Samuel Yeomans
Advisor: Linda Milor
CDC Computerized Cognitive Testing Platform Design Team

Operating Systems in Microcontrollers and other Embedded Devices

Abstract

Embedded operating systems are small operating systems that are installed on embedded devices like microcontrollers. Typically, these operating systems don't consume much processing power or electrical power, and provide the embedded device more functionality and flexibility. There are two main categories of operating systems: real-time and general purpose. Both categories balance the tradeoff between performance metrics (i.e., response time and cycle variation) and additional functionality (e.x., I/O driver support, graphical user interface, file structures, etc.) differently. Real-time systems focus on improved performance metrics at the cost of lower functionality. General purpose systems focus on improved functionality at the cost of performance metrics.

Commercial applications of embedded operating systems and current state-of-the-art

Embedded operating systems can be found in many high-end electronics and consumer appliances. Some high-profile examples include Tesla Motors' Tesla Model 3 [1], Apple's Apple Watch [2], and Samsung's RF28N9780SG smart fridge [3]. Each of these products includes a graphical user interface that allows the user to interact with the device and its file system. It isn't well-known how much time and money Tesla and Apple have invested in developing these custom-built operating systems, but Samsung's smart fridge makes use of Windows IoT [4], which commercially costs $0.30 per month per device to license [5].

Raspberry Pi's Raspbian [6] is perhaps the most popular operating system for microcontrollers among hobbyists and professionals alike. From their website, "Raspbian is a free operating system based on Debian, optimised for the Raspberry Pi hardware. Raspbian comes with over 35,000 packages: precompiled software bundled in a nice format for easy installation on your Raspberry Pi [6]." Raspbian features software such as an IDE, peripheral drivers, a web browser, a media player, and a VNC server among many others. Raspbian is a general purpose OS with a large user-base, which makes it a strong choice for non-real-time applications. Programming on these systems is typically done in Python, but C++, Java, Scratch, and Ruby are also supported out-of-the-box.

ARM’s Mbed chips can make use of the Mbed OS 5, ano open-source OS that supports real-time software execution [7]. While this OS is capable of threading software in real-time applications, it has no GUI or built-in IDE. However, there is an online compiler along with a large community and user base. The online compiler is designed for rapid prototyping of Internet of Things (IoT) devices in C++. To use this OS (as there is no built-in GUI), a developer need only to import the latest officially-supported release into the online compiler and include the library like any other C++ library. Every feature of the OS can be called as a C++ function.

While operating systems can add more functionality to a project, they aren’t always necessary. In fact, neither the Mbed chips discussed above nor the popular Arduino [8] series boards require an operating system to run. Bare-metal controllers (embedded devices that run without operating systems) are perhaps the best example of a real-time controller, but they lack the robustness of systems that include an operating system.

How the underlying technology works

Embedded operating systems are closely tied to the hardware of the systems they run on. In fact, most software development on embedded devices has to be done in parallel with hardware development. The operating system or software code on an embedded device is installed on the internal flash memory of the device itself in many cases [7] [8], but can be installed on an external device such as a micro SD card in other cases [6]. The operating systems of most microcontrollers can access hardware components like timers and GPIO pins using the hardware addresses of the components. Developers typically use some form of abstraction to access these components, either through the operating system itself or a supporting API library.

As is often the case with engineering projects, there are tradeoffs and design decisions to be made when choosing an embedded operating system. Some applications require a system to respond within a limited amount of time (often less than 10 milliseconds), and have a limited amount of cycle variation, or “jitter” (often less than 100 microseconds). Systems that meet these performance metrics are called real-time systems. Real-time systems can be split into three distinct categories: hard real-time (which must meet the performance requirements in every case), soft real-time (which only need to meet the performance requirements most of the time), and non-real-time (or general purpose) [9]. Real-time operating systems exist, and can add some of the capabilities provided by operating systems to a real-time application. These capabilities include threading, file systems and resource management.

Building blocks for implementing this technology

Embedded operating systems run on embedded hardware, often called “microcontrollers”. Microcontrollers can be thought of as miniature computers that are designed to talk to other hardware components of a project. Inside every microcontroller is a processor that has the capability of interpreting programmed instructions in a manner similar to general-purpose desktop processors.

Many projects that involve microcontrollers also require additional hardware. For example, a project may require an Arduino to communicate with another device over Bluetooth. This project will require an external Bluetooth module to be wired into the Arduino, as these microcontrollers do not come with Bluetooth capabilities (though other microcontrollers do, such as the Raspberry Pi). A more involved project may require an Arduino to control a 5 volt DC motor and cause it to rotate in either direction. For this project, a designer may decide to include an H-bridge driver circuit to allow the motor to be driven in reverse.

Conclusion

This project recommends the use of a Raspberry Pi running a standard Raspbian operating system. This configuration will maximize the project’s functionality, as Raspbian includes support for many features the design team can make use of. Raspbian natively supports Ethernet, Bluetooth and Wi-Fi communication standards; includes a VNC to display the desktop user interface remotely; and makes use of the Debian kernel. While Raspbian may not be a real-time operating system, the design constraints for the project don’t require a real-time solution. Pairing this operating system with the Raspberry Pi 3 Model B+ has satisfied the design requirements of many similar projects [10].

References

Manual

[1] "Tesla Model 3 Owner's Manual", *tesla.com*, 2019. [Online]. Available at: https://www.tesla.com/content/dam/tesla/Ownership/Own/Model%203%20Owners%20Manual.pdf. [Accessed: 17- Feb- 2019].

Web Page

[2] "Apple Watch Pricing", apple.com, 2019. [Online]. Available at: https://www.apple.com/shop/buy-watch/apple-watch. [Accessed: 17- Feb- 2019].

Technical Specifications

[3] "RF28N9780SG", Samsung, 2018. [Online]. Available at: https://image-us.samsung.com/SamsungUS/home/home-appliances/refrigerators/4-door-flex/pd/rf28n9780sr-aa/RF28N9780SG_v6_060418.pdf. [Accessed: 03- Mar- 2019].

Web Page

[4] "Overview of Windows 10 IoT - Windows IoT", Docs.microsoft.com, 2019. [Online]. Available at: https://docs.microsoft.com/en-us/windows/iot-core/windows-iot. [Accessed: 17- Feb- 2019].

Online Newspaper Article

[5] Ryan Daws, "Microsoft unleashes Windows 10 IoT Core Services," IoT News, July 19, 2018. [Online], Available: https://www.iottechnews.com/news/2018/jul/19/microsoft-windows-10-iot-core-services/. [Accessed: Mar. 5, 2019].

Web Page

[6] "Raspberry Pi Documentation", Raspberrypi.org, 2019. [Online]. Available at: https://www.raspberrypi.org/documentation/ [Accessed 17- Feb- 2019].

Web Page

[7] "Mbed OS 5 Documentation", os.mbed.com, 2019. [Online]. Available at: https://os.mbed.com/docs/mbed-os/v5.11/introduction/index.html. [Accessed: 17- Feb- 2019].

Web Page

[8] "Arduino - Introduction", arduino.cc, 2019. [Online]. Available at: https://www.arduino.cc/en/Guide/Introduction. [Accessed: 17- Feb- 2019].

Lecture

[9] J. Hamblen. ECE 4180. Class Lecture, Topic: "Real-Time Systems." Georgia Institute of Technology, Atlanta, GA, Jan. 11, 2019.

Web Page

[10] "Raspberry Pi Projects", projects.raspberrypi.org, 2019. [Online]. Available at: https://projects.raspberrypi.org/en/. [Accessed: 17- Feb- 2019].