (i = 0, j = nvert - 1; i < nvert; j = i++)
171   {
172     if (((verty[i] > testy) != (verty[j] > testy)) &&
173     (testx < (vertx[j] - vertx[i]) * (testy - verty[i]) / (verty[j] - verty[i]) + vertx[i]))
            c = !c;
174   }
175   return c;
176 }
177
178 /**
179 * 函    数：判断指定点是否在指定角度内部
180 * 参    数：X Y 指定点的坐标
181 * 参    数：StartAngle EndAngle 起始角度和终止角度，范围：-180-180
```

```
182 *           水平向右为0度，水平向左为180度或-180度，下方为正数，上方为负数，顺时针旋转
183 * 返 回 值：指定点是否在指定角度内部，1：在内部，0：不在内部
184 */
185 uint8_t OLED_IsInAngle(int16_t X, int16_t Y, int16_t StartAngle, int16_t EndAngle)
186 {
187   int16_t PointAngle;
188   PointAngle = atan2(Y, X) / 3.14 * 180;  //计算指定点的弧度，并转换为角度表示
189   if (StartAngle < EndAngle)  //起始角度小于终止角度的情况
190   {
191     /*如果指定角度在起始终止角度之间，则判定指定点在指定角度*/
192     if (PointAngle >= StartAngle && PointAngle <= EndAngle) return 1;
193   }
194   else      //起始角度大于于终止角度的情况
195   {
196     /*如果指定角度大于起始角度或者小于终止角度，则判定指定点在指定角度*/
197     if (PointAngle >= StartAngle || PointAngle <= EndAngle) return 1;
198   }
199   return 0;   //不满足以上条件，则判断判定指定点不在指定角度
200 }
201
202
203 /**
204 * 函     数：将OLED显存数组更新到OLED屏幕
205 * 参     数：无
206 * 返 回 值：无
207 * 说     明：所有的显示函数，都只是对OLED显存数组进行读写
208 *           随后调用OLED_Update函数或OLED_UpdateArea函数
209 *           才会将显存数组的数据发送到OLED硬件，进行显示
210 *           故调用显示函数后，要想真正地呈现在屏幕上，还需调用更新函数
211 */
212 void OLED_Update(void)
213 {
214   uint8_t j;
215   /*遍历每一页*/
216   for (j = 0; j < 8; j ++)
217   {
218     /*设置光标位置为每一页的第一列*/
219     OLED_SetCursor(j, 0);
220     /*连续写入128个数据，将显存数组的数据写入到OLED硬件*/
221     OLED_WriteData(OLED_DisplayBuf[j], 128);
222   }
223 }
224
225 /**
226 * 函     数：将OLED显存数组部分更新到OLED屏幕
227 * 参     数：X 指定区域左上角的横坐标，范围：-32768-32767，屏幕区域：0-127
228 * 参     数：Y 指定区域左上角的纵坐标，范围：-32768-32767，屏幕区域：0-63
```

```
229 * 参     数：Width 指定区域的宽度，范围：0-128
230 * 参     数：Height 指定区域的高度，范围：0-64
231 * 返 回 值：无
232 * 说     明：此函数会至少更新参数指定的区域
233 *             如果更新区域Y轴只包含部分页，则同一页的剩余部分会跟随一起更新
234 * 说     明：所有的显示函数，都只是对OLED显存数组进行读写
235 *             随后调用OLED_Update函数或OLED_UpdateArea函数
236 *             才会将显存数组的数据发送到OLED硬件，进行显示
237 *             故调用显示函数后，要想真正地呈现在屏幕上，还需调用更新函数
238 */
239 void OLED_UpdateArea(int16_t X, int16_t Y, uint8_t Width, uint8_t Height)
240 {
241   int16_t j;
242   int16_t Page, Page1;
243 
244   /*负数坐标在计算页地址时需要加一个偏移*/
245   /*(Y + Height - 1) / 8 + 1的目的是(Y + Height) / 8并向上取整*/
246   Page = Y / 8;
247   Page1 = (Y + Height - 1) / 8 + 1;
248   if (Y < 0)
249   {
250     Page -= 1;
251     Page1 -= 1;
252   }
253 
254   /*遍历指定区域涉及的相关页*/
255   for (j = Page; j < Page1; j ++)
256   {
257     if (X >= 0 && X <= 127 && j >= 0 && j <= 7)   //超出屏幕的内容不显示
258     {
259       /*设置光标位置为相关页的指定列*/
260       OLED_SetCursor(j, X);
261       /*连续写入Width个数据，将显存数组的数据写入到OLED硬件*/
262       OLED_WriteData(&OLED_DisplayBuf[j][X], Width);
263     }
264   }
265 }
266 
267 /**
268 * 函     数：将OLED显存数组全部清零
269 * 参     数：无
270 * 返 回 值：无
271 * 说     明：调用此函数后，要想真正地呈现在屏幕上，还需调用更新函数
272 */
273 void OLED_Clear(void)
274 {
275   uint8_t i, j;
```

```
276   for (j = 0; j < 8; j ++)         //遍历8页
277   {
278     for (i = 0; i < 128; i ++)       //遍历128列
279     {
280       OLED_DisplayBuf[j][i] = 0x00; //将显存数组数据全部清零
281     }
282   }
283 }
284 
285 /**
286 * 函     数：将OLED显存数组部分清零
287 * 参     数：X 指定区域左上角的横坐标，范围：-32768-32767，屏幕区域：0-127
288 * 参     数：Y 指定区域左上角的纵坐标，范围：-32768-32767，屏幕区域：0-63
289 * 参     数：Width 指定区域的宽度，范围：0-128
290 * 参     数：Height 指定区域的高度，范围：0-64
291 * 返 回 值：无
292 * 说     明：调用此函数后，要想真正地呈现在屏幕上，还需调用更新函数
293 */
294 void OLED_ClearArea(int16_t X, int16_t Y, uint8_t Width, uint8_t Height)
295 {
296   int16_t i, j;
297   for (j = Y; j < Y + Height; j ++)   //遍历指定页
298     for (i = X; i < X + Width; i ++)  //遍历指定列
299         if (i >= 0 && i <= 127 && j >=0 && j <= 63)
300       OLED_DisplayBuf[j / 8][i] &= ~(0x01 << (j % 8));
301 }
302 
303 /**
304 * 函     数：将OLED显存数组全部取反
305 * 参     数：无
306 * 返 回 值：无
307 * 说     明：调用此函数后，要想真正地呈现在屏幕上，还需调用更新函数
308 */
309 void OLED_Reverse(void)
310 {
311   uint8_t i, j;
312   for (j = 0; j < 8; j ++)         //遍历8页
313   {
314     for (i = 0; i < 128; i ++)       //遍历128列
315     {
316       OLED_DisplayBuf[j][i] ^= 0xFF;  //将显存数组数据全部取反
317     }
318   }
319 }
320 
321 /**
322 * 函     数：将OLED显存数组部分取反
```

```
323 * 参     数：X 指定区域左上角的横坐标，范围：-32768-32767，屏幕区域：0-127
324 * 参     数：Y 指定区域左上角的纵坐标，范围：-32768-32767，屏幕区域：0-63
325 * 参     数：Width 指定区域的宽度，范围：0-128
326 * 参     数：Height 指定区域的高度，范围：0-64
327 * 返 回 值：无
328 * 说     明：调用此函数后，要想真正地呈现在屏幕上，还需调用更新函数
329 */
330 void OLED_ReverseArea(int16_t X, int16_t Y, uint8_t Width, uint8_t Height)
331 {
332   int16_t i, j;
333 
334   for (j = Y; j < Y + Height; j ++)   //遍历指定页
335   {
336     for (i = X; i < X + Width; i ++)  //遍历指定列
337     {
338       if (i >= 0 && i <= 127 && j >=0 && j <= 63)     //超出屏幕的内容不显示
339       {
340         OLED_DisplayBuf[j / 8][i] ^= 0x01 << (j % 8); //将显存数组指定数据取反
341       }
342     }
343   }
344 }
345 
346 /**
347 * 函     数：OLED显示一个字符
348 * 参     数：X 指定字符左上角的横坐标，范围：-32768-32767，屏幕区域：0-127
349 * 参     数：Y 指定字符左上角的纵坐标，范围：-32768-32767，屏幕区域：0-63
350 * 参     数：Char 指定要显示的字符，范围：ASCII码可见字符
351 * 参     数：FontSize 指定字体大小
352 *           范围：OLED_8X16    宽8像素，高16像素
353 *                 OLED_6X8     宽6像素，高8像素
354 * 返 回 值：无
355 * 说     明：调用此函数后，要想真正地呈现在屏幕上，还需调用更新函数
356 */
357 void OLED_ShowChar(int16_t X, int16_t Y, char Char, uint8_t FontSize)
358 {
359   if (FontSize == OLED_8X16)    //字体为宽8像素，高16像素
360   {
361     /*将ASCII字模库OLED_F8x16的指定数据以8*16的图像格式显示*/
362     OLED_ShowImage(X, Y, 8, 16, OLED_F8x16[Char - ' ']);
363   }
364   else if(FontSize == OLED_6X8) //字体为宽6像素，高8像素
365   {
366     /*将ASCII字模库OLED_F6x8的指定数据以6*8的图像格式显示*/
367     OLED_ShowImage(X, Y, 6, 8, OLED_F6x8[Char - ' ']);
368   }
369 }
```

```
370
371 /**
372 * 函    数：OLED显示字符串
373 * 参    数：X 指定字符串左上角的横坐标，范围：-32768-32767，屏幕区域：0-127
374 * 参    数：Y 指定字符串左上角的纵坐标，范围：-32768-32767，屏幕区域：0-63
375 * 参    数：String 指定要显示的字符串，范围：ASCII码可见字符组成的字符串
376 * 参    数：FontSize 指定字体大小
377 *           范围：OLED_8X16    宽8像素，高16像素
378 *                 OLED_6X8     宽6像素，高8像素
379 * 返 回 值：无
380 * 说    明：调用此函数后，要想真正地呈现在屏幕上，还需调用更新函数
381 */
382 void OLED_ShowString(int16_t X, int16_t Y, char *String, uint8_t FontSize)
383 {
384   uint8_t i;
385   for (i = 0; String[i] != '\0'; i++)   //遍历字符串的每个字符
386   {
387     /*调用OLED_ShowChar函数，依次显示每个字符*/
388     OLED_ShowChar(X + i * FontSize, Y, String[i], FontSize);
389   }
390 }
391
392 void OLED_ShowNum(int16_t X, int16_t Y, uint32_t Number, uint8_t Length, uint8_t FontSize)
393 {
394   uint8_t i;
395   for (i = 0; i < Length; i++)    //遍历数字的每一位
396   {
397     /*调用OLED_ShowChar函数，依次显示每个数字*/
398     /*Number / OLED_Pow(10, Length - i - 1) % 10 可以十进制提取数字的每一位*/
399     /*+ '0' 可将数字转换为字符格式*/
400     OLED_ShowChar(X + i * FontSize, Y, Number / OLED_Pow(10, Length - i - 1) % 10 + '0',
          FontSize);
401   }
402 }
403
404 void OLED_ShowSignedNum(int16_t X, int16_t Y, int32_t Number, uint8_t Length, uint8_t
      FontSize)
405 {
406   uint8_t i;
407   uint32_t Number1;
408
409   if (Number >= 0)            //数字大于等于0
410   {
411     OLED_ShowChar(X, Y, '+', FontSize); //显示+号
412     Number1 = Number;         //Number1直接等于Number
413   }
414   else                  //数字小于0
```

```
415   {
416     OLED_ShowChar(X, Y, '-', FontSize); //显示-号
417     Number1 = -Number;           //Number1等于Number取负
418   }
419
420   for (i = 0; i < Length; i++)      //遍历数字的每一位
421   {
422     /*调用OLED_ShowChar函数，依次显示每个数字*/
423     /*Number1 / OLED_Pow(10, Length - i - 1) % 10 可以十进制提取数字的每一位*/
424     /*+ '0' 可将数字转换为字符格式*/
425     OLED_ShowChar(X + (i + 1) * FontSize, Y, Number1 / OLED_Pow(10, Length - i - 1) % 10 + '
          0', FontSize);
426   }
427 }
428
429 void OLED_ShowHexNum(int16_t X, int16_t Y, uint32_t Number, uint8_t Length, uint8_t FontSize
      )
430 {
431   uint8_t i, SingleNumber;
432   for (i = 0; i < Length; i++)    //遍历数字的每一位
433   {
434     /*以十六进制提取数字的每一位*/
435     SingleNumber = Number / OLED_Pow(16, Length - i - 1) % 16;
436
437     if (SingleNumber < 10)       //单个数字小于10
438     {
439       /*调用OLED_ShowChar函数，显示此数字*/
440       /*+ '0' 可将数字转换为字符格式*/
441       OLED_ShowChar(X + i * FontSize, Y, SingleNumber + '0', FontSize);
442     }
443     else               //单个数字大于10
444     {
445       /*调用OLED_ShowChar函数，显示此数字*/
446       /*+ 'A' 可将数字转换为从A开始的十六进制字符*/
447       OLED_ShowChar(X + i * FontSize, Y, SingleNumber - 10 + 'A', FontSize);
448     }
449   }
450 }
451
452 void OLED_ShowBinNum(int16_t X, int16_t Y, uint32_t Number, uint8_t Length, uint8_t FontSize
      )
453 {
454   uint8_t i;
455   for (i = 0; i < Length; i++)    //遍历数字的每一位
456   {
457     /*调用OLED_ShowChar函数，依次显示每个数字*/
458     /*Number / OLED_Pow(2, Length - i - 1) % 2 可以二进制提取数字的每一位*/
```

```
459     /*+ '0' 可将数字转换为字符格式*/
460     OLED_ShowChar(X + i * FontSize, Y, Number / OLED_Pow(2, Length - i - 1) % 2 + '0',
            FontSize);
461   }
462 }
463
464 /**
465 * 函     数: OLED显示浮点数字（十进制, 小数）
466 * 参     数: X 指定数字左上角的横坐标, 范围: -32768-32767, 屏幕区域: 0-127
467 * 参     数: Y 指定数字左上角的纵坐标, 范围: -32768-32767, 屏幕区域: 0-63
468 * 参     数: Number 指定要显示的数字, 范围: -4294967295.0-4294967295.0
469 * 参     数: IntLength 指定数字的整数位长度, 范围: 0-10
470 * 参     数: FraLength 指定数字的小数位长度, 范围: 0-9, 小数进行四舍五入显示
471 * 参     数: FontSize 指定字体大小
472 */
473 void OLED_ShowFloatNum(int16_t X, int16_t Y, double Number, uint8_t IntLength, uint8_t
        FraLength, uint8_t FontSize)
474 {
475   uint32_t PowNum, IntNum, FraNum;
476
477   if (Number >= 0) OLED_ShowChar(X, Y, '+', FontSize);
478   else
479   {
480     OLED_ShowChar(X, Y, '-', FontSize);
481     Number = -Number;
482   }
483
484   /*提取整数部分和小数部分*/
485   IntNum = Number;
486   Number -= IntNum;
487   PowNum = OLED_Pow(10, FraLength);
488   FraNum = round(Number * PowNum);
489   IntNum += FraNum / PowNum;
490
491   /*显示整数部分*/
492   OLED_ShowNum(X + FontSize, Y, IntNum, IntLength, FontSize);
493
494   /*显示小数点*/
495   OLED_ShowChar(X + (IntLength + 1) * FontSize, Y, '.', FontSize);
496
497   /*显示小数部分*/
498   OLED_ShowNum(X + (IntLength + 2) * FontSize, Y, FraNum, FraLength, FontSize);
499 }
500
501 /**
502 * 函     数: OLED在指定位置画一个点
503 * 参     数: X 指定点的横坐标, 范围: -32768-32767, 屏幕区域: 0-127
```

```
504 * 参     数：Y 指定点的纵坐标，范围：-32768-32767，屏幕区域：0-63
505 * 返 回 值：无
506 */
507 void OLED_DrawPoint(int16_t X, int16_t Y)
508 {
509   if (X >= 0 && X <= 127 && Y >=0 && Y <= 63)   //超出屏幕的内容不显示
510   {
511     /*将显存数组指定位置的一个Bit数据置1*/
512     OLED_DisplayBuf[Y / 8][X] |= 0x01 << (Y % 8);
513   }
514 }
515
516 /**
517 * 函     数：OLED获取指定位置点的值
518 * 参     数：X 指定点的横坐标，范围：-32768-32767，屏幕区域：0-127
519 * 参     数：Y 指定点的纵坐标，范围：-32768-32767，屏幕区域：0-63
520 * 返 回 值：指定位置点是否处于点亮状态，1：点亮，0：熄灭
521 */
522 uint8_t OLED_GetPoint(int16_t X, int16_t Y)
523 {
524   if (X >= 0 && X <= 127 && Y >=0 && Y <= 63)   //超出屏幕的内容不读取
525     /*判断指定位置的数据*/
526     if (OLED_DisplayBuf[Y / 8][X] & 0x01 << (Y % 8)) return 1;
527
528   return 0;
529 }
```

Listing 3: OLED.h

```
1   #ifndef __OLED_H
2   #define __OLED_H
3
4   #include <stdint.h>
5   #include <string.h>
6   #include <math.h>
7   #include <stdio.h>
8   #include <stdarg.h>
9   #include "driver/i2c_master.h"
10  #include "OLED_Data.h"
11
12  #define OLED_8X16 8
13  #define OLED_6X8 6
14
15  /*IsFilled参数数值*/
16  #define OLED_UNFILLED 0
17  #define OLED_FILLED 1
18
19  /*初始化函数*/
```

```
20  void OLED_Init(int port, uint8_t add, int scl, int sda, int speed);
21
22  /*更新函数*/
23  void OLED_Update(void);
24  void OLED_UpdateArea(int16_t X, int16_t Y, uint8_t Width, uint8_t Height);
25
26  /*显存控制函数*/
27  void OLED_Clear(void);
28  void OLED_ClearArea(int16_t X, int16_t Y, uint8_t Width, uint8_t Height);
29  void OLED_Reverse(void);
30  void OLED_ReverseArea(int16_t X, int16_t Y, uint8_t Width, uint8_t Height);
31
32  /*显示函数*/
33  void OLED_ShowChar(int16_t X, int16_t Y, char Char, uint8_t FontSize);
34  void OLED_ShowString(int16_t X, int16_t Y, char *String, uint8_t FontSize);
35  void OLED_ShowNum(int16_t X, int16_t Y, uint32_t Number, uint8_t Length, uint8_t FontSize)
        ;
36  void OLED_ShowSignedNum(int16_t X, int16_t Y, int32_t Number, uint8_t Length, uint8_t
        FontSize);
37  void OLED_ShowHexNum(int16_t X, int16_t Y, uint32_t Number, uint8_t Length, uint8_t
        FontSize);
38  void OLED_ShowBinNum(int16_t X, int16_t Y, uint32_t Number, uint8_t Length, uint8_t
        FontSize);
39  void OLED_ShowFloatNum(int16_t X, int16_t Y, double Number, uint8_t IntLength, uint8_t
        FraLength, uint8_t FontSize);
40
41  #endif
```

附录 C　dht 数据收集程序

Listing 4: dht.c

```
#include "dht.h"

#include <freertos/FreeRTOS.h>
#include <freertos/task.h>
#include <string.h>
#include <esp_log.h>
#include <esp_rom_sys.h>
#include <driver/gpio.h>

// DHT timer precision in microseconds
#define DHT_TIMER_INTERVAL 2
#define DHT_DATA_BITS 40
#define DHT_DATA_BYTES (DHT_DATA_BITS / 8)

static const char *TAG = "dht";

static portMUX_TYPE mux = portMUX_INITIALIZER_UNLOCKED;
#define PORT_ENTER_CRITICAL() portENTER_CRITICAL(&mux)
#define PORT_EXIT_CRITICAL() portEXIT_CRITICAL(&mux)

#define CHECK_ARG(VAL)            \
do                                \
{                                 \
  if (!(VAL))                     \
  return ESP_ERR_INVALID_ARG; \
} while (0)

#define CHECK_LOGE(x, msg, ...)          \
do                                       \
{                                        \
  esp_err_t __err = (x);                 \
  if (__err != ESP_OK)                   \
  {                                      \
    PORT_EXIT_CRITICAL();                \
    ESP_LOGE(TAG, msg, ##__VA_ARGS__); \
    return __err;                        \
  }                                      \
} while (0)

static esp_err_t dht_await_pin_state(gpio_num_t pin, uint32_t timeout, int
    expected_pin_state, uint32_t *duration)
{
  gpio_set_direction(pin, GPIO_MODE_INPUT);
```

```
43   for (uint32_t i = 0; i < timeout; i += DHT_TIMER_INTERVAL)
44   {
45     esp_rom_delay_us(DHT_TIMER_INTERVAL);
46     if (gpio_get_level(pin) == expected_pin_state)
47     {
48       if (duration)
49       *duration = i;
50       return ESP_OK;
51     }
52   }
53   return ESP_ERR_TIMEOUT;
54 }
55 
56 static esp_err_t dht_fetch_data(dht_sensor_type_t sensor_type, gpio_num_t pin, uint8_t data[
       DHT_DATA_BYTES])
57 {
58   uint32_t low_duration, high_duration;
59 
60   gpio_set_direction(pin, GPIO_MODE_OUTPUT_OD);
61   gpio_set_level(pin, 0);
62   esp_rom_delay_us(sensor_type == DHT_TYPE_SI7021 ? 500 : 20000);
63   gpio_set_level(pin, 1);
64 
65   CHECK_LOGE(dht_await_pin_state(pin, 40, 0, NULL), "Phase 'B' timeout");
66   CHECK_LOGE(dht_await_pin_state(pin, 88, 1, NULL), "Phase 'C' timeout");
67   CHECK_LOGE(dht_await_pin_state(pin, 88, 0, NULL), "Phase 'D' timeout");
68 
69   for (int i = 0; i < DHT_DATA_BITS; i++)
70   {
71     CHECK_LOGE(dht_await_pin_state(pin, 65, 1, &low_duration), "Low duration timeout");
72     CHECK_LOGE(dht_await_pin_state(pin, 75, 0, &high_duration), "High duration timeout");
73 
74     uint8_t byte_index = i / 8;
75     uint8_t bit_index = i % 8;
76     if (!bit_index)
77     data[byte_index] = 0;
78 
79     data[byte_index] |= (high_duration > low_duration) << (7 - bit_index);
80   }
81 
82   return ESP_OK;
83 }
84 
85 static inline int16_t dht_convert_data(dht_sensor_type_t sensor_type, uint8_t msb, uint8_t
       lsb)
86 {
87   int16_t result;
```

```
88    if (sensor_type == DHT_TYPE_DHT11)
89    {
90      result = msb * 10;
91    }
92    else
93    {
94      result = (msb & 0x7F) << 8 | lsb;
95      if (msb & 0x80)
96      result = -result;
97    }
98    return result;
99  }
100
101 esp_err_t dht_read_data(dht_sensor_type_t sensor_type, gpio_num_t pin, int16_t *humidity,
        int16_t *temperature)
102 {
103   CHECK_ARG(humidity || temperature);
104
105   uint8_t data[DHT_DATA_BYTES] = {0};
106
107   gpio_set_direction(pin, GPIO_MODE_OUTPUT_OD);
108   gpio_set_level(pin, 1);
109
110   PORT_ENTER_CRITICAL();
111   esp_err_t result = dht_fetch_data(sensor_type, pin, data);
112   PORT_EXIT_CRITICAL();
113
114   gpio_set_direction(pin, GPIO_MODE_OUTPUT_OD);
115   gpio_set_level(pin, 1);
116
117   if (result != ESP_OK)
118   return result;
119
120   if (data[4] != ((data[0] + data[1] + data[2] + data[3]) & 0xFF))
121   {
122     ESP_LOGE(TAG, "Checksum failed");
123     return ESP_ERR_INVALID_CRC;
124   }
125
126   if (humidity)
127   *humidity = dht_convert_data(sensor_type, data[0], data[1]);
128   if (temperature)
129   *temperature = dht_convert_data(sensor_type, data[2], data[3]);
130
131   ESP_LOGD(TAG, "Humidity: %d, Temperature: %d", *humidity, *temperature);
132
133   return ESP_OK;
```

```
134 }
135
136 esp_err_t dht_read_float_data(dht_sensor_type_t sensor_type, gpio_num_t pin, float *humidity
        , float *temperature)
137 {
138   CHECK_ARG(humidity || temperature);
139
140   int16_t int_humidity, int_temperature;
141   esp_err_t result = dht_read_data(sensor_type, pin, humidity ? &int_humidity : NULL,
          temperature ? &int_temperature : NULL);
142   if (result != ESP_OK)
143   return result;
144
145   if (humidity)
146   *humidity = int_humidity / 10.0f;
147   if (temperature)
148   *temperature = int_temperature / 10.0f;
149
150   return ESP_OK;
151 }
```

Listing 5: dht.h

```
1   /**
2   Copyright 2024 Achim Pieters | StudioPieters®
3
4   Permission is hereby granted, free of charge, to any person obtaining a copy
5   of this software and associated documentation files (the "Software"), to deal
6   in the Software without restriction, including without limitation the rights
7   to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
8   copies of the Software, and to permit persons to whom the Software is
9   furnished to do so, subject to the following conditions:
10
11  The above copyright notice and this permission notice shall be included in all
12  copies or substantial portions of the Software.
13
14  THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
15  IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
16  FITNESS FOR A PARTICULAR PURPOSE AND NON INFRINGEMENT. IN NO EVENT SHALL THE
17  AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
18  WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
19  CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
20
21  for more information visit https://www.studiopieters.nl
22  **/
23
24  #ifndef __DHT_H__
25  #define __DHT_H__
```

```
26
27  #include <driver/gpio.h>
28  #include <esp_err.h>
29
30  #ifdef __cplusplus
31  extern "C" {
32    #endif
33
34    typedef enum
35    {
36      DHT_TYPE_DHT11 = 0,
37      DHT_TYPE_AM2301,
38      DHT_TYPE_SI7021
39    } dht_sensor_type_t;
40
41    esp_err_t dht_read_data(dht_sensor_type_t sensor_type, gpio_num_t pin, int16_t *humidity
          , int16_t *temperature);
42    esp_err_t dht_read_float_data(dht_sensor_type_t sensor_type, gpio_num_t pin, float *
          humidity, float *temperature);
43
44    #ifdef __cplusplus
45  }
46  #endif
47
48  #endif  // __DHT_H__
```

附录 D　L9110H 电机驱动程序

Listing 6: mydht11.c

```
#include <stdio.h>
#include "mydht11.h"
#include "freertos/FreeRTOS.h"
#include "esp_log.h" // 确保包含日志头文件
#include <driver/gpio.h>
#include <driver/ledc.h>

#define PWM_CHANNEL LEDC_CHANNEL_0
#define PWM_TIMER LEDC_TIMER_0

static const char *TAG = "L9110_PWM_Control";
void l9110_pwm_init(void)
{
  gpio_config_t io_conf;
  io_conf.intr_type = GPIO_INTR_DISABLE;
  io_conf.mode = GPIO_MODE_OUTPUT;
  io_conf.pin_bit_mask = (1ULL << IN2_PIN);
  io_conf.pull_down_en = 0;
  io_conf.pull_up_en = 0;
  gpio_config(&io_conf);

  gpio_set_level(IN2_PIN,0);
  // 配置 PWM 输出（用于 IN1）
  ledc_timer_config_t ledc_timer = {
    .speed_mode       = LEDC_LOW_SPEED_MODE,
    .timer_num        = PWM_TIMER,
    .duty_resolution  = LEDC_TIMER_8_BIT,  // 分辨率：0~255
    .freq_hz          = 1000,              // PWM 频率 1kHz
    .clk_cfg          = LEDC_AUTO_CLK,
  };
  ledc_timer_config(&ledc_timer);

  ledc_channel_config_t ledc_channel = {
    .channel    = PWM_CHANNEL,
    .duty       = 0,
    .gpio_num   = PWM_PIN,
    .speed_mode = LEDC_LOW_SPEED_MODE,
    .hpoint     = 0,
    .timer_sel  = PWM_TIMER,
  };
  ledc_channel_config(&ledc_channel);

  ESP_LOGI(TAG, "L9110 PWM 初始化完成");
```

```
44  }
45
46  // 设置电机转速（speed: 0~255）
47  void set_motor_speed(uint8_t speed)
48  {
49    if(speed > 255) speed = 255;
50    ledc_set_duty(LEDC_LOW_SPEED_MODE, PWM_CHANNEL, speed);
51    ledc_update_duty(LEDC_LOW_SPEED_MODE, PWM_CHANNEL);
52    ESP_LOGI(TAG, "设置电机速度: %d", speed);
53  }
```

Listing 7: mydht11.h

```
1   #include <stdio.h>
2   #include "freertos/FreeRTOS.h"
3   #include "esp_log.h" // 确保包含日志头文件
4   #include <driver/gpio.h>
5   #include <driver/ledc.h>
6
7   #define IN1_PIN 7   // 连接到 L9110 的 IN1
8   #define IN2_PIN 6   // 连接到 L9110 的 IN2
9   #define PWM_PIN 7   // 使用 IN1 引脚作为 PWM 输出
10  #define PWM_CHANNEL LEDC_CHANNEL_0
11  #define PWM_TIMER LEDC_TIMER_0
12
13  void l9110_pwm_init(void);
14  void set_motor_speed(uint8_t speed);
```

附录 E FuzzyControl 程序

Listing 8: TMF.c

```c
#include <stdio.h>
#include <math.h>

// 定义模糊集合
typedef struct
{
  float low;
  float medium;
  float high;
} FuzzyMembership;

// 三角隶属度函数x: 当前输入值, 用于计算该点的隶属度;
// a: 三角形隶属度函数的左边界;
// b: 三角形的顶点位置（最大值为 1）;
// c: 三角形的右边界。
float triangle(float x, float a, float b, float c)
{
  if (x <= a || x >= c) return 0.0;
  else if (x > a && x <= b) return (x - a) / (b - a);
  else return (c - x) / (c - b);
}

// 模糊化函数: 误差
FuzzyMembership fuzzify_error(float error)
{
  FuzzyMembership membership = {0.0, 0.0, 0.0};

  membership.high = triangle(error, -7.0, -7.0, 0.0);
  membership.medium = triangle(error, -3.0, 0.0, 3.0);
  membership.low = triangle(error, 0.0, 7.0, 7.0);

  return membership;
}
// 模糊化函数: 误差变化
FuzzyMembership fuzzify_delta_error(float delta_error)
{
  FuzzyMembership membership = {0.0, 0.0, 0.0};

  membership.high = triangle(delta_error, -2.0, -2, 0.0);
  membership.medium = triangle(delta_error, -1.0, 0.0, 1.0);
  membership.low = triangle(delta_error, 0.0, 2, 2.0);

  return membership;
```

```
44  }
45
46  // 模糊规则推理
47  FuzzyMembership infer_control(FuzzyMembership error_membership, FuzzyMembership
        delta_error_membership)
48  {
49    FuzzyMembership control_membership = {0.0, 0.0, 0.0};
50
51    // 规则 1: 如果误差低且误差变化低, 则输出低
52    control_membership.low = fmax(control_membership.low, fmax(error_membership.low,
          delta_error_membership.low));
53
54    // 规则 2: 如果误差中等且误差变化中等, 则输出中等
55    control_membership.medium = fmax(control_membership.medium, fmax(error_membership.medium
          , delta_error_membership.medium));
56
57    // 规则 3: 如果误差高且误差变化高, 则输出高
58    control_membership.high = fmax(control_membership.high, fmax(error_membership.high,
          delta_error_membership.high));
59
60    // 规则 4: 如果误差高且误差变化低, 则输出高
61    control_membership.high = fmax(control_membership.high, fmax(error_membership.high,
          delta_error_membership.low));
62
63    // 规则 5: 如果误差低且误差变化高, 则输出低
64    control_membership.low = fmax(control_membership.low, fmax(error_membership.low,
          delta_error_membership.high));
65
66    return control_membership;
67  }
68  // 去模糊化函数
69  float defuzzify(FuzzyMembership control_membership)
70  {
71    float numerator = 0.0;
72    float denominator = 0.0;
73
74    // 低功率部分
75    for (float x = 0.0; x <= 33.3; x += 1.0)
76    {
77      float membership = fmin(control_membership.low, triangle(x, 0.0, 16.65, 33.3));
78      numerator += x * membership;
79      denominator += membership;
80    }
81
82    // 中等功率部分
83    for (float x = 33.3; x <= 66.6; x += 1.0)
84    {
```

```
85        float membership = fmax(control_membership.medium, triangle(x, 33.3, 50.0, 66.6));
86        numerator += x * membership;
87        denominator += membership;
88      }
89
90      // 高功率部分
91      for (float x = 66.6; x <= 100.0; x += 1.0)
92      {
93        float membership = fmax(control_membership.high, triangle(x, 66.6, 83.3, 100.0));
94        numerator += x * membership;
95        denominator += membership;
96      }
97
98      // 如果分母为 0，返回 0；否则返回比例缩放后的值
99      float output = (denominator == 0.0) ? 0.0 : (numerator / denominator);
100
101     // 将输出从 [0, 100] 映射到 [0, 255]
102     return output * 255.0 / 100.0;
103   }
104
105   // 主函数
106   void func(void)
107   {
108     float target_temperature = 25.0;  // 标准温度（目标温度）
109     float current_temperature = 22.0; // 当前温度（假设值）
110     static float last_error = 0.0;    // 上一次误差（初始为 0）
111
112     // 计算误差和误差变化
113     float error = target_temperature - current_temperature;
114     float delta_error = error - last_error;
115
116     // 模糊化
117     FuzzyMembership error_membership = fuzzify_error(error);
118     FuzzyMembership delta_error_membership = fuzzify_delta_error(delta_error);
119
120     // 模糊推理
121     FuzzyMembership control_membership = infer_control(error_membership,
          delta_error_membership);
122
123     // 去模糊化
124     float output = defuzzify(control_membership);
125
126     // 更新上一次误差
127     last_error = error;
128
129     // 输出结果
130     printf("Target Temp: %.1f, Current Temp: %.1f, Error: %.1f, Delta Error: %.1f, Control
```

```
      Output: %.1f\n",
131   target_temperature, current_temperature, error, delta_error, output);
132 }
```

Listing 9: TMF.h

```
1  #define TargetTemp 25.00
2
3  // 定义模糊集合
4  typedef struct {
5    float low;
6    float medium;
7    float high;
8  } FuzzyMembership;
9
10 FuzzyMembership fuzzify_temperature(float temperature);
11 float defuzzify(FuzzyMembership control_membership);
12 void fuzzyControl(float temperature,float e,float deltae);
13 FuzzyMembership infer_control(FuzzyMembership error_membership, FuzzyMembership
       delta_error_membership);
14 FuzzyMembership fuzzify_delta_error(float delta_error);
15 FuzzyMembership fuzzify_error(float error);
16 float triangle(float x, float a, float b, float c);
```