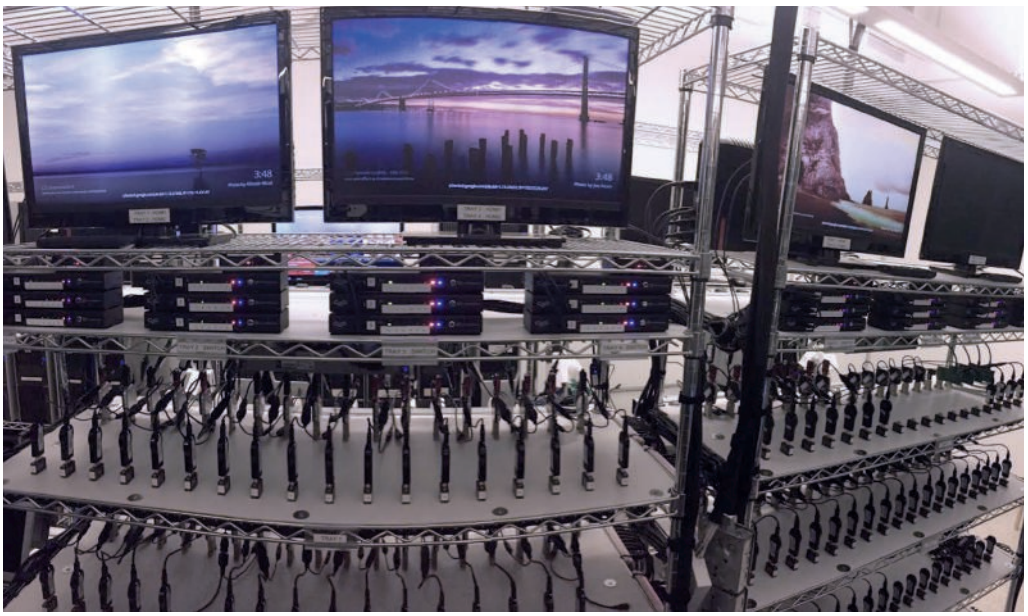
When development never sleeps

Development techniques from the cloud promise faster updates for embedded firmware, as **Chris Edwards** explains



Is software ever finished? As Leonardo da Vinci observed with art 500 years ago, it is not so much finished as worked on until it is time for it to be abandoned.

In the case of traditional embedded systems that state of abandonment arrived at the point the product was shipped, with only some systems supporting some kind of upgrade or bug-fix through a wholesale ROM replacement or, more recently, a flash-memory update.

But times have changed. Partly due to the rise of connected products and a much greater ability to deliver updates and partly as recognition that it almost impossible to deliver bug-free products embedded systems software is much more fluid than ever.

At a variety of conferences, embedded systems developers have talked about how organisations as diverse as Bosch, Dyson, Ericsson and Google have embraced the concepts of continuous integration for embedded devices, sometimes in a shift to agile development practices or to reduce the time taken in traditional waterfall processes.

Adam Arakelian, director of engineering at Dell's EMC storage subsidiary told delegates at Vector Software's 2018 test symposium in Boston, a move to continuous integration was to reduce delays by performing tests as early as possible in a project and to remove the traditional disconnect between development and QA teams. "We were trying to build a shift-left culture. It was about pulling testing back to the left side: when the code is written."

Continuous integration splits into two main forms. One is continuous delivery. This is where developers commit changes to an evolving product in a process where all the integration tests and actions are performed automatically. But someone makes a conscious decision when to distribute each new binary.

Continuous deployment takes that final step of automating the distribution and installation of the updated software as long as all the tests pass. In web-based deployments, different users might find they are on subtly different versions based on when they last logged into a service.

Eight years ago, John Jenkins former project director at Amazon, in a speech at O'Reilly's Velocity conference said he dug into the statistics for at his employer for the month of May 2011 and found "every 11.6 seconds someone is kicking off a deployment" to a production application.

Is such a deployment frequency realistic for embedded? For any device that has to be reflashed with new software each time, it's not a likely prospect. But applications that rely on an IoT infrastructure, it is entirely plausible that parts of the system will be updated rapidly and they will continue to have to work well with software that may have been in place for years – and work around its flaws because the embedded firmware is not easily changed.

Organisations as diverse as Bosch, Dyson and Google have embraced the concepts of continuous integration for embedded devices.

Cloud environment

Regression tests are essential in these environments before pushing out the updates to ensure nothing that already works is broken by the change. Doing those tests manually is an unrealistic prospect. But some help has arrived from the cloud environment, through tools that go under the banner of "development operations", often compressed to just "devops". Many of them come in open-source form, ranging from Docker's software containers to the

Jenkins build-and-test management environment and do not necessarily need to run in the cloud. They can run on developers' own machines for local unit testing.

Docker provides a lightweight form of virtualisation, largely because the software inside the container shares many resources with the underlying operating system. Docker's main advantage, and other tools like it, is that it provides the ability to present a consistent environment to the software build and the tools used to test it. This reduces the chances of software not functioning as expected because of subtle differences from machine to machine. In the devops world, Docker has become popular as a way of instantiating many copies of a build in a matter of seconds across a server farm for parallelising tests in large-scale regression frameworks.

Although the tools were conceived for software development, they have picked up supporters in chip design, where continuous integration is used for "shift left" reasons. Developers are now expected to build unit tests to check their modules behave as expected. Those unit tests form part of a suite of regression tests that build up over the lifetime of the project.

In addition to software-test specialists such as Vector Software, EDA tools suppliers such as Siemens subsidiary Mentor have built plugins for Jenkins that trigger a round of verification runs each night based on the code that has been committed by the source-code control system. The output is a new crop of results that go into a database that shows whether a change has triggered a new round of failures or if parts of the code remain stubbornly untested and need attention.

Chip development takes place largely in a virtual realm of simulation backed up sometimes by emulation, where the design is compiled and run on specialised hardware, as well as by static, formal-verification

tools. One major difference between the embedded software and chip-design projects is that the former rely much more heavily on physical target hardware being available once development of a module moves beyond unit testing and into system-level testing. This is beginning to change as companies such as Jumper work on improving the accuracy of tests on simulated hardware. Jumper has techniques that take into account interrupts and other time-sensitive issues that often lead to a mismatch between testing on simulated and physical hardware. The company claims to have found such a timing bug in one of Nordic Semiconductor's software development kits using its device emulator.

Device farms

For the situations where only the target will do, users of continuous-integration methods turn to hardware-in-the-loop environments that range from one or two prototypes on a bench to large-scale device farms. As the "farm" name suggests, these are collections of target boards that are hooked up to a server somewhere along with some form of emulated I/O.

Google's Chromecast developers built custom test rigs that push firmware builds onto the smart USB sticks, even to the extent of inserting some into isolation boxes so they could try out different WiFi scenarios without disrupting other nearby tests.

With a device farm, the build environment updates the code that sits on the board for each new regression run and pulls off test data once the run completes. Alternatively, individual boards can be run interactively to help developers perform quick unit tests on their own code. For off-the-shelf devices such as smartphones, Amazon Web Services is among those that have deployed readymade device farms for rent. TimeSys has built an environment to try to ease the creation of custom device farms for custom hardware. However, few teams have the resources to create their own farms, with the result that, for the moment, testing for continuous integration with real hardware is performed on a much smaller scale.

The push towards more agile processes supported by continuous integration is likely to see greater support among tools companies.

Embedded software developers have other issues to contend with, such as licensing. Niall Cooling, head of training firm Feabhas which has run courses and conference on agile methods and continuous integration, points out that the licensing dongles used by a number of popular tools do not play well with Docker containers.

A bigger problem though, Cooling adds, is the design of many of the tools assumes an interactive, graphical interface whereas automated test relies on command-line interfaces and scripting. Many valuable features are not yet available to teams that want to pull debug tools into a continuous-integration environment. The result is that continuous integration using tools such as Docker can be restricted to ensuring that builds complete successfully with regression testing remaining a manually intensive process.

As the demand for faster updates and fixes for connected products continues to grow, the push towards more agile processes supported by continuous integration is likely to see greater support among tools companies over time though few will experience the 11-second updates boasted by Amazon as a result.