What is so special about Embedded Software Development?

spilberg.nl/blog/what-is-so-special-about-embedded-software-development/

Bob Peters, Guest blogger / Trendwatcher Embedded Systems

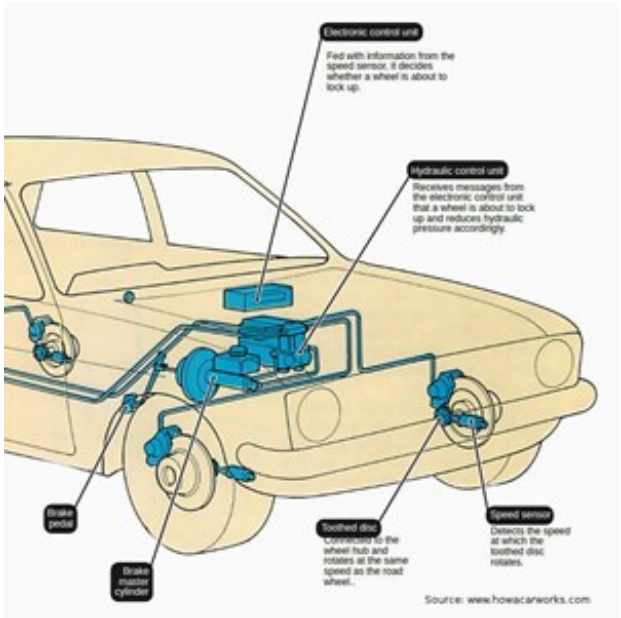
Many people do not (exactly) know what the difference is between Software Development and Embedded Software Development. This can lead to differences in perspective, especially when it comes to effort planning. Most features will cost more time to implement on an Embedded System compared to a PC / Server platform. You might wonder why this is the case? I will try to elaborate on the main differences between Software Development and Embedded Software Development.

Limited resources

The hardware of an Embedded System is most often a custom made, resource constrained, platform with a minimum amount of resources. In a modern PC we talk about Gigabytes (Giga = 1 000 000 000) of RAM, whereas on an Embedded System it is not uncommon to work with 256 kilobytes (kilo = 1000) of RAM or less. Processing power on an Embedded platform runs in the tens or a hundred Megahertz (Mega = 1 000 000) with one or two cores, compared to a PC that has 4 cores running on 3 Gigahertz (Giga = 1 000 000 000) or more. As you can see the differences are multiple orders of magnitude.

Imagine you need to implement a http server, also called web server. It sends the data of webpages to the requesting clients. Memory is scarce on an Embedded System and therefore allocated statically, to overcome the situation of running out of memory at runtime. This means we need to know up front how large buffers need to be, since the web pages do not live in a full-fledged filesystem, but in an in-memory simplistic (filesystem) storage. Most of the time the webpages are preprogrammed on the embedded device, together with rest of the software. Moreover, the number of supported connections is fixed, to minimize unnecessary memory allocation. On a PC the webpages live on a file system and are dynamically created, meaning we make the files at the time we need them, in the size we need them. More memory is allocated if a new connection/request is made. As you can see the Embedded System needs upfront configuration and allocation, hence limited resources forces (System) Architects to think carefully about their design decisions.

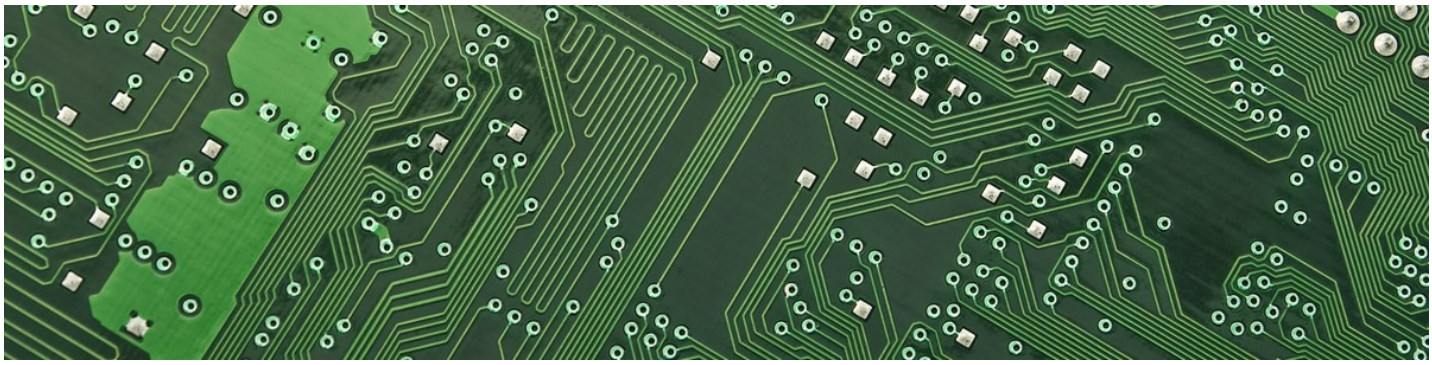
Debugging



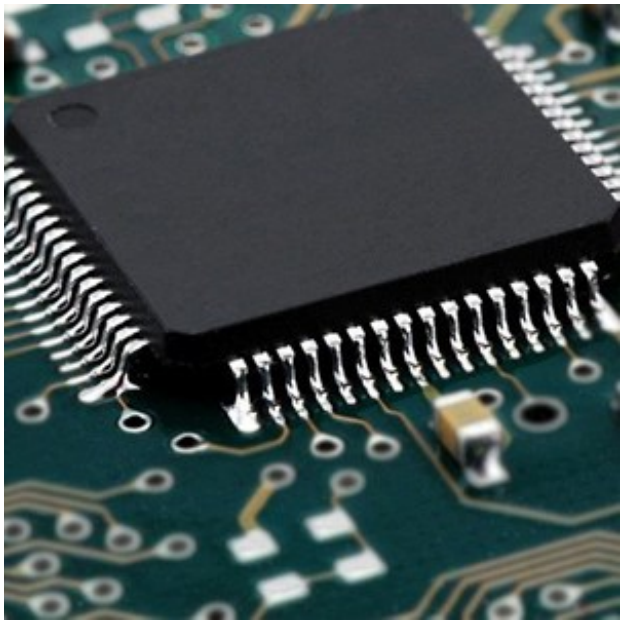
Another difference we will encounter is way we can debug. Embedded Software is closely coupled to the hardware that it is executing on. Software running on a PC, is most of the time loosely coupled to the hardware. If you look at debugging, a Software Engineer uses breakpoints and prints information to the screen. The Embedded Software Developer will use less breakpoints, more printing to a serial port, and knows its way with an oscilloscope or other instruments to observe what the hardware is doing. Most Embedded Platforms have no screen / keyboard, which makes visibility / control for debugging low, and therefore more difficult.

Moreover, a Software Engineer is developing software that can be directly executed on his local machine. Embedded Development is conducted on a PC, the software is compiled and a binary is uploaded to the target platform. The processor architecture of the embedded platform is most of the times different from that of the PC, therefore the PC cannot execute this software natively. This brings extra complexity in debugging, since you are not developing on the target platform directly, which requires

extra tooling like a programmer/debugger.



Real-time Operating System



Embedded Systems come in two flavors, namely with and without an Operating System (OS). If a system is running without an OS, it is referred to as running the code bare metal. Systems that run code bare metal, have a reasonable small code size / functionality. Embedded Systems with an OS, are different from the OSes running on a PC / Server. Embedded Operating Systems have less services and are often Real-time. Many Embedded Systems are executing time critical tasks. Some of these tasks may never be executed too late, else they can cause damage, this category is called hard Real-time. Think for instance of the control systems in a car, like the Anti-lock Braking System (ABS). If the task would miss its deadline it can have catastrophic consequences. That is why timing for these systems needs to be guaranteed. To guarantee timing, the system needs to be deterministic in memory usage, scheduling, context switching, priority etc. You might expect this has quite some impact in the way these systems are designed, developed, and tested compared to software running on a general-purpose platform like a PC.

There are more differences between Software Development and Embedded Software Development, but I think these are the most prominent ones. I hope it gave you some insight on Software Development for Embedded Systems.

This was my first blog post as guest blogger for Spilberg, I hope you enjoyed it! Please feel free to contact me or to visit my blog EmbedWise.com. Next time I will focus on Real Time Operating Systems. Follow Spilberg on Linkedin, Facebook or Twitter and we will keep you posted.

Guest blogger / Trendwatcher Embedded Systems

Bob Peters is an Embedded Systems Enthusiast, Blogger, Developer, Entrepreneurial, interested in startups and always curious. Bob holds a Master degree in Embedded Systems and lives in the Netherlands. Checkout Bob's articles on EmbedWise.com or follow him on Twitter