Ultra Lightweight JavaScript Engine for Internet of Things

Evgeny Gavrin
Samsung Electronics, Russia
e.gavrin@samsung.com

Sung-Jae Lee
Samsung Electronics, South Korea
sj925.lee@samsung.com

Ruben Ayrapetyan
Samsung Electronics, Russia
r.ayrapetyan@samsung.com

Andrey Shitov
Samsung Electronics, Russia
a.shitov@samsung.com

Abstract

The demonstration aims to present JerryScript, a JavaScript engine for the Internet of Things (IoT). This is a lightweight JavaScript engine intended to run on very constrained devices such as microcontrollers, which have only a few kilobytes of RAM available to the engine (<64 KB RAM) and constrained ROM space for the code of the engine (<200 KB ROM). The engine is ECMA-262 5.1 compliant, supports on-device compilation, execution, and provides access to peripherals from JavaScript. It powers the IoT.js project, which provides an interoperable service platform in the world of web-based IoT. This demonstration proves that usage of JavaScript on every constrained device is reasonable and profitable.

Categories and Subject Descriptors D.3.4 [**Programming Languages**]: Processors; D.4.7 [**Operating Systems**]: Organization and Design

Keywords JavaScript engine; Internet of Things; interpreter; microcontrollers; IoT platform; virtual machine

1. Presenters

The demonstration will be conducted by the authors of the JerryScript engine [1] and the IoT.js platform [2].

1.1 Evgeny Gavrin

Leading engineer at Samsung Research center Russia (SRR), PhD student at Moscow State University (MSU), leader of the JerryScript project. Graduated from Saint-Petersburg State University (SpbSU). Areas of interest: virtual machines, dynamic compilation and compiler optimizations.

1.2 Sung-Jae Lee

Principal engineer at Samsung Electronics, leader of the IoT.js project. Graduated from Sungkyunkwan University. Areas of interest: Web technologies, Internet of Things.

1.3 Ruben Ayrapetyan

Engineer at SRR, core developer of the project. Graduated from MEPhI (Moscow Engineering Physics Institute). Areas of interest: Virtual machines, binary translation, interpreters, multithreading.

SPLASH Companion'15, October 25–30, 2015, Pittsburgh, PA, USA
ACM. 978-1-4503-3722-9/15/10...$15.00
http://dx.doi.org/10.1145/2814189.2816270

1.4 Andrey Shitov

Leading engineer at SRR, PhD student at MSU, core developer of the project. Graduated from MSU. Areas of interest: virtual machines, compilation of dynamic languages, type inference, AOT/JIT compilation, compiler optimizations.

2. Showcase

The demonstration is based on a client-server approach. The idea is to control and program very constrained devices remotely using JavaScript. The main concept is to let visitors of the conference interact with IoT devices and show the power of JavaScript for programming these devices.

2.1 Server Part

The server part consists of a Raspberry Pi 2 [RPi2] board, which has a web-server running on it, and a quantity of OpenMote [3] development boards – IoT devices. Every OpenMote board has a number of sensors and actuators connected to it, e.g. LEDs, servos, etc.

The Raspberry Pi 2 board interacts with OpenMote boards wirelessly via ZigBee protocol and serves as a hub, transmitting commands sent by user from the website.

Every OpenMote board has the JerryScript engine on it which is capable of executing JavaScript and accessing the boards' facilities.

2.2 Client Part

To control the IoT devices (OpenMote boards), a visitor can open a web-site where several control options are listed. He can do it either from his own gadget, or from preconfigured tablets that would be available in the demo room.

Simple control commands like "blink LED", "get sensor data" are represented as buttons in the web interface. In addition, the web interface provides the ability to run custom scripts on the devices. A sample script is shown in the edit box, so it can be customized by the visitor, sent to the OpenMote board and executed.

The script triggers LED blinking or servos rotation in a specified manner, so that the visitor can notice the result of his programming.

3. Concept

The demonstration shows an example of very simple and developer friendly application development for IoT.

The development of IoT solutions is very active. There will be a lot of IoT devices soon and these devices will be very small and constrained. The majority of commercial solutions utilize low-level languages, such as C or assembly, for programming such devices. This choice complicates software development, debugging, deployment and updating. Moreover, programming requires knowledge of low-level hardware details of the specific device; hence the developer should be highly qualified. Due to the lack of a common execution environment IoT developers have to handle such issues manually for each hardware target.

We believe that the key component of success in IoT is a lightweight execution environment that should become the core of the future IoT platform [2], providing a way to develop applications using runtime-safe technology while enabling easy programmability, deployment, updating, remote control, etc.

Although the idea is quite simple, the implementation of such an engine is very challenging. Consider the hardware characteristics of the OpenMote board utilized in the demonstration:

Table 1. OpenMote board hardware

MCU	CC2538SF53
Family	ARM Cortex-M3
RAM	32Kb
Flash	512Kb
Frequency	32MHz

Launching a JavaScript engine on such a device is a non-trivial task. The JerryScript engine solves it, showing very low memory consumption. The demo video [4] shows how the JerryScript engine, running on NUCLEO-F401RE microcontroller, can perform on-the-fly execution of JavaScript applications deployed to the board wirelessly through the web interface in a browser.

There are several tricks behind the implementation that allow executing within such constraints: compressed memory pointers, AST-less parser, compact JavaScript object representation, aggressive garbage collection, etc.

4. IoT.js

As mentioned earlier, the JerryScript engine powers IoT.js – a framework for the "Internet of Things". Conceptually, IoT.js and JerryScript are similar to node.js [5] and V8 [6] engine. The key differentiation point of this tandem is the capability of running on very constrained devices with less than 256KB of RAM, where node.js could not even be launched. Additionally, IoT.js and JerryScript have very small binary size ~500KB, in comparison to the 15MB of node.js.

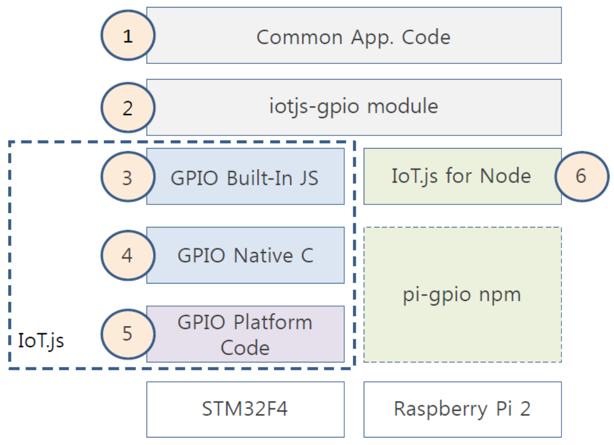

Figure 1. IoT.js module layout

IoT.js has a package management system for feature expansion and on-demand installation to save space on the device. IoT.js packages are backward compatible with node.js, so the developers have the ability to use well-known libraries and development experience to write applications for a variety of devices.

IoT.js hides low-level details, allowing running the same code on different platforms. The demo video [7] shows how the same JS code is launched on RPi2 with node.js, RPi2 with IoT.js and STM32F4 with IoT.js. Figure 1 describes the module layout for this demo. The following components are shown:

1. Demo application code (for both iot.js and node.js)
2. "iotjs-gpio" module to switch between iot.js and node.js depending on the target
3. Built-in gpio.js that calls native gpioctl
4. Native gpioctl code; Binding JS functions to C functions
5. Target platform dependent code; for NuttX/STM32F4
6. "pi-gpio" module for using gpio from node.js

Iotjs-gpio module, which provides the API for GPIO control, hides target-specific details from the developer, so that code for setting up GPIO pin high is very concise at JS level (component 1 in the figure):

```
gpio.write(portpin, on);                                (1)
```

The same platform dependent code (component 5 in the figure) is more complex:

```
struct gpioioctl_write_s wdata;
wdata.port = portpin;
wdata.data = data;                                      (2)
return ioctl(_fd, GPIOIOCTL_WRITE,
             (long unsigned int) &wdata);
```

As a result, developers write cross platform code without meditating on low-level "C" interfaces and choosing which board to support.

5. Summary

The demonstration presents a newly developed, ultra lightweight JavaScript engine, capable of executing applications on very constrained devices, and IoT.js framework, enabling cross-platform development for IoT.

Visitors can see the typical scenario of interacting with IoT devices and try the modern way of programming for the IoT platform.

References

[1] http://samsung.github.io/jerryscript/
[2] http://samsung.github.io/iotjs/
[3] http://www.openmote.com/
[4] http://www.youtube.com/watch?v=zhQW6ywO6sM
[5] https://nodejs.org/
[6] http://code.google.com/p/v8/
[7] http://www.youtube.com/watch?v=FLnT129j64c