开源蓝牙堆栈的好处

新兴的单片机开源代码：Apache Mynewt和NimBLE

产品运营总监, Aditi Hilbert, Runtime
技术总监，Sterling Hughes，Runtime

2017年9月11日

介绍

在开发连接设备的早期阶段，过程总是包括选择正确的处理器（MCU或CPU？）和适用的传输模式（蓝牙？无线上网？LORA？）。 当成本和电池成为开发的两个重要考虑因素时，单片机（MCU）通常是默认的最佳选择。 毋庸置疑，物联网将由这些受限的MCU级的设备组成。MCU的连接选项通常与专有软件开发工具包（SDK）捆绑在一起。

这些软件开发套件（SDK）具有以下优点：

- 预认证堆栈支持硬件配置，减少开发成本;
- 与硬件捆绑在一起的基本文档使启动工作变得简单。

专用的软件开发工具包具备成本低、入门快的优势

类似于 Apache Mynewt 和 NimBLE 的开源软件是长期更好的选择。

但是，当选择长期的软件架构时，依靠单片机供应商提供的SDK在开发一些关键功能时会存在一些明显的缺点，如网络堆栈或软件升级：

- **长期支持**：由于每个供应商提供的 SDK 都不同，相应厂商的产品线对跨硬件的应用程序代码可移植性有局限性。成本考虑或惯性选择往往最终导致被锁定到某一芯片厂商;
- **缺少控制力**：软件堆栈的核心部件通常可用作预编译二进制文件软件，从而限制了灵活性或定制可能;
- **芯片和 MCU 架构依赖性**：SDK 和元件库仅在特定供应商的芯片上才能运行。 但是，即使已经购买了专有源代码，修改，共享和重新分配也有限制。 除了供应商的二进制文件之外，很多发行版本进一步约束你对某些处理器内核的开发。（例如，ARM Cortex-M）;
- **只有原型优化**：很多软件开发工具包被优化用来方便地构建原型, 只能通过工作台上的调试器进行调试。缺乏部署在生产环境中具备成本效应，可远程管理和维护，和可扩展的产品所需的接口。

本白皮书探讨了开源蓝牙低耗能堆栈Apache 2.0软件授权是如何可以消除这些问题，并在不增加成本的前提下促进产品差异化 。

源代码的优点

访问开源代码可以实现调试，修复和增强功能 –———一独立于芯片供应商提供的支持。 “解决自己的问题”是开源与生俱来的特点。 开发者不仅可以更好地了解代码如何运作，他们还可以通过跟踪，设置断点，选择适当的编译器选项等来调试和硬化代码。与固定大小的二进制对比，开发人员可以通过删除应用程序不需要的功能和数据来恢复内存和存储资源。 反之，开发人员也可以进行应用程序性能优化而不是尺寸。 简而言之，源代码让开发者可以访问和控制。

以下是产品开发人员利用手中的源代码解决现实中的一些难题的案例。

RAM和Flash：更多应用空间

开源软件允许用户自由修改代码和选择供应商

其结果是让解决方案变得更加灵活

一个设备制造商选择了非常有限的 BLE SoC，但是想要添加一个应用程序协议来控制设备。Apache Mynewt BLE 堆栈“NimBLE” 允许 Runtime 可以在所选的单片机上，将主机和控制器子系统压缩到 128KB 的闪存和 16 KB 的 RAM 中。该存储器配置剩下足够的空间添加一个来自 BLE 顶部的 Open Connectivity Foundation（OIC 1.1）完整的应用层框架，通过标准的方式发送和管理设备。

更好的调试

一客户需要在短时间内完成开发，但是硬件供应商提供的二进制RTOS中没有中断处理器，导致同步错误和项目延迟。无法禁用 / 启动受影响代码的中断功能，调试工作无法进行。而使用开源代码，中断问题很容易找到并修复。

更多吞吐量

客户有多个下游BLE连接传感器，并需要超过中央可承受的与每个传感器单一连接的总吞吐量。

Runtime帮助客户开发iOS和Android应用程序作为创建5个并发连接的中心，通过5种不同的设备ID进行电子分身和循环，提升五倍吞吐量。 结果，客户避免了移动到常规蓝牙的汇总点的额外成本。

服务质量

一人体传感器网络制造商希望将多个外围设备连接到一个中央，并保证固定大小的应用数据包具备高吞吐量和低延迟。中央调度器必须分配固定时槽给外围设备并保证这些固定时槽可用。如果外围设备安排连续紧凑，启用更快的连接间隔将确保高效数据传输速率和低延迟。Runtime在中央使用具有可配置数据包大小，外围数量和吞吐量限制的Apache Mynewt的NimBLE控制器来提供这种流媒体服务。

该解决方案允许客户获取统计数据，如数量连接，接收的总数，接收的总字节数，数据包/秒和最后10秒间隔的字节/秒等。服务质量（QoS）是用来保证删除旧数据，并通过让每个活动连接的通道地图变成可调试，允许客户避免坏通道。

类似于Apache Mynewt和 NimBLE 的开源软件打破了性能和可配置性的极限

帧交叉存取

一个客户想要一个外围设备在BLE 4.2中来推送不同的发布数据和参数到不同的中心。传输中发布包交叉存取的数量需要灵活性。 由于Apache Mynewt的NimBLE控制器允许BLE 4.2中有多个发布实例，Runtime用它来添加系统配置变量来启用它们配置发布发送数量和频率。 Apache Mynewt控制器支持在扫描或连接操作时进行发布。它能够容纳在任何预定的事件（例如，连接事件）之间的多个发布。

社区和商业实体共同受益

增加位置/邻近服务的并发连接

一客户需要尽可能多的并发连接用于快速存在性检测。Runtime测试显示使用Apache Mynewt堆栈最多可以同时连接32个数据交换。使用单个设置可以方便地指定最大连接数量，允许客户和社区内的其他人超过32个连接的测试。 这样的开源促使一些供应商增加他们的二进制（！）允许的最大连接数，但

仍然有一点差距 —- 超出硬编码限制的连接的灵活性没有了。 可查看红熊实验室Red Bear Lab的演示视频查here.。

Apache Mynewt和NimBLE的其他优势

Apache Mynewt 的高度模块化设计使其 NimBLE 堆栈可分解。就像 Apache Mynewt 中的安全引导程序进化为开源 MCUboot 项目（给 32 位 MCU 提供一个独立于操作系统和硬件的安全启动）一样， NimBLE 堆栈可以被用作其他操作系统的组件。 端口已经在其他 RTOS 进行，共享核心代码库让互操作变得更加容易。

仅仅给开发人员提供源代码，而不提供更方便用户的使用方法，这本身不能促进快速理解和发展。用一个高粒集的系统配置变量作为构建时间选项是一个解决实施自定义行为复杂性的一种简单可测的方式。Apache Mynewt SDK做到了这一点。 此外，它允许这些系统配置变量可以在任何包或库中定义，并允许对根据该类型的优先级分配的值包进行重写。这使得贡献者可以通过低级API更容易地创建新粒集，并简化应用程序开发人员重写更高级别的设置。 这是应用开源、灵活的BLE SDK的一个例子。与捆绑了一个BLE连接栈的RTOS内核相比，它可提高产品开发人员的开发能力。从在模拟器模式下启用详细调试设置到启用完整的收发器深入行为分析统计（包括配置前后），具备丰富的非默认选项和并容易使用。

鼓励采用开源软件的一种许可证

许可授权和社区模式

良好的开源提供了一种有效的方式以最低价格来获得最多听众，其目标是鼓励开发人员使用此技术，而Apache License 2.0正是如此。 这有利于终端产品的商业开发和专有再分配，并让其商业模式不会再出现戏剧性的变化。

一个旨在提升开源软件质量和安全的活跃社区

Apache Mynewt BLE堆栈已经能够构建一个丰富的生态系统，有大量的由独立开发人员开发的不同产品。生成的代码库可以轻松应对各种各样的测试，惠及整个社区。

社区成员积极使用和测试代码也有助于积累资质。将BT SIG Profile Test Suite（PTS）的资格测试和结果放入到开源库中，有助于准备主机子系统认证。 该用户可以将其与任何硬件厂商提供的合格的控制器子系统进行混合和匹配。如果开源项目支持特定的控制器级别的BLE频段，Apache Mynewt也可以，那么社区将有一个合格控制器系统保障。 对产品厂商来说，基于这种硬件的终端产品认证将被简化。 另外，诸如Runtime这种提供商业支持的公司将通过认证服务进一步简化设备制造商的设备认证过程。

在这个蓬勃发展的活跃社区，通过公开代码、快速修复漏洞、分享安全代码实例、甚至进行未经请求的安全评估，开放源码变得更加安全。任何人都可审核该代码 - 社区内外的任何人 – 非隐匿性安全。

结论

一个开放源代码的操作系统使客户可以控制他们的软件架构，并提供独立于所选硬件的可持续软件平台。现实生活中的例子证明了 Apache Mynewt 操作系统与 NimBLE BLE 堆栈相结合的开源的力量和灵活性。

关于**Runtime:**

Runtime 提供连接设备的云管理服务，作为消费者，医疗，商业和工业物联网应用中端对端 IoT 平台的一部分。Runtime 同时支持 Apache Mynewt，一款主要用于优化受限制的基于微控制器系统的网络和远程管理的开源嵌入式操作系统。 Apache Mynewt 包括世界上第一个开放源码，控制器级 BLE 4.2，蓝牙 5 和用于 MCU 的蓝牙网格堆栈，以及用于管理连接设备的高效 OIC 1.1 应用程序框架。