2023 全 国 大 学 生

嵌入式芯片与系统设计

应用赛道

EMBEDDED CHIP AND SYSTEM DESIGN APPLICATION

作品名称	环境卫士
作者姓名	廖月伟、周泰宇、潘星宇

目 录

第一章 设计概述

1.1 设计目的

根据调查发现，清洁人员精力有限，总有不能及时发现的污渍、垃圾（倾倒的奶茶、饮料等）。同时在环卫过程中，需要消耗大量的人力去来回巡逻监管来维持环境清洁。另一方面，当前的扫地机器人算法设计往往是全范围打扫，这样的算法效率低下，难以快速覆盖大范围区域。为解决以上问题，帮助清洁人员提高清洁效率、节省人力物力成本，我们团队采用实时监测、图像识别、物联网、ROS Nav 导航等技术，能迅速识别定位污渍块，垃圾团，并通过网络实时共享给清洁人员，清洁人员只需要对特定的区域进行清扫。运用我们的设计，大片区域的环卫就只需要少量人员来维持。本设计为 RT Thread 方向赛道作品，运用 RT-Thread OS，STM32 Cube AI，以及 RT-Thread AI 工具，目的是让 RTT 系统结合深度学习，在物联网设备中发挥其性能。

1.2 应用领域

对于酒店、学校、医院、办公楼等大型公共场所，这种系统可以通过感应器、摄像头实时检测环境中的污渍块或垃圾团，并定向调配清洁人员或设备进行清洁，从而有效节省资源并提高工作效率。对于家庭环境，系统可以作为智能扫地机器人的辅助，不仅可以完成所有空间的打扫，更能对环境进行实时监控，只针对有污渍和垃圾的区域进行清洁，提高清洁效率，节约能源。同时，拥有自主导航和图像传输功能的移动机器人，也能够作为移动监控或移物机器人。

1.3 主要技术特点

本产品运用 RT-Thread OS，STM32 Cube AI，以及 RT-Thread AI 工具，也即 RT AK，在物联网设备中发挥了嵌入式实时操作系统的性能和特点。

利用板载的 AP6212 wifi 蓝牙模块实现数据上传。Art-pi 读取 imu 模块得到机

器人的姿态，读取 dht11 获得环境温湿度数据，并将数据通过 mqtt 协议上传到 onenet 物联网平台，通过可视化功能使数据可以实时显示。

AI 模型运用 Yolo-Fastest，其模型大小仅为 1.3MB，这使得 Yolo-Fastest 能够在资源受限的设备上运行。此外，Yolo-Fastest 的计算复杂度也较低，其 FLOPS 仅为 0.23B，这意味着 Yolo-Fastest 拥有更快的运行速度。运用 Darknet 框架训练并测试，经过量化后 darknet 模型输出为 tflite 模型格式，再部署在 Art-pi 上。

Art-pi 使用 micro-ros 与 ubuntu 上位机上的 ROS2 进行连接，将机器人里程计以及环境深度信息上传 ROS 终端。在 ROS2 里通过使用 Cartographer 和 Nav2 为机器人提供了实时 SLAM 和自主导航的功能。Cartographer 基于传感器数据进行定位和地图构建，具有全局一致性优化和回环检测等特点。Nav2 支持路径规划、障碍物避障和多导航模式，与 ROS2 集成，可灵活配置和扩展。两者结合使用，机器人能在未知环境中构建地图、实现自主定位和导航，适用于各种应用场景。

1.4 关键性能指标

MPU6050 的加速度度量范围设置为 ±2g，角速度度量范围设置为 ±2000。Dht11 湿度测量范围为 20% ~ 90%RH，温度测量范围为 0 ~ +50 摄氏度，湿度测量精度为 ±5.0%RH，温度测量精度为 ±1.0□，响应时间 <5ms。

对于 Yolo-Fastest 算法的应用，以下是一些核心参数：模型大小：1.3MB 平均精度：61.02 浮点运算次数：0.23B FLOPS Yolo-Fastest. Flite 模型性能表现平均精度：48.26 输入尺寸：320x320x3 浮点运算次数：0.238 模型大小：1.17M

在 ROS2 里通过使用 Cartographer 和 Nav2，在一组测试场景中，机器人的定位误差保持在 ±3 厘米内，算法的平均处理时间为 5.8ms。

1.5 主要创新点

本产品具有 ROS 建图导航、多线程、多界面、多功能的特点。

1.5.1 ROS 建图导航

实时 SLAM 算法优化：机器人在 rt-thread 实时操作系统的帮助下能够在实时场景下更快速、更准确地构建地图并进行自主定位。多传感器融合，激光雷达与惯性测量单元融合进来，提高地图构建和定位的精度和鲁棒性。动态环境建图与导航，产品能有效处理动态环境，及时更新地图和避免障碍物。自主路径规划优化，更高效地生成平滑且安全的路径，避免碰撞和过度转弯。

1.5.2 多线程

通过使用 RT-thread 操作系统，让 MCU 拥有能同时处理多个线程的能力：传感器数据采集、运动控制、通信、用户界面等。rt-thread 的线程管理（调度、同步、优先级管理、中断处理等）的运用，使得机器人各模块各线程能够和谐高效运行，更充分地利用 MCU 资源。同时，线程的优先级管理使得产品优先进行紧急的线程，提高系统的实时性。

1.5.3 多界面

提供丰富的接口和视图来监控和控制系统的运行：微信小程序，网页服务器，onenet 物联网平台、ubuntu 终端以及 LCD 屏幕显示。微信小程序通过蓝牙给用户提供一个便利的方式来配网；同时，wifi 能自动重连，简化了联网过程。网页服务器和 onenet 物联网平台用来可视化传感器数据，界面简洁直观，并设置接口允许用户远程操作机器人。ubuntu ROS 终端运行配置机器人建图导航系统，显示机器人传来的图像信息。LCD 显示此时摄像头的图像数据，同时显示系统各个传感器数据。

1.5.4 多功能

ROS SLAM 建图导航、实时识别、远程监控、数据上传等。产品通过 ROS 实现了 SLAM 技术，使机器人能够在未知环境中自主构建地图，并同时进行定位，从而实现自主导航功能。同时产品配备了实时图像或物体识别功能，利用先进的计算机视觉算法，可以快速、准确地识别目标。另外，产品提供远程监控功能，用

户可以通过手机、平板电脑或计算机等设备，实时查看机器人的工作状态、摄像头画面以及环境信息。无论是在智能家居、服务机器人、物流仓储还是其他领域，这样的多功能产品可以帮助用户实现更智能、高效、便捷的服务和体验。

第二章 系统组成及功能说明

2.1 整体介绍

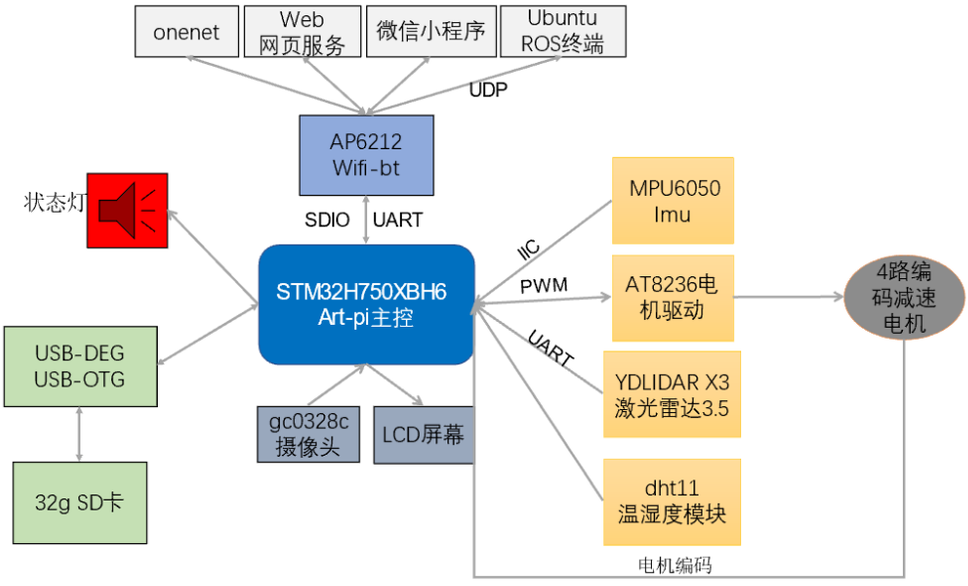

图 2-1 硬件部分

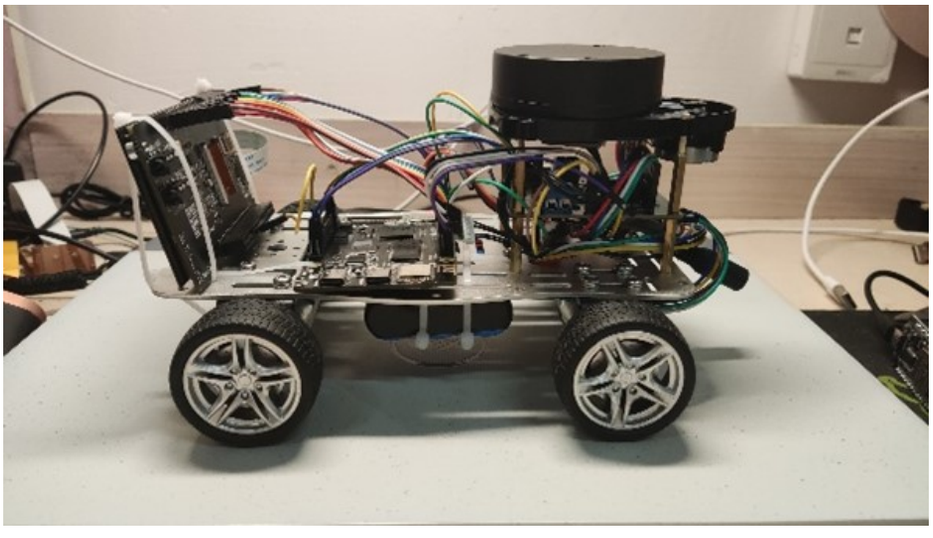

图 2-2 侧视图

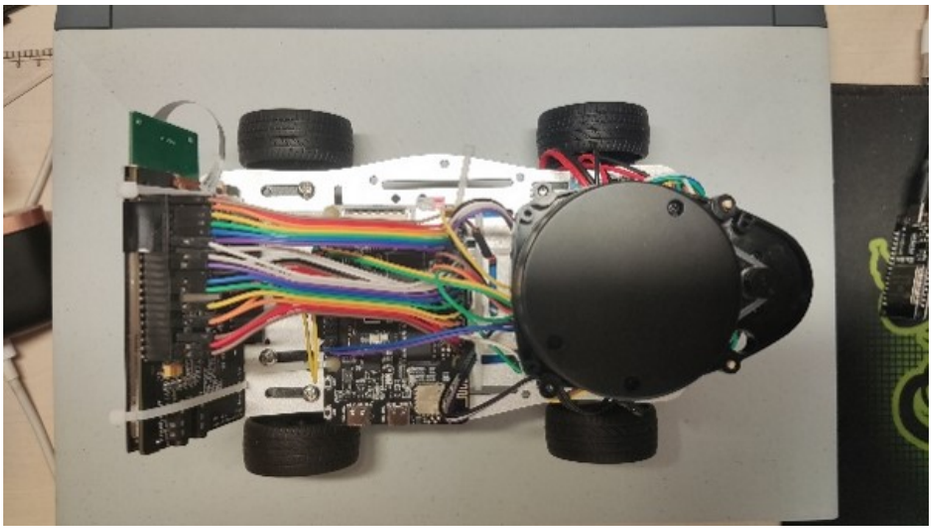

图 2-3 俯视图

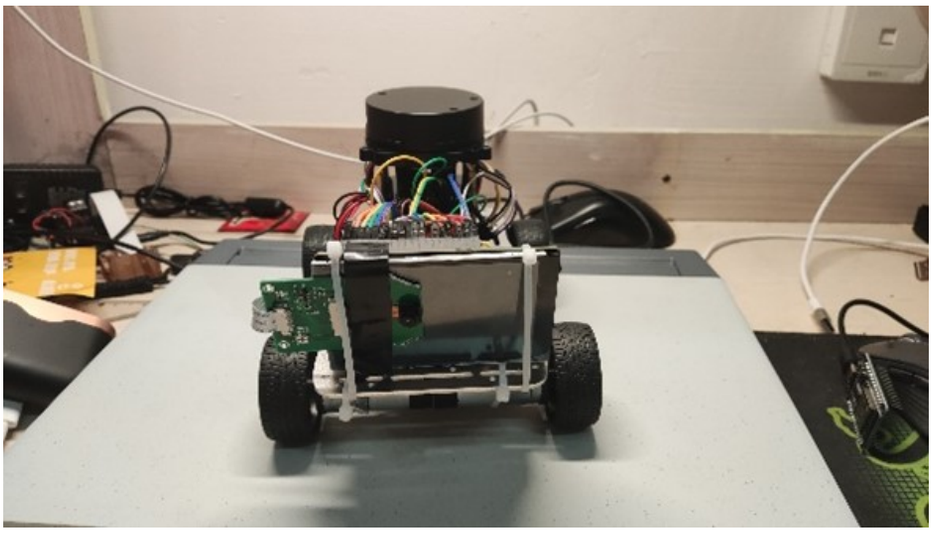

图 2-4 主视图

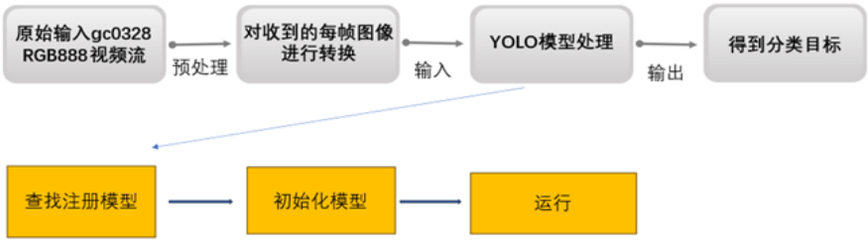

图 2-5 AI 识别实现

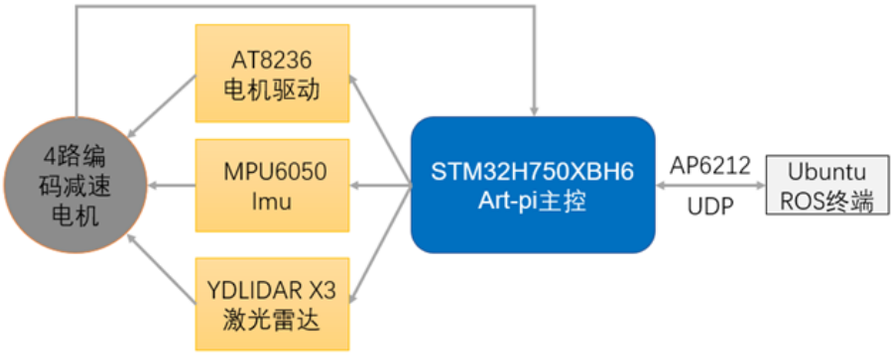

图 2-6 ROS 实现

2.2 各模块介绍

本设计主要分为三大块：ROS，AI，用户界面。

2.2.1 ROS 部分

移动机器人底座：运动控制、传感器数据采集

AT8236：AT8236 模块涉及控制底座上的电机，使机器人能够实现运动、转向等动作。AT8236 电机驱动模块用来驱动底座上的电机，负责控制电机的转速和方向。AT8236 通过接收 PWM 控制信号来控制电机的转速，并根据不同的信号控制方式实现机器人的前进、后退、左转、右转等运动。同时，AT8236 将通过读取编码器的脉冲信号来获取电机的转速和方向信息，并将信号反馈回 Art-pi，实现电机的闭环控制。

YDLIDAR X3 激光雷达：激光雷达用于扫描周围环境并获取周围环境深度信息，从而绘制出环境的二维地图。通过激光雷达数据，机器人可以进行自主定位和导航，避障等功能。根据 YDLIDAR X3 的通信协议，使用 UART 对该模块进行配置和初始化。根据其数据手册对获得的数据进行处理从而获得角度以及对应的深度信息，再填入 ros 中的 sensor_msgs__msg__LaserScan 消息类型中。

mpu6050：根据 MPU6050 的通信协议和寄存器操作手册，编写驱动程序以读取和解析 MPU6050 的数据。通过 I2C 总线与 MPU6050 进行通信，以及数据解析函数，将读取的原始数据转换为加速度和角速度等物理量。将计算得到的 imu 数据填入 ros 中的 sensor_msgs__msg__Imu 消息类型中。

DHT11：设置一个独立线程，读取 DHT11 传回的温湿度的数据。在线程中，首先初始化 DHT11，包括设置 GPIO 口和设置数据传输的协议。编写 dht11 驱动代码，通过调用 DHT11 的读取函数，获取原始温湿度数据，并对其进行处理，将原始数据转换为温度和湿度的数据。

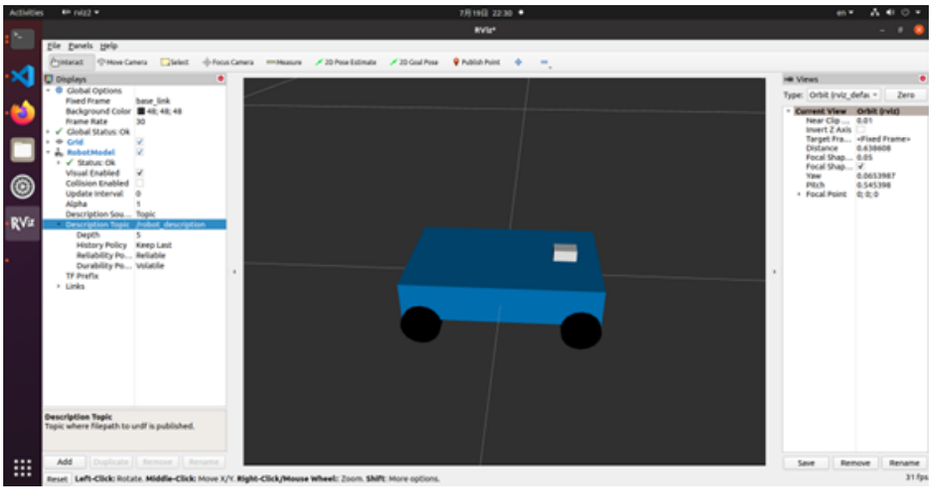

图 2-7 URDF 建立小车模型

ubuntu ROS 终端：建模，建图，导航，如图 2-7 所示

图 2-8 Cartographer 建图

如图 2-8 所示，/cartographer_node 节点从/scan 和/odom 话题接收数据进行计算，输出/submap_list 数据。Art-pi 通过 micro-ros 发布激光雷达的环境深度信息/scan 以及 imu 的机器人姿态信息。通过计算 Art-pi 发布来的信息，Cartographer 建立周围环境地图，并提供给 Nav2 模块进行机器人的移动导航。

Nav2 导航框架: Nav2 实现以安全的方式让移动机器人从 A 点移动到 B 点。Nav2 应用机器人导航，在这个过程中需要完成动态路径规划、计算电机的速度、避免障碍、恢复行为。Nav2 模块发布 Art-pi 订阅的/cmd_vel twist 类型信息，将移动机器人的速度、角速度欲值发送给 Art-pi。经过对此数据的运动学逆解，实现给定线速度和角速度，计算出轮子所需转速。

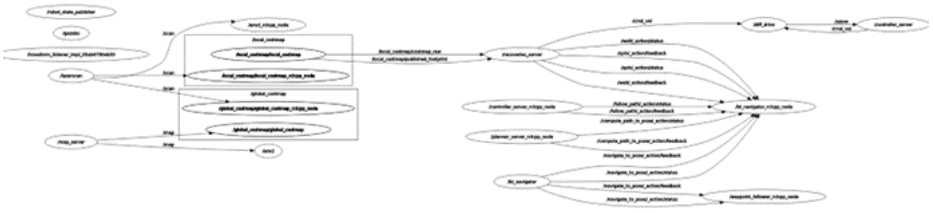

图 2-9 最终 ros rqt 关系图

2.2.2 AI 部分

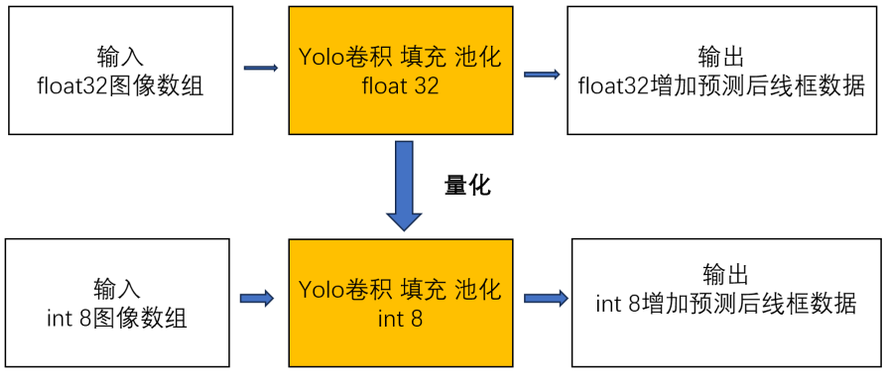

图 2-10 Yolo 模型部署

模型部署：我们巧妙地运用 Darknet，这是一款用 C 和 CUDA 编写的优秀神经网络框架，专门用于实现 YOLO Fastest 模型的构建、训练以及验证。Darknet 拥有低延迟的优势，同时是首批支持 YOLO 算法的框架之一。在模型经历了精细的量化和剪枝过程之后，我们成功地将模型转换为 Keras 和 TensorFlow Lite 兼容的模型格式，即 h5 和 tflite。进一步地，借助于 STM32 Cube AI 和 RT AK 工具的强大功能，我们将模型无缝地部署到 RT Thread，一个具有可伸缩微内核实时操作系统特性的开源嵌入式系统，从而确保了模型在实际运行环境中的高效性能和稳定性。

图 2-11 算法细节

模型推理：我们选用 C 语言作为编程工具，精确地实现了 YOLO 模型的解码流程。在模型推理的过程中，我们计算了各个检测结果的置信度，并根据项目需求设定了适宜的阈值。为了得到最优的输出结果，我们采用了非极大值抑制技术，有效地消除了重复和冗余的检测框，进一步提升了模型的推理准确度。如图 2-11

所示，输入的原始视频流首先经过 RGB 三通道灰度化处理，随后进行双线性插值将摄像头数据转换为神经网络模型所需的输入格式。之后，rt_ai_find 函数查找适配的 AI 模型，rt_ai_init 函数对找到的 AI 模型进行必要的参数配置以及初始化。最后，rt_ai_run 函数执行模型推理，输出优化后的图片以及精确的分类结果，展现了实时计算和高级视觉处理能力的完美结合。

2.2.3 用户界面部分

本产品用户界面共有微信小程序，网页服务器，onenet 物联网平台，ubuntu ROS 终端以及 LCD。

图 2-12 微信小程序

微信小程序：（需打开手机 wifi、蓝牙）如图用户可以通过微信扫码或搜索进入 Art-pi 蓝牙配网小程序。该小程序通过蓝牙给用户提供一个便利的方式来配网；同时 fal 以及 easyflash 的运用，将 wifi 数据储存在掉电不丢失的 flash 中，使得 wifi 能自动重连，用户使用产品时只需要配网一次，简化了用户联网过程。

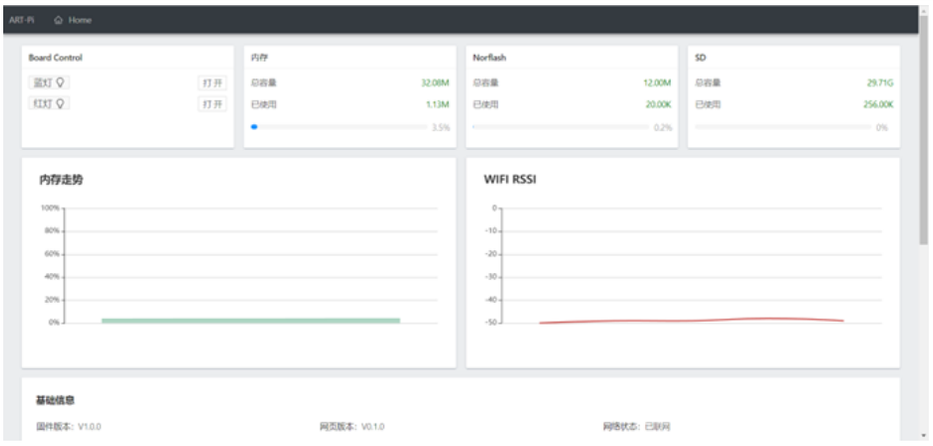

图 2-13 网页服务器

网页服务器：微信小程序联网成功后，可在手机或电脑的浏览器访问我们的 web 网页服务器。在该网页中显示产品此时内存占用、SD 卡容量、网络质量等信息。同时，该 web 网页允许用户对机器人进行一定的控制，实现了用户与机器人之间的信息传递和控制指令的转发。

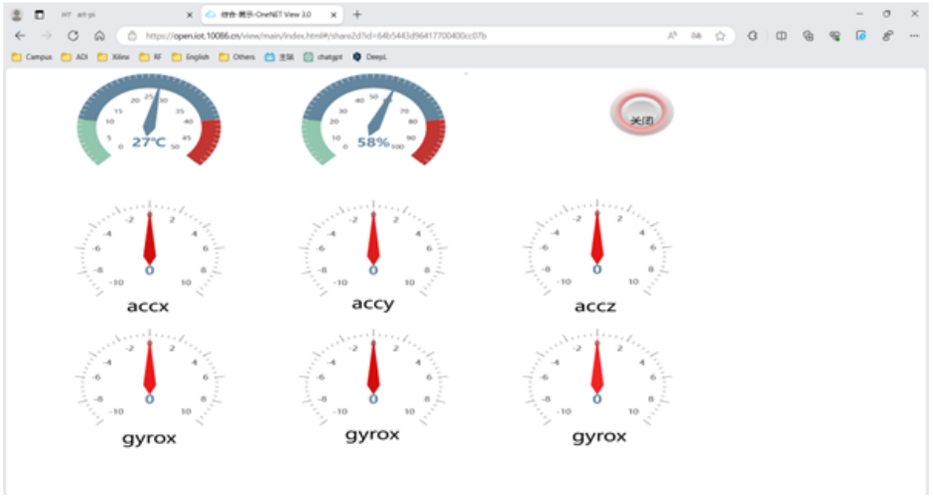

图 2-14 OneNet

OneNet 物联网平台：与 web 网页服务器不同，OneNet 物联网平台并不局限于局域网。通过该物联网平台可以连接机器人的传感器和控制模块，收集机器人的传感器数据，同时也可以通过物联网平台发送指令给机器人，实现远程控制。在 Art-pi 连接网络后，运用 mqtt 协议将各类数据，如各种传感器数据，上传到 onenet 云端，并使用可视化服务，使各类数据更加清晰。并通过 onenet 的 API 订阅来自远端的。

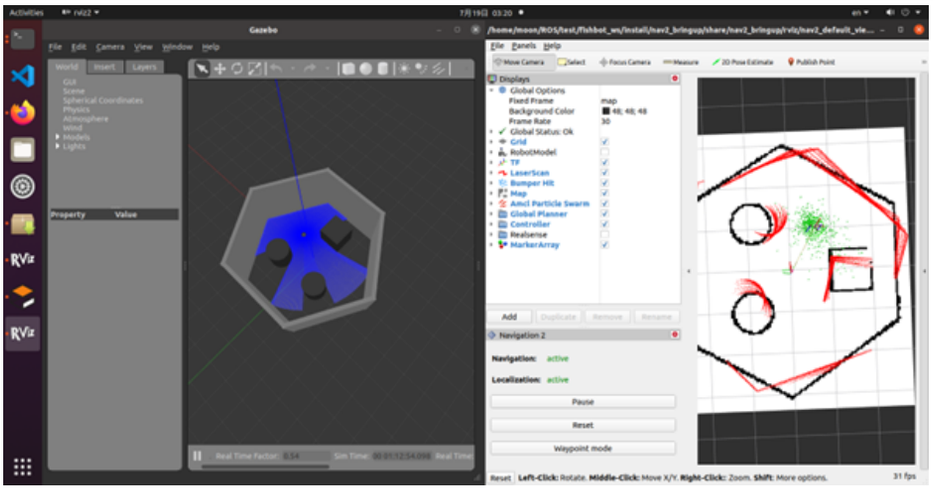

图 2-15 slam 建图

图 2-16 slam 导航

Ubuntu 终端：Ubuntu 终端是 ROS 机器人建图导航的控制和管理终端。通过 ROS 的 cartographer 功能包对机器人周围环境的建图、Nav2 功能包给机器人提供导航服务。在 ubuntu 终端上，可以配置机器人巡航路径。当摄像头识别到垃圾时，它会将图像数据传送到 ubuntu。ubuntu 通过调用 cv2 库对 Art-pi 传来的图片信息进行处理和显示。

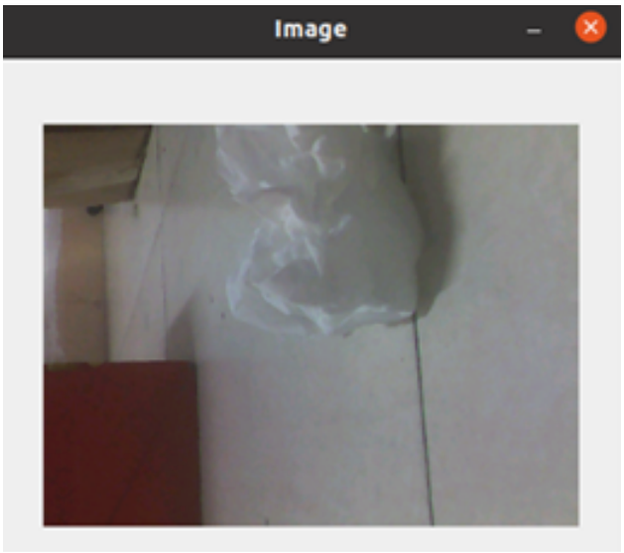

图 2-17 LCD

LCD：ili9488 驱动的 spi LCD 屏幕上显示此时 gc0328c 摄像头传回的图像数据，并显示系统各个传感器数据。同时，ft6236 驱动的触摸屏，实现系统于用户的直接交互。通过 LCD 模块，我们的产品在没有网络支持的环境中，也能将系统的各个参数显示给用户，同时能接收来自用户的命令、配置。

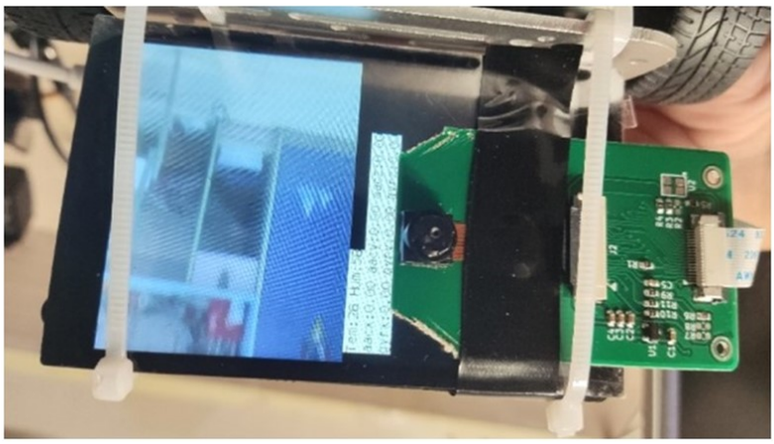

图 2-18 文件系统

文件系统 & usb-otg 读卡器: 在 Art-pi 上搭建了虚拟文件系统，可以通过 ls、cd、cp 等命令对 SD 卡、flash 中的文件进行操作。当 AI 识别到垃圾污物，系统将会把图像数据保存到 SD 卡中。用户可以通过读卡器或者 OTG 接口读取污物照片来进行二次确认。

第三章 完成情况及性能参数

多终端用户操作界面的各类功能基本实现，包括微信小程序，网页服务器，onenet 物联网平台，ubuntu ROS 终端以及 LCD 显示。没有网络可以通过 LCD 屏幕对产品进行操作，局域网内可以通过 web 网页服务器，外网能够通过 onenet 物联网平台接入机器人系统。全面的多终端用户操作界面使得用户在使用机器人时能够获得更加丰富、便捷、灵活的体验和服务。

在使用 ROS，进行 slam 建图，坐标误差在 ±5cm 内，延时在 38ms 左右，可以进一步改进。同时机器人的 imu 模块读取的数据没有进行滤波处理，之后添加卡尔曼滤波等。同时，产品使用 UDP 向 ubuntu 上位机传输数据，在传输大容量数据如激光雷达深度数据包或传输图像等数据时，UDP 传输数据不可靠。由于数据量较大，丢包的可能性增加。另外，UDP 在传输过程中，数据包的到达顺序不受保证，可能导致数据包顺序混乱。

在实验中，我们选择了使用 Yolo-Fastest 模型。Yolo-Fastest 是 Yolo 系列模型的一个变种，这个模型被特别设计来解决在实时目标检测任务中的效率问题。与原版的 Yolo 模型相比，Yolo-Fastest 具有更小的模型规模和更快的推断速度，这使得它非常适合在资源受限的环境中使用。

Yolo-Fastest 使用更少的计算资源实现了相当于其它模型的性能。据测试，Yolo-Fastest 在单张图像的目标检测上，其延时低至几毫秒，而且只需要 0.23B FLOPS（浮点运算次数）即可完成运算。这在很大程度上优化了我们的计算资源分配，提高了整体的运行效率。

Yolo-Fastest 的平均精度为 61.02%，这在实时目标检测任务中算是非常不错的表现。这样的精度可以满足大部分的应用场景，尤其是在那些对实时性需求更高，对精度要求相对较低的情况下。

尽管 Yolo-Fastest 的性能在许多方面都很突出，但我们仍然在使用过程中发现了一些可以改进的地方。比如，它在检测小目标和处理复杂背景时，其性能仍有待提高。为了进一步优化模型性能，我们将考虑对模型进行一些微调操作，如改

变模型架构或者对训练数据进行增强等。

第四章 总结

4.1 扩展之处

1. 添加更多传感器支持：当前系统中只包括了温湿度传感器的监测和上传，但我们可以通过添加更多的传感器模块来监测其他环境参数，如光照强度、气压等，以获取更全面的数据。

2. 机器人控制的多样性：除了小车，可以考虑支持其他类型的机器人或设备，如无人机、机械臂等。通过扩展控制算法和硬件接口，使系统能够控制不同类型的机器人或设备。

3. 用户定制化和扩展性：提供更加用户友好的界面和工具，使用户能够自定义和扩展系统的功能。例如，提供插件或模块化的设计，以便用户可以添加自己的功能模块。

4.2 心得体会

1. 充分利用多线程能力：通过使用 RT-Thread 多线程的特性，我们能够同时处理多个任务，提高系统的效率和性能。合理分配不同任务的线程，可以避免阻塞和提高响应速度，使系统能够更好地利用 MCU 的资源。

2. 功能的整合和协作：在系统中集成多个功能模块时，需要考虑模块之间的协作和数据交互。通过合理的设计和接口定义，我们能够实现温湿度数据的采集、上传和显示，小车的控制和导航，以及图像识别和地图标记的功能。模块之间的协作能够使系统更加完整和高效。

3. 项目实践的价值：通过实际的项目实践，我们能够将之前学到的知识应用到实际情境中，加深对技术的理解和掌握。项目实践还能够提升解决问题的能力、团队协作的能力和创新思维，为日后的开发工作奠定坚实的基础。

第五章 附录

```
rgb2gray(lcd->lcd_info.framebuffer, input_gray, 320, 240);
bilinera_interpolation(input_gray, 240, 320, input_gray_160, 160, 160);
_itof(input_buf, input_gray_160, 160 * 160, 255.0);
rt_ai_run(person_d, NULL, NULL);
boxs = (yolo_box *)rt_ai_output(person_d, 0);
yolo_decode((float *)boxs);
do_nms_sort(&r1, boxs);
p = boxs;
for (int i = 0; i < 125; i++)
{
    p = &boxs[i];
    if ((p->class_score * p->objectness) > 0.2)
    {
        _x1 = (int)(p->x * 320 - (p->w * 320 * 0.5));
        _x1 = _x1 > 0 ? _x1 : 1;
        _x1 = _x1 < 320 ? _x1 : 319;
        _y1 = (int)(p->y * 240 - (p->h * 160 * 1.5 * 0.5));
        _y1 = _y1 > 0 ? _y1 : 1;
        _y1 = _y1 < 240 ? _y1 : 239;
        _x2 = (int)(p->x * 320 + (p->w * 320 * 0.5));
        _x2 = _x2 > 0 ? _x2 : 1;
        _x2 = _x2 < 320 ? _x2 : 319;
        _y2 = (int)(p->y * 240 + (p->h * 160 * 1.5 * 0.5));
        _y2 = _y2 > 0 ? _y2 : 1;
        _y2 = _y2 < 240 ? _y2 : 239;
        lcd_draw_rectangle(_x1, _y1, _x2, _y2);
    }
}
DCMI_Start();
```

图 5-1 AI 模型核心代码

```
void bilinera_interpolation(rt_uint8_t *in_array, short height, short width,
                            rt_uint8_t *out_array, short out_height, short out_width)
{
    double h_times = (double)out_height / (double)height,
           w_times = (double)out_width / (double)width;
    short x1, y1, x2, y2, f11, f12, f21, f22;
    double x, y;

    for (int i = 0; i < out_height; i++)
    {
        for (int j = 0; j < out_width; j++)
        {
            x = j / w_times;
            y = i / h_times;

            x1 = (short)(x - 1);
            x2 = (short)(x + 1);
            y1 = (short)(y + 1);
            y2 = (short)(y - 1);
            f11 = is_in_array(x1, y1, height, width) ? in_array[y1 * width + x1] : 0;
            f12 = is_in_array(x1, y2, height, width) ? in_array[y2 * width + x1] : 0;
            f21 = is_in_array(x2, y1, height, width) ? in_array[y1 * width + x2] : 0;
            f22 = is_in_array(x2, y2, height, width) ? in_array[y2 * width + x2] : 0;
            out_array[i * out_width + j] = (rt_uint8_t)(((f11 * (x2 - x) * (y2 - y)) +
                                                         (f21 * (x - x1) * (y2 - y)) +
                                                         (f12 * (x2 - x) * (y - y1)) +
                                                         (f22 * (x - x1) * (y - y1))) /
                                                        ((x2 - x1) * (y2 - y1)));
        }
    }
}
```

图 5-2 双线性插值

```
void Decode(uint8_t len)
{
    uint8_t index = 0;
//    FSA = ((Rx_buffer_temp[5] << 8) + Rx_buffer_temp[4]) >> 7;
//    LSA = ((Rx_buffer_temp[7] << 8) + Rx_buffer_temp[7]) >> 7;
//    rt_kprintf("FSA =%d,LSA =%d\r\n ", FSA, LSA);

    scan_msg.angle_min = (float)(((Sensor_Data[5] << 8) | Sensor_Data[4]) >> 1) / 64;
    scan_msg.angle_max = (float)(((Sensor_Data[7] << 8) | Sensor_Data[6]) >> 1) / 64;
    if(scan_msg.angle_max < scan_msg.angle_min)
        scan_msg.angle_increment = (scan_msg.angle_max + 360 - scan_msg.angle_min) / len;
    else
        scan_msg.angle_increment = (scan_msg.angle_max - scan_msg.angle_min) / len;
//    rt_kprintf("FSA = %d,LSA = %d\r\n ", scan_msg.angle_min, scan_msg.angle_max);

    scan_msg.scan_time = (float)(current_time - start_scan_time) * 1e-3;
    scan_msg.time_increment = scan_msg.scan_time / len;
    scan_msg.range_min = 0.05;
    scan_msg.range_max = 5;

    while (index < len)
    {
        if (((Sensor_Data[index * 2 + 10] & 0x03) == 2) || ((Sensor_Data[index * 2 + 10] & 0x03) == 3))
            data[index] = 0.0;
        else
            data[index] = (float)((Sensor_Data[index * 2 + 10] >> 2) + (Sensor_Data[index * 2 + 11] << 6)) / 100.0;
        index++;
    }
    scan_msg.header.stamp.sec = current_time * 1e-3;
    scan_msg.header.stamp.nanosec = current_time - scan_msg.header.stamp.sec * 1000;
    scan_msg.ranges.data = (float *)rt_malloc(sizeof(float) * len);
    memcpy(scan_msg.ranges.data, data, len);
    scan_msg.ranges.capacity = len;
    scan_msg.ranges.size = len;
}
```

图 5-3 YDLIDAR X3 驱动以及 ROS /scan 数据的填充

```
if ((data->disk->flag & RT_DEVICE_FLAG_STANDALONE) &&(!(data->disk->open_flag & RT_DEVICE_OFLAG_OPEN)))
{
        if(rt_device_open(data->disk, RT_DEVICE_OFLAG_RDWR) != RT_EOK)
        {
            rt_kprintf("disk open error\n");
            return -RT_ERROR;
        }
}
```

图 5-4 优化 rt-thread 内核，解决模拟 U 盘和文件系统不能共存

```c
static rt_err_t drv_pwm_enable(TIM_HandleTypeDef *htim, struct rt_pwm_configuration *configuration, rt_bool_t enable)
{
    /* Converts the channel number to the channel number of Hal library */
    rt_uint32_t channel = 0x04 * (configuration->channel - 1);

    if(!configuration->complementary)
    {
        if (!enable)
        {
            HAL_TIM_PWM_Stop(htim, channel);
        }
        else
        {
            HAL_TIM_PWM_Start(htim, channel);
        }
    }
    else if(configuration->complementary)
    {
        if (!enable)
        {
            HAL_TIMEx_PWMN_Stop(htim, channel);
        }
        else
        {
            HAL_TIMEx_PWMN_Start(htim, channel);
        }
    }

    return RT_EOK;
}
```

图 5-5 RT-thread 内核 pwm 驱动优化，增加对 CHxN 通道输出的支持

```c
if (sensors & INV_XYZ_GYRO)
{
    gyrox = 2000 * (double)gyro[0] / 32768;
    gyroy = 2000 * (double)gyro[1] / 32768;
    gyroz = 2000 * (double)gyro[2] / 32768;
}
if (sensors & INV_XYZ_ACCEL)
{
    aacx = 2 * G_CD * (double)accel[0] / 32768;
    aacy = 2 * G_CD * (double)accel[1] / 32768;
    aacz = 2 * G_CD * (double)accel[2] / 32768;
}
if (sensors & INV_WXYZ_QUAT)
{
    q0 = quat[0] / q30; // q30格式转换为浮点数
    q1 = quat[1] / q30;
    q2 = quat[2] / q30;
    q3 = quat[3] / q30;
}
```

图 5-6 mpu6050 数据处理：三轴陀螺仪，三轴加速度，dmp 四元数

```python
#!/usr/bin/env python3
import rclpy,cv2
from rclpy.node import Node
from sensor_msgs.msg import Image
from cv_bridge import CvBridge

class ImageSubscriber(Node):
    def __init__(self):
        super().__init__('image_subscriber')
        self.subscription = self.create_subscription(
            Image,
            'pub_image',
            self.image_callback,
            10
        )
        self.bridge = CvBridge()

    def image_callback(self, msg):
        try:
            # 将ROS图像消息转换为OpenCV图像格式
            cv_image = self.bridge.imgmsg_to_cv2(msg, 'bgr8')
        except Exception as e:
            self.get_logger().error('Error converting ROS Image to OpenCV image: %s' % str(e))
            return
        # 显示图像
        cv2.imshow('Image', cv_image)
        cv2.waitKey(1)

def main(args=None):
    rclpy.init(args=args)
    image_subscriber = ImageSubscriber()
    rclpy.spin(image_subscriber)
    image_subscriber.destroy_node()
    rclpy.shutdown()

if __name__ == '__main__':
    main()
```

图 5-7 ubuntu ROS 视频流接收节点

```
<?xml version="1.0"?>
<robot name="smartcar">
    <link name="base_link">
        <visual>
            <geometry>
                <box size="0.25 .16 .05"/>
            </geometry>

            <origin rpy="0 0 1.57075" xyz="0 0 0"/>

            <material name="blue">
                <color rgba="0 .5 .8 1"/>
            </material>
        </visual>
    </link>

    <link name="right_front_wheel">
        <visual>
            <geometry>
                <cylinder length=".02" radius="0.025"/>
            </geometry>

            <material name="black">
                <color rgba="0 0 0 1"/>
            </material>
        </visual>
    </link>

    <joint name="right_front_wheel_joint" type="continuous">
        <axis xyz="0 0 1"/>
        <parent link="base_link"/>
        <child link="right_front_wheel"/>
        <origin rpy="0 1.57075 0" xyz="0.08 0.1 -0.03"/>
        <limit effort="100" velocity="100"/>
        <joint_properties damping="0.0" friction="0.0"/>
    </joint>
```

图 5-8 四轮机器人 urdf 描述文件

```
<link name="right_back_wheel">
<visual>
    <geometry>
    <cylinder length=".02" radius="0.025"/>
    </geometry>
    <material name="black">
    <color rgba="0 0 0 1"/>
    </material>
</visual>
</link>

<joint name="right_back_wheel_joint" type="continuous">
    <axis xyz="0 0 1"/>
    <parent link="base_link"/>
    <child link="right_back_wheel"/>
    <origin rpy="0 1.57075 0" xyz="0.08 -0.1 -0.03"/>
    <limit effort="100" velocity="100"/>
    <joint_properties damping="0.0" friction="0.0"/>
</joint>

<link name="left_front_wheel">
    <visual>
        <geometry>
            <cylinder length=".02" radius="0.025"/>
        </geometry>
        <material name="black">
            <color rgba="0 0 0 1"/>
        </material>
    </visual>
</link>
```

图 5-9 四轮机器人 urdf 描述文件

```
<joint name="left_front_wheel_joint" type="continuous">
    <axis xyz="0 0 1"/>
    <parent link="base_link"/>
    <child link="left_front_wheel"/>
    <origin rpy="0 1.57075 0" xyz="-0.08 0.1 -0.03"/>
    <limit effort="100" velocity="100"/>
    <joint_properties damping="0.0" friction="0.0"/>
</joint>

<link name="left_back_wheel">
    <visual>
        <geometry>
            <cylinder length=".02" radius="0.025"/>
        </geometry>
        <material name="black">
            <color rgba="0 0 0 1"/>
        </material>
    </visual>
</link>

<joint name="left_back_wheel_joint" type="continuous">
    <axis xyz="0 0 1"/>
    <parent link="base_link"/>
    <child link="left_back_wheel"/>
    <origin rpy="0 1.57075 0" xyz="-0.08 -0.1 -0.03"/>
    <limit effort="100" velocity="100"/>
    <joint_properties damping="0.0" friction="0.0"/>
</joint>
```

图 5-10 四轮机器人 urdf 描述文件

```
    <link name="head">
        <visual>
            <geometry>
                <box size=".02 .03 .03"/>
            </geometry>
            <material name="white">
                <color rgba="1 1 1 1"/>
            </material>
        </visual>
    </link>

    <joint name="tobox" type="fixed">
        <parent link="base_link"/>
        <child link="head"/>
        <origin xyz="0 0.08 0.025"/>
    </joint>
</robot>
```

图 5-11 四轮机器人 urdf 描述文件

参考文献

[1] *RT-Thread Club Article*, Accessed on 20th July, 2023, https://club.rt-thread.org/ask/article/87c3db5fc4f12a37.html.

[2] *RT-Thread Studio*, Accessed on 20th July, 2023, https://www.rt-thread.org/document/site/#/development-tools/rtthread-studio/README.

[3] *RT-Thread Standard*, Accessed on 20th July, 2023, https://www.rt-thread.org/document/site/#/rt-thread-version/rt-thread-standard/README.

[4] *Art-Pi Documentation*, Accessed on 20th July, 2023, https://Art-pi.gitee.io/website/docs/#/.

[5] *Art-Pi Search on RT-Thread Club*, Accessed on 20th July, 2023, https://club.rt-thread.org/ask/search.html?q=Art-pi.

[6] *RT-AK on GitHub*, Accessed on 20th July, 2023, https://github.com/RT-Thread/RT-AK.

[7] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A., *You Only Look Once: Unified, Real-Time Object Detection*, In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, (2016), pp. 779-788.

[8] Redmon, J. and Farhadi, A., *YOLOv3: An Incremental Improvement*, arXiv preprint, (2018), arXiv:1804.02767.

[9] Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., Liu, D., Mu, Y., Tan, M., Wang, X., Liu, W., and Xiao, B., *Deep High-Resolution Representation Learning for Visual Recognition*, IEEE transactions on pattern analysis and machine intelligence, (2020).