相关代码: configs\environmental-sensors\bmp_aht_esp32c3-espnow.yaml configs\environmental-sensors\voc-esp32c3-espnow.yaml configs\displays\display9341-espnow.yaml configs\displays\envpanel-espnow-c3.yaml

基于 ESP-NOW 的环境监测可视化终端设计与实现（参考文档）

摘要

本文实现了一套基于 **ESP32-C3** 的环境监测可视化终端：上游节点通过 **ESP-NOW** 发送温湿度、气压、TVOC、甲醛、CO_2 等数据；下游接收端在 **ILI9341 SPI 彩屏（240×320）** 上实时显示。通过分离采集与显示节点，采集端可在无网环境独立运行，在不同地点采集数据并统一在中心展示。界面提供**数据新鲜度提示**与**阈值分级着色**。

系统同时集成 **SNTP / Home Assistant 时间同步** 与 **MQTT 上行**。后端网站（https://iot2.14790897.xyz/）对 MQTT 上传进行 Token 鉴权，鉴权通过后将数据写入 PostgreSQL，前端按需读取展示。方案兼具**低时延、低功耗、免配网**与**离线可用（ESP-NOW 本地通信）**的特点，适用于家庭与办公场景的本地环境质量看板。

关键词： ESP32-C3；ESP-NOW；ESPHome；ILI9341；环境监测；本地可视化

---

0. 快速上手（最小可用示例）

以下示例基于仓库现有工程裁剪而来，便于快速验证链路。

0.1 接收端（ESP32-C3 + ILI9341）最小示例

```yaml
esphome:
  name: envpanel-espnow-c3

esp32:
  board: airm2m_core_esp32c3
  framework:
    type: esp-idf

wifi:
  ssid: !secret wifi_ssid
  password: !secret wifi_password

logger:
api:
captive_portal:

time:
  - platform: sntp
    id: sntp_time
    servers: [ntp.aliyun.com, time.windows.com]
    timezone: Asia/Shanghai

espnow:
```

```
  id: espnow_component
  channel: 1            # 与发送端一致
  auto_add_peer: true
  on_broadcast:
    then:
      - lambda: |-
          // 支持 JSON 载荷（推荐）与 12/24 字节二进制帧（兼容）
          if (size >= 2 && data[0] == '{' && data[size-1] == '}') {
            std::string json(reinterpret_cast<const char*>(data), size);
            ESP_LOGI("espnow", "JSON: %s", json.c_str());
          }

globals:
  - id: last_update_time
    type: unsigned long
    initial_value: '0'

packages:
  display9341: !include ../displays/display9341-espnow.yaml
```

完整实现见 `configs/displays/envpanel-espnow-c3.yaml:1`。

0.2 发送端（VOC/CO_2/甲醛）最小示例

```
esphome:
  name: voc-esp32c3-espnow

esp32:
  board: airm2m_core_esp32c3
  framework:
    type: arduino

espnow:
  id: espnow_component
  channel: 1
  auto_add_peer: true

globals:
  - id: last_espnow_payload
    type: std::string
    initial_value: '"{}"'

script:
  - id: send_payload
    then:
      - espnow.broadcast:
          id: espnow_component
          data: !lambda |-
            const std::string &p = id(last_espnow_payload);
            return std::vector<uint8_t>(p.begin(), p.end());

interval:
```

```
  - interval: 10s
    then:
      - lambda: |-
          // 构造 JSON（示例值）
          std::string json = "{";
          json += "\"tvoc\":0.123,";
          json += "\"formaldehyde\":0.045,";
          json += "\"co2\":0.800"; // mg/m³
          json += "}";
          id(last_espnow_payload) = json;
      - script.execute: send_payload
```

完整实现见 `configs/environmental-sensors/voc-esp32c3-espnow.yaml:1`。

---

1. 系统总体设计

1.1 功能目标

- 无线接收（ESP-NOW）多种环境传感器数据（温度、湿度、气压、TVOC、甲醛、CO_2）。
- 在 2.4" ILI9341 彩屏上实时展示，并按阈值进行颜色提示。
- 显示数据新鲜度（近 60 s 为“在线”，超时提示“数据超时”）。
- 同步时间（SNTP 或通过 Home Assistant），并可选 MQTT 上行。

1.2 架构与数据流

```
传感器节点(ESP-NOW 发送端) → 2.4GHz广播
                    ↓
              ESP32-C3 接收端
  (解析ESP-NOW帧→更新全局量/传感器实体→显示/MQTT)
                    ↓
          ILI9341 屏幕本地可视化
```

- 上游：任意 ESP-NOW 发送器（如 C3/C6/S2）打包并发送结构化数据帧。
- 下游：ESP32-C3 接收端解析并更新到对应的 `sensor`/`globals`，触发显示刷新。
- 联网：将接收到的数据序列化为 JSON 上行 MQTT（项目中已预留/示例化）。

---

2. 硬件设计

2.1 处理与通信

- 主控：ESP32-C3（RISC-V，2.4 GHz）
- 无线：ESP-NOW（2.4 GHz，低延迟、免配对的轻量级广播/点对点）

2.2 显示子系统

- 控制器：ILI9341（SPI）
- 接口：`clk_pin: GPIO2`，`mosi_pin: GPIO3`，`cs_pin: GPIO7`，`dc_pin: GPIO6`，`reset_pin: GPIO10`

- 分辨率：240×320，`data_rate: 10MHz`（SPI）

---

3. 软件设计（ESPHome）

项目采用 **ESPHome** 进行固件描述与生成，主要包含两部分：（1）接收端主工程：`configs/displays/envpanel-espnow-c3.yaml`（2）显示布局与组件封装：`configs/displays/display9341-espnow.yaml`

3.1 时间同步

- SNTP（C3 直连 NTP 服务器）

```
time:
  - platform: sntp
    id: sntp_time
    servers:
      - ntp.aliyun.com
      - time.windows.com
    timezone: Asia/Shanghai
```

3.2 SPI 与 ILI9341 显示

`display9341-espnow.yaml` 中完成 SPI 与屏幕平台配置，并在 `lambda` 中绘制 UI。

```
spi:
  clk_pin: GPIO2
  mosi_pin: GPIO3

display:
  - platform: mipi_spi
    model: ILI9341
    cs_pin: GPIO7
    dc_pin: GPIO6
    reset_pin: GPIO10
    data_rate: 10MHz
    rotation: 0
    invert_colors: false
    dimensions: 240x320
    lambda: |-
      it.fill(Color::BLACK);
      // 布局参数与字体
      // 读取各传感器状态并打印
      // 根据阈值设置颜色
      // 显示“数据是否超时”的新鲜度提示
      // 显示时钟
```

常见阈值着色示例（片段）：

```
auto color_by_range = [&](float v, float lo_g, float hi_g, float lo_r, float hi_r)
{
  if (v < lo_r || v > hi_r) return Color(255,0,0);   // 红
  if (v >= lo_g && v <= hi_g) return Color(0,255,0);  // 绿
  return Color(255,255,0);                            // 黄
};
it.printf(8, 64, font_main, color_by_range(id(received_temperature), 18, 26, 16,
30),
          "温度: %.1f°C", id(received_temperature));
```

3.2.1 字体与中文显示

```
font:
  - file: "../../resources/static/NotoSansSC-Regular.ttf"
    id: font_main
    size: 16
    bpp: 1
    glyphs: "0123456789.%°Cmg/hPa³TVO甲醛环境监测传感器数据在线离温湿气压更新秒前已有
超时度连接未:- slESNWp"
```

> 说明：`glyphs` 字符集应覆盖界面上使用的所有汉字与符号，减少字库体积、避免乱码。

3.3 UI 与告警阈值

阈值区间示例：

- 温度：<16 或 >30 → 红；18–26 → 绿；其余 → 黄
- 湿度：<30 或 >70 → 红；40–60 → 绿；其余 → 黄
- 气压：950–1050 hPa → 青；超范围 → 黄
- TVOC：<0.3 mg/m³ → 绿；0.3–0.6 → 黄；≥0.6 → 红
- 甲醛：<0.08 mg/m³ → 绿；0.08–0.10 → 黄；≥0.10 → 红
- CO_2：内部以 mg/m³ 存储，显示时 ×1000 为 ppm

3.4 数据新鲜度判定

通过最后一次数据更新时间 `last_update_time` 与 `millis()` 的差值判断：

- < 60 s → “更新: Xs 前”
- ≥ 60 s → “数据超时”（红色）

这能直观判断上游 ESP-NOW 数据链路是否健康。

也可将其暴露为二进制传感器，便于 HA 订阅与联动（见接收端实现）：

```
binary_sensor:
  - platform: template
    name: "ESP-NOW Data Fresh"
    id: data_fresh
```

```
    lambda: |-
      if (id(last_update_time) == 0) return false;
      return (millis() - id(last_update_time)) < 60000;
    filters:
      - delayed_on: 1s
      - delayed_off: 5s
```

显示层判定片段（便于复用）：

```
const bool data_fresh = (millis() - id(last_update_time)) < 60000;
if (data_fresh) {
  const unsigned long seconds_ago = (millis() - id(last_update_time)) / 1000;
  it.printf(8, 200, time_font, Color(200,200,200), "更新: %lus 前", seconds_ago);
} else {
  it.print(8, 200, time_font, Color(255,0,0), "数据超时");
}
```

3.5 ESP-NOW 数据接收与解析（接收端）

接收端在 `envpanel-espnow-c3.yaml` 中：

- 维护多个全局变量（如 `received_temperature`、`received_humidity` 等）。
- 在 ESP-NOW 回调中解析并更新这些量，同时更新时间戳 `last_update_time`。
- 这些变量再通过 `sensor.template` 或直接在 `display.lambda` 中读取显示。

3.5.1 帧格式

- JSON 文本：字段名直观、易扩展。示例字段：`tvoc`、`formaldehyde`、`co2`、`timestamp`、`air_quality` 等。
-

接收端先判断首尾是否为 `{/}`，若是则按 JSON 处理

3.6 MQTT 上行（JSON 序列化）

项目中示例了将收到的数据序列化为 JSON，并（在需要时）上报至 MQTT 服务器（片段）：

```
DynamicJsonDocument doc(256);
auto params = doc["params"].to<JsonObject>();
auto timestamp = id(sntp_time).now().timestamp; // 或 ha_time

if (!isnan(id(received_temperature))) { auto t =
params["temperature"].to<JsonObject>(); t["value"]=id(received_temperature);
t["time"]=timestamp; }
// humidity / pressure / tvoc / formaldehyde...
if (!isnan(id(received_co2)) && id(received_co2) > 0.0f) {
  auto c = params["co2"].to<JsonObject>();
  c["value"] = id(received_co2) * 1000; // mg/m³ → ppm
  c["time"] = timestamp;
}
```

```
serializeJson(doc, json_msg);
ESP_LOGI("mqtt_upload", "MQTT数据上传: %s", json_msg.c_str());
```

MQTT 基础配置示例（TLS）：

```
mqtt:
  broker: !secret mqtt_broker
  port: 8883
  username: !secret mqtt_username
  password: !secret mqtt_password
  certificate_authority: !secret certificate
  on_connect:
    - lambda: 'ESP_LOGI("mqtt", "MQTT连接成功");'
```

---

4. 可靠性与可维护性设计

4.1 链路健康性

- 通过“数据新鲜度”及时提示上游掉线。
- 在 UI 增加 Wi-Fi / HA 连接状态，便于定位是本地显示掉线还是上游中断。

4.2 容错与越界处理

- 代码中对 `NaN` 与异常值做了分支显示，避免 UI 崩溃。
- CO_2 单位换算时防止负值/无效值显示。

4.3 实时性能

- ESP-NOW 接收为事件驱动，屏幕刷新在主循环绘制，10 MHz SPI 数据率可覆盖 1 Hz 级别刷新。

---

5. 安全性与隐私

- ESP-NOW 在本地 2.4 GHz 下广播/配对通信，默认不出公网，降低隐私泄露风险。
- MQTT 采用 TLS 以确保链路安全。

---

8. 关键配置摘录（便于论文引用）

8.1 字体与中文字符集

```
font:
  - file: "../../resources/static/NotoSansSC-Regular.ttf"
    id: font_main
    size: 16
    bpp: 1
    glyphs: "0123456789.%°Cmg/hPa³TVO甲醛环境监测传感器数据在线离温湿气压更新秒前已有
超时度连接未:- slESNWp"
```

8.2 时间源

```
time:
  - platform: sntp
    id: sntp_time
    servers: [ntp.aliyun.com, time.windows.com]
    timezone: Asia/Shanghai
```

8.3 数据新鲜度判断（显示层摘录思路）

```
const bool data_fresh = (millis() - id(last_update_time)) < 60000;
if (data_fresh) {
  const unsigned long seconds_ago = (millis() - id(last_update_time)) / 1000;
  it.printf(8, 200, time_font, Color(200,200,200), "更新: %lus 前", seconds_ago);
} else {
  it.print(8, 200, time_font, Color(255,0,0), "数据超时");
}
```

---

9. 结论

本文基于 ESP32-C3 与 ESPHome 实现了一个低耦合、低延迟、易部署的本地环境监测可视化终端。利用 ESP-NOW 做上游链路，使系统摆脱路由器/云依赖；通过 ILI9341 屏幕与阈值着色提升可读性；以时间同步与数据新鲜度提示提高可靠性与可维护性。实验表明，该方案在家庭与办公室场景下具有良好的稳定性与扩展潜力。