The Internet of Things Ecosystem

To understand which languages you can apply to your IoT projects, you must first understand the IoT ecosystem. This knowledge is important because the processor architectures and resources available to your software or firmware will differ greatly at each level.

Let’s start by defining a simple taxonomy of...
Figure 1. Taxonomy for Internet of Things devices

At the bottom are the edge devices. These devices interact with the world and represent things like wearables and other connected devices. The devices source and create data and interact with the world through actuators.

Next are the gateways. These devices can be intermediary devices that move data to other systems for processing. The gateways can also aggregate data from many edge-devices and provide a control path to edge devices.
Finally, there’s the cloud. The cloud is a scalable set of compute, network, and storage resources that provide the ability to store, analyze, and visualize data from edge devices and gateways.

Examples of these levels include the Intel® Galileo board and the Intel® Curie™ Compute Module for edge devices, Intel® IoT Gateways for the gateway level, and the Wind River* Helix* Lab Cloud for the cloud level.

With the IoT ecosystem divided into its layers, let’s look at which languages you can apply at each level.

**Edge Device**

Edge devices, such as wearables, are typically constrained-resource, embedded systems because of the space and power constraints in which they function. Devices like the Intel® Curie™ module are the size of a button, and can be powered by a small, coin-sized battery (see Figure 2). Given the minimal resources of the Intel® Curie™ module, the typical languages suitable for its use include Assembly and C. Although C is the lingua franca of embedded firmware development, there are times when you need to squeeze as many instructions as possible into a device. In such cases, Assembly is your best choice. The downside is that development
Another example of an edge device is the Intel® Edison board (see Figure 3), which you can use in the wearables space or in general IoT products. Differing from the Intel® Curie™ module, which incorporates a microcontroller, the Intel® Edison board includes a dual-core Intel® Atom™ processor with considerably more computing power (the size of an SD card). Because the Intel® Edison board runs Linux*, the C language is an ideal choice here, but you could use other languages, as well, including Python* and Node.js*. Python* is ideal for
quick prototyping and production deployment, but can lag in performance compared to native compiled C. If you use Intel® XDK, you can also run Node.js* (JavaScript*) with Node-RED*. Node-RED* makes it easy to build and run data flows, offering a graphical approach to development. Knowledge of the JavaScript* language makes this environment even more powerful.

![Figure 3. The Intel® Edison board](image)

**Gateway**

At the gateway level, the compute capability rises greatly because gateways are responsible for communication and analysis of data from many devices through several
Intel® IoT Gateways provide a variety of designs that scale from a single-core Intel® Quark™ system on a chip to a quad-core Intel® Atom™ or Intel® Core™ processor (see Figure 4). These platforms support either Wind River Linux® 7 or the Snappy Ubuntu® Core (Linux*).

**Figure 4.** An Intel® IoT gateway

In addition to support for C and C++ (which is also ideal for higher-performance devices), you can use Python*, but at greater execution speeds. Node.js* with JavaScript*
Cloud

When you reach the cloud, the computing capability increases drastically, as do the language choices. In the cloud, you’ll find servers enabled through power-efficient Intel Atom and Intel Core processors as well as highest-compute-density Intel® Xeon® processors. Apps written in the cloud serve a variety of needs, and as such, the languages you use here can differ greatly. To process the massive amounts of data that IoT edge devices create, you can use big-data frameworks like Apache Hadoop*. On top of Hadoop, query languages such as Apache Hive* enable computation over massive data sets with Structured Query Language (SQL)-like queries. Apache Pig* is also useful for experimentation with large data sets in the Pig Latin scripting language.

Data analytics and visualization are other key applications within the cloud that many programming languages enable. A popular language and environment for statistical computing and visualization is the R language, which has recently grown in popularity. The Julia language is another option here. Julia is a high-performance, dynamic language designed with cloud computing in mind.
You use several languages to build web-services to monetize data in the cloud, including JavaScript*, Node.js*, or server-side and client-side Java*. Given many frameworks and supporting languages (such as Rails and Ruby), the language opportunities in this space is large and growing.

Summary

Choosing a programming language for a project requires consideration of the target environment (including the processor) as well as the resources available to it. Developing software in the cloud opens many possibilities, given the scale of resources available, but developing embedded firmware for smaller microcontrollers requires greater control to minimize instruction count and maximize execution speed and resource management.

The following list briefly summarizes the languages discussed in this article, and identifies their key use models:

- **Assembly**: Developing firmware in the native instruction set provides the greatest control in resource-constrained systems such as edge-devices.

- **C/C++**: One step above Assembly, C and C++ enable construction of resource-constrained code but with readability and
maintainability. C’s prominence allows you to find it in every use model. Python*: An interpreted language that simplifies prototyping, you can also use it for production. Python* supports a massive number of libraries and modules so that you can get more done with less code. It’s useful in more powerful edge devices, gateways, and even the cloud.

- **JavaScript*/Node.js*: A popular language and runtime, each enables the development of scalable network applications and can be applied across use models.

- **Node-RED*: Developing visually with Node-RED* makes it easy to build data flows that include sensors and actuators. If you know JavaScript*, these flows can be even more powerful. You can apply this language at gateways and within more powerful edge devices, such as those that the Edison board powers.

- **HiveQL**: If you’re using Hive (built on Hadoop), you can use HiveQL to process massive data sets in cloud environments.

- **Pig**: Using Pig Latin to process big data enables quick script development and simple experimentation of data sets within the cloud.

- **R**: An increasingly popular statistical computing language for the processing and visualization of data, you can apply data mining with R in cloud environments. R is also open source.
Julia: Another high-performance language for processing and visualizing data in cloud environments, Julia is also open source. It was designed with parallelism in mind and includes key elements for distributed computation.

• **Java**: A popular server-side and client-side language for the Web, making it useful in the cloud for both web and web services development.

For more such intel IoT resources and tools from Intel, please visit the [Intel® Developer Zone](https://software.intel.com/en-us/articles/selecting-programming-languages-for-the-iot)

Source:https://software.intel.com/en-us/articles/selecting-programming-languages-for-the-iot
OTHER POPULAR DEALS

Xolo ERA 1X -4G with VoLTE...
Rs. 3999
Buy Now

Swipe Elite 3- 4G with...
Rs. 4399
Buy Now

Lenovo Z2 Plus (Black, 32 GB)(3...
Rs. 10520
Buy Now
Interesting Galleries

Best Alternative Apps for your...
IBM Cloud Challenge
Contest | Win Prizes Worth Rs. 40000