恩智浦机器人开发平台AI ROBOT ™软件系统解析

张晓东
资深系统工程师

2021年10月

SECURE CONNECTIONS
FOR A SMARTER WORLD

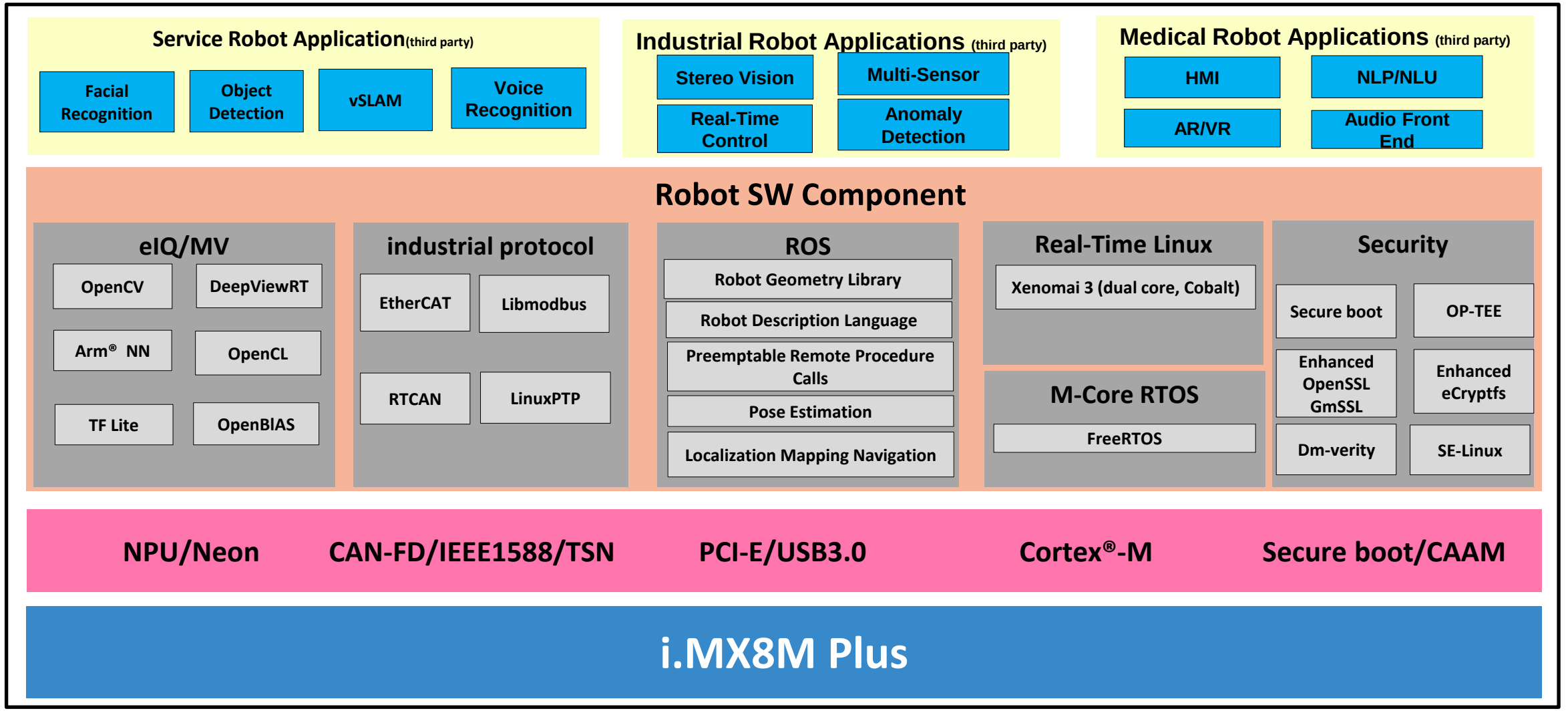
PUBLIC

讨论内容

- 软件架构
- eIQ和机器视觉
- 机器人操作系统ROS
- 实时Linux Xenomai和工业网络
- 可选的Linux系统(Yocto， Ubuntu，自有系统)
- Demo介绍

软件架构

机器人平台软件架构



Highlight : Smart, Security, Real-Rime, Reliability

EIQ和机器视觉

eIQ™ 应用于边缘智能的工具箱和推理引擎

机器学习库和开发工具
面向恩智浦的应用处理器和MCU

部署开源推理引擎

推理引擎的集成与优化：

- i.MX MPU: TensorFlow Lite, Arm NN, ONNX, OpenCV
- i.MX RT MCU: Glow, TensorFlow Lite, Arm CMSIS-NN

经典的ML算法，如支持向量机（SVM）、决策树和随机森林

集成到Yocto Linux BSP 和 MCUXpresso SDK

NXP提供的免费工具，无需单独的SDK或可下载的软件发布：

- i.MX: Yocto里的meta-imx-ml层
- MCU: MCUXpresso SDK的中间件

支持易用性的各种文档资料

端到端的演示示例，例如摄像头 → 推理引擎

文档: eIQ 用户手册，软件发行说明，Demo用户手册

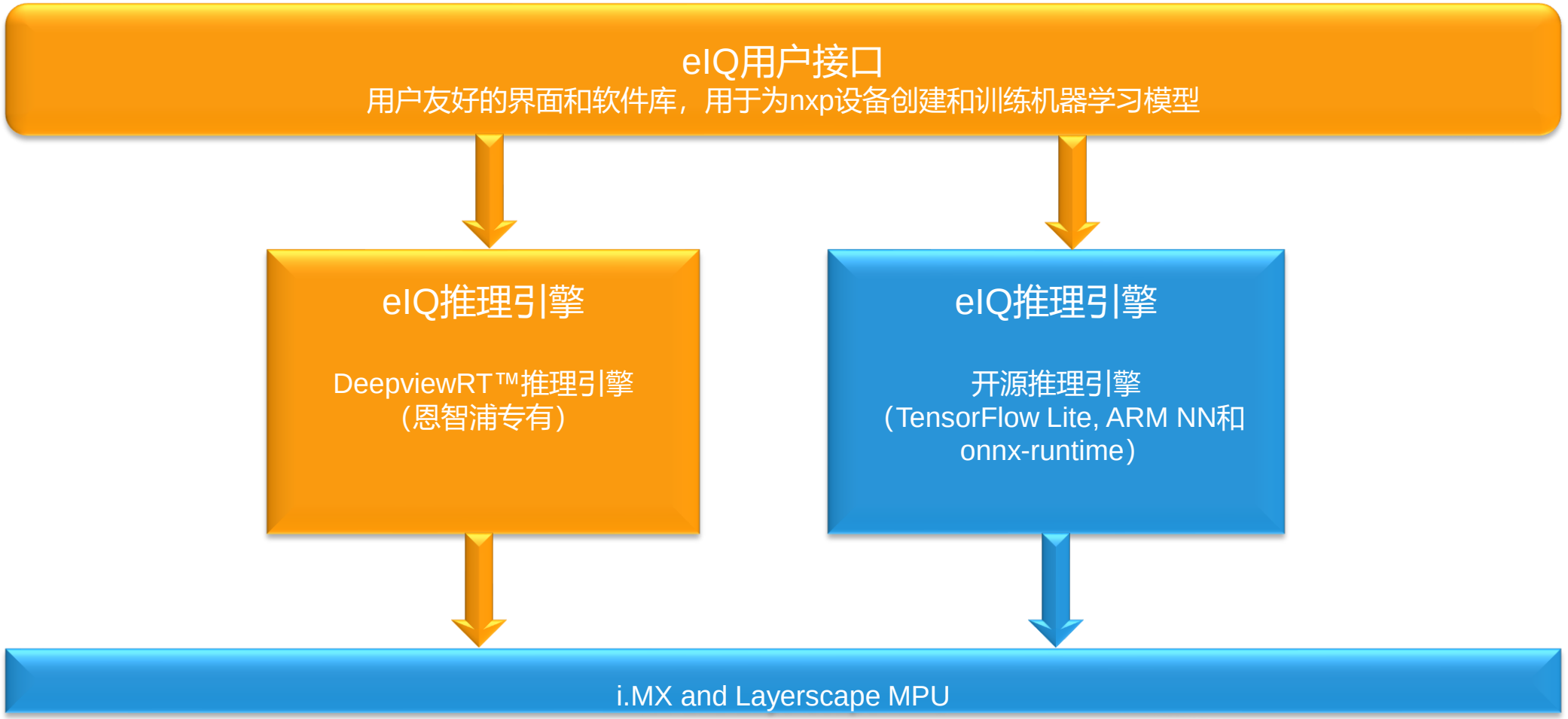
基于流行神经网络框架的预训练模型的使用指南，例如 TensorFlow, PyTorch

技术培训资料，例如讲座，动手操作，视频

https://www.nxp.com/design/software/development-software/eiq-ml-development-environment:EIQ

https://community.nxp.com/community/eiq/overview

eIQ™机器学习软件开发架构



NXP交付产品

NXP交付，包含第三方IP

EDGEVerse eIQ™机器学习软件开发环境

嵌入式开发人员

数据分析师

机器学习专家

PyTorch Caffe2 ONNX

eIQ 工具集

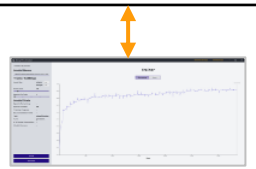
eIQ 用户接口

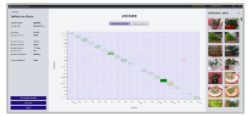
模型优化工具

提供用户自己的数据处理流程


数据整理


模型选择


模型训练、量化、优化


模型验证、部署

提供用户自己的模型工作流程

PyTorch Caffe2 ONNX

eIQ inference

Applications running on EdgeVerse processors

异常检测 | 声音识别 | 人脸识别 | 多目标分类 | 复杂情况诊断

eIQ inference with…

TFLite | TFMicro | Glow | DeepViewRT™

Arm® Cortex®-M, DSP

TFLite | Arm NN | ONNX Runtime | DeepViewRT™

Arm Cortex-A, GPU, DSP, NPU, microNPU

什么是机器视觉？

- 机器视觉是将数据从静态图片或摄像机捕获的视频转换为决策或新描述形式的过程。这种转换是为了实现某些特定目的，如人脸识别的门禁系统。

- 机器视觉是多学科的交叉技术，机器视觉系统通常包括图像处理、控制、电光源照明、光学成像、传感器、模拟与数字视频技术、计算机软硬件技术(图像增强和分析算法及硬件、摄像头接口、 系统软件等)

- 机器视觉算法：图像处理，特征检测与匹配，图像分割与拼接，对象识别

- 机器视觉应用：光学字符识别，产品缺陷检测，新零售，医学成像，汽车安全(ADAS)，安防监控，等等。

什么是OPENCV?

- 非常流行的开源机器视觉库。1999年，英特尔的加里·布拉德斯基发布了OpenCV，为计算机视觉和人工智能提供坚实的基础架构，并希望能够加速此领域每个人的工作

- 底层主要由C和C ++编写，同时提供模块支持Python，Java，MATLAB和其他语言。软件平台支持Linux，Windows和Mac OSX和移动平台Android和iOS

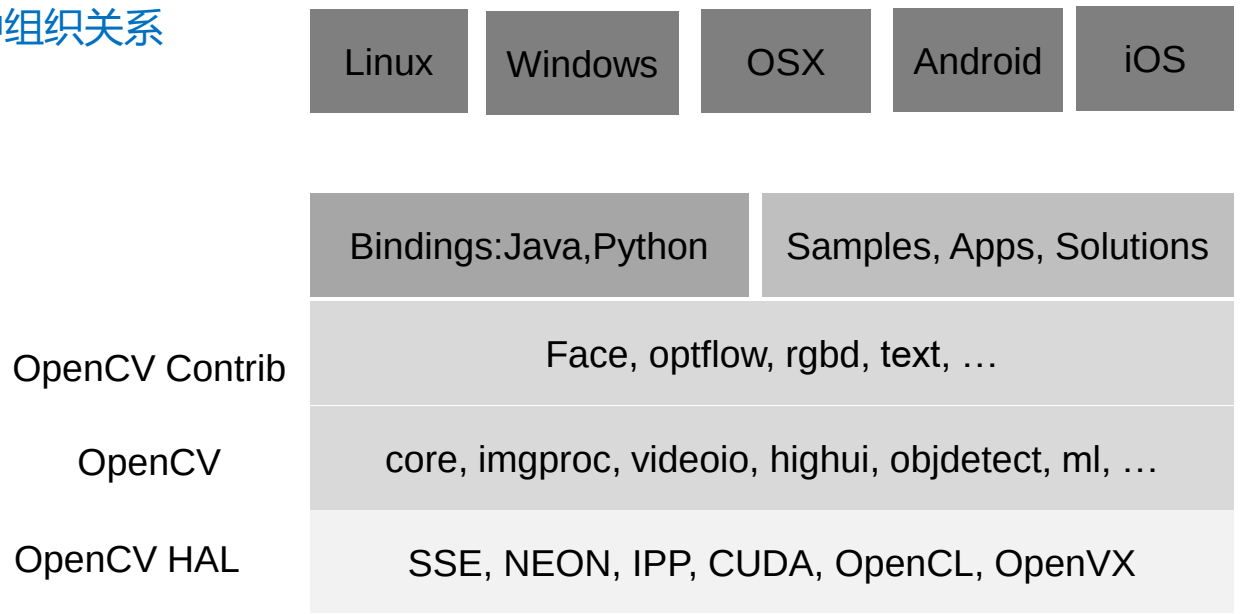
- 在x86平台可以使用英特尔的IPP库加速，ARM平台使用Neon加速，此外支持CUDA(英伟达GPU)、OpenCL和OpenVX加速

- 包含500多种功能，涵盖了包括工厂产品检测，医学成像，安全性，用户UI，相机校准，立体视觉和机器人技术。 还包含通用机器学习库（ML模块，专注于统计模式识别和聚类）和深度学习库（ DNN模块，专注于神经网络计算）

OPENCV的组成结构以及包含的模块

- OpenCV 是由很多模块组成的，这些模块可以分成很多层
  - 最底层是基于硬件加速层（HAL）的各种硬件优化。
  - 再上一层是OpenCV的主要模块，以及opencv_contrib 模块所包含的 由其他开发人员所贡献的代码，其包含大多数高层级的函数功能。这些就是OpenCV的核心。
  - 接下来是语言绑定和示例应用程序。
  - 处于最上层的是 OpenCV 和各个操作系统的交互
- AI Robot软件平台提供OpenCV3.x(ROS1)和OpenCV4.x(ROS2)

- 下图显示了 OpenCV 的这种组织关系



机器人操作系统ROS

ROS简介

- ROS历史
  - ROS系统是起源于2007年斯坦福大学人工智能实验室的项目与机器人技术公司Willow Garage(柳树车库)的个人机器人项目（Personal Robots Program）之间的合作，2008年之后就由Willow Garage来进行推动。其后由于ROS的一些不足以及技术进步和新的需求，开源社区着手开发ROS2.0。2015年8月第一个ROS2的alpha版本落地，2016年12月ROS2的beta版本正式发布，目前ROS2最新版本是Galactic。
- ROS1

  ROS的运行架构是一种使用ROS通信模块实现模块间P2P的松耦合的网络连接的处理架构，它执行若干种类型的通讯，包括基于服务的同步RPC（远程过程调用）通讯、基于Topic的异步数据流通讯，还有参数服务器上的数据存储。
  - 点对点设计
  - 多语言支持
  - 精简与集成
  - 工具包丰富
  - 免费并且开源
- ROS2
  - 支持操作系统包括Linux (Ubuntu 、Debian 、Red Hat 、Fedora)、Windows、Mac、RTOS
  - ROS2的通讯系统是基于DDS
  - 更新的发布/订阅模型

  https://www.guyuehome.com/ 古月居

  http://wiki.ros.org/cn/ROS/Introduction 中文ROS1文档

  http://docs.ros.org/en/galactic/ 英文ROS2文档

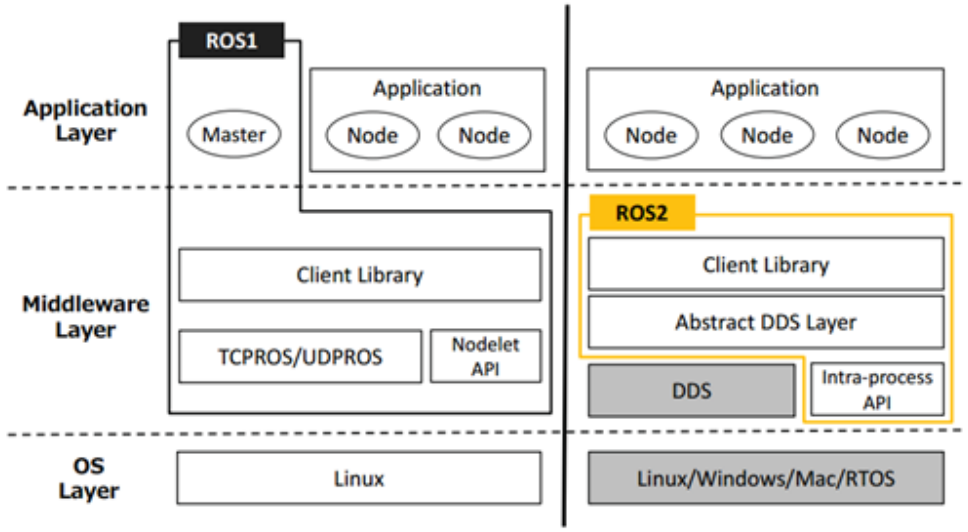


Figure 1: ROS1/ROS2 architecture.

AI ROBOT上的ROS

- 如何配置ROS编译环境?
  - 当前支持ROS1 (kinetic, melodic)和ROS2(dashing, eloquent, foxy)。下一版5.10加入ROS1 noetic和ROS2 galactic, rolling支持
    - DISTRO=imx-robot-xwayland MACHINE=imx8mp-ai-robot source setup-imx-robot.sh -r kinetic -b imx8mp-ai-robot-kinetic
    - DISTRO=imx-robot-xwayland MACHINE=imx8mp-ai-robot source setup-imx-robot.sh -r melodic -b imx8mp-ai-robot-melodic
    - DISTRO=imx-robot-xwayland MACHINE=imx8mp-ai-robot source setup-imx-robot.sh -r dashing -b imx8mp-ai-robot-dashing
    - DISTRO=imx-robot-xwayland MACHINE=imx8mp-ai-robot source setup-imx-robot.sh -r eloquent -b imx8mp-ai-robot-eloquent
    - DISTRO=imx-robot-xwayland MACHINE=imx8mp-ai-robot source setup-imx-robot.sh -r foxy -b imx8mp-ai-robot-foxy

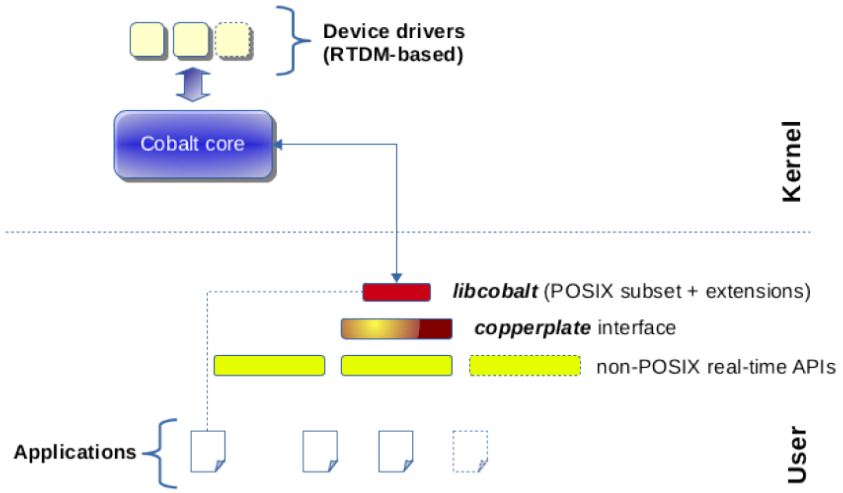
- 不同image配置介绍
  - bitbake imx-robot-sdk （包含大部分可以在Yocto系统运行的ROS包以及全部i.MX软件组件和GNU编译工具链，git，cmake。用户可以在AI Robot板上编译和安装自己的ROS包）
  - bitbake imx-robot-agv （包含最核心的ROS包，sensor-msgs，socketcan-bridge，思岚激光雷达ROS驱动以及orb-slam2 demo和大部分i.MX软件组件）
  - bitbake imx-robot-system （包含机器人系统需要的基本ROS包， Igh-Ethercat软件以及全部i.MX软件组件）
  - bitbake imx-robot-system-rt （在imx-robot-system基础上加入实时Linux支持，可以选择Xenomai或者Preempt-RT）
  - bitbake imx-robot-core （包含最核心的ROS包和基本i.MX软件组件）
  - bitbake imx-robot-core-rt （在imx-robot-core基础上加入实时Linux支持，可以选择Xenomai或者Preempt-RT ）

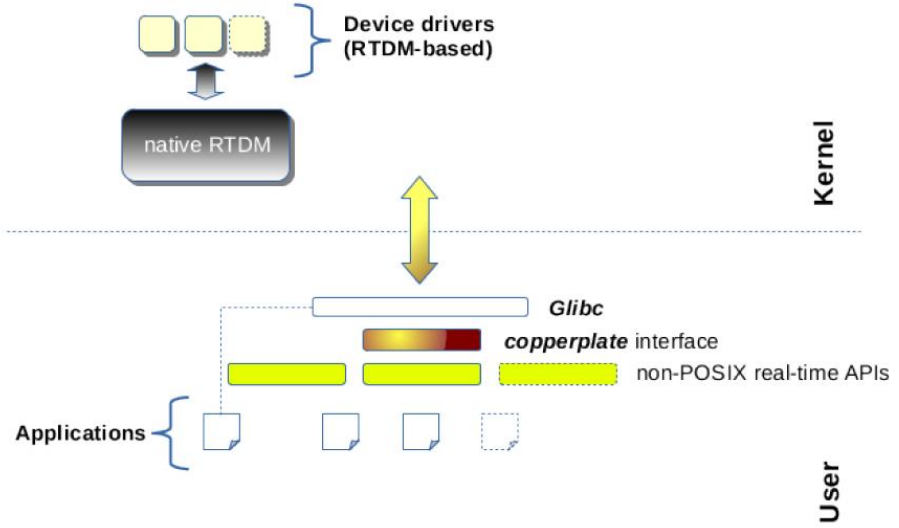
实时LINUX XENOMAI和工业网络

Xenomai 3介绍

- 由于Linux系统一开始就被设计成GPOS（通用操作系统），它的目的是构建一个完整、稳定的开源操作系统，尽量缩短系统的平均响应时间，提高吞吐量，注重操作系统的整体功能需求，达到更好地平均性能。所以标准Linux并不提供硬实时性。为解决linux不具有硬实时的问题，诞生了几种基于Linux的硬实时解决方案，分为两类：
  - 直接修改Linux内核源代码。对Linux内核代码进行细微修改并不对内核作大规模的变动，在遵循GPL协议的情况下，直接修改内核源代码将Linux改造成一个完全可抢占的实时系统。其缺点是：通过修改Linux内核，难以保证实时进程的执行不会遭到非实时进程所进行的不可预测活动的干扰。该方法的代表是RT-patch(Real Preemption Patch)。
  - 双内核法。添加一个实时内核，在内核空间与Linux内核并存，把标准的Linux内核作为一个普通进程在实时内核上调度。其优点是可以做到硬实时，并且能很方便地实现一种新的调度策略。常用的双内核法有RT-Linux、RTAI(Real-Time Application Interface)和 Xenomai。

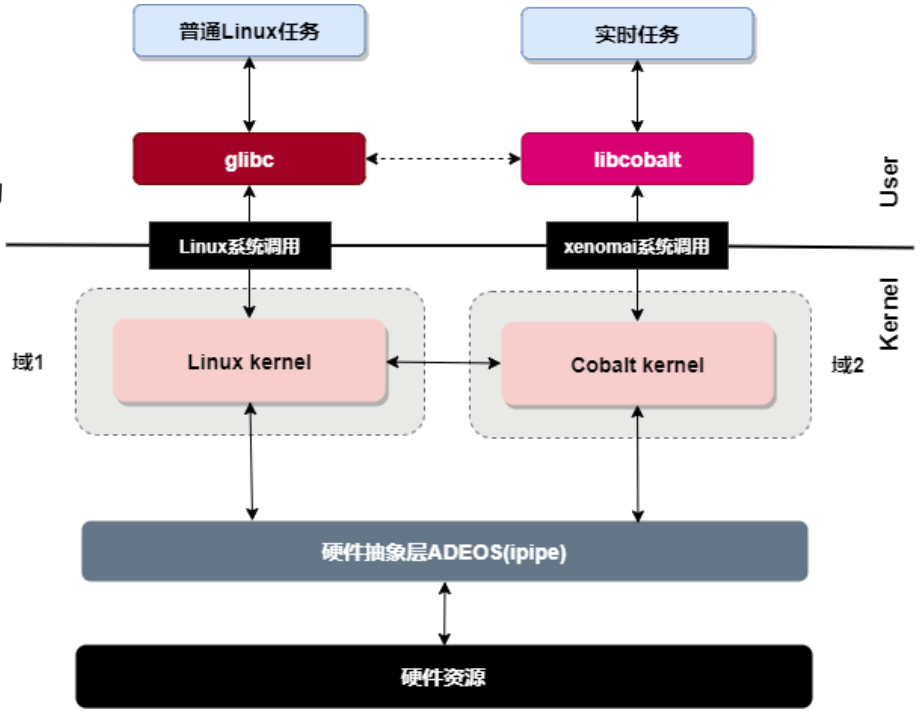
- 从xenomai3开始支持两种方式构建Linux实时系统，分别是Cobalt 和 Mercury
  - Cobalt： 添加一个实时核，双核结构，具有实时内核Cobalt、实时驱动模型RTDM、实时应用POSIX接口库libcobalt，基于libcobalt的其他API skins，如Alchemy API、VxWorks® emulator、pSOS® emulator等。
  - Mercury： 基于直接修改Linux内核源代码的PREEMPT-RT，应用空间在glibc之上，添加Xenomai API库，如下图所示。在不支持cobalt内核时，可使用该方法运行Xenomai应用。



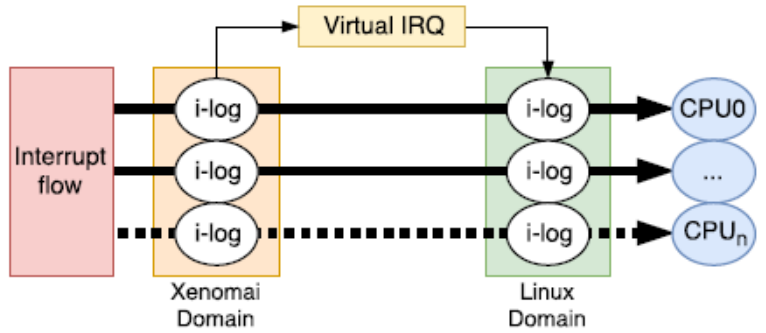


Xenomai3 体系结构

- 内核空间方面，在标准Linux基础上添加一个实时内核Cobalt，得益于基于ADEOS（Adaptive Domain Environment for Operating System），使Cobalt在内核空间与Linux内核并存，并把标准的Linux内核作为实时内核中的一个idle进程在实时内核上调度。
  - 2000年，Karim发表了一篇名为《操作系统的自适应域环境》的论文（即Adeos,Adaptive Domain Environment of Operating System），该论文描述了一种简单而智能的方案，用于在同一系统上运行的多个内核之间共享公共硬件资源。他通过“pipeline”抽象来说明在x86硬件上共享中断的基本机制，根据整个系统给定的优先级，依次向每个内核传入中断。他倡导一种对硬件中断进行优先级排序的新方法，以便可以开发基于Linux内核的实时扩展，而无需使用当时已被某些专有RTOS供应商申请授予专利方法。
  - 2005年6月17日，Philippe Gerum发布用于Linux内核的I-pipe，I-pipe基于ADEOS，但是I-pipe更精简，并且只处理中断，xenomai3使用I-pipe。
  - ADEOS/I-pipe核心思想是Domain,也就是范围的意思，Linux内核有Linux内核的范围，Cobalt内核有Cobalt内核的范围。两个内核管理各自范围内的应用、驱动、中断；两个domain之间有优先级之分， Cobalt内核优先级高于Linux内核；I-pipe优先处理高优先级域的中断，来保证高优先级域的实时性。此外,高优先级域可以通过I-pipe向低优先级域发送各类事件等。



- 中断处理方面，I-Pipe(interrupt Pipeline)分发Linux和Xenomai之间的中断，并以Domain优先级顺序传递中断。I-Pipe传递中断如下图所示，对于实时内核注册的中断，会直接得到处理。对于Linux的中断，先将中断记录在i-log，等实时任务让出CPU后，Linux得到运行，该中断才得到处理。



- 设备驱动方面， Xenomai提供实时驱动框架模型RTDM(Real-Time Driver Model)，专门用于Cobalt，基于RTDM进行实时设备驱动开发，为实时应用提供实时驱动。RTDM将驱动分为2类：
  - 字符设备(open/close, read, write, ioctl)，如UART, UDD, SPI, Memory
  - 协议设备(socket, bind, send, recv, etc)，如UDP/TCP, CAN, IPC, 以太网

- 用户空间方面，添加针对实时应用优化的库--libcobalt，libcobalt提供POSIX接口给应用空间实时任务使用，应用通过libcobalt让实时内核cobalt提供服务。

[i.MX ARM64平台的实时Linux方案Xenomai]

工业网络

- EtherCAT
  - 支持EtherCAT（以太网控制自动化技术），并集成了Igh-EtherCAT master协议栈。
  - 目前，开源代码的EtherCAT master协议栈实现有RT-Lab开发的SOEM和EtherLab的Igh-EtherCAT。使用SOEM比使用IGH EtherCAT主机更简单，但Igh-EtherCAT的实现更完整。
  - Igh-EtherCAT体系结构
    - 主模块(master)：这是包含一个或多个EtherCAT主实例、“设备接口”和“应用程序接口”的内核模块。
    - 设备模块：这些是支持EtherCAT的以太网设备驱动程序模块，通过设备接口将其设备提供给EtherCAT主机。这些经过修改的网络驱动程序可以并行处理用于EtherCAT操作的网络设备和“正常”以太网设备。主设备可以接受特定设备，然后可以发送和接收EtherCAT帧。像往常一样，主模块拒绝的以太网设备连接到内核的网络堆栈。
    - 应用：使用EtherCAT master的程序（通常用于与EtherCAT从机循环交换过程数据）。这些程序不是EtherCAT主代码的一部分，但需要由用户生成或编写。

- RTCAN
  - 基于BSD Socket的CAN设备的开源硬实时协议栈
  - 兼容RTDM架构的CAN设备驱动，支持i.MX系列SOC的flexcan控制器
  - 实用程序rtcanconfig用于设置CAN控制器， rtcanrecv和rtcansend用于CAN消息的接收

- Modbus
  - Modbus协议是一个master/slave架构的一种串行通信协议。
  - 有一个节点是master节点，其他使用Modbus协议参与通信的节点是slave节点。每一个slave设备都有一个唯一的地址。
  - 协议公开并且无版权要求, 易于部署和维护
  - Libmodbus是一个根据Modbus协议实现的开源C库，支持RTU（串行）和TCP（以太网）通信，旨在提供Modbus协议的快速而健壮的实现。

可选的LINUX系统
--YOCTO, UBUNTU，自有系统

基本信息

- 如何获取硬件资料和BSP patch
  - 需要先在NXP官网注册账号，然后发送申请账号时的邮箱给marketing，申请获取代码的权限
- BSP Patch的内容
  - 所有的patch都放入名为meta-imx8mp-ai-robot(5.4.70-2.3.3和5.10.35)的Yocto layer里，主要包括uboot，ATF和kernel的改动。Kernel的第三方驱动包括IMU，smart PA，4G模块，MIPI屏，USB Hub，LVDS屏等。
- 如何获取技术支持
  - 按照正常的NXP支持方式，需要通过community和FAE获取技术支持。技术支持仅限于官方发布的BSP，不包含第三方的驱动程序。BSP patch里的第三方驱动代码仅限于AI Robot参考板演示。ROS部分技术支持仅限于Yocto编译，ROS代码的任何问题建议从ROS官方获取技术支持。

构建基于YOCTO的ROOTFS

- 如何获取基于i.MX Yocto的ROS
  - 可以从链接 i.MX 机器人平台 i.MX Robot Platform 获取，目前匹配AI Robot的最新版本是meta-robot-platform-v1.2-L5.4.70-2.3.3，当前支持ROS1 (kinetic, melodic) and ROS2(dashing, eloquent, foxy)
- 如何编译包含ROS的image
  - 首先下载5.4.70-2.3.3 release

repo init -u https://source.codeaurora.org/external/imx/imx-manifest -b imx-linux-zeus -m imx-5.4.70-2.3.3.xml

  - 然后从NXP官网获取meta-imx8mp-ai-robot压缩包，解压到<当前工作目录>/ imx-linux-zeus/source
  - 创建symbol link: setup-imx-robot.sh -> sources/meta-robot-platform/imx/meta-robot/tools/setup-imx-robot.sh
  - 运行 ***DISTRO=imx-robot-xwayland MACHINE=imx8mp-ai-robot source setup-imx-robot.sh -r melodic -b imx8mp-ai-robot-melodic***
  - 编辑<当前工作目录>/ imx-linux-zeus/imx8mp-ai-robot-melodic/conf/目录下的bblayers.conf，添加下面一行：

BBLAYERS += "${BSPDIR}/sources/meta-imx8mp-ai-robot"

  - 运行 bitbake imx-robot-agv

构建UBUNTU 20.04

- 如何编译Ubuntu 20.04 PoC image
  - 首先下载Ubuntu 20.04 PoC的Yocto编译环境

  repo init -u https://source.codeaurora.org/external/imx/imx-manifest -b imx-linux-hardknott -m imx-5.10.35-2.0.0_desktop.xml
  - 然后从NXP官网获取meta-imx8mp-ai-robot压缩包的5.10.35版本，解压到<当前工作目录>/imx-linux-hardknott/source
  - 在imx-linux-hardknott/sources/meta-nxp-desktop/conf/machine下创建imx8mpairobotdesktop.conf，文件内容如下：

    ```
    MACHINE="imx8mp-ai-robot"
    require conf/machine/imx8mp-ai-robot.conf
    require conf/machine/include/ubuntubasics.inc
    ```
  - 运行 ***DISTRO=imx-desktop-xwayland MACHINE=imx8mpairobotdesktop source imx-setup-desktop.sh -b build-ai-robot-desktop***
  - 编辑<当前工作目录>/imx-linux-hardknott/build-ai-robot-desktop/conf/目录下的bblayers.conf，添加下面一行：

  BBLAYERS += "${BSPDIR}/sources/meta-imx8mp-ai-robot"
  - 运行 bitbake imx-image-desktop
  - 生成image文件在<当前工作目录>/imx-linux-hardknott/build-ai-robot-desktop/tmp/deploy/images/imx8mp-ai-robot/
- 相关文档

  https://www.nxp.com/webapp/Download?colCode=Ubuntu_20.04_POC-beta1-Docs&location=null

已有自己的RootFS

- 如何编译imx-boot（SPL, U-Boot, Arm Trusted Firmware, DDR firmware）
  - 首先下载5.4.70-2.3.3 release (5.10.35类似)

  repo init -u https://source.codeaurora.org/external/imx/imx-manifest -b imx-linux-zeus -m imx-5.4.70-2.3.3.xml
  - 然后从NXP官网获取meta-imx8mp-ai-robot压缩包的5.4.70-2.3.3版本，解压到<当前工作目录>/ imx-linux-zeus/source
  - 运行 ***source sources/meta-imx8mp-ai-robot/setup/setup-env-ai-robot -b build-imx8mp-ai-robot***
  - 运行 ***bitbake imx-boot***
  - 生成的image在<当前工作目录>/imx-linux-zeus/build-imx8mp-ai-robot/tmp/deploy/images/imx8mp-ai-robot/， 获取imx-boot-imx8mp-ai-robot-sd.bin-flash_ai_robot ，然后用uuu下载到板子的emmc里。命令是uuu -b emmc imx-boot-imx8mp-ai-robot-sd.bin-flash_ai_robot
- 如何编译Kernel Image and dtb
  - 运行 ***bitbake linux-imx***
  - 如果重新配置Linux内核，运行***bitbake -c menuconfig linux-imx***，然后运行***bitbake -c compile -f linux-imx***和***bitbake -c deploy linux-imx***编译内核
  - Linux内核源代码在<当前工作目录>/imx-linux-zeus/build-imx8mp-ai-robot/tmp/work-shared/imx8mp-ai-robot/kernel-source/
  - 生成的内核image的相关的dtb文件在<当前工作目录>/imx-linux-zeus/build-imx8mp-ai-robot/tmp/deploy/images/imx8mp-ai-robot/

更新Linux内核image和dtb文件

- 通过uboot更新
  - 用TypeC线将电脑连接到AI Robot板子，然后启动板子进入uboot命令行。
  - 输入命令：ums 0 mmc 2
  - 然后，你的电脑里会有一个磁盘，打开该磁盘可以看到内核image和dtb文件。
  - 复制你的内核image和dtb文件，并替换旧的。
  - 运行Ctrl+c命令，将停止ums，然后在uboot命令行中输入boot将起的新的内核。

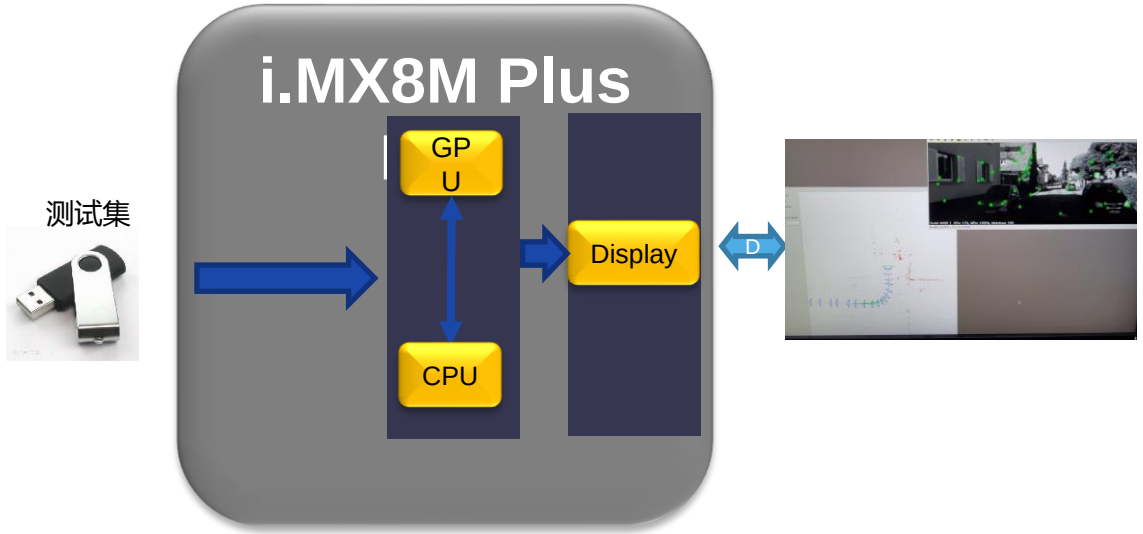
DEMO介绍

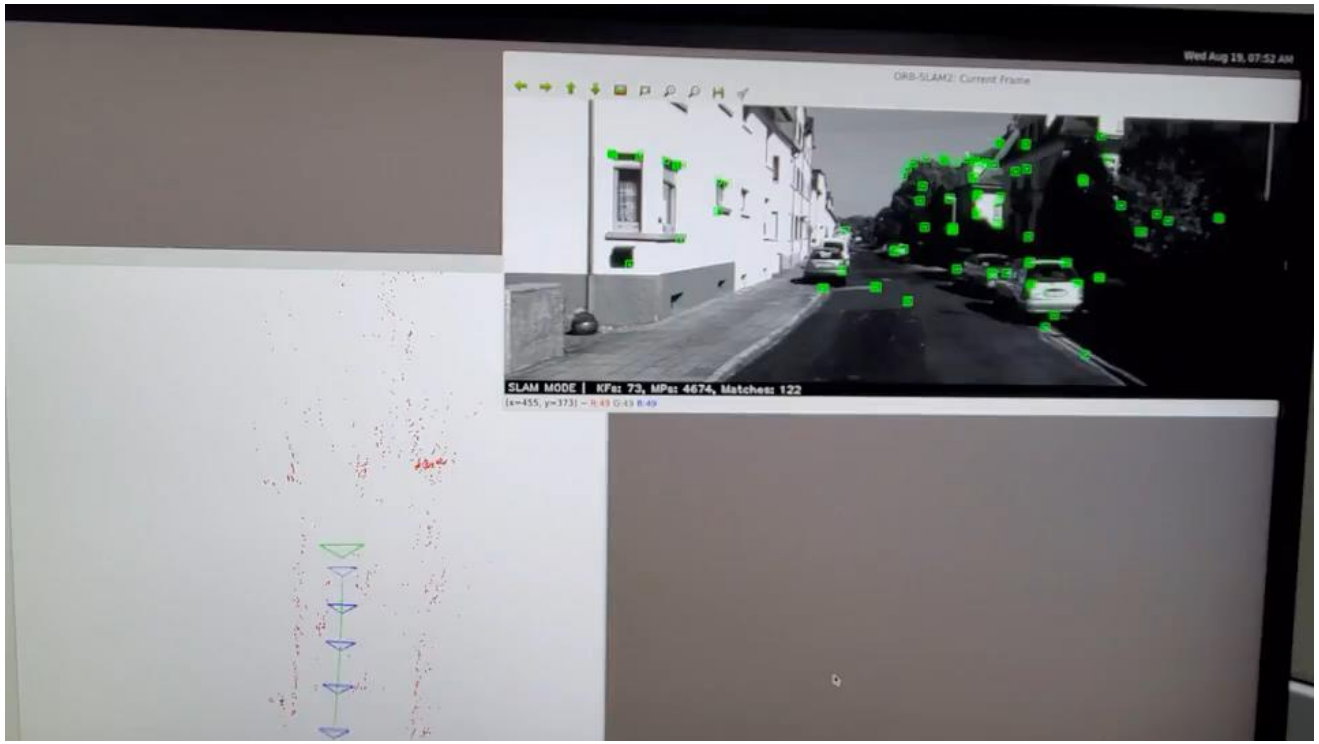
视觉SLAM ORB-SLAM2

ORB-SLAM2是一个用于单目、立体和RGB-D相机的实时SLAM库，用于计算相机轨迹和稀疏3D重建（在具有真实比例的立体和RGB-D情况下）。它能够实时检测环路并重新定位摄像机。

- 移植Panglin库，使用OpenGL ES API替换OpenGL API，用GLFW替换GLEW。
- 下一步移植ORB-SLAM3，以及优化前端视觉里程计模块，使用GPU等其它计算资源加速。

系统框图:





AI ROBOT 避障

基于深度学习的实时小车智能避障

软件架构：

模型：基于Alexnet模型，把全连接改为二分类输出，即有障碍物和无障碍物。模型量化时，保留最后一层二分类层，量化其余层。

数据采集：基于OpenCV采集数据集

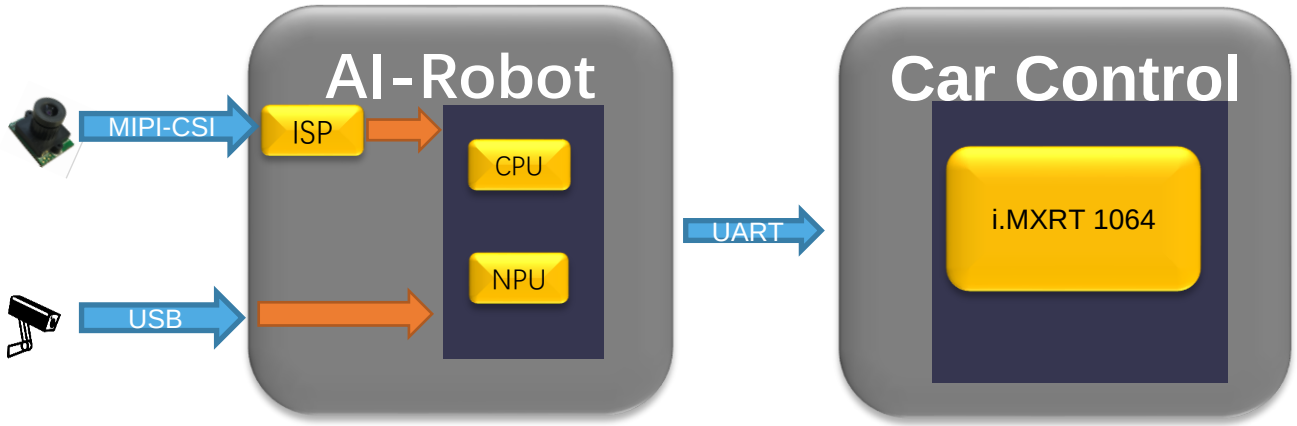
训练：使用Pytorch训练模型

推理引擎：TFLite

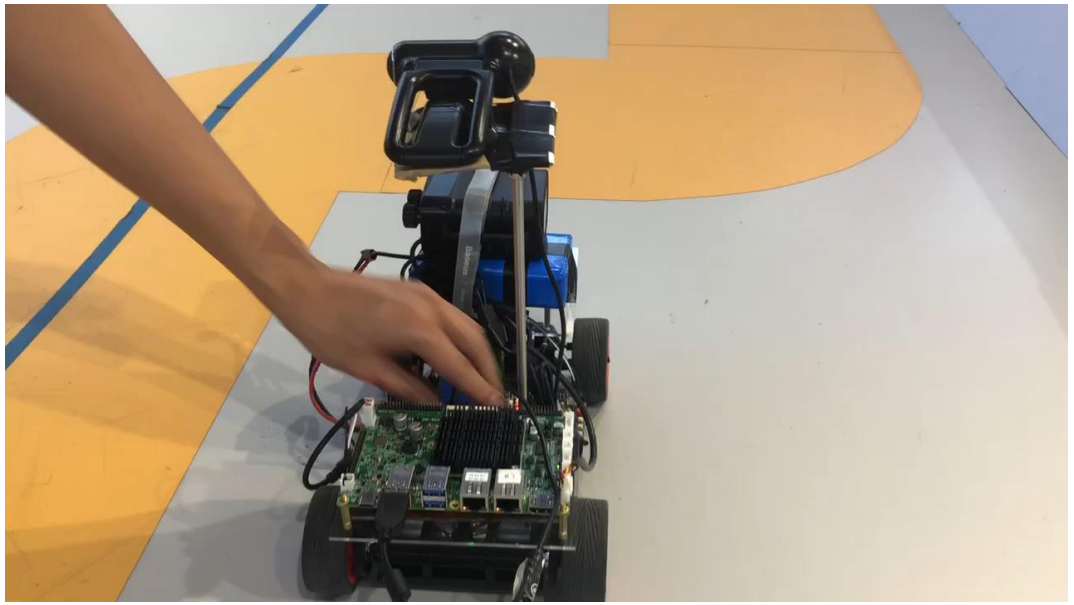
推理时间（NPU）： 7.8ms

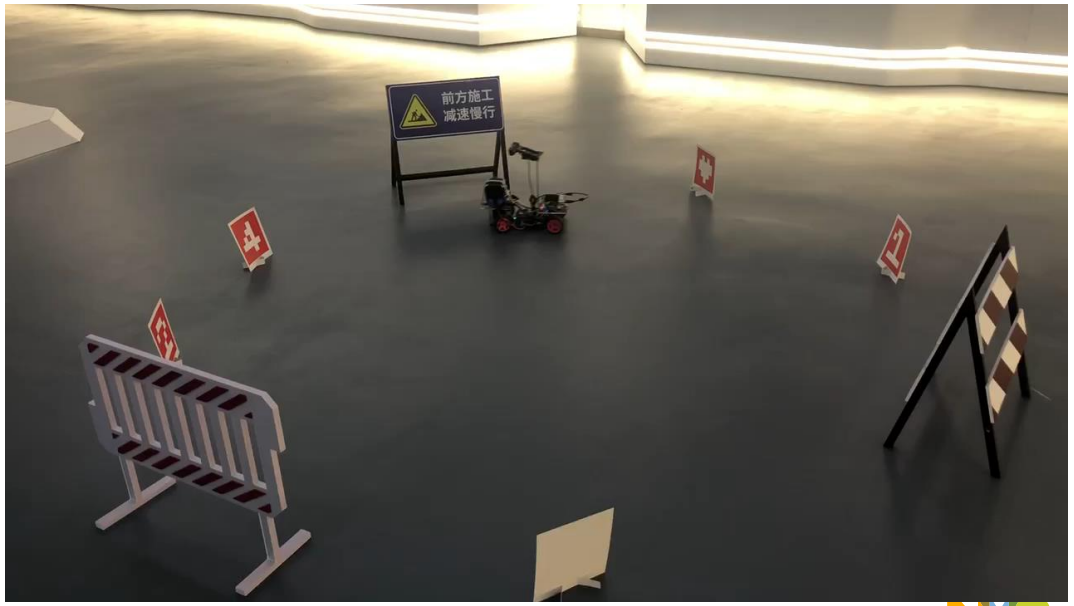
https://github.com/BinJiao/airobot-nxp

系统框图：



MIPI、USB摄像头可选





SECURE CONNECTIONS
FOR A SMARTER WORLD