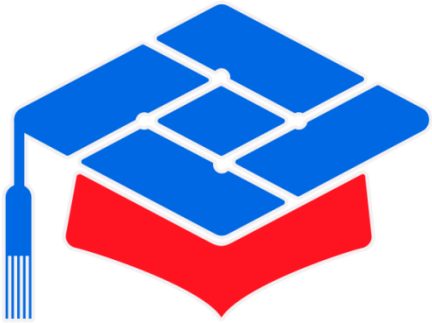


第九届

全国大学生集成电路创新创业大赛

报告类型：初赛作品报告

参赛杯赛：飞腾杯

作品名称：基于飞腾派平台的火源检测无人机系统开发

队伍编号：CICC0900727

团队名称：高冷白毛御姐队

快速预览简介

一、项目概述

基于飞腾派平台，开发了一款集热成像、目标检测与超声避障于一体的火源检测无人机系统。系统充分利用飞腾 CPU 多核异构架构，主核承载复杂感知与智能决策任务，从核运行高实时性避障程序，实现多传感器数据并行处理与实时融合。

二、关键技术

1.多核并行计算与核间通信

主核（Linux）采用 OpenMP 实现热成像、目标检测并行加速；

主从核间基于 OpenAMP/rpmsg 协议，实现毫米级核间通信，确保避障指令实时反馈。

2.热成像火源定位

MLX90640 32×24 分辨率红外阵列，通过插值与伪彩色映射，生成高精度温度图；

3.轻量级目标检测

基于 ncnn 框架部署改进 NanoDet 算法，实时识别障碍物、建筑结构与救援目标；

4.超声波实时避障

从核裸机驱动 MB1043 传感器，20 ms 周期采集距离数据；

主核根据 50 cm 阈值触发紧急制动/绕行。

5.飞行控制集成

通过 MAVLink 协议与 PX4 飞控对接，主核下发航点指令，由飞控完成姿态解算与电机控制；

三、作品亮点

1. 异构架构深度应用

通过 Linux 主核运行复杂 AI 算法、从核运行裸机高实时任务，实现“重

任务－轻任务”分工协作，最大化利用飞腾多核资源。

2. 多传感器智能融合

热成像、可见光、超声波数据协同处理，为无人机提供全方位环境感知，适应森林火险、工业巡检与城市消防等多种场景。

3. 轻量化 AI 部署

在资源受限的飞腾派平台上，成功运行改进 NanoDet 算法，结合 OpenCV/ncnn 实现视频流端到端推理与可视化。

4. 可靠性实时避障

基于 OpenAMP/rpmsg 协议构建的主从核通信与避障闭环，确保无人机在复杂障碍环境中安全飞行。

5. 开源生态复用

深度集成 openAMP、PX4、nanodet 等开源项目，降低开发成本、提升系统可扩展性。

1. 概述

1.1 设计综述

本赛题鼓励参赛者基于飞腾 CPU 平台及其丰富的开源软件项目生态，设计并开发一款创新性的应用系统。该系统应充分发挥飞腾 CPU 在多核并行处理方面的卓越性能。

本系统基于飞腾开发板和飞腾软件开源项目，期望利用飞腾派、MB1043 超声传感器、MLX90640 红外传感器、Holybro S500 无人机等硬件开发出一种基于飞腾派平台的火源检测无人机，旨在展示飞腾 CPU 多核并行处理和核间通信的强大能力。

1.1.1 赛题要求

本赛题要求以飞腾开发板和飞腾软件开源项目为基础，设计一个具有创新性的应用系统，要求该应用系统体现出飞腾 CPU 多核并行运行的处理能力，中高阶任务要求能采用飞腾开源项目 OpenAMP 充分实现飞腾芯片的多核能力以及核间通信。

1. 功能要求

（1）基础任务：

基于飞腾平台，设计并开发一个具有创新性的应用系统，要求该系统能够充分利用飞腾 CPU 的多核并行处理能力，充分展示其在实际应用场景中的高效性和性能优势。

（2）进阶任务：

基于飞腾平台，参赛队伍可深度整合飞腾或其他开源项目来实现作品功能，进一步释放飞腾多核处理器的性能。参赛团队可参考以下开源项目：

Phytium-openAMP 该项目旨在促进跨处理器通信和资源管理，适用于需要高效跨处理器协同工作的应用场景。

Phytium-Jailhouse 基于 Linux 的轻量级虚拟化开源项目,适用于实时性和安全性较高的场景。

参赛队伍可根据具体情况选择其他适宜的开源项目。

1.1.2 系统需求分析

本系统基于飞腾开发板和飞腾软件开源项目，期望开发出一种基于飞腾派平台的火源检测无人机，旨在展示飞腾 CPU 多核并行处理和核间通信的强大能力。

1. 预期实现功能

（1）并行处理

系统基于飞腾开发板，采用 OpenMP 进行并行加速，主核运行 Linux 系统负责热成像模块和视觉模块的运行，以及对从核的控制及相关信息的接收。从核运行裸机程序进行高实时性任务如超声测距避障的处理。通过多核协同工作，提高系统整体处理效率。

（2）超声避障

利用 OpenAMP 开源项目及 rpmsg 通信协议，实现主核与从核之间的数据传输。在主核程序启动时，通过/dev/rpmsg0 发送测距指令，从核通过 UART2 接口与 MB1043 交互，实现数据采集与解析，再返回距离数据到主核，为下一步无人机飞行提供决策依据，形成数据处理闭环。

（3）热成像

系统利用 MLX90640 红外传感器实现热成像。利用迈来芯官方所给固件库进行在飞腾派上的移植，同时在通过红外传感器获取温度数据后使用插值算法与伪彩色转换，将其转化为温度矩阵和热像图并显示在屏幕上，为无人机预警提供依据。

（4）无人机

系统采用 S500 高强度碳纤维机架作为飞行平台，搭载 PX4 开源飞控系统，通过飞腾开发板主核的 MAVLink 协议与飞控通信。主核实时处理目标检测、热成像及避障数据后生成飞行指令，通过串口发送至 PX4 飞控，由飞控完成高频率的姿态解算与电机控制。基于 PX4 的悬停定位与航点追踪功能，实现无人机在复杂环境下的稳定飞行与火源精准追踪，充分发挥飞腾多核架构在任务调度与实时控制中的优势。

（5）目标检测

主核配置并运行 Phytium-Pi-OS 系统，配置 ncnn 框架与 opencv 库，利用并改进轻量级目标检测算法 nanodet，进行视频数据的捕获与处理，识别场景中的

各种物体，为无人机飞行决策提供依据。

1.1.3 系统设计摘要

本系统基于飞腾开发板和飞腾软件开源项目，期望开发出一种基于飞腾派平台的火源检测无人机，旨在展示飞腾 CPU 多核并行处理和核间通信的强大能力。系统主要包括两个部分：无人机部分与机载电脑（飞腾派）部分。其中机载电脑部分又可分为以下几个模块：热成像模块、目标检测模块、超声避障模块。同时我们使用 OpenMP 进行并行加速，并使用 OpenAMP 实现核间通信，确保各核能够高效地协同工作。

本系统中主核运行 Linux 系统，从核运行裸机程序，以实现高效的多核协作处理。在主核上，安装并配置了 Phytium-Pi-OS 系统，确保其支持 OpenMP 和 OpenAMP；使用 OpenCV 库和 ncnn 框架，利用并改进开源项目 nanodet，在主核上进行视频捕获和目标检测，识别场景中的各种物体，为无人机飞行决策提供依据；使用 Opencv 和 adafruit 库中的有关工具在主核上实现火源识别和检测，将捕获的温度数据进行处理和识别;利用迈来芯官方所给固件库进行在飞腾派上的移植，同时在通过红外传感器获取温度数据后使用插值算法与伪彩色转换，将其转化为温度矩阵和热像图并显示在屏幕上，为无人机预警提供依据。在从核上，利用了飞腾公司提供的 standalone-SDK 和 OpenAMP 开源项目，配置并运行了裸机程序；利用 OpenAMP 开源项目及 rpmsg 通信协议，实现主核与从核之间的数据传输。在主核程序启动时，通过/dev/rpmsg0 发送测距指令，从核通过 UART2 接口与 MB1043 交互，实现数据采集与解析，再返回距离数据到主核，为下一步无人机飞行提供决策依据，形成数据处理闭环。

系统采用基于 PX4 开源飞控的 S500 六轴无人机作为飞行载体，通过飞控的 MAVLink 协议与飞腾开发板主核建立实时通信。主核综合热成像数据、目标检测结果及超声避障信息生成航点指令，经串口发送至 PX4 飞控，由飞控的 PID 控制器完成毫秒级电机调速。通过 PX4 的多模态控制架构（支持定点悬停、自动航点追踪及紧急避障模式），实现协同设计。

1.2 应用场景

1.2.1 森林火灾巡检

无人机在森林火灾监测中，主核通过热成像模块实时扫描地表温度，结合梯度算法识别异常高温区域。从核通过 MB1043 超声波传感器以 20ms 的周期采集障碍物距离数据，当检测到障碍物距离小于 50cm 时，主核立即控制无人机悬停或绕行。主核采用 OpenMP 并行加速热成像与视觉任务，从核裸机程序保障避障响应时间极低。

1.2.2 工业设施高温预警

在化工厂、变电站等场景中，无人机通过热成像模块实时绘制设备温度分布图，标记过热点（阈值≥80℃），目标检测模块可以联动识别管道破裂或油渍泄漏。无人机在密集管线中飞行时，从核通过 MB1043 传感器动态获取前后方距离数据，主核结合实时视觉与超声反馈生成三维避障路径。

1.2.3 城市消防与紧急救援

在高层建筑火灾中，无人机穿透浓烟环境，热成像模块精准定位受困人员体温信号（35-42℃特征波段），目标检测模块可以识别窗户与楼梯结构以规划救援通道。从核以 50Hz 频率采集超声数据，确保室内狭窄空间内无碰撞风险。

2．系统架构设计

2.1 系统结构选择与模块划分

2.1.1 系统结构选择

本系统基于 Holybro S500 无人机构建套装与飞腾派，其中飞腾派部分采用基于飞腾多核处理器的多核异构架构，主核运行 Linux 操作系统，从核运行裸机程序。这种结构选择能够充分发挥飞腾 CPU 的多核并行处理能力，确保系统具有高性能和高实时性。

系统的总体架构与飞腾派部分架构入下图所示：

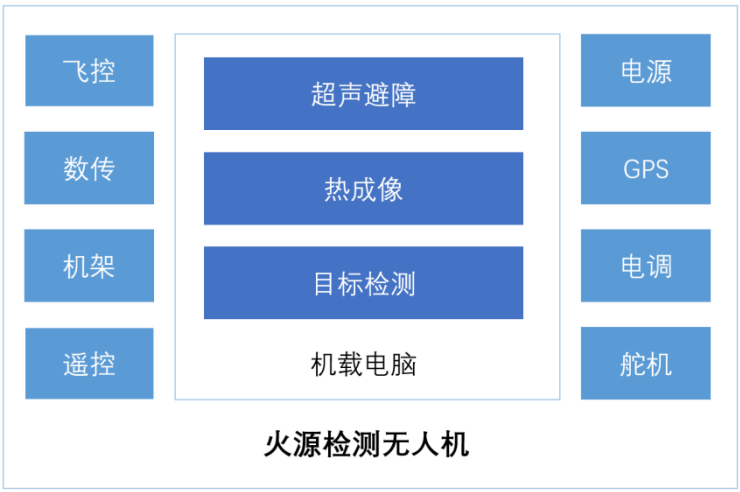


图 1 系统总体架构设计

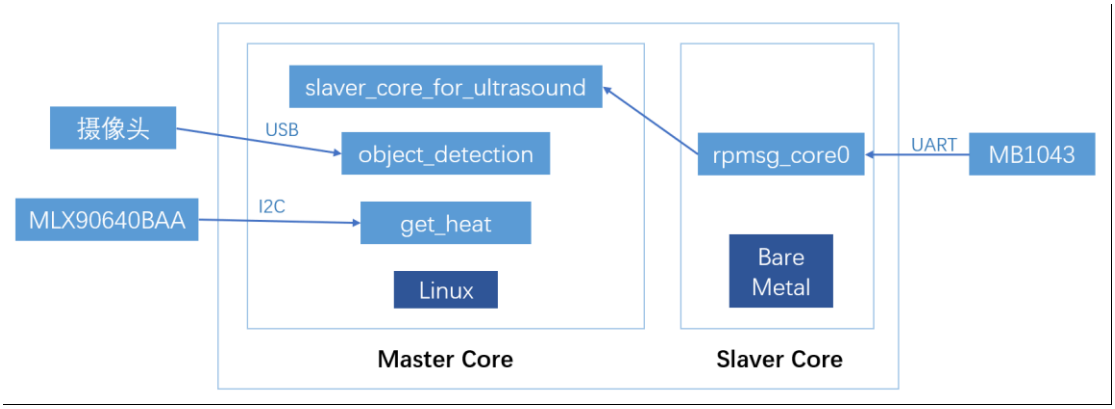


图 2 飞腾派部分架构设计

2.1.2 模块划分

系统分为无人机与机载电脑（飞腾派）两部分：

（1）机载电脑部分：

热成像（get_heat）模块：通过 MLX90640 红外传感器实时捕捉环境温度分布，识别异常高温区域。

目标检测模块（object_detection）：基于 ncnn 框架改进轻量级开源算法 Nanodet，结合 OpenCV 进行实时视频分析，识别火源及障碍物。

超声避障（slaver_core_for_ultrasound）模块：从核通过 MB1043 传感器采集距离数据，利用 OpenAMP 框架实现主从核间低延迟通信，确保避障实时性。

（2）无人机部分：基于 Holybro S500 无人机平台，接收飞腾派指令执行飞

行控制，实现动态路径规划与紧急避障。

系统通过 OpenMP 加速主核多任务并行处理，并通过 OpenAMP 实现核间通信，确保热成像、目标检测与避障任务的高效协同。

2.2 技术选型

2.2.1 硬件选型

飞腾开发板

- 多核异构架构支持 Linux 与裸机程序协同运行
- 原生集成 OpenAMP 框架，核间通信延迟<1ms

MLX90640ESF 红外传感器

关键参数：

- 分辨率：32×24 像素
- 测温范围：-40℃~300℃

输出方式：I2C 接口（400kHz）

选型理由：

- 支持非接触式宽域温度检测，适用于火源定位
- 低功耗设计（峰值电流<2mA），适配无人机移动平台

MB1043 超声传感器

关键参数：

- 测距范围：30cm~500cm
- 接口类型：UART（9600bps，8N1）

选型理由：

- 数据帧结构简单，便于裸机程序快速解析

Holybro S500 无人机开发套件

- 轴距：500mm
- 最大载重：1.2kg
- 飞行时间：25 分钟（空载）
- 支持 PX4 开源飞控系统

2.2.2 软件选型

（1）操作系统：主核运行 Phytium-Pi-OS，从核运行裸机程序。

（2）开发库与框架：

• OpenMP：用于多核并行计算加速。

• OpenAMP：实现多核间的通信与协作。

• OpenCV：用于视频捕获与数据预处理。

• ncnn：轻量级神经网络推理框架

• nanodet：轻量化目标检测

• phytium-standalone-sdk：从核固件开发（UART/GPIO 驱动、OPENAMP 配置）

• melexis/mlx90640-library：MLX90640 传感器官方固件库

2.3 接口描述

2.3.1 主从核间接口

通信协议：使用 OpenAMP 框架和 rpmsg 协议，采用轮询机制。

数据传输格式：。

2.3.2 模块间接口

（1）slaver_core_for_ultrasound 模块接口

输入：主核的命令字 DEVICE_CORE_GET_DISTANCE

输出：回传的前方最近的障碍物的距离

（2）object_detection 模块接口

输入：实时视频流（通过 USB 摄像头或机载摄像头捕获的 RGB 图像帧）。

输出：处理后的视频，检测到的目标会用框圈住。

（3）get_heat 模块接口

输入：通过 I2C 总线从 MLX90640 热成像传感器读取的原始帧数据

输出：温度矩阵：处理后生成的 24×32 像素温度矩阵。

伪彩色图像：通过 OpenCV 生成的 320×240 分辨率热成像图。

3. 系统详细设计

3.1 系统连线图

3.1.1 开发板部分连线说明

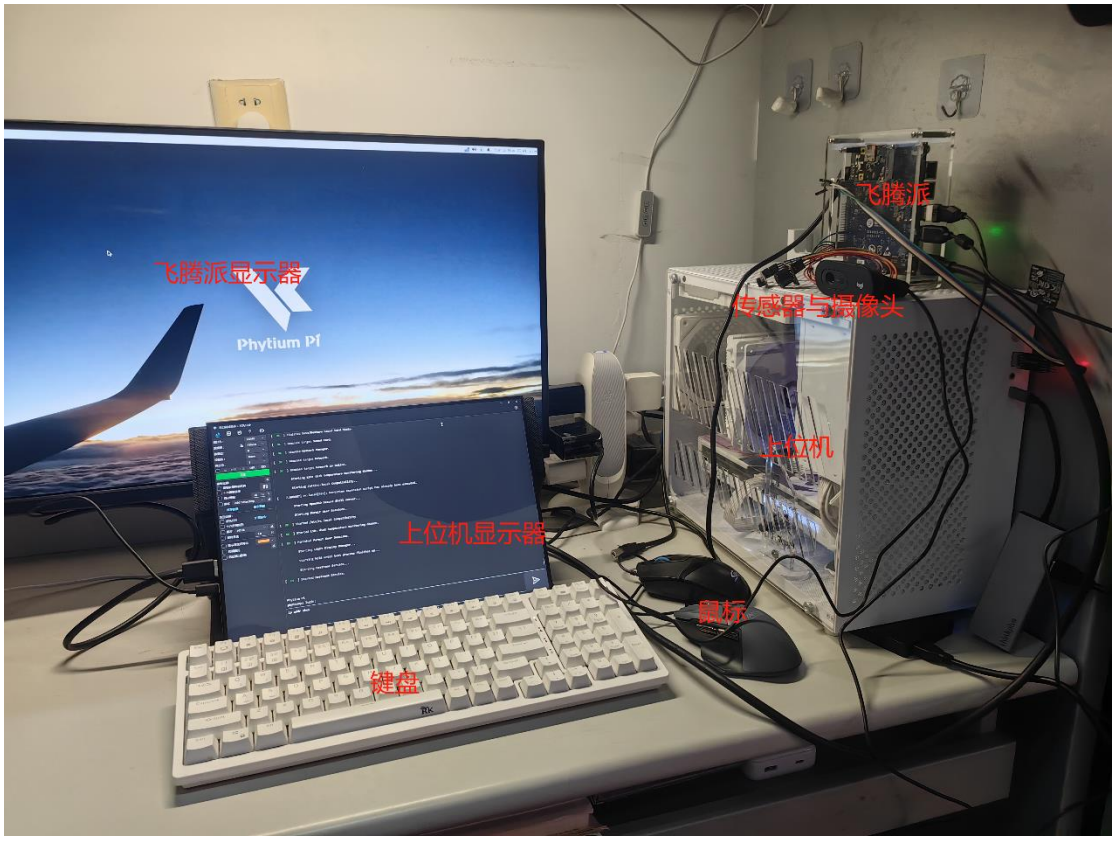


图 3 飞腾派部分开发示意图

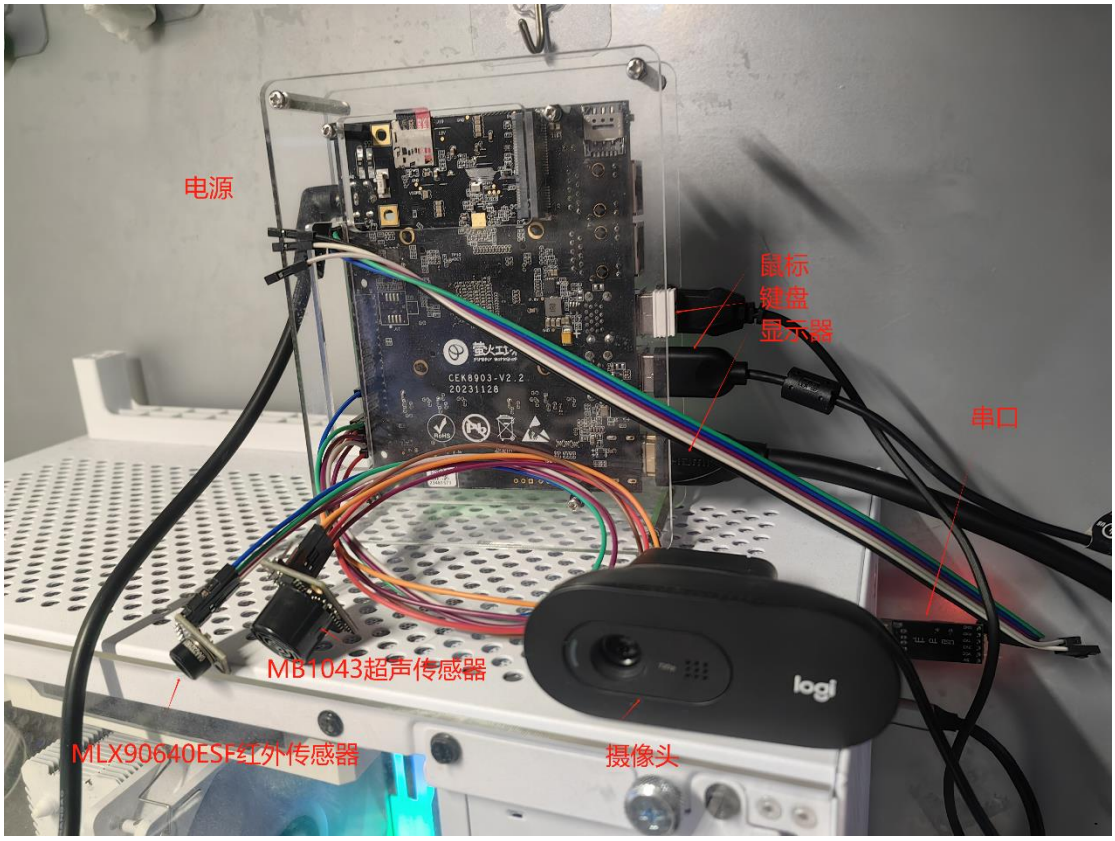


图 4 飞腾派连线

如图所示，开发板连接有

1）键盘

2）鼠标

3）显示器

4）摄像头

5）电源

6）MB1043 超声传感器

占用 J1 多功能接口区：J1-1 Vcc 3.3V,J1-6 GND 用以供电；J1-8 UART_TXD，J1-10 UART_RXD 用以与传感器收发数据。

7）MLX90640 红外传感器

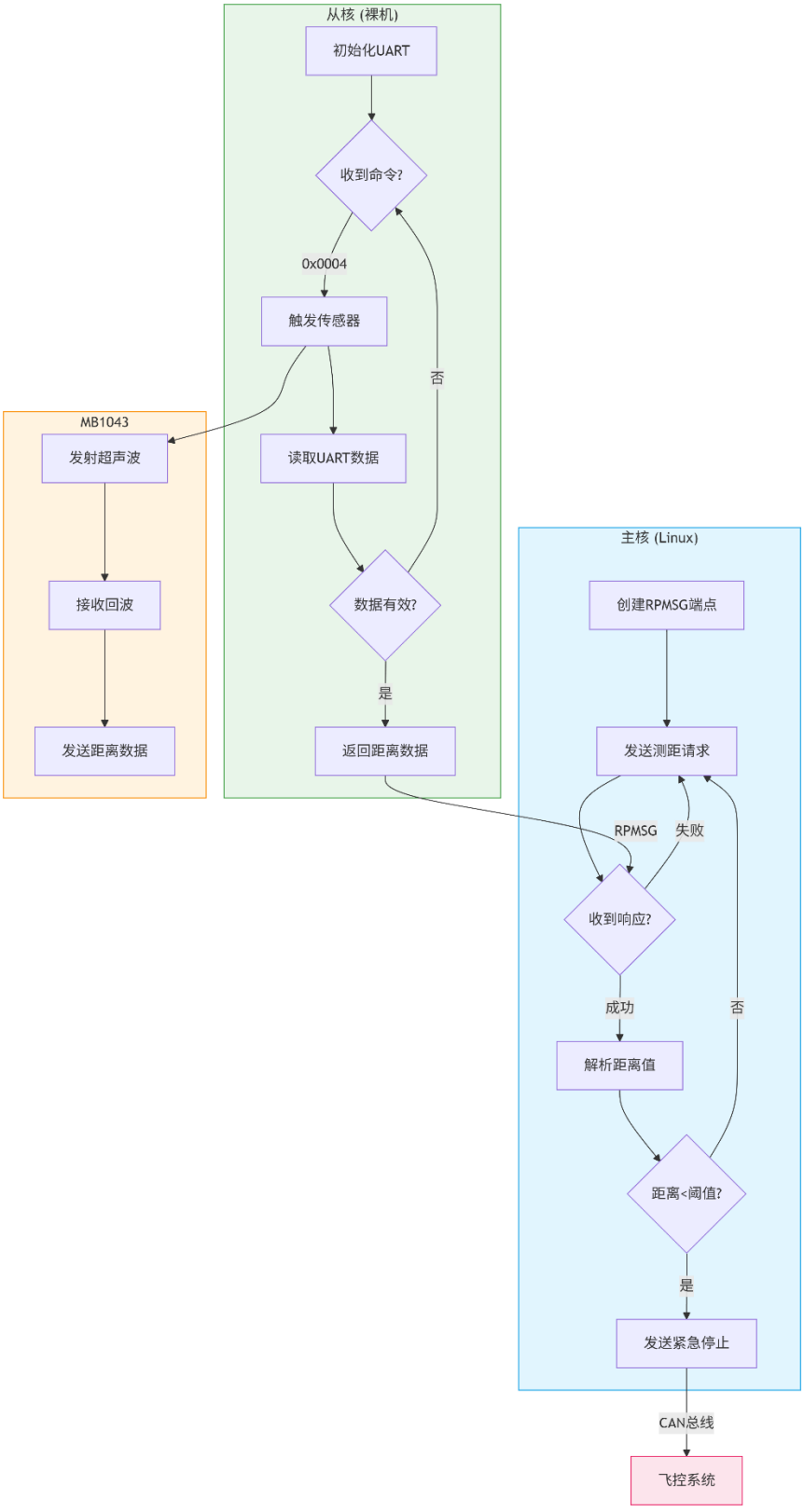
3.1.2 无人机部分说明

图 5 无人机部分调试示意图

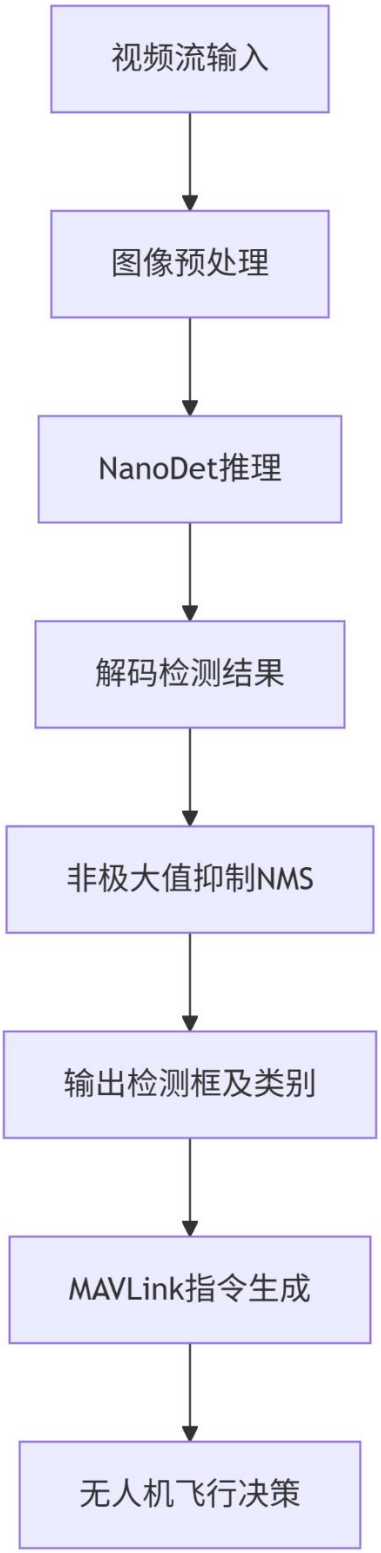
基于 Holybro S500 无人机平台，接收飞腾派指令执行飞行控制，实现动态路径规划与紧急避障。

3.2 软件流程图

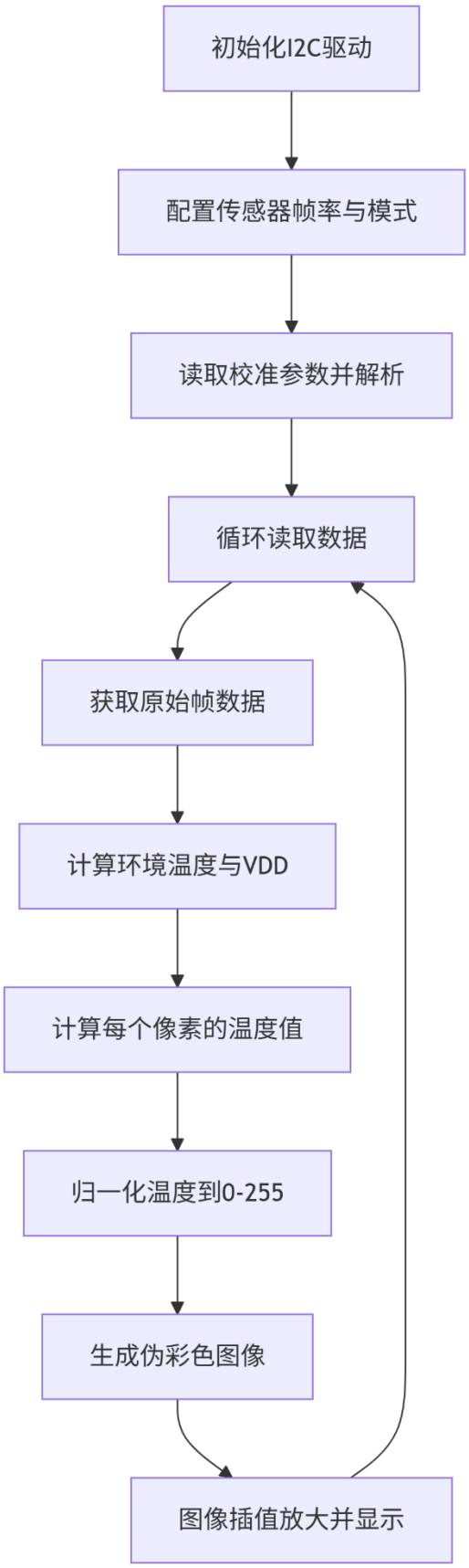
3.2.1 slaver_core_for_ultrasound



3.2.2 object_detection



3.2.3 get_heat


初始化I2C驱动
配置传感器帧率与模式
读取校准参数并解析
循环读取数据
获取原始帧数据
计算环境温度与VDD
计算每个像素的温度值
归一化温度到0-255
生成伪彩色图像
图像插值放大并显示

3.3 关键代码分析

3.3.1 slaver_core_for_ultrasound

1. 主核测距指令发送（rpmsg.c）

主核通过/dev/rpmsg0 设备发送 DEVICE_CORE_GET_DISTANCE 命令至从核，触发超声测距操作。write_full 函数确保完整发送 6 字节请求数据（4 字节命令字 + 2 字节长度占位）。

关键参数：DEVICE_CORE_GET_DISTANCE（协议定义命令字），rpmsg_fd（核间通信设备句柄）。

```
// 构造距离查询请求
struct {
    uint32_t command;
    uint16_t length;
} request = {
    .command = DEVICE_CORE_GET_DISTANCE, // 命令字：0x0004
    .length = 0                          // 数据长度固定为 0（无附加数据）
};

// 发送请求至从核
ret = write_full(rpmsg_fd, &request, 6); // 6 字节：4 字节命令 + 2 字节长度
if (ret < 0) {
    perror("write_full failed.");
    goto err1;
}
```

2. 从核 UART 初始化与数据读取（slaver_00_example.c）

从核通过 UART2 接口与 MB1043 传感器通信，配置为 8N1 格式，波特率 9600。FPl011Receive 函数以轮询方式读取单字节数据，检测到\r 结束符后终止读取。

关键参数：MB1043_UART_ID（UART 硬件实例 ID），MB1043_TIMEOUT_MS（超时阈值）。

```
// 初始化 UART2 接口（MB1043 传感器）
FPl011Config config_value;
const FPl011Config *config_p = FPl011LookupConfig(MB1043_UART_ID); // UART 实例 ID=2
```

```
memcpy(&config_value, config_p, sizeof(FPl011Config));
config_value.baudrate = MB1043_BAUD_RATE; // 波特率 9600

// 配置 UART 数据格式（8N1）
FPl011Format format = {
    .data_bits = FPL011_FORMAT_WORDLENGTH_8BIT,
    .parity = FPL011_FORMAT_NO_PARITY,
    .stopbits = FPL011_FORMAT_1_STOP_BIT
};
FPl011SetDataFormat(&uart_inst, &format);

// 轮询读取传感器数据
char buffer[10] = {0};
uint32_t bytes_read = 0;
while (bytes_read < sizeof(buffer) - 1) {
    uint32_t recv_cnt = FPl011Receive(&uart_inst, &buffer[bytes_read], 1);
    if (buffer[bytes_read-1] == '\r') break; // 检测到结束符
    if (--timeout == 0) {                    // 超时处理（100ms）
        SLAVE_DEBUG_E("MB1043 read timeout");
        break;
    }
    fsleep_millisec(1);
}
```

3. 距离数据解析与响应（slaver_00_example.c）

从核解析传感器原始数据（如 R1234\r），转换为浮点型距离值（123.4 cm）。响应数据包含状态码（status）和实际距离（distance），通过 rpmsg_send 返回主核。

关键校验：数据长度≥5 字节时标记为成功，否则标记为超时。

```
// 解析传感器返回的 ASCII 数据（如"R1234\r"）
buffer[bytes_read] = '\0';
float distance = (buffer[1]-'0') * 100.0f +
                 (buffer[2]-'0') * 10.0f +
                 (buffer[3]-'0') * 1.0f +
                 (buffer[4]-'0') * 0.1f; // 示例值：123.4 cm

// 组装响应数据结构
struct {
    uint32_t command;
    uint16_t length;
    uint8_t status;
```

```
    float distance;
    uint8_t reserved;
} response = {
    .command = DEVICE_CORE_GET_DISTANCE,
    .length = 6,
    .status = (bytes_read >= 5) ? 0 : 1, // 0:成功，1:超时
    .distance = distance
};

// 通过 rpmsg 返回数据至主核
ret = rpmsg_send(ept, &response, sizeof(response));
if (ret < 0) {
    SLAVE_DEBUG_E("Send distance failed");
}
```

4. 主核避障决策逻辑（伪代码）

   主核根据距离阈值（50cm）判断是否触发避障动作（如紧急停止）。

```
// 接收从核返回的距离数据
DistanceResponse *resp = (DistanceResponse *)r_buf;
if (resp->status == 0) {
    if (resp->distance <= 50.0f) { // 阈值：50cm
        send_emergency_stop();     // 触发紧急停止
        printf("Obstacle detected! Distance: %.1f cm\n", resp->distance);
    }
} else {
    printf("Sensor error: %d\n", resp->status);
}
```

3.3.2 object_detection

1. 目标检测核心逻辑（detect 函数）

   完成图像预处理、模型推理、结果解码及后处理。

```
std::vector<BoxInfo> NanoDet::detect(cv::Mat image, float score_threshold, float nms_threshold) {
    ncnn::Mat input;
    preprocess(image, input); // 图像标准化与尺寸调整
    auto ex = Net->create_extractor();
    ex.input("data", input);  // 输入预处理后的张量
    ncnn::Mat out;
    ex.extract("output", out); // 执行推理
    std::vector<CenterPrior> center_priors;
```

```
    generate_grid_center_priors(input_size[0], input_size[1], strides, center_priors); // 生成锚点
    std::vector<std::vector<BoxInfo>> results(num_class);
    decode_infer(out, center_priors, score_threshold, results); // 解码输出
    std::vector<BoxInfo> dets;
    for (auto& cls_results : results) {
        nms(cls_results, nms_threshold); // 非极大值抑制
        dets.insert(dets.end(), cls_results.begin(), cls_results.end());
    }
    return dets;
}
```

2. 解码与边界框生成（decode_infer 函数）

将模型输出转换为物理坐标下的检测框。

```
void NanoDet::decode_infer(ncnn::Mat& feats, std::vector<CenterPrior>& priors, float threshold, std::vector<std::vector<BoxInfo>>& results) {
    for (int idx = 0; idx < priors.size(); idx++) {
        const float* scores = feats.row(idx);
        int label = std::max_element(scores, scores + num_class) - scores; // 取最高置信度类别
        float score = scores[label];
        if (score < threshold) continue;

        const float* bbox_pred = feats.row(idx) + num_class;
        BoxInfo box = disPred2Bbox(bbox_pred, label, score, priors[idx].x, priors[idx].y, priors[idx].stride);
        results[label].push_back(box); // 保存有效检测结果
    }
}
```

3. 非极大值抑制（nms 函数）

功能：去除冗余检测框，保留最优结果。

```
void NanoDet::nms(std::vector<BoxInfo>& boxes, float nms_threshold) {
    std::sort(boxes.begin(), boxes.end(), [](BoxInfo a, BoxInfo b) { return a.score > b.score; });
    for (int i = 0; i < boxes.size(); i++) {
        for (int j = i + 1; j < boxes.size();) {
            float iou = calculate_iou(boxes[i], boxes[j]); // 计算交并比
            if (iou >= nms_threshold) boxes.erase(boxes.begin() + j);
            else j++;
        }
```

```
    }
}
```

3.3.3 get_heat

1. 温度计算核心算法

函数 MLX90640_CalculateTo 通过以下步骤实现温度转换：

传感器数据补偿：根据环境温度、发射率和校准参数修正原始 ADC 值。

辐射率计算：结合像素灵敏度（alpha）和环境辐射（taTr）计算辐射能量。

温度迭代求解：通过四次方根公式反推温度值，分段补偿非线性误差。

```
void MLX90640_CalculateTo(uint16_t *frameData, const paramsMLX90640 *params, float emissivity, float tr, float *result) {
    // ... 补偿计算
    Sx = pow(alphaCompensated, 3) * (irData + alphaCompensated * taTr);
    To = sqrt(sqrt(irData / (alphaCompesated * (1 - params->ksTo[1] * 273.15) + Sx)) - 273.15;
    // ... 分段补偿
}
```

2. 伪彩色映射算法

函数 GrayToPseColor 将归一化灰度值映射为 BGR 伪彩色：

线性分段映射：根据灰度值在 0-255 区间内的位置，动态分配 R/G/B 通道强度。

```
void GrayToPseColor(uint8_t method, uint8_t grayValue, uint8_t &colorR, uint8_t &colorG, uint8_t &colorB) {
    colorR = abs(0 - grayValue);    // 低灰度区红色增强
    colorG = abs(127 - grayValue);  // 中灰度区绿色过渡
    colorB = abs(255 - grayValue);  // 高灰度区蓝色增强
}
```

3. I2C 数据读取优化

MLX90640_I2CReadCombined 使用 Linux 的 I2C_RDWR 接口实现高效批量读取：

组合事务：通过 ioctl 一次性发送寄存器地址并读取多字节数据，减少总线占用时间。

分块回退机制：若系统不支持组合事务，则自动切换为分块读取

（MLX90640_I2CReadChunked）。

```
int MLX90640_I2CRead(uint8_t addr, uint16_t reg, uint16_t cnt, uint16_t *data) {
    if (fd < 0 && MLX90640_I2CInit() < 0) {
        fprintf(stderr, "I2CInit failed\n");
        return -1;
    }
    // 尝试 combined 事务
    if (MLX90640_I2CReadCombined(addr, reg, cnt, data) == 0) {
        return 0;
    }
    // 若 combined 失败，回退到分块读取
    return MLX90640_I2CReadChunked(addr, reg, cnt, data);
}
```

4 系统测试与分析

4.1 飞腾派测试部分

4.1.1 超声避障功能测试。

我们首先用 python 进行编程，实现了基于 MB1043 超声波传感器的实时数据采集、卡尔曼滤波优化及无人机自主避障控制，直接在飞腾派开发板上 linux 环境下对传感器进行测试，验证传感器的可用性。

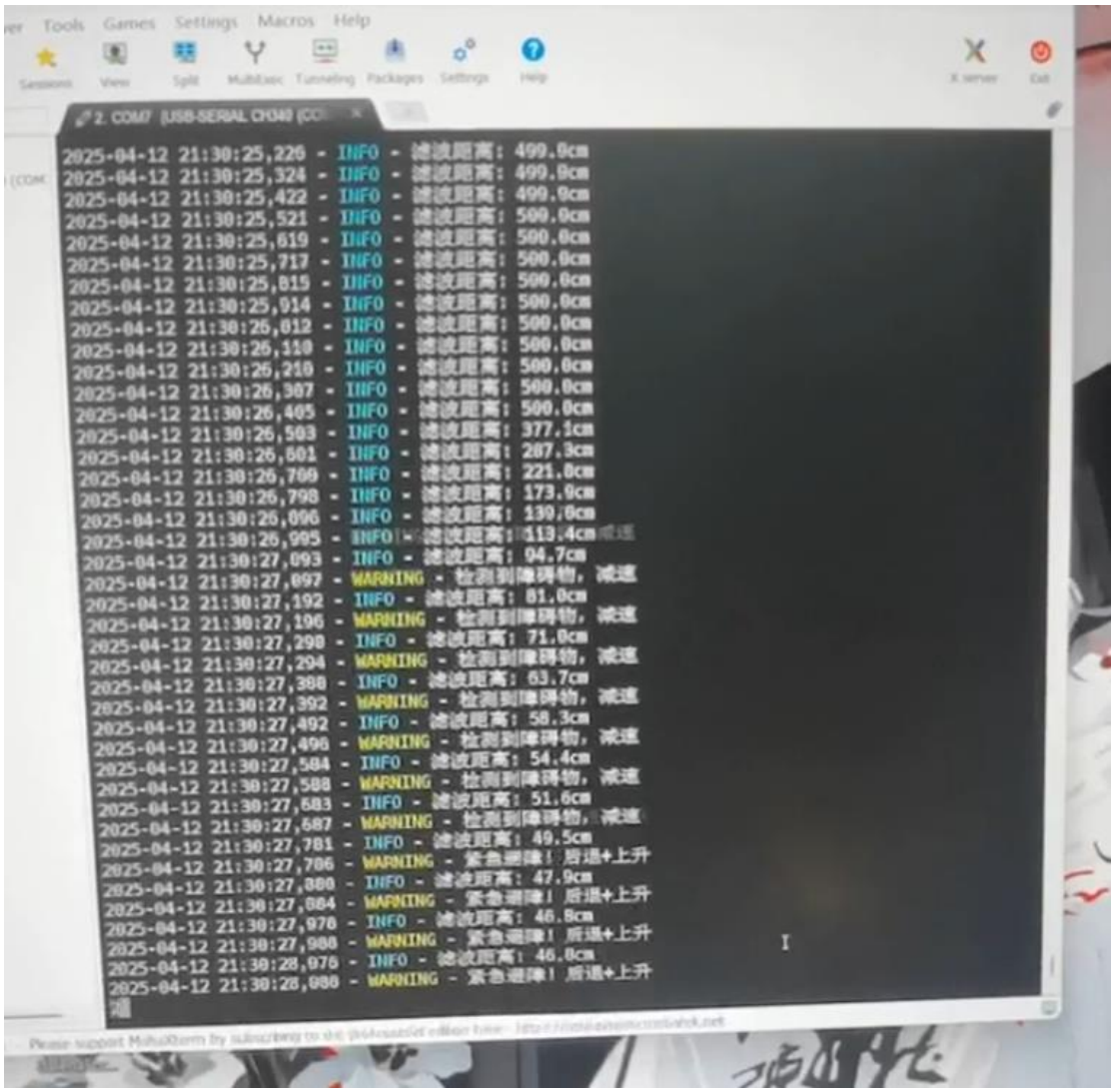


图 6 MB1043 传感器测试视频截图

然后我们结合飞腾官方提供的 standalone-SDK 工具包中提供的 FPL011 库和 openAMP 的相关应用实例，构建 slaver_core_for_ultrasound 工程，对超声避障部分进行移植，实现主核控制从核，从核控制 UART2 引脚读取 MB1043 的数据，然后利用 rpmsg 将数据回传到主核程序上。然后在板子上进行相关测试：

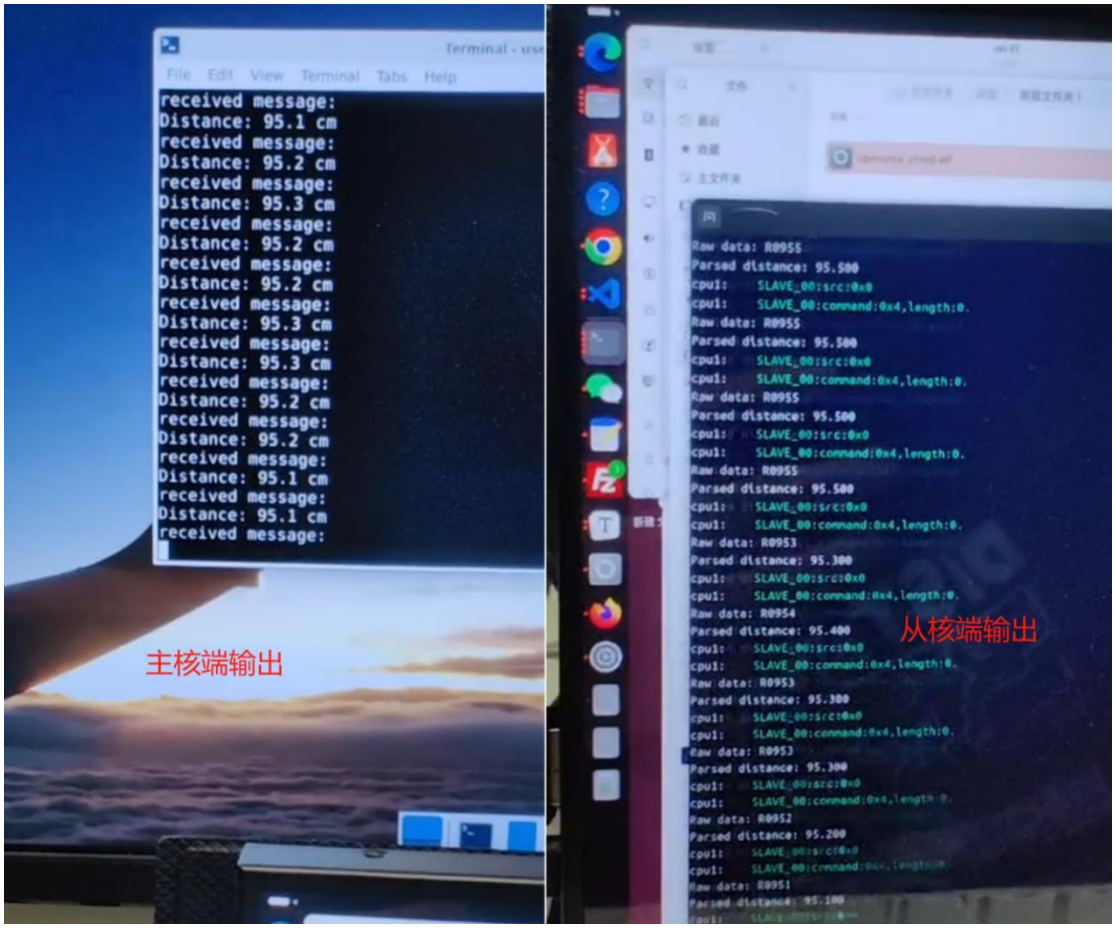


图 7 超声避障模块测试视频截图

4.1.2 热成像功能测试

我们利用 MLX90640 红外传感器实现热成像。下面我们在开发板上运行测试程序，进行相关演示。下面两个图分别是测试者将文化摄像头对准自己，手持手机所拍摄到的热成像图，以及测试者将摄像头对准手手所得到的热成像图。

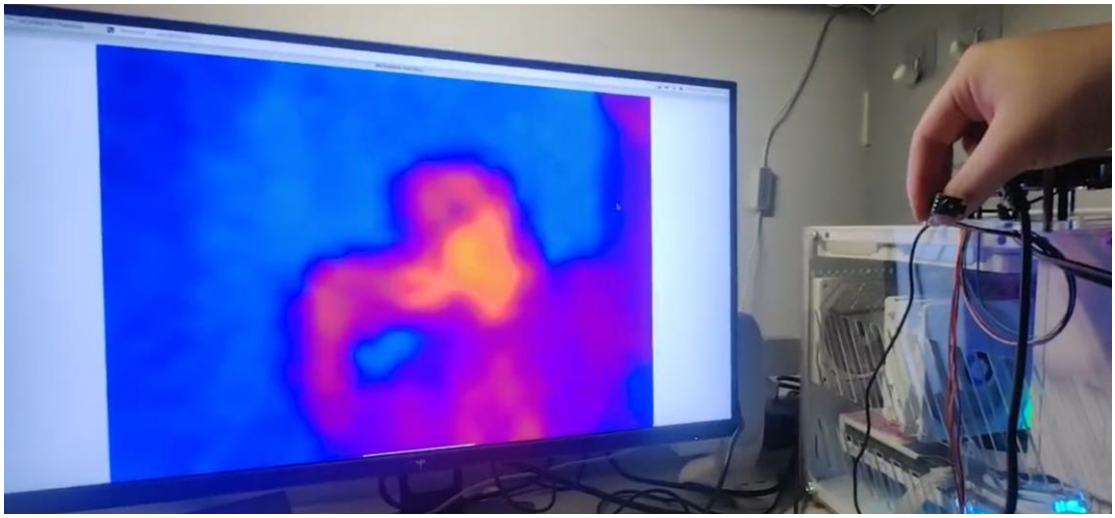


图 8 红外传感器测试视频截图

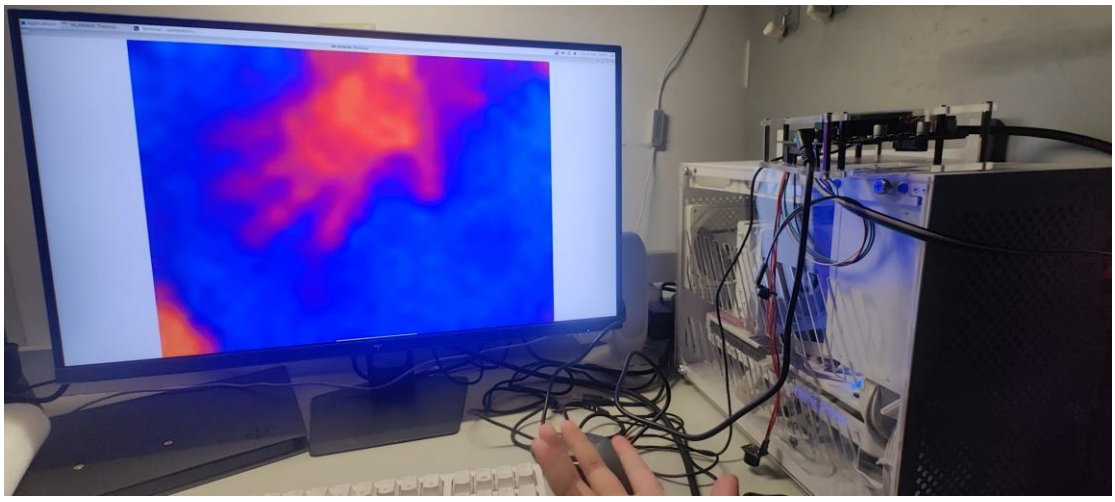


图 9 红外传感器测试视频截图

4.1.3 目标检测功能测试

我们在开源的轻量级目标检测算法上进行改进，使用 openmp 进行并行化操作，以期直接能够实时在我们的开发板上运行。

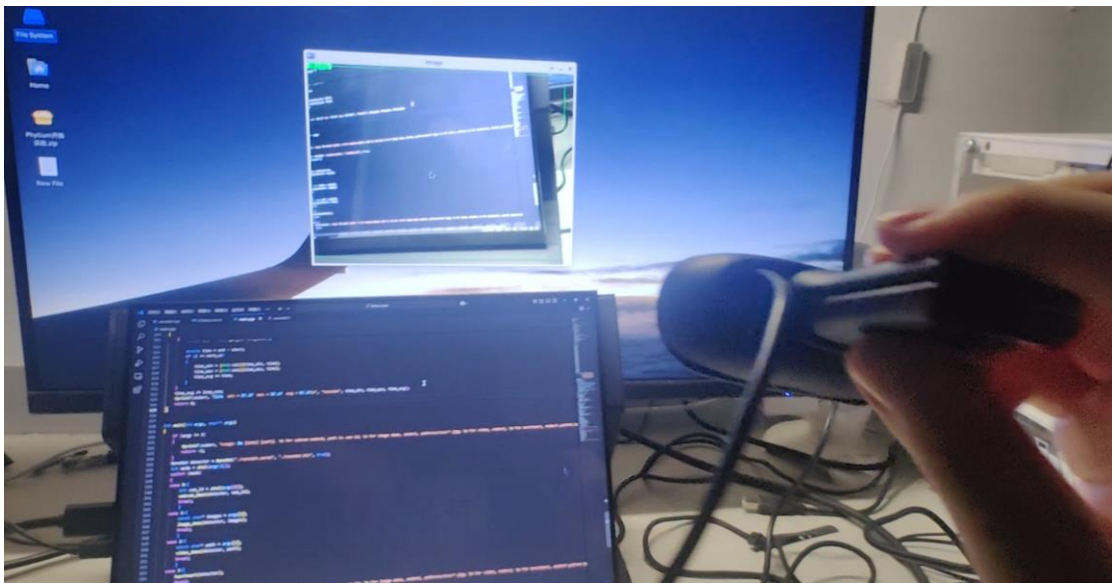


图 10 目标检测测试视频场景截图

图 11 目标检测测试视频截图

4.2 测试结果

上述测试结果说明我们已经完成了目标检测、红外热成像及超声避障的相关功能，其中红外传感器可以获得实时的热像图，目标检测可以对周围的障碍物以及相关环境进行实时的检测，而超声避障则可以接收障碍物与无人机的距离，并通过 openamp 框架将获得的数据传回到主核上，以便进行下一步的决策。初步达到了我们的所期望的目标。

4.2 无人机测试部分

4.2.1 测试步骤

1. 配置地面站

使用 QGroundControl（QGC）地面站软件配置 PX4 飞控参数，确保通信链路正常。设置 FS-iA6B 接收机与遥控器对频，并校准遥控通道（油门、横滚、俯仰、偏航）。通过 3DR Radio X6 数传模块建立飞控与地面站的远程数据交互，确保遥测信号稳定。

2. 手动模式测试（验证机）

首先在 Manual（手动）模式下测试小型无人机，验证基础飞行功能：

(1)垂直升降测试：缓慢推拉油门，观察无人机响应速度及稳定性。

(2)悬停测试：在低空（1.5m 高度）手动悬停，检查姿态自稳能力。

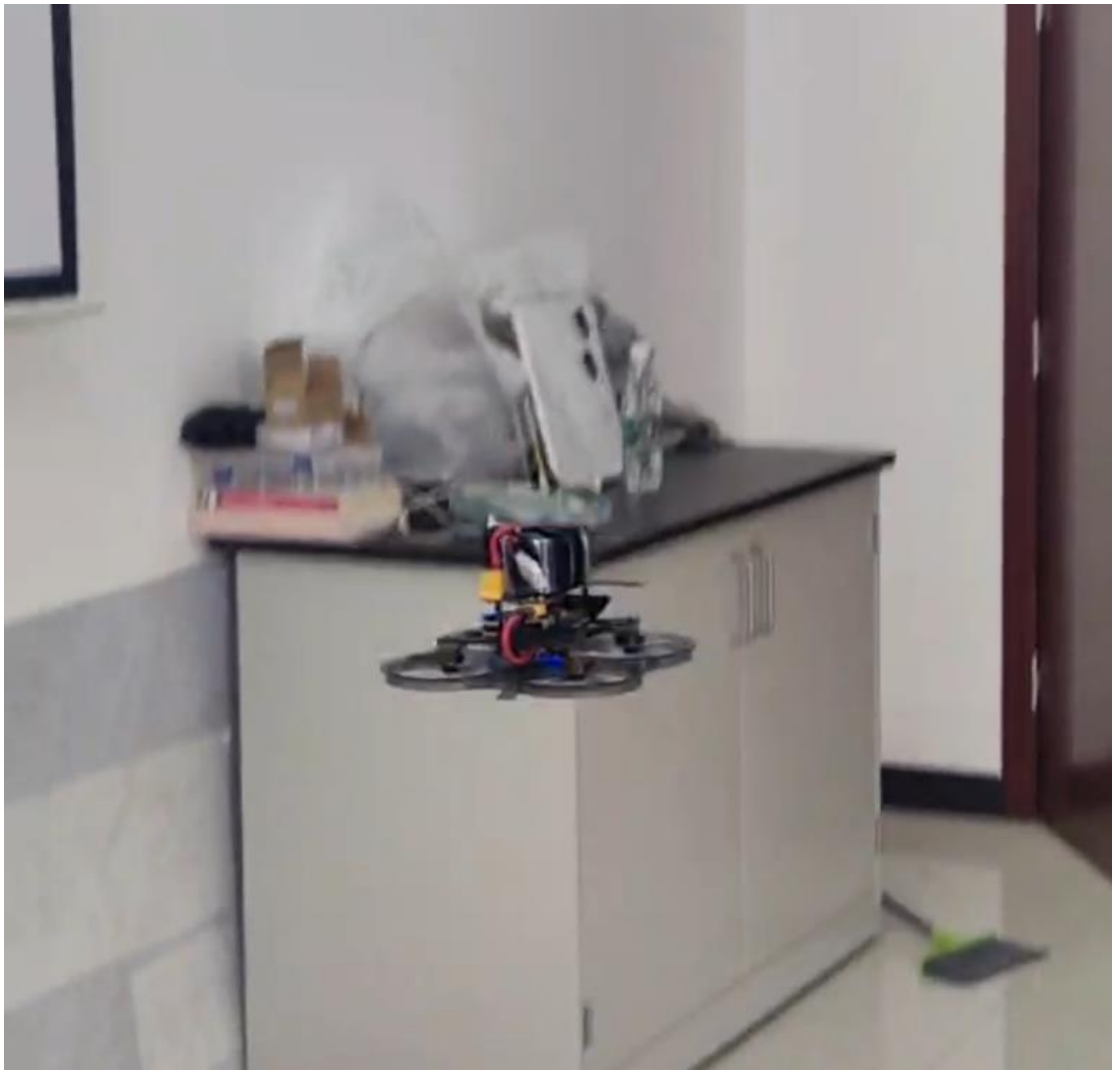


图 12 验证机飞行视频截图

3. S500 无人机功能测试

切换至 Stabilize（自稳）模式，测试 S500 无人机的飞行性能：

(1)定点悬停：依赖飞控内置的 IMU 与气压计数据，验证悬停精度。

(2)航向控制：通过遥控器偏航输入，测试无人机转向响应。

(3)数传链路验证：在地面站实时监测飞行数据（如高度、电池电压、GPS 状态）。

4.2.2 测试结果

1.基础飞行性能

(1)小型验证机在手动模式下响应灵敏，悬停误差≤±0.3m；

(2)S500 在自稳模式下悬停精度达±0.1m（无风环境），满足设计要求。

2.系统协同性

数传链路在 50m 半径内稳定传输。

图 13 S500 调试视频截图

5 创新点分析与总结

5.1 创新点分析

5.1.1 基于飞腾派的多核异构系统设计与应用

充分利用了飞腾 CPU 的多核架构，设计了主核运行 Linux 系统处理复杂任务（如热成像分析、NanoDet 目标检测）、从核运行裸机程序处理高实时性任务（如超声波测距与避障）的异构系统 。这种分工有效地发挥了飞腾平台多核并行处理的性能优势，保证了复杂感知与实时控制的并行不悖。

应用 OpenAMP 等技术实现核间通信与资源管理，确保了主从核之间高效协同工作，这对于要求高实时性与复杂数据处理的无人机应用是一项重要探索 。

5.1.2 多传感器融合与智能化感知

整合了热成像传感器、超声波传感器以及可见光摄像头（用于目标检测），实现了无人机对环境的多维度感知。热成像用于特定目标（如火源）的远距离探测 ，超声波用于近距离避障 ，而目标检测则提供了对环境中更广泛目标的识别能力 。

将多种传感器数据在飞腾派上进行融合处理，为无人机的自主决策提供了更全面、更可靠的环境信息，提升了系统的智能化水平。

5.1.3 轻量级目标检测算法的嵌入式应用与优化

在资源相对受限的机载嵌入式平台（飞腾派）上，成功部署并运行了轻量级目标检测算法。这对于在边缘端实现实时智能分析具有重要意义。

5.1.4 实时避障与自主飞行能力的集成：

将超声波测距数据与无人机飞行控制系统紧密结合，从核处理的实时距离信息能够及时反馈给主核乃至飞行控制系统，为无人机在复杂环境中的安全飞行提供了保障。

5.2 系统优化

算法优化：方法仍然存在如实时性能稍弱，获取数据的速度稍慢慢，可针对实时性进行优化，如，针对目标检测模块在特定场景下的检测精度和速度进行优化；优化热成像数据的处理算法，提高火源识别的准确性等。

多核调度与负载均衡：进一步优化飞腾派各核心的任务分配和调度策略，最大限度地发挥并行处理能力，降低系统延迟。

功耗优化：针对无人机续航需求，对硬件选型、软件算法及系统运行策略进行低功耗设计和优化。

通信效率：优化无人机与地面站、飞腾派内部各模块以及各核心间的通信协议和数据传输效率。

鲁棒性与容错性：增强系统在各种干扰或部分传感器失效情况下的稳定性和容错能力。

5.3 总结

本项目成功研制了一套基于飞腾派平台的智能无人机系统原型，该系统集成了无人机飞行平台与搭载飞腾派的机载智能处理单元 。机载单元通过高效利用飞腾 CPU 的多核异构特性，实现了对热成像、超声波测距以及目标检测等多模态感知信息的并行处理与分析 。

实验测试结果表明，本系统能够有效完成热成像数据采集与分析、超声波实时测距与避障决策、以及进行目标检测等核心功能。主从核的协同工作模式充分发挥了飞腾平台的计算潜力，为无人机赋予了较强的环境感知与自主决策能力。特别是在火源检测等特定应用场景下，多传感器的信息融合为快速响应和准确定位提供了技术基础。

综上所述，本项目不仅验证了飞腾派平台在嵌入式人工智能和无人机载应用中的可行性与优势，也为后续开发更高级别的智能无人机系统积累了宝贵经验。未来的工作可以围绕算法的深度优化、更高程度的自主飞行控制以及针对特定行业应用的定制化功能拓展等方面展开，以期实现更广泛的实际应用价值。

参考资料网址

https://edu.phytium.com.cn/classroom/3/threads
https://gitee.com/phytium_embedded/phytium-standalone-sdk
https://github.com/RangiLyu/nanodet/
https://gitee.com/phytium_embedded