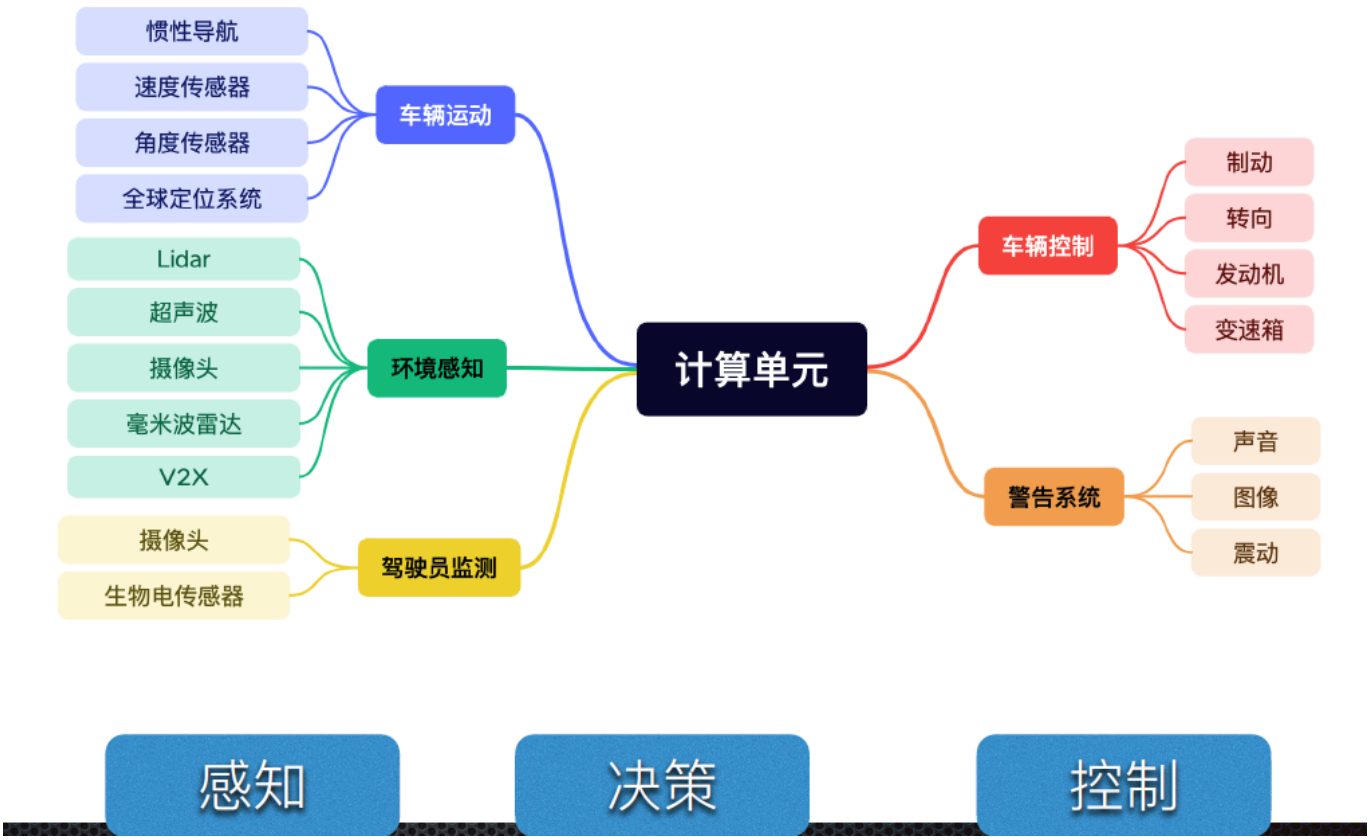
ES重点 By Mas

一、硬件部分总结与应用示例

1. 感知-决策-控制



2. 物联网设备根据能力和性能的分类

- 高端设备：车载
- 中端物联网设备：包括Raspberry Pi等单板机和智能手机
- 低端物联网设备：资源有限，无法运行传统操作系统。依据内存容量标准化为0级，1级，2级
    - 流行的包括：

      Arduino, Econotag, Zolertia Z1, IoT - lab M3节点, Open-Mote节点, TelosB motes

3. 车载计算平台

- 在自动驾驶汽车的计算单元部分，需考量整体车规、电磁干扰和震动方面设计，以及ISO-26262标准的要求
- 芯片计算单元：如CPU, GPU, FPGA, ASIC等
- 所有的CPU, GPU, FPGA, MCU和总线都要做冗余来防止单点故障
- 工控机：
  - Intel CPU+Nvidia GPU的解决方案，考虑硬件供电、散热、接口
- 芯片厂家的无人驾驶计算平台：
  - 英伟达DRIVE，德州仪器基于DSP的TDA2X Soc，恩智浦的BlueBox
  - 采用开发板设计，不适合直接应用，但厂家提供了SDK，可以开箱即用
- 自研的计算平台：
  - 如Tesla的FSD
  - 自研芯片非常适合无人驾驶，但研发成本太高

4. 环境感知

- 感知盲区：人类驾驶员受限于视野范围，存在诸多驾驶盲区；智能驾驶车辆安装的传感器对于特定时刻也有感知盲区，每一时刻都会有新的盲区
- 智能驾驶车辆的感知：
  - 状态感知：交通状态感知+车身状态感知
    - 交通状态感知：被动环境传感器，如**相机、麦克风**+主动环境传感器，如**激光雷达、毫米波雷达、超声波雷达**
    - 车身状态感知：GPS，北斗卫星导航系统，惯性导航系统；获取行驶速度、姿态方位等信息
- 常用传感器：
  - 相机：成本低、刷新快；依赖天气、被动
  - 激光雷达：大小距离，分辨率高，不受光照影响；怕雾霾、昂贵

- 毫米波雷达：电磁波+多普勒效应，体积小，检测多个物体；角度受限
- GPS（全球卫星定位系统）+IMU（惯性测量单元）辅助定位，弥补GPS刷新率不足

二、嵌入式软件架构综述

1. 嵌入式软件架构模式

- 非结构化单体架构：易构建，难移植；与应用耦合
- 分层架构：最常用，划分为若干独立的层

- 事件驱动架构：中断、消息队列、信号量和事件，可扩展、高内聚低耦合，但何时都有额外开销和复杂性
- 微服务架构：低耦合，易于维护测试；复杂、额外开销、抖动、增加开发时间和预算

2. 实时嵌入式软件常用的设计模式

- 单核
- 多核
- 发布和订阅模型
- RTOS模式

- 处理中断和低功耗设计

3. 获取数据的设计模式

- 线性数据存储
- 乒乓缓冲/双缓冲
- 环形/循环缓冲区
- 带有信号量的循环缓冲区
- 带有事件标志的循环缓冲区
- 消息队列

4. RTOS应用程序设计模式

- 资源同步：决定对共享资源的访问是否安全
  - 方式：中断锁定、抢占锁定、互斥锁
- 活动同步：决定执行是否已经达到特定状态
  - 单向同步、双向同步、广播模式、发布订阅模式、低功耗设计模式

三、建模

1. 建模、设计、分析

- 建模是通过模拟来加深对系统理解的过程
- 设计是创建系统的结构。指定系统如何完成其功能。
- 分析是通过剖析获得对系统更深入理解的过程。具体说明系统为什么能/不能完成模型认为其能完成的功能

2. 模型及特征

- 模型：对所研究的系统、过程或概念的表达形式。目的是给出系统的抽象视图，每个模型表示一组对象及其之间的关系

- 特征：简单、经得起理论检验、高表现力、提供逻辑推理能力、可执行、可综合、能够适应不同任务、描述不存在歧义且易于理解和修改

3. 常见建模技术和建模语言

- 技术：系统动态的抽象模型
  - 建模物理现象 - ODE常微分方程
  - 反馈控制系统 - 时域建模
  - 模态行为建模 - FSMs，混合自动机
  - 传感器和执行器建模 - 标定、噪声
  - 软件建模 - 并发，实时模型
  - 网络建模 - 延迟、错误率、丢包
- 语言：具有明确定义和标准语法
  - 图形、文本
  - 面向文档、仿真或执行
  - 专注于体系架构层面内容、实现层面内容（C程序、状态图、数据流、V图承诺）

4. 嵌入式系统模型的用途

- 通过现代建模软件工具，离线仿真进行设计和执行初始验证
- 使用模型作为所有后续开发阶段的基础
- 建模将降低出错风险，通过在整个开发过程中执行验证和确认测试来缩短开发周期
- 更快、更可靠的进行设计评估和预测
- 可以在性能和可靠性方面改进设计
- 可重用性以及对物理原型的依赖的减少，降低资源成本
- 代码自动生成技术减少开发错误和开销

5. 何时为嵌入式系统建立模型

- 任务和安全关键型应用
- 高度复杂的应用程序和系统
- 大型开发团队
- 没有其他选择（没有原型）

6. 反应式系统

6.1 概念

能够持续地与环境进行交互，并且即时地进行响应。大多数实时系统都是反应式，如看门狗、通信网络、家庭应用（洗衣机、微波炉、洗碗机等）

6.2 特征

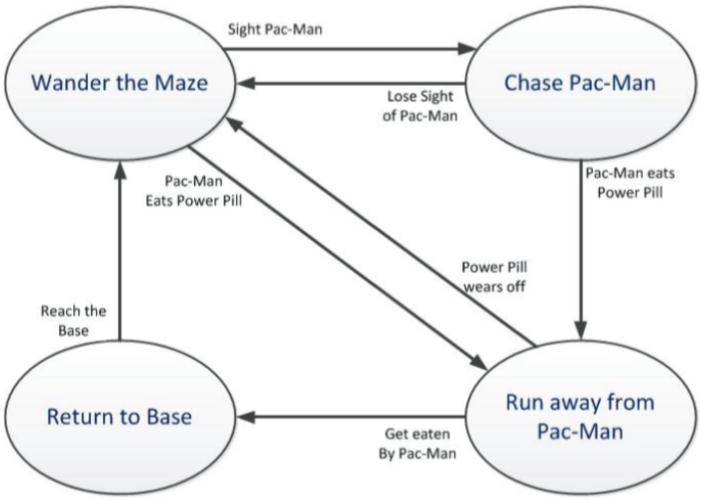
事件驱动。系统接收外部事件后进行处理然后产生相应。反应式系统可定义为系统可能的输入/输出时间序列的集合、条件、动作和时序约束。

6.3 建模

有限状态机、行为有限状态机、协同设计有限状态机、UML状态图、程序状态及、通信交互进程等

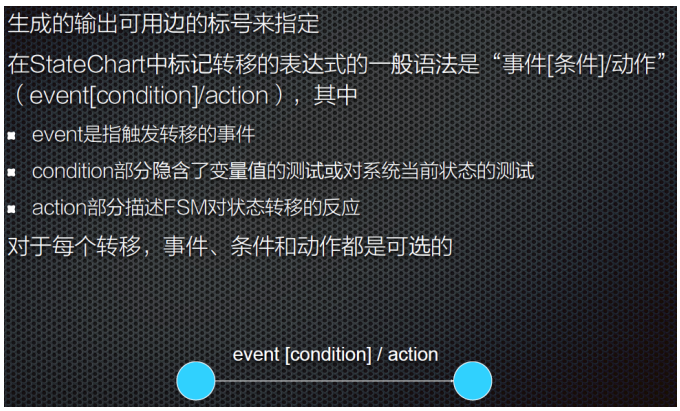
7. FSM

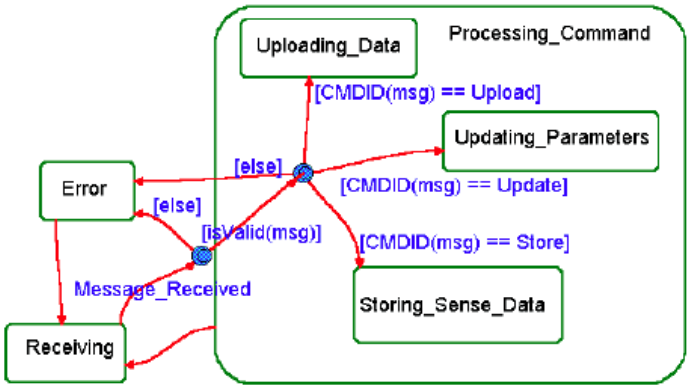
- 有限状态机、有限状态自动机、状态机：有限各状态以及在这些状态之间的转移和动作等行为的数学计算模型
- 输入符号、输出符号、状态、初始状态、状态转移函数、输出函数
- DFA和NFA：等价，所以常用NFA因为更简洁



8. 层次FSM：状态图

- 状态图（StateCharts）语言由DavidHare于1987年提出，支持层次结构模型及并发
- 支持：
  - 将状态重复分解为AND/OR子状态：嵌套状态、并发性、正交组件
  - 动作（可能有参数）
  - 活动（只要状态处于活动状态就执行）
  - 判定式
  - 历史
  - 同步（即时广播）通信机制





四、嵌入式操作系统概述

1. 实时系统、术语、分类

- 实时系统：指计算的正确性不仅取决于程序的逻辑正确性，也取决于结果产生的时间，如果系统时间约束条件得不到满足，将会发生系统出错
- 术语：
  - 确定性：一个系统始终会为某个已知输入产生相同的输出
  - 非确定性：输出具有随机变化特征
  - 截止时限（ddl）：必须完成某项任务的有限时间窗口，指明计算何时必须结束
- 分类：
  - 硬实时：系统有一组严格的截止时限，切磋过一个就会认为失败（如飞机传感器、自动驾驶）
  - 软实时：系统试图满足截止时限要求，但错过了不会认为失败。可能降低服务质量来改进其响应能力（如音视频传输软件的延迟）

    - 准实时：系统将截止时限后的信息/计算视为无效。与软实时一样，错过不会失败，但可能降低服务质量（如财务预测系统、机器人装配线）

2. RTOS和GPOS

- 相似功能：
    - 多任务级别
    - 软硬件资源管理
    - 为应用提供基本的OS服务
    - 从软件应用抽象硬件
- RTOS分离出的不同功能：
    - 嵌入式应用上下文中具有更好的可靠性
    - 满足应用需要的裁剪能力
    - 更快
    - 减少内存需求
    - 为实时嵌入式系统提供可裁剪的调度策略
    - 支持无盘化嵌入式系统，允许从ROM和RAM上引导运行
    - 对不同硬件平台具有更好的可移植性

3. RTOS关键要求

- 操作系统的时间行为必须是可预测的
    - 任何调度策略都必须是确定性的
    - 禁止中断的时间必须尽可能短（避免关键事件处理过程的不可预测延迟）
- 操作系统必须管理线程和进程调度
- 一些系统要求管理时间（京都可能不同，与环境的连接（GPS，移动网络等）获取精确时间信息）
- 快速

- 可靠
- 简洁紧凑

4. 为何使用RTOS

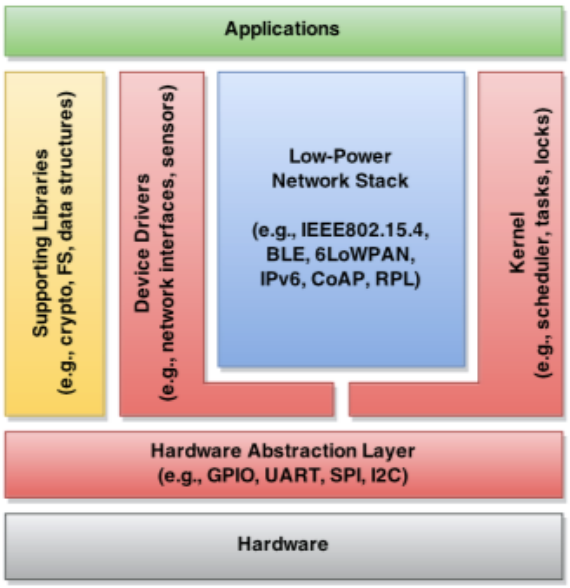
- 可被服用的标准软件组件
- 灵活性
- 响应时间

5. RTOS类别

- 快速专有内核
- 对标准操作系统的实时扩展
    - RT_PREEMPT：Linux内核补丁，将调度器修改为完全可抢占
    - Xenomai：符合POSIX标准的协同内核，Linux内核被视为实时内核调度器的空闲任务
    - RTAI：协同内核的替代解决方案

6. 物联网操作系统

- RTOS、Linux、机器人和路由器操作系统、新型物联网操作系统、边缘计算操作系统平台
- IoT源于大量廉价、小巧、节能的通信设备或物件，从硬件看是异构硬件组成的
- 要求：
    - 内存占用小，支持异构硬件
    - 网络连接、节能
    - 实时性
    - 安全性
- 通用架构：



五、实时调度

1. 实时系统所需的调度策略

- 优先级排序约束
- 时间约束
  - 截止时限
  - 要求不早于某一特定时间进行
  - 相互依赖/相互合作形成一个应用程序
  - 可能不相关，只是共享处理器资源

2. 调度决策

- 分配：该由哪个处理器执行任务
- 排序：每个处理器按什么顺序执行任务
- 定时：每一项任务什么时候执行

3. 任务模型

- 调度程序做出很多的有关任务的假设形成的假设集
- 假设调度开始之前直到所有要执行的任务
- 支持任务到达
- 支持任务反复执行（无休止或周期性）
- 任务突发性，重复出现但时间不规律，时间间隔有一个下限
- 任务间有优先序
- 先决条件（互斥锁等）

4. 时限约束

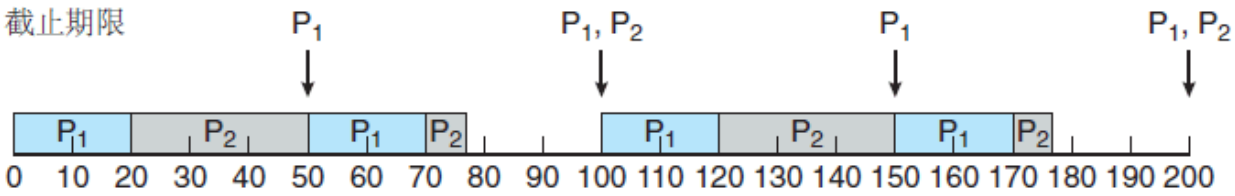
- 硬截止时限：错过系统将失败（该调度称为硬实时调度）
- 软截止时限：不需要严格执行的设计决策
    - 软实时调度
    - 补偿性措施：近似数据或特殊安全模式
    - 简单措施避免数据扩散：置之不理或插入空白值

5. 调度程序的度量

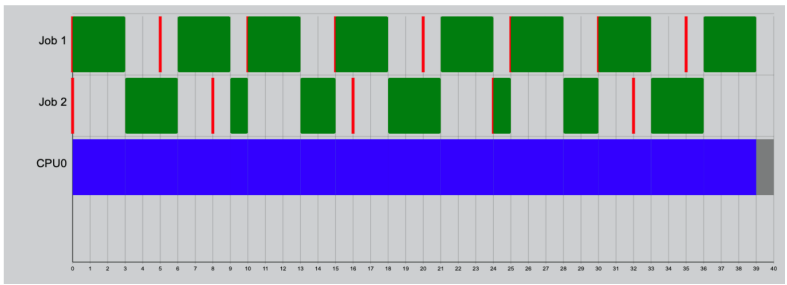
- 可行调度：任务符合时限
- cpu利用率：使用率小于等于100%
- 调度开销：制定调度决策所需的时间
- 延迟：
    - 最大延迟：$L_{max} = \underset{i \in T}{max}(f_i - d_i)$
    - 可行调度为0或负数，软实时程序允许不大的正数
- 总完成时间：$M = \underset{i \in T}{max} f_i - \underset{i \in T}{min} r_i$
    - 性能指标而非实时要求

6. RMS、EDF及改进

6.1 RMS（单调速率调度）

- 假设：
  - 假设n个独立的周期性任务，硬截止时限$D_i = T_i$
  - 执行时间C_i（没有互斥锁、信号量、阻塞I/O、优先序）
  - 固定优先级，忽略不计上下文切换的时间
  - 抢占式调度
- 原理：基于任务周期指定优先级，周期越短优先级越高
- 例子：
  - $T_1 = 50, C_1 = 20 \quad T_2 = 100, C_2 = 35$
  - $T_1 < T_2$，因此任务1优先级更高
  - 

6.2 EDF（最早截止时限优先）

- 给定具有任意到达时间的独立任务集，任何时刻执行绝对截止时限最小的任务
- 在最大延迟最小化上最优
- 例子：
  - $T_1 = 5, C_1 = 3 \quad T_2 = 8, C_2 = 3$
  - 

- 每个红线即为截止时限集刷新的时间点

6.3 EDF改进：具有优先序的EDF

- 截止时间不晚于后置任务的最早开始时间
- 开始时间不早于前置任务的最晚结束时间

六、物联网概述

1. 定义和术语

- 物联网
  - 通过射频识别（RFID）、红外感应、全球定位系统、激光扫描器等信息传感设备，按约定的协议，把任何物品与互联网相连接，进行信息交换和通信，以实现智能化识别、定位、跟踪、监控、管理的一种网络概念
  - 是一种计算设备、机械、数字机器相互关联的系统，具备通用唯一识别码（UID），并具有通过网络传输数据的能力，无需人与人或是人与设备的交互
- 术语
  - 设备：在物联网中具有强制性通信能力和选择性传感、激励、数据捕获、数据存储和数据处理能力的设备
  - 物：物联网中，“物”指物理世界（物理装置）或信息世界（虚拟事物）中的对象，可以被标志并整合入通信网

2. IoT特征

- 只能：从生成的数据中提取知识
- 架构：一个支持许多其他架构的混合架构
- 复杂的系统：一组动态变化的对象
- 规模：可伸缩性
- 时间：数十亿并行和同时发生的时间
- 空间：定位

- 一切都是服务：将资源作为服务消费

3. IoT的优势和不足：

- 优势：
  - 技术优化：有助于技术的改进和提高（汽车传感器收集数据并提高）
  - 改进的数据采集：促进对数据的即时行动
  - 减少浪费：提供实时信息可使资源被有效管理（发动机出问题->跟踪溯源至硬件）
  - 提高客户参与度：允许客户发现问题和改进流程来改善体验
- 不足：
  - 安全：连接设备的生态系统可能缺乏足够的认证控制
  - 隐私：在没有用户积极参与的情况下暴露大量的个人数据
  - 灵活性：与其他不同系统的集成不方便
  - 复杂性：设计、部署、维护都很复杂
  - 合规性：有一套自己的规则，但其本身仍然很复杂

4. 应用：

- 消费级：
  - 智能家居（智能灌溉、智能车库门、智能灯、智能安全系统）
  - 可穿戴设备（健康手表、智能服装）
  - 宠物类（宠物定位、只能狗狗门）
- 企业级：
  - 零售、金融、营销（定向广告、资产跟踪、冷库监控、数字标牌）
  - 医疗保健（家庭护理、预防性学习模型、药物研究、AI手术、跌倒指示灯）
- 工业级：
  - 智慧城市（天气、废物管理、交通灯、照明、停车场）

- 智慧安防（警务、监控、移动物体）
- 电网、能源、制造、工业、交通、物流、供应链、农业……

七、IoT技术

1. 多样化的技术环境

- 硬件（端设备）
- IDE（用于开发设备软件、固件和API）
- 通信（RFID、NFC、6LowPAN、Zig Bee、蓝牙、WiFi、2/4/5G）
- IoT协议（CoAP、RESTful HTTP、MQTT、XMPP）
- 软件（RIOT OS、Contiki OS、Eclipse IoT）
- 云平台/数据中心
- 机器学习算法和软件

2. 物联网软件、硬件、连接

- 软件：若干物联网技术以接近成熟，包括边缘人工智能、基于物联网的流分析、监督和非监督机器学习
- 硬件：归类为相当成熟或主流，包括cpu、mcu、gpu、安全芯片、fpga和边缘网关
- 连接：eSIM、mesh网络、5G和WIFI6，接近成熟

3. 进步最快的技术

- 智能传感器
  - 过去三年出现高潮，旨在解决延迟、数据吞吐量和安全性问题
  - 嵌入数据处理功能
  - 由可穿戴医疗设备、基于人工智能的制造质量控制灯应用推动
- Wi-Fi 6

    - 相当成熟，吞吐能力几乎是Wi-Fi 5的4倍
- gpu
    - 优化gpu以训练AI深度学习模型，为IoT应用同时处理多个计算，并将gpu引入数据中心（并行处理）

4. IoT面临的挑战

- 技术挑战
    - 安全、互联、寿命和兼容性、标准化、智能分析与动作
- 商业挑战
- 社会性问题

八、IoT平台关键技术

1. 设备管理

1.1 概念

- 设备管理服务可以帮助对所有连接的设备在全球范围内进行规模化注册、查看以及远程管理
- 在设备接入基础上，提供了更丰富完备的设备管理能力，简化海量设备管理复杂性，节省人工操作，提升管理效率

1.2 主要功能

生命周期、设备分组、设备影子、物模型、数据解析、数据存储、在线调试、固件升级、远程配置、实时监控等

1.2.1 物模型

- 是对设备在云端的功能描述，包括设备的属性、数据、服务、事件
- 通过定义一种物的描述语言来描述（TSL），采用JSON格式

1.2.2 设备影子

- 物联网平台提供，用于缓存设备状态，是一个JSON文档
- 每个设备有且仅有一个设备影子，可通过MQTT获取和设置设备影子来同步状态

1.2.3 **数字孪生**

（数字映射、数字镜像）

- 概念：充分利用物理模型、传感器更新、运行历史等数据，集成多学科、多物理量、多尺度、多概率的仿真过程，在虚拟空间中完成映射，从而反应相对的尸体装备的全生命周期过程
- 价值：
  - 可见性：实现机器操作的可见性，以及制造工厂或机场中大型的胡练习用可见性
  - 预测性：使用多种建模技术来预测机器未来状态
  - 假设分析：通过适当接口与模型进行交互，并对模型询问假设问题来模拟现实无法创建的条件
  - 对行为进行理解和解释的记录与沟通机制：对单独的机器或机器集合行为进行理解和解释
  - 连接不同系统：如后端业务应用

2. 边缘计算

2.1 基于云的IoT解决方案不足

- 首先，对于大规模边缘的多源异构数据处理要求，无法在集中式计算线性增长的计算能力下得到满足
- 其次，数据在用户和云数据中心之间的长距离传输导致高网络时延和计算资源浪费
- 再次，大多数终端用户处于网络边缘，使用资源有限的移动设备，具有较低存储和计算能力以及有限的电池，需要将一些任务分摊到网络边缘端
- 最后，云计算中数据安全和隐私保护在远程传输和外包机制将面临大挑战，边缘计算可以降低隐私泄露风险

2.2 概念

- 将主要处理和数据存储放在网络边缘节点的分布式计算形式

- 2.0：包括云边缘、边缘云、边缘网关三类落地形态

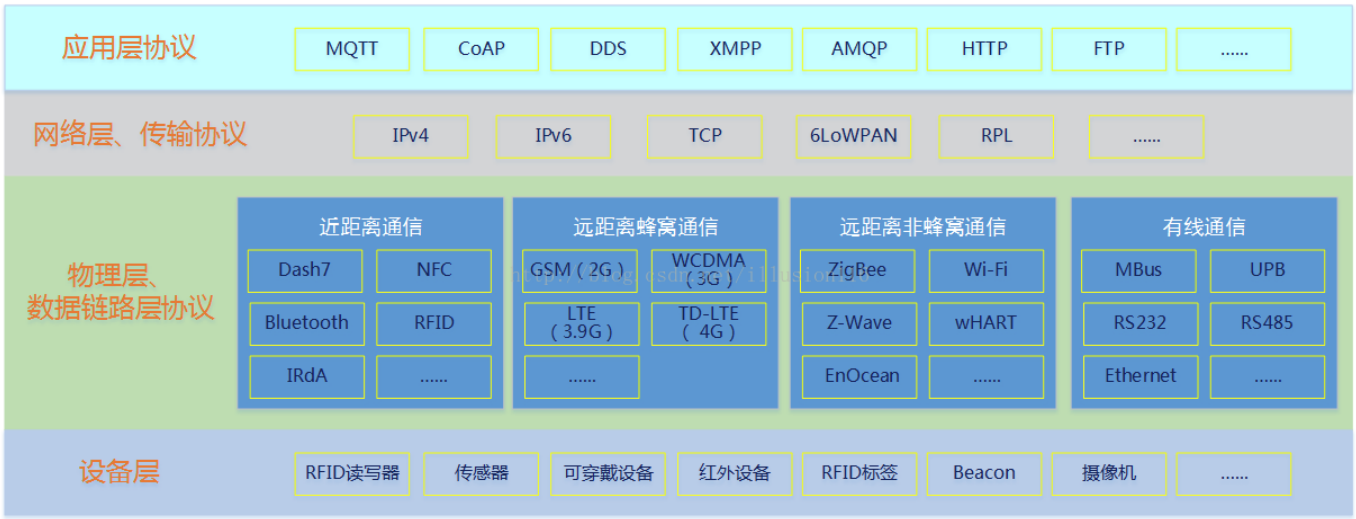
2.3 优势

减少带宽、减少延时、改善隐私提升安全性（预处理数据，不向云端提供详细的、隐私的、不必要的数据）

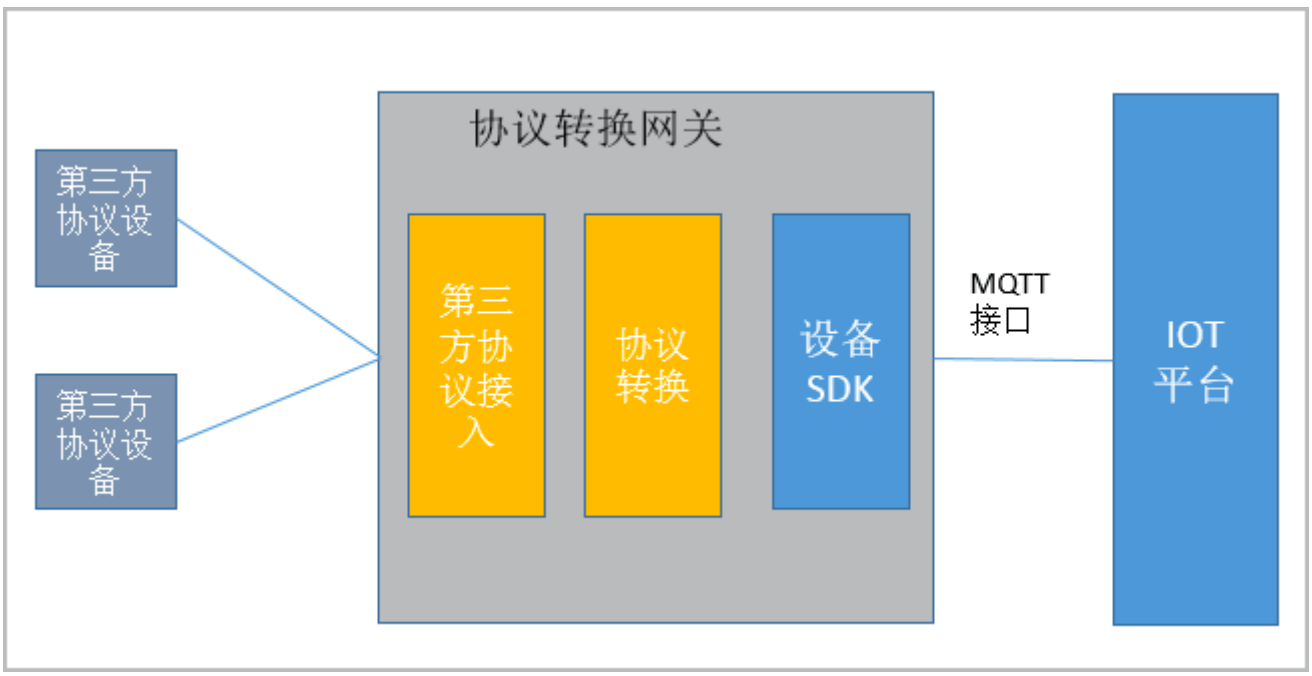
2.4 用途

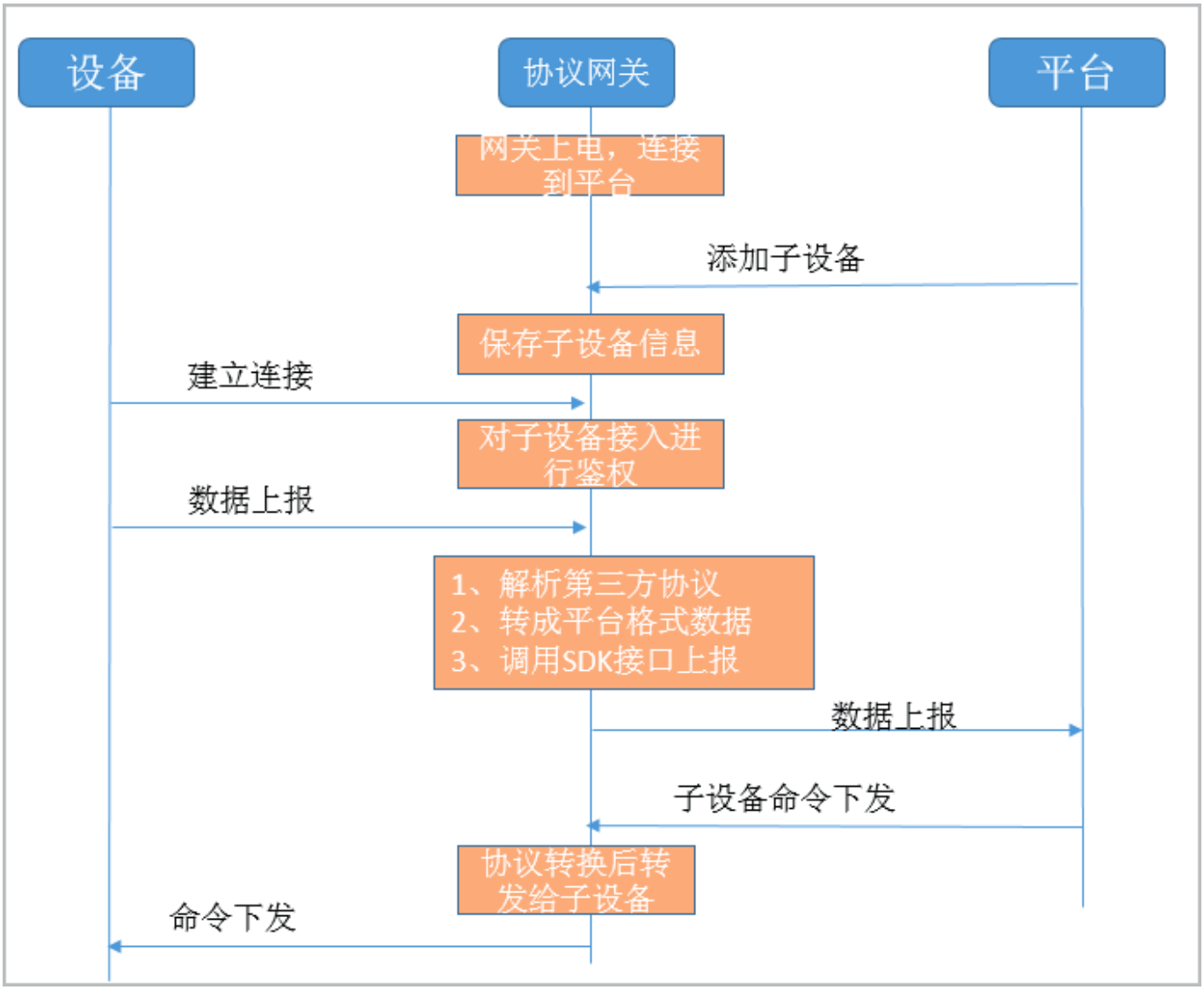
智能交通系统、工业自动化、自动驾驶车辆导航

3. 通信协议



多协议接入方案：使用协议转换网关




设备
协议网关
平台
网关上电，连接到平台
添加子设备
保存子设备信息
建立连接
对子设备接入进行鉴权
数据上报
1、解析第三方协议
2、转成平台格式数据
3、调用SDK接口上报
数据上报
子设备命令下发
协议转换后转发给子设备
命令下发