Introduction to IoT.js

Tilmann Scheller
Principal Compiler Engineer
t.scheller@samsung.com

Samsung Open Source Group
Samsung Research UK

OpenIoT Summit 2016
San Diego, USA, April 4 – 6, 2016
Overview

• Introduction
• JerryScript
• IoT.js
• Memory consumption/Performance
• Demo
• Future work
• Summary
Introduction
What is IoT.js?

- A lightweight version of Node.js
- Attempt to bring the success of Node.js to the embedded world
- Retains backwards compatibility with Node.js as much as possible
- Runs on top of JerryScript
## IoT Hardware

<table>
<thead>
<tr>
<th>IoT Hardware</th>
<th>Memory spec</th>
<th>IoT Hardware</th>
<th>Memory spec</th>
</tr>
</thead>
</table>
| PanStamps             | -RAM : 2KB  
                       -ROM : 33KB  
                       (Flash:32KB, EEPROM:1KB) | Nanode                | -RAM : 32KB  
                       -ROM                                                |
| TinyDuino             | -RAM : 2KB  
                       -ROM : 33KB  
                       (Flash:32KB, EEPROM:1KB) | Arduino Yun           | -RAM : 2.5KB  
                       -ROM  
                       (Flash:32KB, EEPROM:1KB)                           |
| Arduino Uno           | -RAM : 2KB  
                       -ROM : 33KB  
                       (Flash:32KB, EEPROM:1KB) | mbed – LPC1768        | -RAM : 32KB  
                       -ROM  
                       (Flash:32KB, EEPROM:1KB)                           |
| RFduino               | -RAM  
                       -ROM                                                | Wi-Go Module          | -RAM  
                       -ROM : 2MB                                          |
| XinoRF                | -RAM : 2KB  
                       -ROM : 33KB  
                       (Flash:32KB, EEPROM:1KB) | pcDuino               | -RAM : 1GB  
                       -ROM : 2GB                                          |
| OpenKontrol Gateway   | -RAM : 32KB  
                       -ROM                                                | OpenPicus Flyport WiFi| -RAM  
                       -ROM : 16MB                                          |
| Pinoccio              | -RAM : 32KB  
                       -ROM                                                | Hackberry             | -RAM : 512MB/1GB  
                       -ROM : 4GB                                             |
| Raspberry Pi          | -RAM : 512MB  
                       -ROM                                                | UDOO                  | -RAM : 1GB  
                       -ROM                                                |
| BeagleBone Black      | -RAM : 512MB  
                       -ROM                                                | Libelium Wasmote      | -RAM  
                       -ROM                                                |
| CubieBoard            | -RAM : 512MB/1GB  
                       -ROM                                                | The Rascal            | -RAM : 64MB  
                       -ROM                                                |

What is JerryScript?

- A really lightweight JavaScript engine
- Has a base footprint of 10KB of RAM
- Optimized for microcontrollers
- Developed from scratch by Samsung
Why JavaScript on microcontrollers?

- There's a huge pool of JavaScript developers
- Opens up the possibility for web developers to easily write software for embedded devices
- Performance overhead of JavaScript less of an issue for control tasks
- Increased productivity, shorter time to market
- Ability to load code dynamically over the network
- Security: Executing JavaScript code is safer than executing arbitrary native code
Open source

- Actively developed on GitHub
- JerryScript and IoT.js are both open source released under the Apache 2.0 license
- Building up a community around IoT.js/JerryScript
JerryScript
JerryScript History

- Development started in June 2014
- Released as open source in June 2015
- JerryScript passed 100% of the test262 conformance test suite in August 2015
- Rewritten compact byte code implementation landed in January 2016
- Current focus on performance optimization
JerryScript

- Heavily optimized for a low memory footprint
- Interpreter-only
- Compact object representation
- Compressed pointers
- No AST, directly creating byte code
- Compact byte code heavily optimized for low memory consumption
JerryScript Portability

- Extremely portable
- Self-contained
- Small C library
- Can run bare-metal
- Supports the STM32F4, ESP8266 boards
- Runs on Linux/OS X as well
JerryScript

- Written in C99
- About 74KLOC
- Code size ~190KB when compiled with GCC LTO for ARM Thumb-2
- Implements the entire ECMAScript 5.1 standard, passes 100% of the test262 conformance test suite
- C API for embedding JerryScript
- Byte code snapshot feature
JerryScript C API

```c
#include <string.h>
#include "jerry.h"

int main (int argc, char * argv[]) {
    char script [] = "print ('Hello, World!');"

    jerry_completion_code_t code = jerry_run_simple (script,
                                                    strlen (script),
                                                    JERRY_FLAG_EMPTY);
}
```
JerryScript C API

```c
int main (int argc, char * argv[]) {
    char script1 [] = "var s = 'Hello, World!';";
    char script2 [] = "print (s);";

    // Initialize engine
    jerry_init (JERRY_FLAG_EMPTY);

    jerry_api_value_t eval_ret;

    // Evaluate script1
    jerry_api_eval (script1, strlen (script1),
                    false, false, &eval_ret);
    // Free JavaScript value, returned by eval
    jerry_api_release_value (&eval_ret);

    // Evaluate script2
    jerry_api_eval (script2, strlen (script2),
                    false, false, &eval_ret);
    // Free JavaScript value, returned by eval
    jerry_api_release_value (&eval_ret);

    // Cleanup engine
    jerry_cleanup ();
}
```
IoT.js
IoT.js

- Lightweight version of Node.js
- Focus on resource-constrained devices
- Tries to retain backwards compatibility
- Mostly written in JavaScript
IoT.js Architecture

Application

IoT.js API

IoT.js core

Javascript Binding Libuv Binding

JerryScript libtuv

Operating System
IoT.js Modules

- Assert
- Buffer
- DNS
- Events
- File System
- HTTP

- Net
- Modules
- Process
- Stream
- Timers
- GPIO
Target hardware

- STM32F4 developer board
- Cortex-M4F clocked at 168 MHz
- 192KB of RAM
- 1MB of flash memory
NuttX

- Real-time operating system
- Open source (BSD license)
- Scales from 8-bit microcontrollers to 32-bit microcontrollers
- Supports the STM32F4 developer board
libtv

- libuv - Support library for asynchronous I/O
- libtv - Stripped down version of Libuv
- Runs on NuttX and Linux
- Experimental support for mbed OS
Memory consumption/ Performance
IoT.js - Memory consumption

<table>
<thead>
<tr>
<th>Test name</th>
<th>IoT.js</th>
<th>Node.js</th>
</tr>
</thead>
<tbody>
<tr>
<td>require('http')</td>
<td>40KB</td>
<td>6.70MB</td>
</tr>
<tr>
<td>test_httpserver.js</td>
<td>80KB</td>
<td>8.63MB</td>
</tr>
</tbody>
</table>

Measured on a Raspberry Pi 2
SunSpider 1.0.2 - Memory consumption

Max RSS in KB (lower is better)

Measured on a Raspberry Pi 2
SunSpider 1.0.2 - Performance

Execution time in seconds (lower is better)

Measured on a Raspberry Pi 2
ubench - Memory consumption

Measured on a Raspberry Pi 2
ubench - Performance

Measured on a Raspberry Pi 2

Execution time in seconds (lower is better)
Demo
Tetris Demo

- Implementation of the classic Tetris game
- Each device drives one LED matrix as display
- Implemented as a Node.js module
- Both devices are running exactly the same JavaScript code
Demo

Raspberry Pi 2
(1GB RAM, 8GB Flash)

Tetris
Node.js
V8
Linux

LED Matrix

I2C

USB Keypad

STM32F4 board
(192KB RAM, 1MB Flash)

Tetris
IoT.js
JerryScript
NuttX

LED Matrix

I2C

Switches via GPIO
Tetris Demo

https://youtu.be/q4iBeOAC7XI
Pong Demo

- Implementation of the classic Pong game
- Display shared across two devices
- Each device drives one LED matrix
- Implemented as a Node.js module
- "AI" opponent running on the microcontroller
Demo

Raspberry Pi 2
(1GB RAM, 8GB Flash)

Pong Client
Node.js
V8
Linux

STM32F4 board
(192KB RAM, 1MB Flash)

Pong Server
IoT.js
JerryScript
NuttX

USB Keypad

LED Matrix

I2C

Ethernet
Pong Demo
Future work
Future work

- Close performance gap between JerryScript and Duktape
- Further performance and memory optimizations
- Enhance JerryScript C API
- Add more IoT.js modules
- Package manager for IoT.js
- Support more devices
Summary
Summary

- Significantly lowers barrier of entry for JavaScript development targeting heavily constrained embedded devices
- Speeds up development
- Active community
- Looking for bug reports and feedback
Thank you.
Contact Information:

Tilmann Scheller
t.scheller@samsung.com

Samsung Open Source Group
Samsung Research UK