2017嵌入式软件行业现状及概述

[2017-10-28](http://www.yellowmax2001.com/2017/10/28/2017%E5%B5%8C%E5%85%A5%E5%BC%8F%E8%BD%AF%E4%BB%B6%E8%A1%8C%E4%B8%9A%E7%8E%B0%E7%8A%B6%E5%8F%8A%E6%A6%82%E8%BF%B0/)

导读：

本文章介绍了截止2017年为止的嵌入式软件行业现状，包括就业方向，行业产品类型，待遇等等。另外还简单概述了下嵌入式软件所需的知识技能以及未来可能的技能发展走向。本文章远不如专业的行业报告一般精准、专业，仅仅是根据个人见闻来提供一个比较整体的嵌入式软件行业印象，抛砖引玉，以供参考。（文末有2017年集成电路行业薪资表哦）。

**嵌入式软件定义**

**嵌入式软件**

其实嵌入式软件这个名词还是比较难以准确定义的，一般把“用于控制、监视或者辅助操作机器和设备的装置”称为嵌入式系统，系统包括一系列软硬件设施等，而嵌入式软件姑且可以认为是嵌入式系统的软件开发这部分。比如：手机的系统开发（Android、IOS等），无人机的系统控制软件、驱动开发，路由器的软件系统开发等等。广义上，可以把单片机（STM32）、SOC等等硬件构成的控制系统中软件开发部分都称之为嵌入式软件开发。

**嵌入式系统架构**

**硬件**

硬件架构以ARM为主，有很多人可能分不清ARM这个单词到底代表什么意思。感觉好像很多地方都有ARM这个词，而且意思还不甚一致，这里稍微解释下。

* ARM公司
* 首先ARM是一家英国公司（全名**Advanced RISC Machines**），于1991年成立英国剑桥，由苹果、诺基亚等等合资成立，主要业务是出售芯片设计技术的授权，也就是常说的IP，它的盈利方式有点类似于高通，但是高通有很大一部分收入来自于专利墙，从这点来说ARM还是差了不少。ARM的IP包括物理IP（嵌入式内存IP、逻辑IP、接口IP等），图像与多媒体（熟悉的Mali系列GPU以及video、显示等），其它还有系统IP（CoreLink(核心)连接内存控制器、CoreLink(核心)连接系统控制器、TrustZone安全IP），无线IP等等。
* ARM处理器
* 一颗SOC包括处理器内核、GPU（可选）、各种片上外设（USB、I2C、ISP等等），嵌入式领域的SOC处理器内核主要就是ARM、高通（Kyro等）、还有苹果的。前期ARM的处理器型号有ARM7/9/11，例如熟悉的三星的S3C2440就是ARM9的处理器内核，这是比较早期的架构了，从ARM11开始就分出了三个不同的分支，现在我们最熟悉的是ARM公司的三种不同系列的处理器，Cortex-A/R/M系列。
  + Cortex-A：主打高性能，为各大嵌入式操作系统（Linux、Android等）提供基础硬件，常见于手机、平板、IPC等嵌入式设备。最新有A75、A53等等，再往前有A7等，像联发科的最新X30处理器用的就是双核A75、四核A53、四核A35。高通的处理器芯片早期是Krait，从820开始转为Kryo架构，与ARM的处理器架构有所区别，不过其指令集用的还是ARM指令集。
  + Cortex-R：主打快速响应，用于实时性要求强的场合。这一系列并不像M/A系列那样为人熟知，它通常用于硬盘控制器、网络设备或打印机、汽车控制系统等。
  + Cortex-M：更小、更低功耗，用于离散处理与微控制器。常见的使用该系列处理器内核的有K60（飞思卡尔用的较多，有Cortex-M3/M4等）,STM32（工业控制用的较多，有Cortex-M0/M3/M4等），像有些比较熟悉的产品中都用到了STM32，比如米家的小米扫地机器人（全志R16-CortexA7，STM32F103-CortexM3），大疆无人机（很多个STM32联合使用）。
* ARM指令集
* ARM指令集是ARM平台处理器的操作指令系统，指令集定义了数据处理、加载、处理与存储等等一些系列指令，有了指令集，处理器才能按照特定的要求去执行相关的动作，完成指定的操作。ARM的部分指令集版本与部分处理器版本的对应关系表如下表：

|  |  |
| --- | --- |
| **指令集版本** | **处理器版本** |
| ARMv1 | ARM1 |
| ARMv2 | ARM2,ARM3 |
| ARMv3 | ARM6,ARM7 |
| ARMv4 | ARM7TDMI,ARM9TDMI |
| ARMv5 | ARM9E,ARM10E |
| ARMv6 | ARM11,Cortex-M |
| ARMv7 | Cortex-A/R/M |
| ARMv8 | Cortex-A50 |

* 架构
  + 哈佛架构 一种将程序指令存储和数据存储分开的存储器结构，它的主要特点是将程序和数据存储在不同的存储空间中，即程序存储器和数据存储器是两个独立的存储器，每个存储器独立编址、独立访问。用到哈佛架构的处理器有ARM9/10/11以及之后。
  + 哈佛架构
  + 冯诺依曼（普林斯顿架构） 一种将程序指令存储器和数据存储器合并在一起的存储器结构。取指令和取操作数都在同一总线上，指令与代码在统一地址空间分布，也就是说有时候并分不出来某段地址上面是放的数据还是命令，通过分时复用的方式进行；缺点是在高速运行时，不能达到同时取指令和取操作数，从而形成了传输过程的瓶颈。由于程序指令存储地址和数据存储地址指向同一个存储器的不同物理位置，因此程序指令和数据的宽度相同。用到冯诺依曼架构的处理器有ARM7及之前的。
  + 冯诺依曼架构

**软件**

嵌入式行业的软件系统众多，包含操作系统、图形UI系统等等，其中较为常见的有Linux、RTOS类、Android、QT等等，下面根据不同的类型选取几个较为常见的软件系统进行简要概述。

* 操作系统
  + 非实时操作系统
  + linux：也有实时的linux系统，这里不做介绍。嵌入式里面应用最广泛的非实时操作系统非linux莫属，小到智能手表，大到汽车电子等等都有linux的身影。该系统于1991年由芬兰的linus torvalds实现，经过多年的发展已经成为功能极为强大的成熟的操作系统，并且源代码完全公开，商用完全免费。
  + WinCE：（Windows Embedded Compact），微软出品，有着比较强大的图形用户界面，专门为掌上电脑以及嵌入式设备设计。WinCE的实时性比linux更好一些，但是相比linux来说，它的可裁减、可移植性都弱了不少，并且linux拥有来自全球黑客的共同协作开发，从某方面来讲它的开发支持力度要比WinCE大。核心源代码开放，但是GUI不开放，商业使用需要一定费用。 现在在嵌入式消费电子领域，linux的占用量比WinCE要大，linux的开发者也比WinCE要多，相对来说，linux的职位、市场需求比WinCE要大，也就是说，学linux更容易找到工作哈。
  + 嵌入式实时操作系统（RTOS）
  + FreeRTOS：创始人RichardBarry，该操作系统是一个轻量级的实时操作系统，功能包括任务管理、内存管理、信号量、队列、软件定时器、协程等，在[FreeRTOS官网](http://www.freertos.org/)：[http://www.freertos.org/，这里可以找到任何FreeRTOS的学习资料与源代码。](http://www.freertos.org/%EF%BC%8C%E8%BF%99%E9%87%8C%E5%8F%AF%E4%BB%A5%E6%89%BE%E5%88%B0%E4%BB%BB%E4%BD%95FreeRTOS%E7%9A%84%E5%AD%A6%E4%B9%A0%E8%B5%84%E6%96%99%E4%B8%8E%E6%BA%90%E4%BB%A3%E7%A0%81%E3%80%82)
  + μC/OS：（Micro-Controller Operating System）截至目前，最新版本出到III，该系统前两个版本是完全开源的，第三个版本不完全开源，虽说源代码是开源的，但是如果要用于商业用途的话是要收费的（但是据说个人创业者免费，这个都要跟官网的协议进行同步确认），不过目前来说该系统的最新版本还是比FreeRTOS的功能更加强大一点，不过没有前者更加轻量。该系统是一个可以基于ROM运行的、可裁剪的、抢占式、实时多任务内核，具有高度可移植性，适用于微处理器与控制器。
  + VxWorks：该系统其实在消费类的嵌入式产品中并不是很常见，但是在军工、航天、自动化系统中比较常见，该系统拥有强大的开发套件与开发调试环境，其严格的测试流程保证了系统有着较高的稳定性，实时性较linux强太多。不过该系统的授权费用极为昂贵，各种组件都是单独拆开分别进行授权。该系统的入门较linux更为简单，但是网络上的资料极其少（估计是不开源的缘故），所以普通的智能家居、安防、消费电子产品领域，还是linux占大头，该系统则用于一些实时性、稳定性要求更高、对成本不是过于敏感的场合，比如航空航天、军工之类的设备上面。 上述前两个系统都可以运行在类似STM32以及ARM7/9内核处理器上面，从商业角度来讲，如果追求免费轻量可以使用FreeRTOS，如果追求功能的完备性，可以使用μC/OS。当然也可以作为个人学习使用，对于个人来说，两者都有可供获取的完整的源代码，并且免费使用，网络上（包括各自的官网）也有很多的学习资料可供选择，嵌入式学习是FreeRTOS以及μC/OS会是个非常不错的选择。有些四轴飞行器上面就有用到FreeRTOS，前两个实时操作系统对硬件的要求没有VxWorks高，在一些较为简单的、低成本硬件系统中使用前两者是一个不错的选择。
* UI
  + QT：跨平台的C++图形用户界面应用程序开发框架，于1991年由QT Company开发，可以用于开发GUI程序，先后被诺基亚与Digia收购。具有强大的跨平台普适性，可以支持windows、mac、ios、android等。在传统软件与企业应用领域用的较多。
  + Android：Android是基于linux内核开发的一套GUI系统，由google公司完成，目前为止出到Android O（8），现在有非常多的平板、盒子、手机都用到该系统，如果说以前的Android系统还不是那么好用，那么从6.0开始，到最新的版本已经非常好用了，当然，这个跟很多因素也有关，不仅仅是系统本身的进步。该系统是开源免费用于商业的，同时不是跟linux内核一样使用GPL协议，通过HAL将linux内核与上层framework隔开，既使用了linux内核，同时又保证了android相关的厂商不受GPL协议（产品用拥有到GPL协议的代码，则产品的代码也必须完全永久开源免费）的传染性所影响，保证了商家的利益，也为该系统的推广普及铺平道路。

关于Android系统：在初始的时候是google搞出来跟ios相爱相杀的，话说google有时候就是喜欢搞事情，谷歌做的很多项目起初完全就是没有奔着赚钱去的（谷歌公司的特性决定了它可以这么任性），android起初可能就是这样一个项目，只不过最后成功了。早期的android系统都说比较卡，这与几个方面有关：

* android的策略：android对屏幕的响应级别是比较低的，ios是优先响应屏幕事件。
* 系统碎片化严重：这个碎片化不是指内存碎片化，由于使用android系统的厂商众多，如小米、华为、三星等等，还不算无数“山寨厂商”，它们的屏幕大小，硬件都不甚一致，所以一个android程序或者android系统本身就需要适配如此多的硬件平台，这就注定要牺牲某些性能而导致系统的不流畅，更不用说有些厂商的硬件配置本身就跟不上，不像ios一样，硬件配置高，并且硬件风格极为统一。
* APP参差不齐：android系统本身的开放性，使得大量质量不合格的app运行与系统当中，而有的app干脆用常驻后台，开它个7、8个后台线程来满足自己的“特殊需求”，这也是跟国内用不了谷歌有关，ios的app有统一的发布渠道、统一的审核，这些都要比android规范很多。

不过现在android已经比以前好多了，系统碎片化的问题虽然还有，但是在慢慢规范，这个也与国内手机厂商的崛起有关，硬件配置慢慢跟上，app慢慢规范化、统一化。

**IC设计**

嵌入式有一个完整的上下游产业链，以手机为例，整个产业链的简要图示如下所示（当然省略了很多中间步骤，简化说明）：

手机上下游产业链

上图少了很多的东西，比如IP提供者不仅ARM，ARM提供处理器核心，还有别的IP也有其它的公司提供，比如方案不仅只有汇顶科技的指纹识别，也有其它方面的方案等等。**嵌入式软件工程师主要的职位需求基本就在黄色区域圈出来的部分**，这里简要介绍下IC设计方面的知识。

**公司**

IC设计类的公司基本分为三类，分别是IDM，Fabless，Foundry。

IP：知识产权，例如ARM核，A53之类。

SOC：片上系统，比如骁龙835、联发科X30、全志A20这些。

IC：集成电路，通常指一个行业，集成电路行业。

* IDM
* 全称Integrated Device Manufacturer，整合元件制造厂商，这种厂商的整个技术实力非常强大，技术领域横跨机械制造、软件开发、硬件架构设计、光学电子等等，放眼全世界也就那么几个。这种公司从IP开发，SOC芯片设计、芯片代工、封装测试再到销售全部囊括在内，属于垂直整合设计类公司，比较熟悉的有三星、英特尔这些牛气的大厂。由于整个的芯片制造过程涉及到的工业制造技术非常复杂，技术种类繁多，所以这种类型的公司通常具有很深的技术背景，门槛非常高，并且比较难以复制。
* Fabless
* 这类公司一般只管部分IP设计开发以及整个的SOC设计，技术实力弱点的公司可能用到的IP是购买的多，而技术实力强一点的，IP自己设计的较多，但不管怎样，最终的SOC设计肯定是公司自己完成的。这类公司就比较多了，比如：高通、联发科、海思、全志等等。这类公司的技术实力也不差，不要以为SOC设计仅仅是买几个IP，然后像搭积木一样毫不费力，其中涉及到各种功耗控制，IP协调等等技术，同样一些IP，有些公司出的SOC功耗就是低，有的公司就不行，芯片可以实现更加高级的制程。这类公司没有自己的代工厂，设计出来的SOC需要送到代工厂加工成芯片实体。
* Foundry
* 这类公司就负责芯片的代工，不做SOC的设计工作，这类公司的工业制造技术实力比较强大，代表着尖端的机械设计制造水平。这类公司也比较少，比如：中国台湾有台积电，大陆有中芯国际这些，都属于Foundry类的IC公司。这种公司不同于一般的类似服装加工这类劳动密集型的工厂，由于本身行业产品的特殊性，使得加工制造一颗芯片与加工制造一件衣服的难度完全不是在同一等级，纳米级的微观尺度加工制造本身就带来了很多技术难题。不过这类公司的重点不在于我们通常理解的软件设计，而在于机械制造上面，如何去制造更加低功耗的、更高级工艺的芯片是它们关注的重点。

**芯片制程**

芯片制程早期有微米级别，到近期最常见的有28nm,22nm,16nm,14nm,10nm。截至目前，最新有高通的10nm制程的SOC。那个XXnm可以理解为大规模集成电路中晶体管的尺寸，为什么要追求更小的尺寸呢？简单说就是由于晶体管的作用，电子传递时间随尺寸的减少而减少（22nm之前的可以看作简单的正比例关系），所以尺寸越小，整体芯片的CPU时钟频率就越高，性能就越好。另外尺寸更小，就表明了在晶圆面积一定的情况下可以容纳更多的晶体管（更多的功能电路），同时整体的功耗也会降低，成本随之下降，这也是为什么各大SOC厂商、芯片代工厂都追求更低尺寸的制造工艺的原因。

题外话：我们经常看到的黑色芯片并不是真正的晶圆电路部分，那些黑色的是外面的树脂（常用环氧树脂）封装，引出的管脚内部才接到真正的“芯片”上面，通常情况下，内部的晶圆电路要比经过环氧树脂封装之后的体积要小得多。

**行业**

**职位方向**

通常来讲，嵌入式软件主要有几个方向：应用，驱动，算法，系统。

* 应用
* 有协议开发，UI开发、业务逻辑开发等等。比如使用QT开发一个汽车智能中控的图形用户界面程序，使用安卓开发一个OTT盒子的应用程序。我们常说的嵌入式应用开发并不包括安卓APP或者IOS APP这些，那些属于移动开发范畴，一般不把这些看作嵌入式的应用开发。比如一款运动相机，它上面有录像、截图、回放等等功能，而这些功能可以分为一个个的应用（也可能这些都是同一个应用），这些功能常常基于linux系统应用编程实现，实现这些应用的过程就是嵌入式中的应用开发。其余的还有像GPS导航终端设备中的导航程序开发，蓝牙设备上面的连接程序，古时候MP3/MP4播放器的播控程序等等都属于嵌入式应用开发的范畴。有些常见的协议开发也属于应用范畴，比如蓝牙协议开发，TCP/IP应用层协议开发，Zigbee模块开发等等。
* 驱动
* 驱动常见于芯片原厂，说的比较多的是linux驱动开发工程师。一般驱动开发主要工作是做BSP（板级支持包，围绕SOC片上系统硬件来做），也有做外设驱动的（sensor-摄像头，网卡等等），对于一家芯片原厂（Fabless类型的公司）来说，一个新出的芯片是需要开发人员做从bootloader到linux内核的适配的，其中包含有USB、flash、I2C、Camera、Framebuffer、SPI、串口、DMA、网卡等等模块的驱动，不管是在bootloader下的驱动还是集成到linux内核里面的驱动的开发都可以说是驱动开发。这些要求要对硬件的各种时序协议有较多的开发经验，比如USB协议、I2C时序、MIPI/CSI（屏、Camera）等等协议与时序要求。还需要对linux内核的架构熟练掌握，一个驱动要想适配到linux内核里面，最好还是按照内核相关模块的框架来实现的好，完全按照自己的方式来实现相关的驱动的话易用性、扩展性、兼容性都可能比较差。内核里面有I2C子系统，SPI子系统，输入子系统，V4L2视频框架，USB子系统，网络等块设备驱动框架等等，每个驱动工程师需要对其中的某一块或者某几块非常熟悉才能够胜任驱动开发的工作。
* 算法
* 一款嵌入式产品可能会包含有很多的算法，比如对一款运动相机来说，它需要有视频编解码算法、图像处理算法（ISP）、防抖算法。而在AI人工智能兴起之后，一些嵌入式产品如安防相机也需要集成人脸识别、车牌识别、运动检测、景深识别等等算法。更多的时候，算法工程师的工作不是去设计一个新的算法，比如视频编解码中的H264、H265算法已经比较成熟了，不需要自己再去费老大劲搞出来一个性能并不如已有算法好的算法，况且大部分公司的财力、技术并不允许自行实现某些高性能算法。更多的时候，算法工程师要做的是在某一个平台上面去实现一个算法，针对某一个平台去优化某个算法，这些是软件层面的，而有些芯片是直接将编解码模块做成硬件放在芯片内部的，这部分我倒不清楚具体是软件还是硬件的职能，亦或是两者结合。人工智能潮流的到来让嵌入式领域也需要更多的优秀算法工程是参与，比如语音识别（常用语智能音箱，智能家居等等），小米的扫地机器人，京东的叮咚音箱，未来还有更多的空调家电类也会集成语音识别功能，而语音识别需要多种音频有关的算法工程师通力协作来共同完成一款功能齐全、优秀的产品。安防产业中就更不用说了，大量的视频算法相关的工程师都是急缺职位。随着人工智能的逐渐发展普及，嵌入式领域的算法工程师需求量会越来越大，而且工资也一定不会低。
* 系统
* 这个了解的不多，理解比较浅显，举一个了解过的简单的例子，就拿一款安防产品来说：产品的功能以及规格由产品经理等确定，然后整个系统的设计需要系统工程师的框架规划，比如采用什么操作系统？如何建立代码工程（包括版本管理工具的选择、编译系统的搭建、代码框架的构思），整个产品使用什么框架进行设计（视频类的开源框架有OpenMAX，OpenGL等等，也可自行设计），开发语言的选择（什么时候用C，什么时候用C++），功能模块的划分（去耦合）等等。

**产品类别**

嵌入式领域的产品众多，列举部分进行简介：

* OTT盒子
* Over The Top。作用是获取网络视频资讯，在终端进行显示播放。不同于老式的IPTV机顶盒，老式的机顶盒连接的不是互联网，而是电视网，相当于一个局域网，老式的机顶盒仅显示终端限于电视，而OTT盒子直接连接到互联网，显示终端可以是通过wifi或者蓝牙等协议连接的显示设备，并且有的还可以支持多屏显示，目前为止国内对OTT盒子的管控比较严格，比如OTT在国内不允许观看直播节目，以及不允许通过USB等媒介安装应用等，不过OTT盒子的总体需求量是在逐年上升的，在国外OTT盒子算是比较大的一个市场了。国内常见的盒子芯片厂商有海思、全志、瑞芯微、炬力等。盒子终端设备厂商有天猫、京东、三大电信运营商等。
* 路由器
* 这个都很熟悉了，现在几乎每家都需要，属于硬性需求，路由器的开发对网络这块的要求比较高。
* IPC
* IP Camera（网络录像机），常见于安防类产品，比如小区监控，交通灯以及大街小巷无处不在的摄像头等等，主体功能是实时采集视像、音频信息并通过互联网网路发往服务器进行保存，有的也可以进行音频双向传播。老式的模拟录像机已经逐渐被淘汰，取代代之的是数字、智能录相机，现在主流的IPC都在往4K分辨率发展，也已经有8K分辨率的解决方案了。随着人工智能的发展，现在IPC里面逐渐集成了人脸识别，车牌识别，移动侦测，景深探测等等AI智能算法。在最新2017年的深圳安博会现场，有多家安防相关的公司进行展览。有做智能算法的，如商汤、旷视、云从，商汤之前在深圳的文博会现场采集了20万人次的人像数据，同时比对出了20多个有前科的人员，这几个公司的智能算法都是国内甚至国际前列，属于独角兽级别的创业公司。有做芯片方案的，比如海思、全志，它们提供IPC的芯片以及软件解决方案，现在的芯片大都支持4K录像，集成人工智能算法。也有做终端IPC产品的，如海康、大华、宇视（合称海大宇，市场占有率80%以上）等等，海康的市值从2016年9月到目前为止已经上涨了千亿以上，侧面也说明了有人工智能加持的安防产品需求增长迅速。随着智慧城市的提出以及不断建设，安防产品的需求量将会进一步加大，有预测，到2022年，安防产品的市场规模会达到万亿人民币级别（现在已经接近6000亿了）。
* 智能音箱
* 亚马逊不一定是最早做智能音箱的，但是它的echo音箱的的确确带了一波潮流，echo发布于2014年，经过几年的发展，已经成为最火的智能家居产品之一。在国内比较知名的智能音箱产品有京东的叮咚音箱，小米的智能音箱，国内销量最大的是叮咚音箱。智能音箱开始的时候乍一看并不觉得有多大的市场需求量，当然这个跟多方面有关：一个是它还显得不够“智能”，能做的事情还比较少，也就是产品做的不够好；另一方面也是由于现在智能家居的产品普及率还远没有手机那么广，换言之，就是智能家居这个氛围、意识在国内还没有完全形成。但是随着技术的发展，生活水平的提高，生活习惯的改变，智能音箱的普及率以及需求也会越来越高。目前智能音箱主要的技术模块有：语音识别（包括硬件模块与软件语音识别算法），主控芯片，音箱产品终端平台设计。语音识别模块知名的有科大讯飞、思必驰等；主控芯片知名提供商有联发科（亚马逊echo）、全志科技（京东叮咚音箱）、瑞芯微等；音箱厂家漫步者、Doss等。
* 扫地机器人
* 这个也算是未来家庭必备了吧（懒人必备）。扫地机器人看起来好像并没有什么技术含量，不就扫个地么？其实扫地机器人有许多方面的规格要达成，并且不同厂家不同价位的扫地机器人的各方面规格达成与实际表现也不同。最容易想得到的就是路径规划能力了，不同公司的机器人路径规划算法种类不同，优劣程度也不同，常见的有两种，一种可以称之为随机规划，也就是走到哪扫到哪，完全没有精确的路径规划，清洁力度随着时间的增加而非线性增加。还有一种是有较为合理的路径规划的，它首先会对整个室内做一次布局扫描，然后根据室内布局进行路径规划，一般是U型环回前进的，这样不会出现愣头愣脑瞎撞一气的低效清洁的状况。另外障碍物的检测也是比较重要的模块，有的机器检测到障碍物直接怼上去才知道转弯，而有的则是离障碍物还有一段距离就转弯（会导致清扫有死角，不干净），有的是遇到障碍物提前减速，然后轻轻撞上之后再转弯（清扫无死角，不会损坏家具机器）。有的机器面对有坡的地方会清扫不力，别的还有狭小空间的清扫能力，遇到缠绕物体（电线、绳子）的逃脱能力等等都是衡量其优秀与否的标准。而这些都需要优秀的算法作为支撑。扫地机器人的品牌有iRobot、科沃斯、小米等等。
* 行车记录仪、后视镜、汽车中控
* 对于有车人士来说是刚需。近年来碰瓷事件频发，有时候有个行车记录仪少了很多麻烦，不仅是这个（毕竟小概率），行车记录仪重点还是在于交通事故等等的录像备份，有车人士必不可少。行车记录仪的主要几个参数就是照片分辨率、镜头视场宽度、帧率与噪声。主流的视场角在130~170度之间，因为事故发生时前撞和侧撞的概率比较大，需要比较广角的镜头才能够满足需求。分辨率与帧率啥的就不用说了。做汽车电子比较著名的芯片厂商有海思、安霸、全志、联咏等。终端产品也有非常多的公司，不太了解就不多说了。
* SDV
* 也即运动相机，比如GoPro、小米、小蚁等等。现在主流的运动相机都往4K30帧/60帧这个配置发展，其实运动相机在国内的需求并不是太大，因为大家大部分都忙着上班、上学，再说国内的运动风气还是远没有那些欧美发达国家那么浓厚，所以SDV在国外的市场需求还是可以的。现在的SDV大都要集成防抖功能，不管是软件防抖还是光学防抖，如果是软件防抖的话对处理器的性能以及软件算法的要求会比较高，否则软件防抖根本没有视觉上的效果。

可以看出来，上面有很大一部分产品都是需要做视频支持的，不管是录像还是播放，所以视频处理算法像H264、H265、MJPEG编解码这些还是非常有应用需求的，所以与之相关的职位也不少，另外对于没有上安卓系统的产品来说，比如IPC，SDV，行车记录仪等等（这些都没有很高的GUI界面要求，所以用不着上安卓系统），还需要在linux系统上面做视频组件的开发，所以单单视频这一块就可以产生巨大的职位需求。

* 其它
* 如共享单车、共享充电宝等一众共享产品，未来肯定还会有更多的共享电子产品。智能家居里面的儿童机器人、教育机器人等等。

**所需技能**

如果是驱动开发的话就需要对linux、RTOS等系统还有硬件的要求就比较高，驱动开发首先需要能够看得懂各种原理图、技术手册，对英文的要求也是比较高。像linux驱动开发就需要对linux内核各种子系统进行深入了解，就拿上面那些产品中的涉及到视频的来说，linux的V4L2架构的熟知是必不可少的，因为这是整个软件视频流的源头，做视频开发是绕不开这一块的。

* 驱动通用技能
* 能看懂各种硬件原理图与技术手册，包括时序图、寄存器、管脚接线等等，作为驱动工程师时常要与硬件打交道，免不了调各个模块的时候用到万用表、示波器、时序分析仪一类的工具，这些也要会用。另外对各种协议（不同的职位方向只要知道不同侧重点即可，不必全通）如I2C、SPI、MIPI-CSI、USB、网络栈等等。另外英文水平得要够好，否则大部分全英的技术手册，各种spec文档根本看不懂的。就拿一个sensor（比如大名鼎鼎的索尼imx系列摄像头）驱动工程师来说，就需要掌握如下技能：
  + 不同接口标准与协议的了解：sensor的数据接口标准与协议有LVDS、MIPI、Parallel（并口），命令交互接口标准与协议有I2C、SPI、HISPI等。
  + sensor的datasheet阅读与调试工具使用：每种接口需要有与之相配的协议时序来驱动，在调试过程当中难免会出现咋调都没反应的情况，此时就需要示波器、时序分析仪、万用表来一步一步排除硬件问题或者时序驱动问题（实际上正确的步骤应该是先确定硬件以及时序，然后再是调试出图）。
  + linux的V4L2框架：该框架是专门用于linux的视频流处理框架，几乎所有运行在linux系统中的sensor都不可避免要用到这个框架，在内核当中omap3isp是一个很好的参考例程。
  + ISP以及图像调优：虽说这部分的工作不一定跟上面一部分是同一个工程师负责，但是ISP驱动框架还是需要掌握的，不同的IC都有自己的ISP模块，大同小异，但是需要知道如何去编写一个V4L2框架下的ISP模块驱动。

其它的像简单数据结构如双向链表也是需要掌握的，linux内核里面有大量的地方用到双向链表数据结构。C语言指针的各种灵活运用也是必须掌握的。

* 应用
* 数据结构、各种应用框架、少量的算法知识（如排序算法）、系统级应用编程、网络协议编程、软件结构设计、模块化思想、脚本的灵活运用、linux命令行的使用、编译器的熟练使用、C/C++,JAVA等语言的掌握等等。拿一个产品举例，比如IPC（不是针对职位来说，针对这个产品来说）：
  + 视频播控以及用户界面等框架：比如OpenMAX、OpenGL（用于图像处理）等开源视频框架，一个IPC产品必不可少的组成部分就是视频录制与处理，这些都有可能用得到上面两个开源框架。用户界面像QT、MINI GUI等等也是需要掌握的，这个在一些需要图形用户界面的产品里面是必须的。
  + linux应用编程与命令行：linux系统环境高级编程，包括多线程、文件编程、网络编程、信号量等。各种命令行操作如文件删除、新建、拷贝、nfs等网络指令、内存操作命令、用户管理等等。
  + 网络协议：IPC通常情况下需要通过网线来进行远程视频传输，这就要求对各种网络传输协议要掌握（包括应用层、传输层等），比如TCP/IP、HTTP、RTSP以及ONVIF这种网络传输框架等等。
  + 数据结构与算法：像链表、数组这种就不用说了，还有些设计方法与模型也要掌握，比如最常见的有生产者消费者模型必须得掌握，还有写哈希表、快速查找表等数据结构也是必要的。算法的话在嵌入式应用上面要求并不是很高，常用的排序、搜索等算法掌握就行了。
  + 模块化设计：软件功能的模块化设计，去耦合，一个好的软件框架要能够达到很方便的就能够增加、减少某个功能而对其它功能不产生影响，要能够达到对同类型的功能进行聚合归并管理，减少代码量，提高复用率。对于整个软件结构需要设计的有条例，不能够说这坨代码让人看上去一脸懵，甚至编写者过一段时间都理不清软件逻辑，这样整个软件的可维护性、可升级、可重构性都会大打折扣。
  + 各种语言的掌握：像C/C++这些是必须要用到的，不管是应用方案层代码还是GUI层代码，C++是必不可少的开发语言，像linux系统编程就必须得掌握C语言。当然，有些时候也会用到JAVA，比如有的QT程序开发。

其他的算法这些我就不分类说了，像视频这块就需要掌握H264,H265,MJPEG,JPEG的编解码，ISP的3A算法，图像识别算法（人脸检测、车牌识别等），软件防抖等等。应用层的各种格式有时候也需要掌握，比如MP4、AVI、RMVB、MKV等等的封装与解封装。音频的话我知道的有音频编解码像MPEG编解码（常见于mp3封装格式），还有G711，G721系列，AC1系列等等，与音频质量有关的像音频滤波降噪，音质增强，杜比音效等等，随着AI时代的到来，音频识别算法也日渐急需。

* 通用技能
  + git等版本管理：像SVN，git是比较热门的版本管理工具，对于软件代码工程来说，如果没有版本管理，那将是一场灾难，试想一个数千万行的代码工程如果没有版本管理，那么代码的合并、代码的提交、代码的审查、回退等等会多么困难，整个代码工程会变得混乱不堪，光是代码版本管理就能耗尽整个团队的精力。git与SVN不同，前者是分布式的代码版本管理协作，无需联网，从服务器仓库下载下来直接在本地编辑，本地提交完了之后同步到服务器上即可，随处可用，但是SVN就不行，它是集中式的协作模式，必须依靠中央服务器版本库才能够正常工作。
  + linux命令行操作、脚本编写：由于驱动、应用等等开发都有很大一部分时间是在linux机器上完成的，所以linux上面的命令操作必须掌握。脚本的话是用于自动化工作的，比如编译过程，有了脚本可以使得整个编译过程变得非常方便，如果是单个文件操作，成千上百个文件的编译过程会把人搞死的。另外代码工程的编译系统的建立极度依赖于脚本以及makefile，而这些又依赖于linux系统，所以这些技能是必须掌握的。
  + 良好的编程习惯：包括但不限于良好的变量、函数命名习惯（一致、清晰易懂），让人看到变量或者函数名就知道这个变量是什么类型的，主要作用是什么，这个函数是干嘛用的。恰到好处的注释，代码的关键地方要加上注释，比如函数作用，参数与返回值必须加以说明，结构体等等变量的作用必须加以说明，函数或者结构体使用时的特别注意事项要加以说明，代码中比较难懂的逻辑要加以说明，最好能用函数名或者变量名或者代码本身来兼具注释的功能。良好的逻辑实现，不投机取巧耍花招（虽然看起来很酷，但是代码易读性、易维护性变差）。代码风格统一，比如花括号’{‘的位置（if ( i < 0 ) {，换行的都是异教徒哈哈），空格的使用，一行不能够太长等等。每个公司都有自己的编程规范，按照编程规范来，有不足向公司提出改进，不要一意孤行。

**未来发展方向**

嵌入式也算是个比较古老的行业了，可以说在古时候电脑被发明的时候就有所谓嵌入式这个行业了，随着不断地发展，嵌入式软件有一个比较明显的趋向：系统化、AI智能化、物联网。

* 系统化
* 现在的电子产品都在朝着系统化的道路前进，现在甚至连钟表、手环等产品里面都跑着一个系统，不管是RTOS类的还是linux类的，亦或是自家开发的，总之，嵌入式产品越来越多的使用到了操作系统级别的软件平台，而不再是以前的裸机裸跑的情况了。
* AI智能化：相比今年来各大行业都被AI刷了一波节奏，有的是噱头，有的则是实实在在的智能化，有相关的落地产品。而嵌入式就属于后者，这里说的AI话不是指强人工智能，而是指在嵌入式软件当中越来越多地用到了人脸识别、语音识别、环境侦测等等智能算法，未来还不排除会加入深度学习（实际上智能手机已经在慢慢加入硬件级别的自主学习算法）。AI智能化的产品应用有四旋翼飞行器、智能音箱、智能安防、手机、机器人等等。
* 物联网：万物互联、智能家居现在看起来还是感觉比较遥远，但是已经能够看到方向、摸到入口了。ARM为物联网时代的到来还专门成立了物联网平台以及为物联网服务的操作系统。而国内的一些行业、厂商也在为物联网、智能家居而努力着，比如海尔、格里、美的的智能家电，小米的扫地机器人，智能音箱，家庭化小型机器人等等产品的研发与生产

最后，来一张2017集成电路行业薪资表（来源网络，出处见水印），打打鸡血，嵌入式的同行们一起努力哇：

2017集成电路行业薪资表

**轶事**

我最早接触嵌入式算是大学二年级，那个时候做过飞思卡尔、全国电赛等等。我参加的是第十届飞思卡尔智能车竞赛，等到了第十一届的时候飞思卡尔被恩智浦收购了（那一届叫恩智浦杯），而等到再往后恩智浦又被高通收购了（又改回飞思卡尔），现在想想也是有点感慨。

git编写者也是linux系统的创造者linus（太牛逼了），github是基于git的一个代码托管网站，linus本人也进驻了，有自己的代码仓库，在里面可以看到linux、git最新版本的源代码（代码世界的两大顶梁柱级别的工具），而github也慢慢发展为最受欢迎的开源代码托管网站以及社区。最初linux是被自由软件基金会GNU选中作为其开源软件的一员而慢慢发展壮大（也算是一部分机缘巧合），GNU致力于软件的开源、免费，是真正的永久开源、免费，这个相比对于各大码农来说是非常友好的，也是一大部分码农向往的愿景，像GCC，Emac，gdb等等牛逼的软件工具都出自于GNU这个组织。

ARM近年来也在开发一项技术，叫做ACP，这个技术可以使得IC上面的外设可以不通过CPU即可相互访问对方的内存，想下DMA吧，ACP要做到不止于DMA，当然这项技术什么时候、能否问世还有待商榷。

如果觉得本文章不错，请关注微信公众号-YellowMax多多支持，查看更多文章

欢迎转发、关注、点赞一波

本文章摘自http://www.yellowmax2001.com/2017/10/28/2017嵌入式软件行业现状及概述/