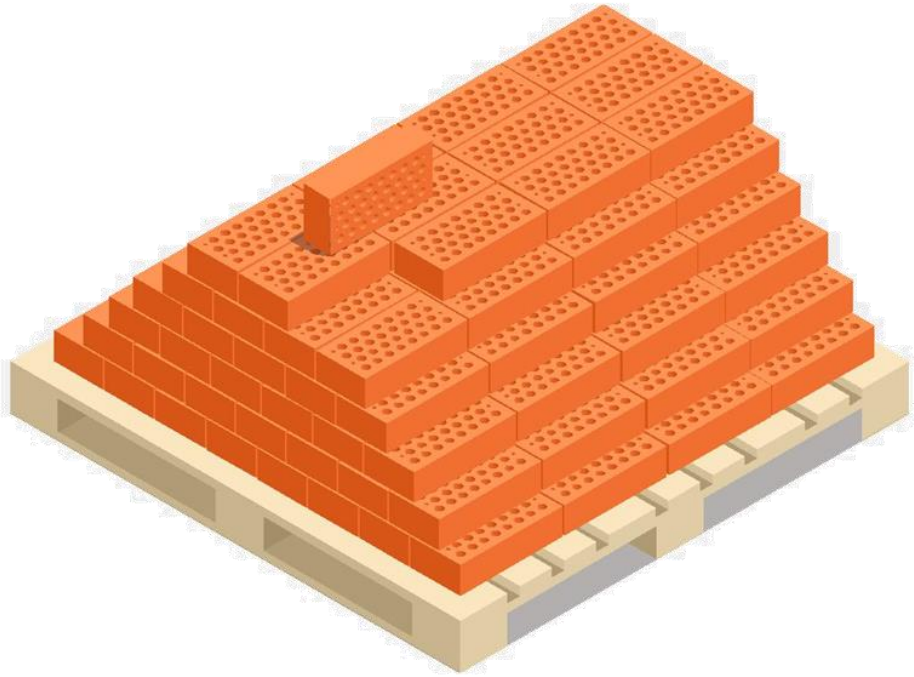


为你的设计
选择正确的软件架构

作者：Alf Mossevig，Silicon Labs 软件开发流程产品经理

为你的设计选择正确的软件架构

目录

软件架构

操作系统

商用与开源解决方案的比较

候选方案

结语

我们都曾在职业生涯的某个时间点思考过这个问题：你已经把一个想法仔细思考过一阵子，逐渐得出可行的结论，现在，你想要创建一个合适的项目，看是要更进一步探索这个想法或是将其产品化。但是，应该从哪种软件架构入手呢？Espruino？Arduino？microPython？Segger embOS？MicriumuC/OS-II？以及在 uC/OS-II 和 uC/OS-III 之间又有什么区别呢？究竟该采用初始成本较低的开源架构，还是选择需要支付前期费用的商业解决方案，来加速你的设计过程呢？

在本白皮书中，我们将讨论你在为微控制器或无线微控制器项目选择一套优秀的软件时，所需要考虑的各种要素。

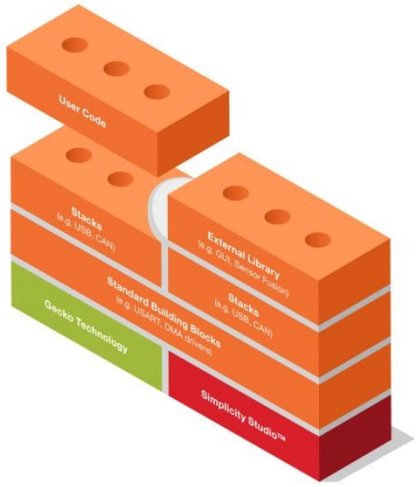
软件架构

在本文件中，我们将经常使用“软件架构”这个词，在我们开始之前，我们应该先看看它真正的意思究竟是什么。在本文件中，将其解释为“编写软件的一种特定方式”。例如，Arduino 提供编写程序代码的一种特定方式，允许软件的片段可以跨越多个项目被重新使用。

软件架构是由几个不同的部分所组成，并由以下组件所定义：程序语言、应用程序编程接口（API），以及某些工具集的连接。例如，像是在 Arduino 和 Espruino 的案例中，软件架构可以被紧密地连接到工具，或是像在 Micrium 和 FreeRTOS 的案例中则是会被分离。

操作系统

因此，该如何选择一个软件架构呢？首先，我们需要对一些名词解释的更明确一点，我们已经开始提到*操作系统*（OS）这个名词。你真正需要写的是可以执行你的特定要求的程序代码，这些程序代码将可以把你的产品与其他产品区分出来。但是，你仍然需要依靠软件的其他部分，像是 ADC 的驱动程序或 SD 卡的文件系统栈。这些软件的部分通常被称为*软件构件*。



在此用一个比较实质的比喻方式，你可以想象你的软件构件就像是砖块，然后把操作系统视为水泥。在操作系统中定义了砖块的形状，以及它们将如何与其他的砖块互动，因此当你添加更多构件到软件之中时，他们将可以继续完美地协同工作。这听起来相当不错，但你真的需要一个操作系统吗？增加操作系统也带来了额外的负荷，它将会消耗数千字节的闪存，为事件的响应增加了延迟的时间，并且还需要花费些许的时间学习如何在操作系统环境中编写程序。

决定采用哪个软件架构，将会决定你的想法是否可以按时间、按预算地推动，或是将你的投资放在风险之中。

一般的经验法则是，如果你的目标闪存容量是 128 KB 或更高，并且或是需要通讯功能，便需要一些栈（USB、以太网络、SDIO、CAN、Wi-Fi、BLE），长期来看，你最好还是使用操作系统。

在操作系统中最重要的其中一件事情便是调度程序（scheduler）。调度程序是用在为可能会争夺相同资源的不同任务、分配资源和处理时间的组件。在一般情况下，调度程序有两种作业的方式，而这正是“实时”在实时操作系统（RTOS）的意义所在。实时意味着在一个特定的时间内，会有一个特定的任务将会被执行。假设你得到一个你需要处理的射频封包，无论你的设备目前正在做什么事，实时操作系统的核心会先离开它目前所做的任务，先完成这一个高优先等级的任务。这种类型的行为在处理器的利用上并不是最有效率的方式，但例如在马达控制应用中，它在通讯栈与应用中最需要重视的便是反应时间的议题。

商用与开源解决方案的比较

举例来说，如果你已经想通了是否需要采用实时操作系统，并开始组建软件需求。你可能知道，你需要一个 USB 栈和以太网络栈，搭配外部 MAC/PHY 驱动程序来一起将设备连接到互联网。但是，你该从哪里开始呢？你真的只需要为你首选的微控制器下载最新的 FreeRTOS 模板，并继续下载开源软件并放到装置中就可以了吗？或者你只是需要去找有你所需软件的商业供货商，并获得完整的软件组合比较好呢？

为了做出更明智的决定，我们经常谈论要给选定的解决方案一个总体拥有成本（TCO）的概念。所谓的总体拥有成本包含的不仅是你为软件付出的货币价值，还包括花费在寻找解决方案、组装不同的构件，并将不同的构件整合到你的项目，以及开发、测试和生产的工作时间。

在一般情况下，我们看到的是商业解决方案的总体拥有成本（TCO），将比自己组建开放原始码组件的解决方案要来的更低一些。但既然是商业解决方案便涉及到初始成本，这些厂商通常要求在使用解决方案的前期，取决于你所需要的组件，便必须先支付 1 万到 10 万美元之间的费用。在另一方面，下载 FreeRTOS 并开始组装自己的解决方案，在某些拥有密集资源的应用中，其所花费的金钱，相对会更便宜一些。

决定你的项目的最佳解决方案，将高度依赖于它是否是最容易让你赚到现金，或是能够为总体拥有成本的工时来“买单”。

候选方案

所以，当你已经看到这里，你的大脑中可能已经开始呐喊：“只要给我一个可以让我开始使用的架构就好了！”

可惜没有那么快，肯定有一些方案的选项是比其他选项更好，微控制器的应用非常多样性，肯定没有那种一个尺寸便适合所有需求的解决方案。我们先来了解许多最流行的操作系统和软件架构，并分别仔细地讨论：

操作系统

所有在这一个章节中提到的操作系统都具有实时能力（RTOS）。

商用解决方案

- **Micrium uC/OS-II 与 uC/OS-III：**这是在微控制器业界最流行的两个实时操作系统，特别是因为其创新的商业模式，因为 Micrium 公司允许你下载完整的软件套件，并开始发展，等你真正开始产生营收后，才需要开始支付解决方案的费用。他们在安全至上的系统中拥有重要的地位，并且他们的大部分软件构件都已经通过认证。

- **Segger embOS：**嵌入式软件市场的新进入者，但这并不意味着他们是新手。该软件产品已经开发了超过 20 年的时间，并已经使用在他们自己的硬件产品之中，因此对装置的支持程度非常好，并配有一个优秀的驱动程序库。

- **Express Logic ThreadX：**由行业的资深人士所创办，该公司专注在所有关于性能的事物上，并挤压出元器件中每一个时钟周期的效能。它通常被看作是操作系统中的劳斯莱斯，并已经有很多认证被使用在安全至上的系统之中。

开源解决方案

- **FreeRTOS：**FreeRTOS 跟 Micrium uC/OS 一样，都是在同行业中最常被采用的实时操作系统之一。它有一个庞大的社群，有很多人都在为软件做出贡献，像是 TCP/IP 栈，但做为开源软件，便意味着没有公司会负责整合，因此需要更多的工作来创建一个整合的解决方案。

- 也有一些公司在 FreeRTOS 的生态系统中，专门从事将差异化的软件构件提供给那些需要整合协助的客户，例如，**Wittenstein 高完整性系统公司**提供称为 SAFERTOS 安全认证的 FreeRTOS 替换核心，以及 **HCC 嵌入式公司**提供可以与任何实时操作系统工作的 USB、以太网络和文件系统。

- **mbed OS**：mbed OS 解决一些你通常会在开源软件所遇到的痛点，因为 ARM 这一家公司会负责整合的工作。然而，它仍然是处于萌芽阶段，所以如果你想要有些贡献就可以去这里。

- **RIOT OS**：RIOT OS 被冠以“物联网中最友好的操作系统”，它是以通讯概念为基础所建立起来的操作系统。这使得它即使在面对困难的通讯问题时，仍然精简且高效率。然而它仍然还在积极发展的阶段当中，所以你得计划多花几个小时来进行除错。

架构

有一些操作系统的功能就像是将砖块黏合在一起的水泥一样，会与发展框架紧密地结合在一起，因此一般不能使用软件本身做为项目的一部分，你必须围绕着它来进行整个开发流程。这些框架往往是使用比 C++ 更高阶的语言所编写，通常可以在实时操作系统上运行。

- **mbed**：mbed 也出现在这里，这时则做为快速成型的项目。它是用 C++编写，并对大多数微控制器和电路板有绝佳的支持，拥有一个庞大的元件程序库，和一个采用网页架构的漂亮集成开发环境（IDE）。目前，在它准备全面部署之前，它仍然需要一点成熟的时间，但它很适合硬件原型的开发。

- **Espruino**：Espruino 是在微控制器上运行的实时 JavaScript 解释器。它允许你能够动态地更改程序代码，甚至不需要读写微控制器便可以撰写程序代码。在它开始量产之前，仍然需要一些时间来发展，但它很适合用于硬件原型，并呈现出成为一个不可忽视软件架构的巨大潜力。

- **microPython**：microPython 所能做的事与 Espruino 大致相同，差别仅在于其使用 Python 来代替 JavaScript。它发展的概念，便是让你从产品开发的一开始到量产，都能够支持预先编译的程序代码，并使用 C 语言来编写时序至上的程序代码。目前仍在开发当中。

- **microEJ**：microEJ 是一个采用 Java 架构的框架，让你轻松地为你的设备打造好看的图形化应用程序。它已经在许多智能手表和一些物联网（IoT）设备中使用。

结语

如果你想要着手进行设备的开发，而无需安装单一的程序，像是 mbed 和 microPython 这类的框架，便是入门的好方法。但是，如果你要建立更大的部署，采用一个纯粹的实时操作系统将会是更好的选择。如果使用工作时间而不是金钱来当作软件的投资，对你的公司来说会更为轻松的话，像是 FreeRTOS 或 RIOT 这类非商业解决方案便有其优势。如果你的公司能负担得起前期投资，像是 Segger、Express Logic 和 Micrium 的商业解决方案，将大大降低软件开发风险和缩短产品上市的时间。在商业解决方案中，特别像是 Micrium 的稳定性和认证的程序代码基础，广泛普及的部署，良好的元器件支持，开放的原始码，以及适合大多数经济效益的商业模式，更使其在商业解决方案中显得特别突出。