Embedded C#:

Will it Beat Out C and Python in the Embedded World?

Gabriel Stroe

Lawrence Technological University

Introduction

Embedded Systems Today

* Today, the embedded world is dominated by C/C++ for traditional style embedded development and Python with the rise in popularity of the Raspberry Pi (Mazzie, Montelisciani, Baldi, Fantoni, 2015).
* 95% of embedded system applications are written in C/C++
* C/C++ is used in vast majority of business applications with several compilers out there depending on the microprocessor. For example there is Keil or the GNU GCC toolchain for arm processors, or Microchips XC compilers for their Peripheral Interface Controllers (PICs).
* C when compiled has very little overhead compared to other modern day languages. Its runtime and memory usage is very efficient (Herity, 2015).
* C is predominantly used over C++ most likely due to two reasons. The first is that most embedded software engineers in the industry today were only taught C and assembly during their college career, and the second being that there are a fair amount of misconceptions in the embedded world about C++ (Herity, 2015).
* The general misconception is the C++ has large overhead costs that require more clock cycles for the same code and that it also requires more memory (Herity, 2015).
* This used to be true 15 years ago but today have been proven false as C++ is so close to C it can be seen as an expanded version of C with extra features, many of which are very useful for embedded applications like classes, overloading methods, and namespaces (Herity, 2015).
* Generally overhead created by C++ comes from a few features (such as virtual classes) creating reference tables that cost more memory and CPU cycles to go through.
* Generally the C++ lack of uses in the embedded world stems from a lack of knowledge by the majority of users out there. This can change if as new embedded developers, who are more familiar C++, join the embedded development workforce.
* Also with the cost of faster microprocessors with more memory has dropped significantly in recent years (Herity, 2015). This really reduces the impact of any overhead that C++ might add.
* Ever since the Raspberry Pi was released the use of python in embedded applications has gone significantly, especially for the hobbyist community.
* Python is actually become the most popular introductory language to be taught in Computer Science Programs today (Radcliffe, 2016).
* Because of its simplicity python is very easy to learn which allows more and more people to developing with it and the low cost of a Raspberry Pi or an Ardunio, many hobbyists have developed applications with python and are now starting to migrate into the industry (Radcliffe, 2016).
* This is really starting to erode at the C/C++’s dominance in the embedded world. Just two years after its release, at least 8 start ups companies created products with the Raspberry Pi as it’s microcontroller and most of them use Python as their primary language (Marinos, 2017).
* The main downside to using Python for embedded system applications is that due to it being an interpreted language running on a virtual machine, its runtime is significantly slower. Virtualizing introduces a lot of overhead even for processors that are designed for virtualization (Radcliffe, 2016).
* To combat this some developers are developing low level extensions like Cython that speeds up mathematical operations (Radcliffe, 2016).
* In addition Just-in-time Compilers (JIT) are being created to compile some of the python code to machine code. This runs in parallel with the python interpreter to help speed its decision making especially when traveling in a loop (Radcliffe, 2016).

C# in Embedded Systems Today

* Generally when programmers think of embedded system languages C# is one of the last ones they think of. C# today is generally considering a strictly windows language with focus on client side GUIs and web based API thanks to ASP.net.
* This is changing though, with the recent release of .net core, C# is going multiplatform and has considerable reduced some of its overhead, largely because with .net core standard library collection is more modular then with the .net framework (De la Torre, 2016).
* In addition Microsoft in 2007 released the .net Micro Framework which can be compiled on ARM based processor with at least 256 KB of general flash memory and 64 KB or RAM (Miller, 2015).
* It runs on a more efficient version of Microsoft’s Common Language Runtime (CLR) (Miller, 2015).
* Generally, development with the .net micro framework is rather small and is mostly done by hobbyist in already familiar to the .net platform looking for to get into the embedded world (Miller, 2015).
* Another method for using C# on an embedded system is to use it with Windows 10 IOT running on a Raspberry PI or Arduino. This is actually done using the .net core libraries.
* This seems to be the most commonly method for using C# in an embedded system today, although it’s still relatively niche when compared to python and especially C/C++ (Dobric, 2015).
* One major downside to .net’s CLR is that it isn’t actually a compiler, it’s an interpreter in its current iteration. Although a JIT compiler with the CLR is currently in development (Dobric, 2015).
* This means that C# has the same issues that python has in the embedded world. It’s just so much slower then C or even C++. Also because since everything in C# is an object, even when a compiler is developed, it is very likely that C# will have the same issues that C++ used to have before some more recent optimizations. C++ Compilers used to have very poorly optimized machine code that the C++ got compiled too which was generally linked with its class feature (Herity, 2015). C# will likely have this issue have this but with every instantiated object.
* Lastly the memory overhead of C# is fairly large. A benchmark from Computer Language Benchmarks shows that while C# with .net core is only 33% slower in terms of computation time, it is about 2000% larger when it comes to memory usage. That is a massive jump.

References

The Computer Language Benchmarks. (n.d.). Retrieved April 22, 2017, from https://benchmarksgame.alioth.debian.org/u64q/compare.php?lang=csharpcore&lang2=gpp

De la Torre, C. (2016, June 27). Cesar de la Torre [Microsoft] – BLOG Cesar de la Torre [Microsoft] – BLOG. Retrieved April 22, 2017, from https://blogs.msdn.microsoft.com/cesardelatorre/2016/06/27/net-core-1-0-net-framework-xamarin-the-whatand-when-to-use-it/

Dobric, D. (2015, September 22). How to run .NET Core Application on Pi2 - Win10 IoT Core? Retrieved April 22, 2017, from http://developers.de/blogs/damir\_dobric/archive/2015/09/22/how-to-run-net-core-application-on-pi2-win10-iot-core.aspx

Herity, D. (2015, February 17). Modern C in embedded systems – Part 1: Myth and Reality. Retrieved April 17, 2017, from http://www.embedded.com/design/programming-languages-and-tools/4438660/Modern-C--in-embedded-systems---Part-1--Myth-and-Reality

Marinos, A. (2017, January 27). Two years of Pi: Units sold and Raspberry startups! Retrieved April 22, 2017, from https://resin.io/blog/two-years-of-pi-units-sold-and-raspberry-startups/

Mazzei, D., Montelisciani, G., Baldi, G., & Fantoni, G. (2015). Changing the programming paradigm for the embedded in the IoT domain [Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on]. New York, NY: IEEE. doi:10.1109/WF-IoT.2015.7389059

Miller, C. (2015, March). The Microsoft .NET Framework in Embedded Applications. Retrieved April 22, 2017, from https://msdn.microsoft.com/en-us/magazine/dn913183.aspx

Radcliffe, T. (2016, August 29). Python vs. C/C in embedded systems. Retrieved April 19, 2017, from https://opensource.com/life/16/8/python-vs-cc-embedded-systems