



Embedded Software System 概述



嵌入式软件开发路线图

- 系统越来越复杂，嵌入式系统的软件开发逐渐变成了软件专家的工作。。
- 对电子硬件很了解，也愿意做一些和硬件密切相关的工
作。
- 最近10-15年，伴随着嵌入式软件规模和复杂性的爆炸性增长，专门的软件开发团队进入了视野。软件团队在不断壮大，但这种增长不仅仅是因为完成所需的代码。
嵌入式软件团队中有不同领域的专家，包括网络、用户界面设计、硬件设备、传统嵌入式软件设计等方面。



嵌入式软件与桌面软件的对比-I

- 内存
 - 有限，影响编程语言的选择以及开发工具的使用（比如编译器优化）
- CPU处理能力
 - 成本和功耗的考虑，ES不得不采用保守的设计方案，其中的CPU往往只是恰好满足要求。
- 操作系统
 - OS (Windows/Linux) 、 RTOS、专有OS或裸机

嵌入式软件与桌面软件的对比-II

- 实时行为
 - 实时系统不一定会运行得很快，但一定是可预测的—通常的术语叫做确定性
 - 实时性的要求对于OS的选择和程序设计都有影响
- 开发流程
 - ES往往没有足够的资源进行软件开发。交叉开发对于工具的选择有很大影响。
 - 开发过程也不同，编辑/编译/调试的循环是一样的，但执行程序这一步却非常复杂，需要把代码转移到目标机上或者在某种环境下运行。

嵌入式软件与桌面软件的对比-III

- 执行流程
 - 多数嵌入式设备从开机开始就会运行某个程序，该程序会一直运行到系统关机。这个程序可能存储在ROM，也可能是从非易失性存储器转移到RAM中执行。
- 每一个嵌入式设备都是不同的
 - 可能是技术层面的-----不同的CPU架构、内存、外设、应用程序和操作系统
 - 也可能是商业运作层面的，比如手机开发、生产的商业模式就和核磁共振扫描仪完全不同。



嵌入式软件与桌面软件的对比-IV

- 嵌入式软件工具和组件
 - 与桌面软就开发一样，嵌入式软件工程师需要运用工具来构建和测试软件。同时，也很可能会引入可重用的软件组件
- 开发工具
 - 从表面上看，嵌入式软件的开发过程和桌面软件很像：编译代码模块，然后在调试器的控制下执行。然而，并不这么简单，有着显著的不同。
 - 交叉编译器
 - 有时候嵌入式应用中会直接包含小部分的汇编代码
 - 嵌入式链接器除了将多个对象模块和函数库例程整合在一起以外，链接器还负责在内存中正确定位代码和数据。嵌入式系统的内存映射十分复杂，对于精确性的要求也十分严格。链接器必需能够灵活地应对这些要求。
 - 嵌入式调试器并不是一个单一的组件，而是一系列对应不同运行环境的工具。借由在本地或者指令集模拟器中运行代码，代码可以仅在宿主机上运行。调试器也可以和一个目标板相连接（JTAG或其他方式，如以太网），并执行代码。
 - 其他一些更有针对性的工具，典型的选项包括分析实时性能、代码执行时设备功耗情况的分析器。

嵌入式软件与桌面软件的对比-V

- 软件组件
 - 最简单和显而易见的可重用的软件组件是库。一些库是由编译器提供的，嵌入式编译器提供的库应当是适合嵌入式环境的：可以重入、可以被存储在ROM中。
 - 其他可用的库，特别是针对C++的库，如标准模板库STL。这些库可能并不适合嵌入式应用，在使用时要多加小心。
 - 与硬件交互式难点之一。在OS内部设备驱动提供了这样的接口
 - ES的互联性不断提高，需要不同的网络支持，TCP/IP或总线CAN\I2C，其他网络技术SNMP、Zigbee、Bluetooth、WiFi和USB。最重要的是应该检查有效性，确保符合标准，并能和其他系统互通互联。
 - 嵌入式设备对于有组织的数据存储是有需求的，比如能够应对电源故障和支持多线程。
 - LCD（往往是触摸式的）成本减低让其能够被应用到嵌入式设计中。软件于是迎来了是一项挑战：支持复杂的图像和用户界面。

Who Designs Hardware?

- May have an embedded engineer who does both software and hardware
- More likely to have two teams
 - small number of engineers who have knowledge of both
- Decisions about hardware can have a lasting effect on software

Software/Hardware Trade-offs

- Microprocessor selection
 - may affect software efficiency
- Memory size and mix
 - late commitment desirable
 - ROM/RAM interchangeability useful
- Peripheral Implementation
 - sometimes replaceable by software [e.g. timer]

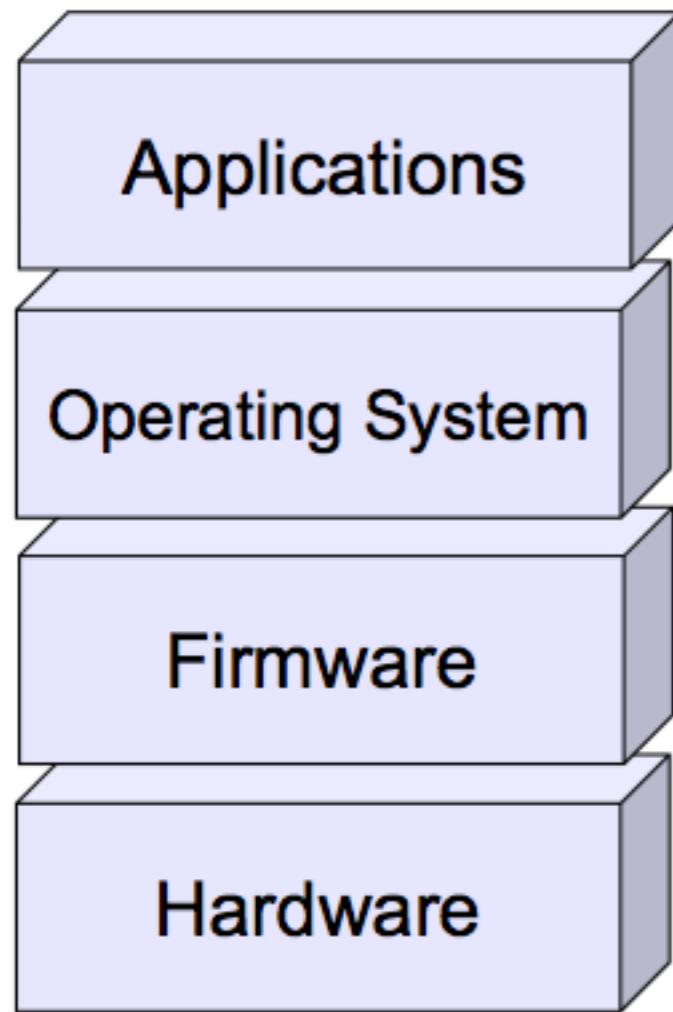
Debug Hardware

- In-circuit emulators
 - not widely available
- Monitor debuggers
 - need communications channel
 - need RAM/ROM swap
- On-chip debug
 - need on-board connector
 - need RAM/ROM swap

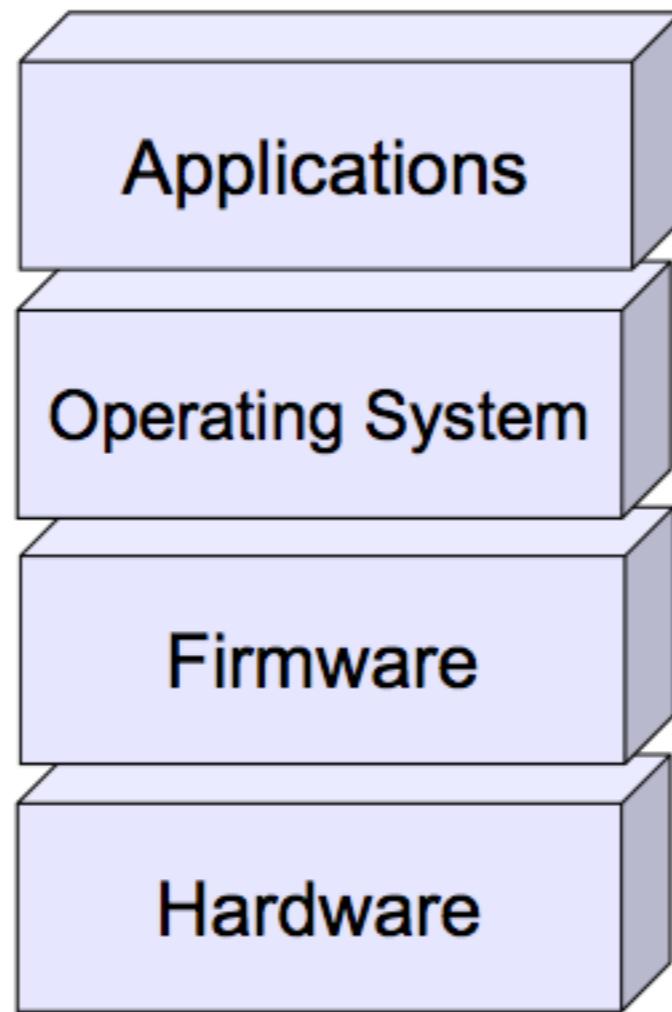
Self-test Support

- I/O circuits
 - “loop back” may be useful
- On-board switches
 - good for configuration or mode selection
- Status displays
 - may be characters or just LED
 - LED can have at least 3 states:
 - on, off, flashing

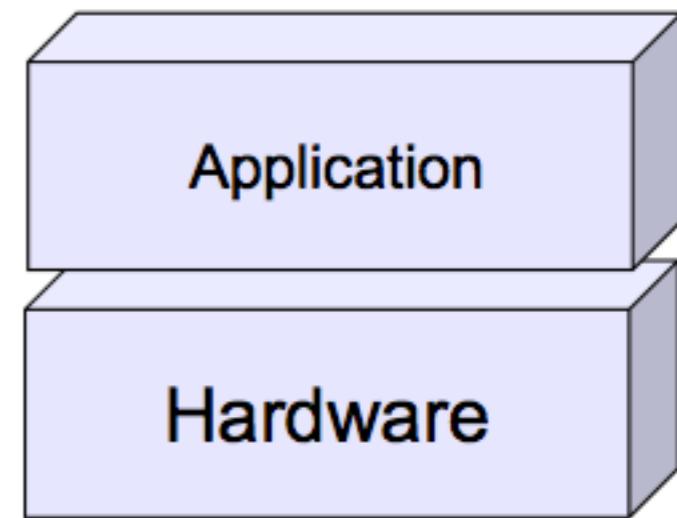
Software Layers



Desktop Computer

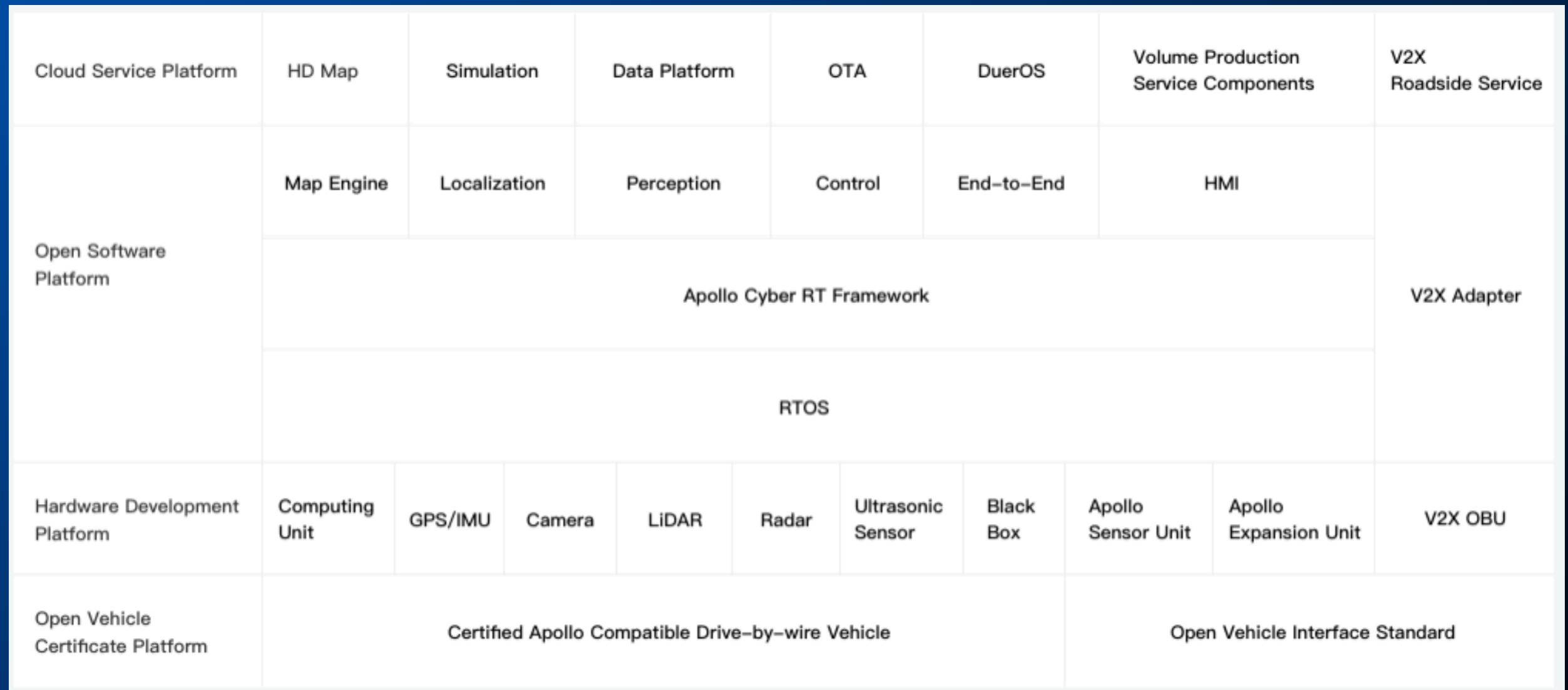


Complex Embedded Computer



Simple Embedded Computer

Apollo 3.5 Architecture



https://apollo.auto/apollo3.5_en.html

为什么要为实时系统建立模型

- 辅助测试和完善最终系统
- 更重要的是，模型利用它所知的系统属性来描述整个系统，并能够被用于对系统特性的进一步研究
- 实时工程师使用程序模型来开发软件和硬件，以便能将整个实时系统全盘考虑
- 模型使得工程师能够预测程序的运行，从而满足系统的性能需求和功能需求

模型间的差异

- 一些模型易于编写，但调试不易
- 难于编写，但调试容易
- 一些模型使得程序运行得更快，但需要付出更多内存资源消耗的代价
- 模型准确性与鲁棒性

嵌入式系统的可视化程序模型

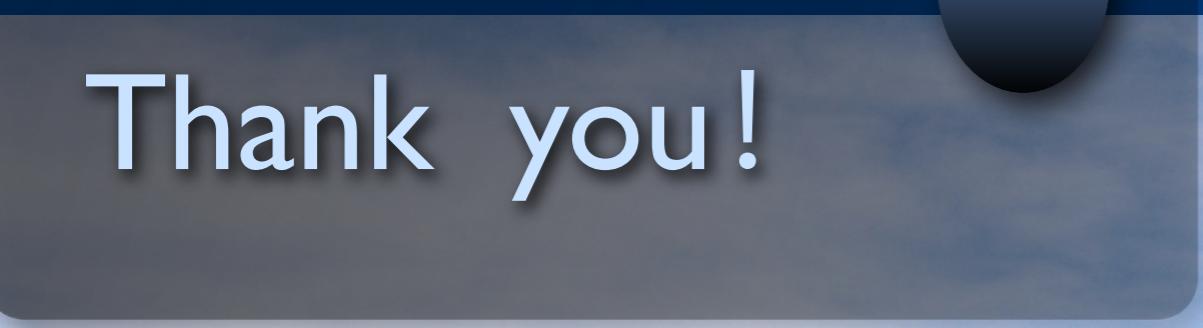
- 实时系统有两种基本的程序模型
 - 将实时应用视为单个执行线程
 - 将实时应用视为多个执行线程
- 取决于需要构建什么类型的系统

单线程程序模型优缺点

- 优点
 - 编程和再编程非常快速简单
 - 改变系统响应的特性的同时，往模型上添加新功能插件也相当容易
- 缺点
 - 在于应用领域的限制
 - 难以做到安全地再编程
 - 很难应用到不同行为或不同环境的运行系统中去。

多线程程序模型的优缺点

- 优点
 - 允许将系统工作划分为几个逻辑阶段，然后编写相互独立的程序来处理各自的工作。
 - 所有处理过程并行。
 - 如果有更高吞吐量的需求，工程师可以在任务中引入新的通信和协作模型
- 缺点
 - 可能引入资源竞争



Thank you!

