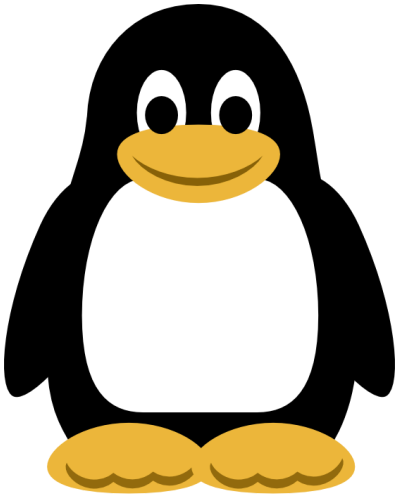
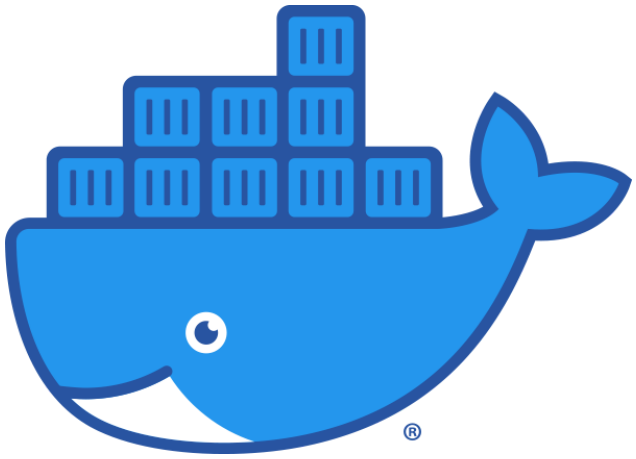
BUILDING EMBEDDED SOLUTIONS WITH MODERN SOFTWARE TOOLS





BUILDING EMBEDDED SOFTWARE ONE CONTAINER AT A TIME

Written by Brad Graves
Contributions by Dr. Hal Holmes

When project managers hear a project involves writing embedded software there is an expectation that a software developer with very specialized skills will be required to work on the project. Embedded software development historically requires a developer with both an electrical engineering and computer science background, writing software and firmware in a low-level language like C or C++, acquiring expensive software development tools, and typically the developed user interface will be rudimentary by modern GUI standards.

While many embedded software projects still share these characteristics, there have emerged new technologies that make it possible to perform embedded software development using modern software development tools and improve the productivity of embedded development exponentially. This case study describes RND Group's experiences with writing software for embedded devices using open-source tools, high-level programming languages, modern UI toolkits, and containerization technology that facilitates developing and testing a high percentage of the software before hardware is available.

WHAT IS AN EMBEDDED SYSTEM?

An embedded system shares many of the same architectural characteristics of a general-purpose system – there is an operating system, device interfaces / device drivers, business logic, and a user interface. The main difference between an embedded system is that it is designed and built for a specific purpose and many of the features found in a general-purpose system, like networking, security, configuration, and various applications are removed or scaled back. The paring down of the features in the embedded system allows the software to run on more constrained processors with less memory and reduced storage and power consumption requirements. Embedded systems are very common in smaller sized devices, such as handheld or portable units and especially devices that must be capable of operating on battery power.

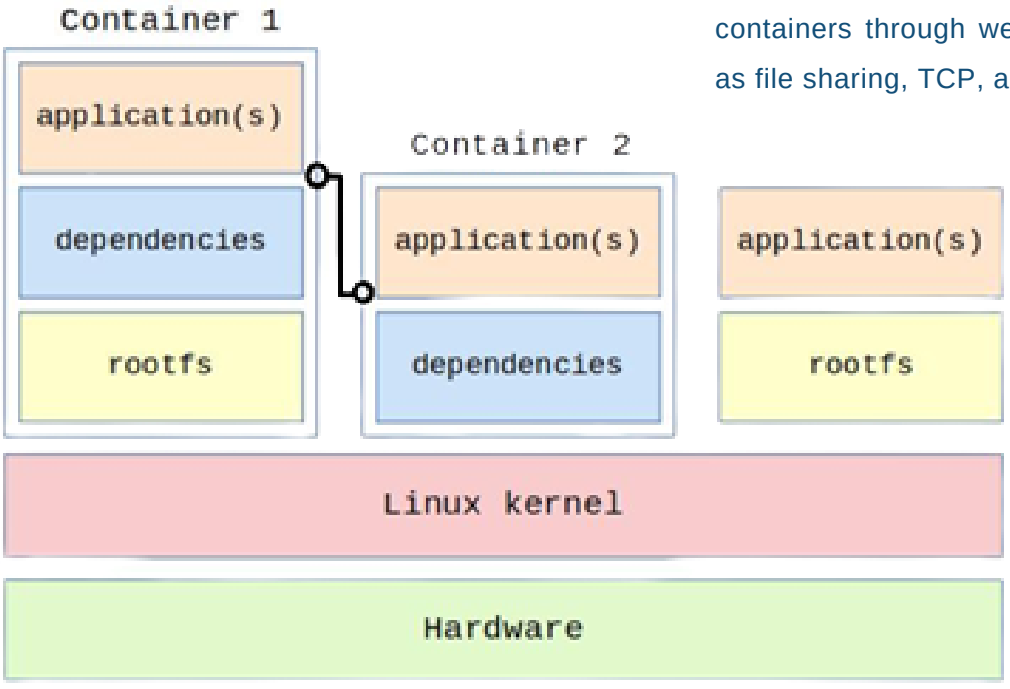
There are many operating systems that can be used for embedded systems development, but the leading choice today is Linux. Linux is an open-source operating system and there are various versions of Linux that have been customized to perform well as embedded operating systems, including Android, Maemo, BusyBox, Yocto / Open Embedded, Buildroot, and Mobilinux.

DOCKER + EMBEDDED LINUX

Docker is a software containerization technology that uses virtualization features built into operating systems such and Linux and Windows to support the creation of software packages. These software packages bundle a full stack of software – application, system services, file system, and device services – that runs in its own separate virtual environment. Docker containers offer numerous benefits, including being much lighter weight than traditional virtualization technologies, the ability to build and extend a Docker container from other pre-existing Docker containers, and a guarantee that the software package in the Docker container performs the same way regardless of what hardware it is deployed on.

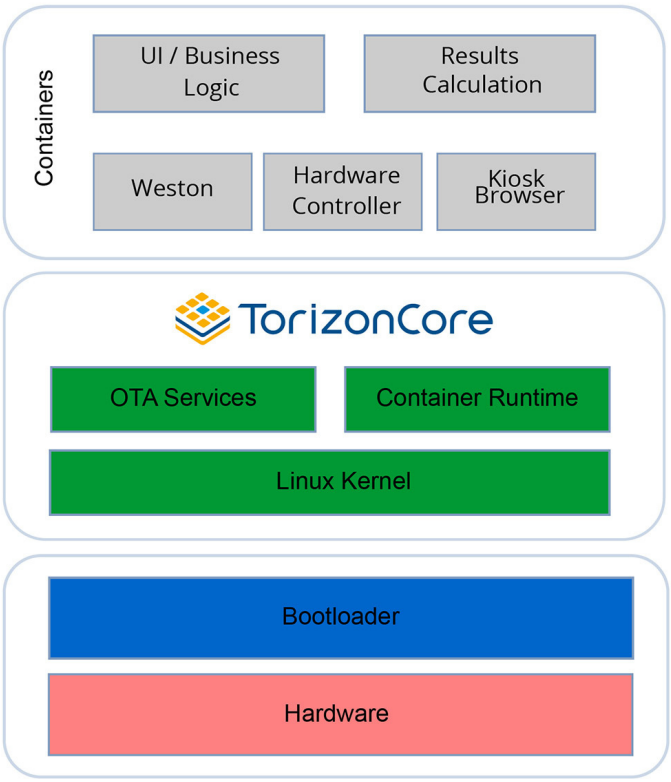
Adoption and use of Docker has grown exponentially in the 8 years since its first release - including in the medical device software space. RND has been using Docker for several of its customer projects over the past 2-3 years.

In a natural evolution and convergence of technologies, there are now versions of embedded Linux that include Docker infrastructure support, including Linux microPlatform from Foundries.io, **TorizonCore from Toradex**, and balenaOS from balena. These products provide a Linux distribution and free embedded software developers from the need to build customized Linux images and, instead, focus on developing applications that run in containers. As illustrated below, multiple containers can be created, if desired, with containers communicating with other containers through well-known channels, such as file sharing, TCP, and HTTP.



EXAMPLE PROJECT ARCHITECTURE

RND has recently delivered two different medical device software systems using Docker containers running on TorizonCore embedded Linux distribution. These systems include a handheld PCR testing device and a desktop blood testing device. Both of these devices are portable and run 8+ hours on battery power.



The software architectures used for these two projects are similar and are depicted above.

The containers in this architecture include:

- **UI / Business Logic**
  - Written in C# and .Net Core, this is a web server that includes features like run workflow management, results display, QC run management, user / account management, audit trail, and reporting. The UI is written in React, Angular, or Razor and is rendered by the Kiosk Browser container.
- **Results Processor**
  - Written in C# or Python, this container isolates assay-specific results calculation logic and is called by the UI / Business Logic container.
- **Weston**
  - Provided by Toradex, this container is a window manager that orchestrates startup of display windows, such as the one produced by the Kiosk Browser container.
- **Kiosk Brower**
  - Provided by Toradex, this container is an HTML web browser that connects to the webserver provided by the UI / Business Logic container.
- **Hardware Controller**
  - Written in C# and .Net Core, this container provides hardware services and interacts with the Linux kernel and custom device drives for hardware control. The UI / Business Logic container has a command/response interface to communicate with this container.

Toradex
Swiss. Embedded. Computing.

There are a number of advantages of using a software architecture like the one described above, including:

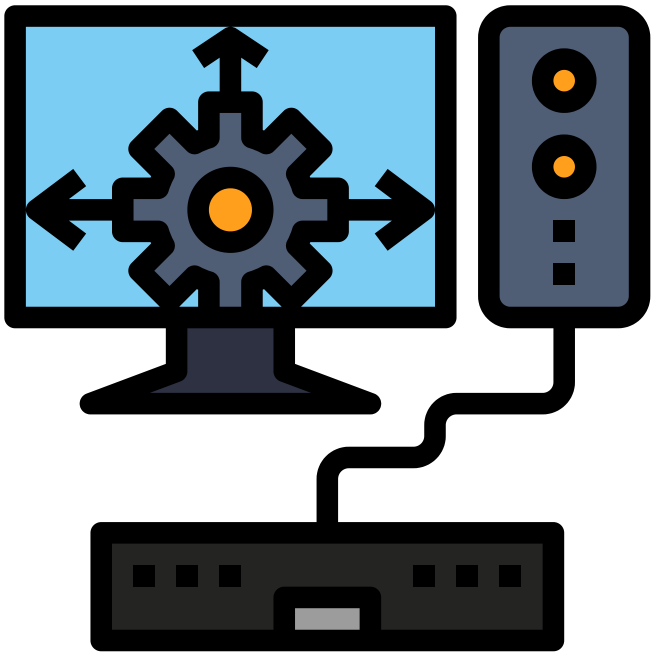
1. Use of low cost, open-source languages, frameworks, and operating systems allows development and deployment with no run-time royalty costs.

2. Use of a high-level, commonly known language like C# allows developers to concentrate on writing application logic and allows organizations to have access to a larger pool of development talent who know C#.

3. Modular breakdown of the software into separate containers allows developing different parts of the application independently and allows for each component/container to focus on a well-defined set of responsibilities.

4. Simulation support is facilitated inherently by the well-defined interfaces between containers. For example, the Hardware Controller container can be replaced to simulate the hardware behavior and development of the other containers can proceed in parallel to hardware development.

5. Cross-platform support is facilitated in two ways. The use of .Net Core allows code to be written and tested on Linux or Windows outside of a Docker container. And, when deployed inside a Docker container, which are operating system-specific, the Docker containers can be hosted and tested on a variety of operating systems, including Linux and Windows.



CONSERVATION X LABS

CUSTOMER TESTIMONIAL

"The software architecture RND created for our handheld medical device provided us great flexibility in development. We were able to quickly deploy a demanding user interface, and the architecture allowed us to rapidly iterate and modify our UI, key functionality, and implement new features throughout our development and user testing. This software format also allowed us to first test updates on the desktop and with carrier boards while we were still working out our hardware and complete instruments were scarce or unavailable. Complimenting this architecture with RND's adept and agile project management enabled us to move our embedded software from requirements to reality more efficiently than I thought possible."

DR. HAL HOLMES
— CHIEF ENGINEER
CONSERVATION X LABS
THYLACINE BIOSCIENCES

FINAL THOUGHTS

RND has been pleased with the stability and maturity of using Docker and the TorizonCore Linux distribution. Toradex provides a larger number of pre-built containers and pre-built Linux kernel images. These tools and building blocks have significantly reduced the amount of time that is typically required to build customized Linux distributions using Yocto or similar tools for embedded projects. RND has a number of developers experienced in C# and in developing web user interfaces, so the ability to use these tools when creating software for an embedded Linux system allows RND to readily staff these projects.

THE RND GROUP
A GENER8 COMPANY

Visit Us Online:
www.RNDgroup.com
www.gener8.net

If your company is contemplating developing medical device software and is interested in learning more about the advantages of using embedded Linux for your platform, please ***contact The RND Group today.***