IoT Opens Door to Driverless Control of Data-Acquisition Hardware

The emergence of the Internet of Things (IoT) represents a significant step in the evolution of hardware and software and their interaction. In a typical IoT application, a computing device with a browser connects to an edge device such as a smart thermostat or wearable health monitor. The computing device may be running any of several operating systems, so operating-system-dependent drivers are not appropriate for IoT applications, which rely on driverless-control strategies. Driverless control is the latest stage in the evolution of programming techniques extending back to 1949 when assembly language provided a text-based level of abstraction above the 1s and 0s of machine code.

Although the term IoT brings to mind low-channel-count, low-bandwidth smart-home and related consumer-oriented applications, the underlying technology also finds use in industrial applications. In the industrial space, the edge device can become a high-channel-count, high-speed data-acquisition system that collects high-quality data. In such an application, driverless control can save customers significant time and money.

Software Evolution

Historically, suppliers of data-acquisition equipment typically provided turnkey bespoke systems with custom proprietary software. Such systems could be challenging to scale up as customers' needs changed. If one vendor could not meet all of a customer's needs, that customer might have been faced with purchasing multiple turnkey systems from different vendors that did not play well together.

The emergence of personal-computer operating systems enables more flexibility and scalability. Customers can assemble a system with hardware from one or more vendors and alter it as needed. Each vendor provides drivers compatible with the Interchangeable Virtual Instrument (IVI) standard. The drivers would let the operating system access the hardware's functionality. These drivers are operating-system dependent, and customers would need to install and update them. The drivers serve as an interface between user-defined commands, often in the form of SCPI (Standard Commands for Programmable Instruments), and the edge device, such as a data-acquisition instrument. SCPI was merged into the IVI standard in 2003, and the combination of SCPI and IVI drivers has become the typical approach to communicating with and controlling test equipment.

With the IoT eliminating the need for drivers, any device with a browser—including a Windows or Linux PC, an Apple Macintosh, a smartphone, or even a single-board computer—can take direct control of an edge device. With this approach, any edge device that can acquire data from sensors can publish that data to the Internet or to a private network over an Ethernet connection, effectively eliminating middleware and anything else that stands between your application software and the hardware.

HTTP, the Internet Backbone

The backbone of the Internet is HTTP, which was devised to display data in a web browser. A subset within HTTP called Representational State Transfer (REST) is the de facto standard for accessing resources over the web, enabling the sending and receiving of commands such as GET, PUT, and POST—commands that are self-explanatory even for nonprogrammers. An application programming interface (API) that conforms to the constraints of REST HTTP is called a RESTful API. With your REST HTTP-based data-acquisition instrument connected to the Internet via Ethernet, you can take full advantage of web services. For example, send your acquired data directly to a file server or create a web page that displays the data.

REST HTTP offers yet another advantage. A driver typically comes with a manual that lists and describes available commands. In contrast, REST HTTP offers discoverability, a feature that lets you access all the relevant information without a manual—sending a GET command will cause the instrument to report back the available commands plus information such as instrument serial number through your browser. The specific commands you will see will depend on the instrument you are querying. You can also configure what data you are seeing—requesting just the data from channel 3, for example—all by calling different HTTP addresses.

The driverless approach provides full instrument functionality, offering more flexibility and lower-level access to instrumentation than traditional drivers. You may notice a small amount of latency, which is inherent in how the Internet works. Drivers, too, can cause slow response times because they typically impose some overhead to support many use cases, including ones you do not need. In addition, for real-time applications, REST HTTP supports deterministic response times, which are not guaranteed with a Windows driver. You can develop your program on a shadow PC for real-time applications and then directly port your code to your real-time target.

Programming-Language Evolution

To augment your application, you can use one of many programming languages that have been developed since the introduction of assembly language, including C, C++, Microsoft Visual Basic, Java, JavaScript, or C#. A key consideration when choosing a language is to make sure its popularity enables you to easily employ programmers fluent in that language. As an alternative to a commercial programming environments, you could consider a free, open-source programming environment like Python, created as a high-level, general-purpose programming language in 1990. Python relies on modules that perform specific functions and on a massive community that can create and maintain these modules and provides support. Python, widely adopted in the educational community, offers a clean, simple, uncluttered syntax. As of July 2022 the TIOBE organization rated it the most popular programming language, followed by C, Java, and C++, with assembly language in eighth place.

If you do not want to write code, you can use Node-RED. This community-driven, open-source programming environment allows integration with any web-connected hardware, API, or web page. Similar to NI LabVIEW, it employs a graphical, block-diagram approach to software development, eliminating the need to write lines of code. Node-RED requires only a browser to serve as a programming environment and visual data display. To create special functions not directly supported by Node-RED, you can add functions written in JavaScript, C#, C++, Python, or other programming languages.

Communication Interfaces

Several communications standards have found use in test-and-measurement applications, including RS-232, an early standard that still finds use in legacy applications. GPIB, initially the Hewlett-Packard Instrument Bus, adds features specific to test-and-measurement applications and has been codified as the IEEE 488 standard. More modern communications standards include several generations of USB, with USB 4.x offering data rates to 40 Gb/s. USB can be useful for connecting a bench instrument to a laptop. RS-232, GPIB, and USB operate over relatively short distances: 15 m, 20 m, and 5 m, respectively.

Ethernet is at the pinnacle of communications interfaces, a key Internet technology that commonly carries the Internet Protocol (IP). Used extensively in local-area networks (LANs), it offers cable lengths to 100 m, but with repeaters and routers, the distance it can span is essentially unlimited. It finds wide use in test-and-measurement applications, particularly in conjunction with instruments that comply with the LAN eXtensions for Instrumentation (LXI) standard.

REST HTTP for Thermocouple and Strain-Gauge Instruments

AMETEK Programmable Power's VTI Instruments brand (VTI) employs REST to provide driverless control for its products, including EX1401 isolated thermocouple and voltage measurement instrument and EX1403A bridge- resistance and strain-gauge measurement instrument. Both instruments feature 16 channels and 24-bit resolution. The EX1401 provides typical accuracies of ±0.20°C and sample rates to 20-kS/s, while the EX1403A reaches sample rates of 128 kS/s. At first glance, these instruments would seem far removed from an IoT device like a smart thermostat that measures one temperature and controls one heating or cooling unit. Nevertheless, they have Ethernet interfaces and support driverless control, meeting the definition of an IoT device.

Although REST HTTP is sufficiently flexible to accommodate various data types, including HTML and XML, VTI has chosen to use JSON (JavaScript Object Notation) because it is compact and fast to parse. VTI's RESTful API supports discoverability—a GET request to an API directory will return a list of subdirectories and endpoints, minimizing the need for documentation.

Figure 1. A typical Python IDE shows the discovery of available commands.

Figure 1 shows the VTI RESTful HTTP interface for an EX1403A instrument imported into a Python environment. Typing "rest." displays functions specific to the instrument being called. To get the same intuitive programming experience with an IVI driver for Windows, you could install a module called comtypes, a lightweight Python COM package that allows COM type calls to the driver. This functionality is not available at all under Linux.

VTI has published two libraries to assist you in implementing driverless control of its instruments from a Python environment. The first, JSON REST, provides support for all JSON/REST communications and includes a wrapper layer to make VTI products particularly easy to use. The second, vtivrt, implements the VMEbus International Trade Association's VITA 49 high-speed data-streaming interface used by VTI's REST-based digitizer instruments. The data-streaming interface is not REST-based but is used alongside the REST interface as the only method available for reading large quantities of high-speed data. VTI's REST APIs come with examples to help speed test-system development (**Figure 2**).

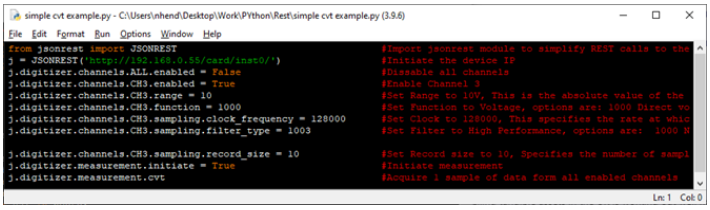

Figure 2. A RESTful HTTP example of acquiring data from the EX1403A.

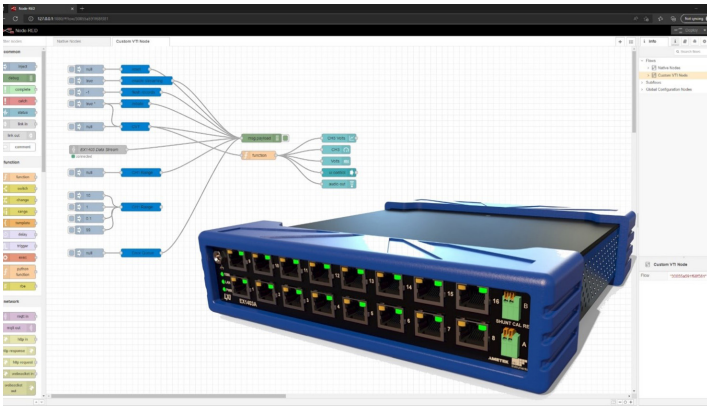

Figure 3. A Node-RED programming block diagram with an EX1403A instrument.

VTI also supports using its RESTful HTTP APIs in conjunction with Node-RED. The company offers a set of nodes you can use in the Node-RED environment, allowing you to control VTI hardware without writing code. You can also use Node-RED in a multivendor environment and publish the front-panel display on a LAN where multiple test engineers can access it. You can quickly and easily modify VTI's nodes, as shown in **Figure 3** for the EX1403A. For path, enter the HTTP address that points to the page that controls or displays the status of a particular EX1403A function.

Conclusion

Platform-independent driverless control using a RESTful API can save you time and money when setting up a data-acquisition system. You need only a device with a browser. You can augment your system with free, community-driven, open-source programming languages like Python, or you can use a tool like Node-RED that eliminates the need to write code. VTI Instruments fully supports REST for its EX1401 isolated thermocouple and voltage measurement instrument and EX1403A bridge and strain gauge measurement instrument while simultaneously providing traditional drivers for multiple operating systems. Customers have successfully employed VTI's RESTful API, including in real-time applications. The company stands ready to assist you in migrating from traditional drivers to REST HTTP.