ieee·usa today's engineer online

feature

02.08

# Help Wanted: Embedded Engineers
## Why the United States is losing its edge in embedded systems…

By Mike Anderson, Chief Scientist, The PTR Group, Inc.

**Embedded Systems are Everywhere**

I read a few years ago that the average individual in an industrialized nation came into contact with over 48 embedded systems every day. With all of the cellular phones, portable media player/MP3 devices, global positioning system (GPS) units, set-top boxes, digital video recorders, automobile telematics systems and digital televisions that were introduced over the past couple of years, I suspect that this number is probably on the low side today. We are surrounded by embedded systems comprised of custom hardware and software designs.

So, just what is an embedded system? Who develops them? What is special about embedded systems and the people who build them? And, more importantly, why is it growing harder to find these people to hire them? I'll try to answer these questions and more within this piece.

An embedded system can be characterized as any device in which you inherently know there must be a computer in there someplace, but you're just not sure where. This is not to be confused with real-time systems. Systems that have real-time deadlines may or may not be embedded, and not all embedded systems have real-time deadlines. There is considerable overlap for sure, but they are not one and the same.

For instance, a GPS locator has a computer inside. I suspect that everyone knows that. But, we typically don't attach a keyboard and mouse to it although it's probably one of the most computer-like of the embedded systems since the user must input addresses in some way. Nor does the user typically vaporize if the device takes 30 seconds to acquire a satellite fix rather than 15 seconds. The GPS is an embedded device, but we probably wouldn't classify it as a real-time device.

On the other hand, anti-lock brakes are computer controlled as well. Our input into the anti-lock brakes of the car is limited to the brake pedal. The rest of the operation is completely based on other sensors and the environment. It is clearly an embedded system. However, the anti-lock brake system is a real-time system as well. If it doesn't function within the real-time deadline constraints, then people get injured.

Nonetheless, these two examples have much in common. Both operate in a constrained resource environment. Both must be economically viable. Both must interact with the user and their surroundings (satellite downlink in the case of the GPS and the slickness of the surface for the anti-lock brakes). Both have hardware and firmware/software that are running on fairly low-power devices. It is because of many of the similarities that embedded systems and real-time systems are frequently grouped together.

**Embedded Systems Design and Engineering Economics**

Embedded systems development typically requires a different mindset than we find in the desktop environments. Embedded systems are frequently resource-limited. These systems have low-power processors, possible battery operation, and limited memory and storage. And yet, there are increasing demands from consumers for high-end features such as video/audio playback, cameras and satellite connectivity ala GPS. While trying to satisfy these requirements, the embedded system developer also needs to consider the cost of the bill of materials (BOM) for the unit. We need to keep in mind that there may be literally millions of a particular cell phone or GPS unit produced. Embedded designers need to pack in lots of features, but keep the costs down so the units are affordable.

Due to the desire to keep costs down, embedded developers need to squeeze every last bit of performance from the hardware. Higher processor speeds generate more heat and that heat has to be dumped somehow. We certainly do not want a cell phone with a fan built in. Nor do we want a device that gets so hot that we cannot handle it comfortably. So, at least some embedded designers need to understand thermal loading and MIPS/watt ratios. Nonetheless, virtually all embedded system developers need an intimate knowledge of the hardware and how the software will interact with it. This knowledge is often times down to the hardware-register level of the device.

Another, often neglected, aspect to the engineering economics of embedded systems relates to the programming languages we use. Languages that consume lots of RAM because of virtual machines or code generation cause the designer to increase the chip count and/or memory density. Higher memory densities often generate more heat and consume more power. This makes the device more expensive to produce and less desirable to the consumer.

For example, a single C++ catch/throw on the PowerPC can add as much as 2,500+ instructions to the code size. And the use of dynamic constructors and destructors will cause many underlying operating systems to lock and unlock interrupts as the memory is allocated and de-allocated when the application enters and exits scope. That's not to say that C++ is a bad language or that it is not acceptable in embedded systems. But rather that the embedded developer needs to know the effects on the overall system cost of the languages and

programming techniques that are used.

So, where does the embedded system designer gain this type of knowledge? Currently, companies are paying to have developers learn these concepts on the job. Unfortunately, this means that they will make lots of mistakes on the job as well — it's just part of the learning process. But, these mistakes hurt U.S. industry and can cost jobs and reputations depending on the severity of the errors. Another aspect of this on-the-job training is that once the company has invested in training a developer, that developer is now much more valuable and likely to be hired away by a competitor, which means the investment in training is lost to the company who paid for it.

What many in the embedded systems community would like to see is that new graduates already have some of this knowledge from their college/university education. The establishment of embedded systems development specializations in higher learning institutions would go a long way in preparing students who choose this path to be immediately useful in development roles when they leave the university/college setting. Consequently, the cost of hiring a "fresh-out" is greatly reduced because the new hire doesn't need as much training in embedded development concepts, but rather just the specialization of the application. For example, they need to learn how cellular phones work rather than how to boot and run from flash memory.

The ability of a new hire to "hit the ground running" could save U.S. embedded systems companies millions of dollars each year. This is money that could be reinvested in additional research and development to make them more competitive in the world market, or simply yield a higher return to their investors. Regardless of what is done with the "savings," the incoming engineer would make U.S. companies more productive.

**The U.S. Embedded System Developer Shortage Crisis**

This brings us to the current state of the U.S. education system. There has been considerable press as of late concerning the dwindling numbers of engineers that are graduating from higher learning institutions. If we are to believe some sources, engineering, in general, is facing a crisis of sorts.

Various studies have produced reams of data on the growing shortage of engineers here in the U.S. These statistics are then used as leverage to try to raise the number of H-1B visas granted in the U.S. to make up for the shortfall. Others may argue that the decline in numbers of engineers is merely a manipulation of the statistics to justify bringing in relatively cheap labor from outside the U.S. and depressing wages. I'm inclined to believe that there is perhaps a grain of truth to both viewpoints. However, I feel that it is not really the numbers of graduates, but rather their understanding of computing in general that is alarming.

Engineering is not easy. Being a successful engineer takes discipline and a desire to "make things work." And we rely on colleges and universities to provide the foundation of education so that graduating engineers can be successfully utilized to make the products society has come to depend on such as the cellular phones, GPS locators, MP3 players, anti-lock brakes and much more.

Electrical engineers, computer engineers and computer scientists are all likely candidates for embedded-systems engineering positions. The electrical and computer engineers typically have some background in computer hardware and computer architecture. And even if they are not trained in software development, they can usually be brought up to speed in a few months. Unfortunately, it is the computer science (CS) graduates that are failing to make the grade, so to speak. As a CS major myself, I can tell you that what I learned as CS in my masters program 20 years ago was much more hardware/architecture focused and bears little resemblance to today's CS curricula.

Today, a CS education is more like information technology and focuses on database and Web design with a smattering of Java and data structures thrown in for good measure. Unfortunately, such coursework prepares graduates for types of jobs that are commonly offshored. I have even overheard parents telling their children not to go into computers/engineering because their jobs will simply be sent outside of the U.S. and they will end up unemployed! This kind of mindset might be largely responsible for any shortage in students entering engineering, whether perceived or actual.

However, it has been my experience that the folks being brought in on H-1Bs are no more educated in embedded systems development techniques than our home-grown engineers. In fact, as someone who has provided embedded systems training seminars over the past decade, I can attest to the fact that very few engineering graduates, regardless of their country of origin, understand the issues of embedded systems development. In addition, all of the training investment simply disappears if the H-1B holder goes back to their home country. This becomes a brain-drain of sorts and makes the U.S. less competitive. If the United States is going to allow entry to H-1B visa holders, then we need to provide mechanisms for extensions for key technology areas to maintain the brain trust.

So, does the United States have a shortage of engineers? Well, if we are talking about engineers who *understand* how computers work, then I'm afraid we do. I can understand not wanting to get into the nitty-gritty of bus transceiver logic, etc. Some folks like that level of design, and I'm happy that we have them. But, as embedded systems developers, we must all understand the effects of memory management units (MMUs), caches, instruction pipeline flushes, scheduling policies, context switches and the like, to be effective in creating working embedded systems.

As for today's CS programs, it seems that long gone are the computer architecture classes, writing code in assembly language (or even C at this point) and engineering software economics. In fact, a large number of CS majors apparently believe that everything can be implemented in a virtual machine and that both memory and central processing unit (CPU) cycles are infinite. Based on a typical x86-centric desktop computer view, these observations are perhaps less unrealistic. However, the net result of this perspective is the code bloat we perceive in certain operating systems and applications. Certainly, our "lean and mean" perception of what it means to be an embedded system is not consistent with the infinite CPU and memory viewpoint.

The educators and curricula developers are quick to claim that they are simply teaching their students what the industry is demanding. This is often based on a quick look at the local want ads or a search of employment sites on the Internet. Java, PHP and HTML developers all seem to be

in high demand. However, these are again skill sets that are easily outsourced. Proximity to the hardware and the hardware designers as well as test equipment is precisely why it is so much harder to outsource embedded development jobs.

The misreading of demand is not entirely the fault of the colleges and universities. No one has measured comprehensively the demand for embedded systems skill sets. This is because embedded systems are pervasive and integrated into so many products. The automotive, entertainment, telecommunications, and aerospace industries all need embedded systems developers. Unfortunately, it's hard to decipher the need for embedded systems skill sets from a job request that says something like "Wanted: Automotive Telematics Engineers" unless you understand what that entails.

At the recent Ubuntu Live conference, I had an opportunity to sit and discuss the state of embedded development curricula with many educators. I asked why so many electrical engineering, computer engineering and CS majors are matriculating without an understanding of embedded systems. Some of the answers were quite troubling, but those are a topic for a later discussion. However, the most prevalent answer was related to the cost of setting up an embedded system lab, the lack of embedded systems textbooks and the cost of producing an embedded systems curriculum.

The lab cost issue is a red herring based on woefully out-of-date information. Many instructors felt that labs needed expensive logic analyzers and target computers, ala the old Motorola Exorcisor, which their departments simply could not afford. Many were unaware that complete target processors like the XScale PXA-270 with 32 MBs RAM and 8 MBs of flash could be had for a price comparable to a set of textbooks. And, much embedded work can be accomplished with a simple, parallel-port based Joint Test Action Group (JTAG) interface device that can be purchased for less than $100.

The lack of a textbook to teach from is more difficult to address owing to the time needed to develop such a book. However, there are several examples of books about the embedded Linux space that could be used as a text for such a curriculum. These books can be used in the interim while dedicated embedded systems texts are developed.

**A Call to Action**

If the United States is to maintain a leadership role in technological innovation, we need to help our education system understand the skills that their students will require to be successful. This means the development of significant embedded systems programs at our colleges and universities. And, the most significant issue for these schools will be the time and funding it will take to develop an embedded systems curriculum. On this topic, organizations like the IEEE, the Association for Computing Machinery (ACM) and the U.S. Department of Labor need to work in concert with leading corporations to fund the development of curricula to help the United States regain our edge in engineering in general, and embedded systems in particular.

And this is not to say that there aren't good embedded systems programs in a few colleges and universities. Unfortunately, they are more the exception than the rule. We, as an industry, need to

help higher-learning institutions understand the importance of this "close to the metal" type of computing and encourage them to expand their successes and replicate them at other colleges and universities.

We must involve industry leaders such as Intel, FreeScale, Texas Instruments, Analog Devices and others, in coordination with engineering organizations such as the IEEE and ACM, to help the education community to understand the importance of teaching embedded systems theory and development techniques/concepts to their interested students. This coordination may also mean the development of an embedded systems academic program that can be simply handed to universities and colleges to help remove the barrier of developing the curricula from scratch.

Since there is no big push for engineers like we had during the Apollo space program years, industry, the U.S. Government and higher learning institutions also need to work harder to encourage younger students to enter engineering. There are some great success stories with many of the robotics competitions and similar projects for high school students. We need to build on these to capture the imaginations of younger students.

What might be needed here is a summit of all of the interested parties. Let's get some dialog going and elevate it to a national priority. Perhaps someone in the U.S. Senate, such as Sen. Chuck Grassley (R-Iowa) or Sen. Dick Durbin (D-Illinois), who are both significant opponents of H-1Bs, could be champions of such a meeting. For that matter, maybe we need a national "mission," a la the Apollo space missions, to get things back on track and regain momentum. Perhaps a project that focuses on eco-friendly technologies, which tend to rely on embedded systems, is the right way to go. Who knows, we might be able to reduce our dependence on foreign oil and create a generation of engineering innovators at the same time.

We must act soon. Many of the "greybeards" of embedded systems development are getting close to retirement age. We must try to capture their collective knowledge before it's lost and pass it on to the next generation of engineers. The U.S. embedded systems industry has a systemic problem that needs a holistic solution before we lose our technical edge. Of course, that's just my opinion. What do you think?

*Mike Anderson is chief scientist at The PTR Group, Inc. where he is responsible for providing embedded systems consulting and courseware creation. Mike has more than 30 years experience in the areas of computing and embedded systems development and was a former chairperson of the VxWorks User's Group. He holds a Bachelor's degree in Mathematics from the University of South Florida and a Master's degree in Computer Science from George Mason University.*

Expert advice on embedded systems and edge computers: These control machines and processes and help humans make smarter decisions.Â  Improving the process of patching code in vulnerable embedded systems is a major cybersecurity concern because much of the code currently running is vulnerable to hackers. By Kelsey Schnieders Lefever. HMI, OI August 5, 2020. Embedded HMIs excel in automation applications. Delivering the right human-machine interface (HMI) experience is critical whenever people need to interact with automation. By Bill Dehner. Embedded Systems, Edge Computing July 29, 2020. Is it time to look at edge computing? With the rise of 5G in manufacturing, effective edge computing requires consideration of industrial An embedded system is a computer systemâ€”a combination of a computer processor, computer memory, and input/output peripheral devicesâ€”that has a dedicated function within a larger mechanical or electrical system. It is embedded as part of a complete device often including electrical or electronic hardware and mechanical parts. Because an embedded system typically controls physical operations of the machine that it is embedded within, it often has real-time computing constraints. Embedded systems control Edge computing offers organizations a number of important benefits and has the potential to transform the way they do business and deliver services.Â  What is Edge Computing? Traditional cloud computing networks are highly centralized, with data being gathered on the outermost edges and transmitted back to the main servers for processing.