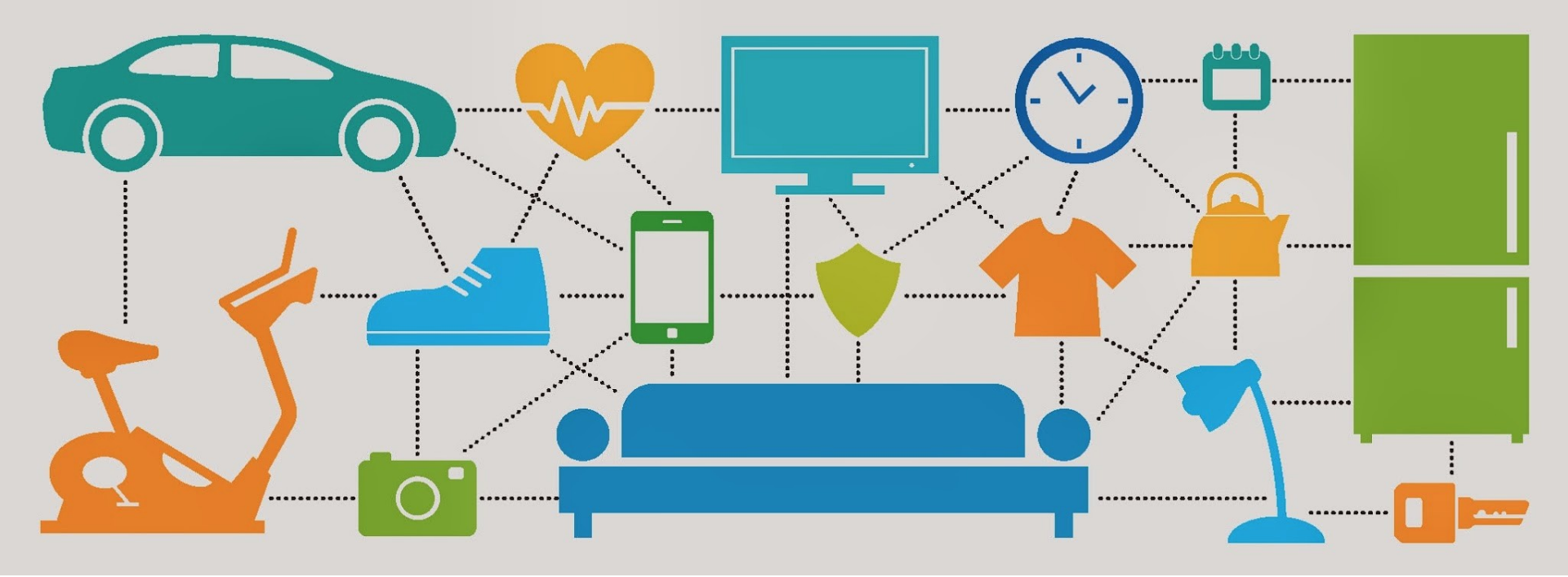


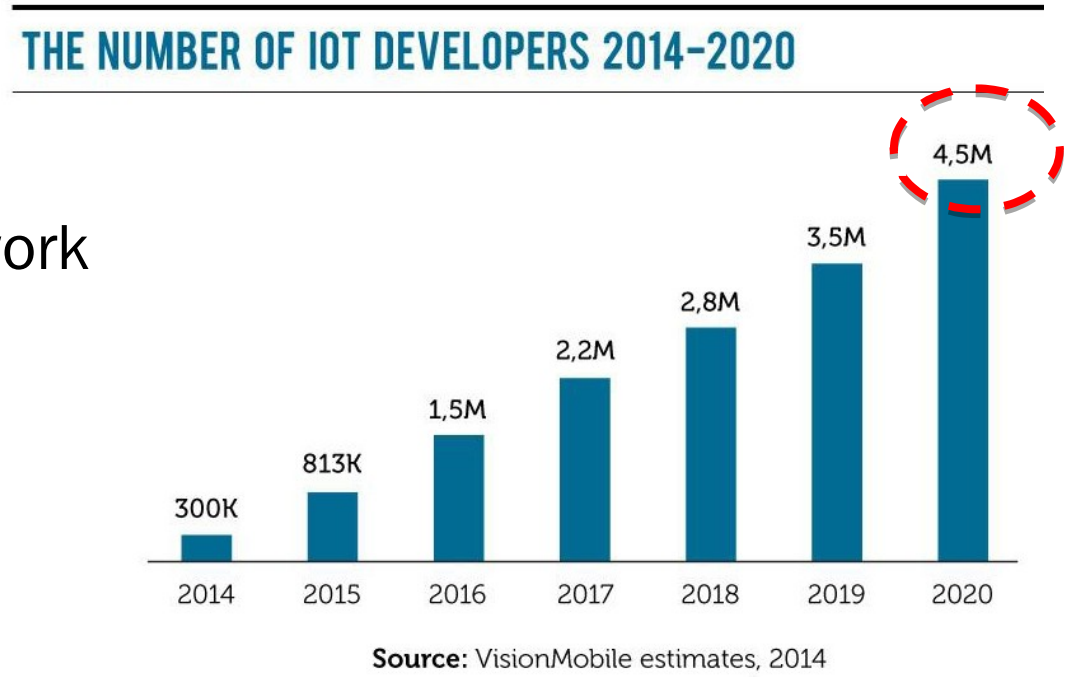
Introduction to IoT.js

World of Connected Devices

"Number of Connected objects expected to reach 50 billion by 2020", Cisco 2014
"There is expected to be 75 billion connected devices by 2020", Intel 2014

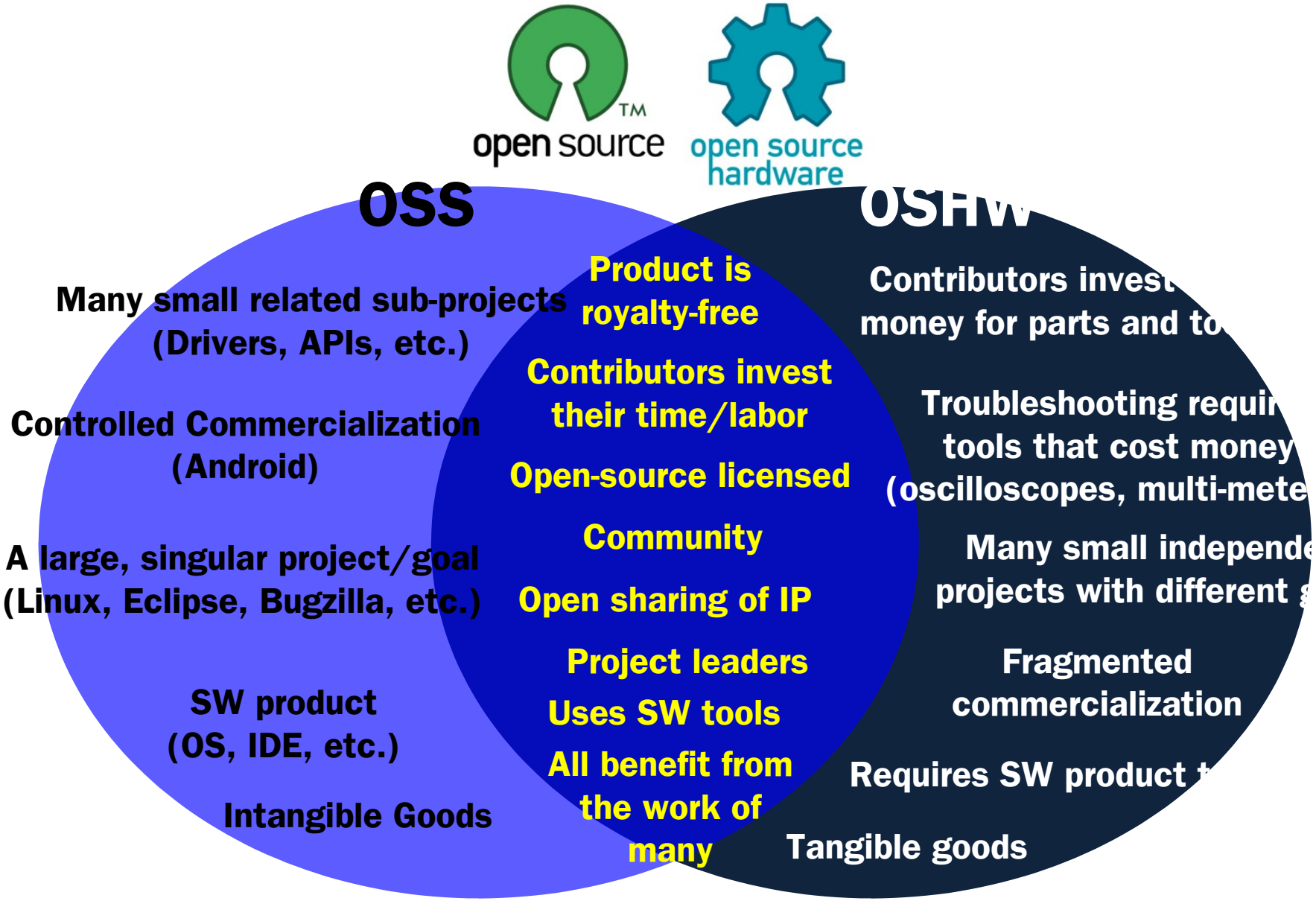
Number of connected devices is continuously growing!

- Becoming cheaper
- Connected to the network
- Sensing data
- Acting on output

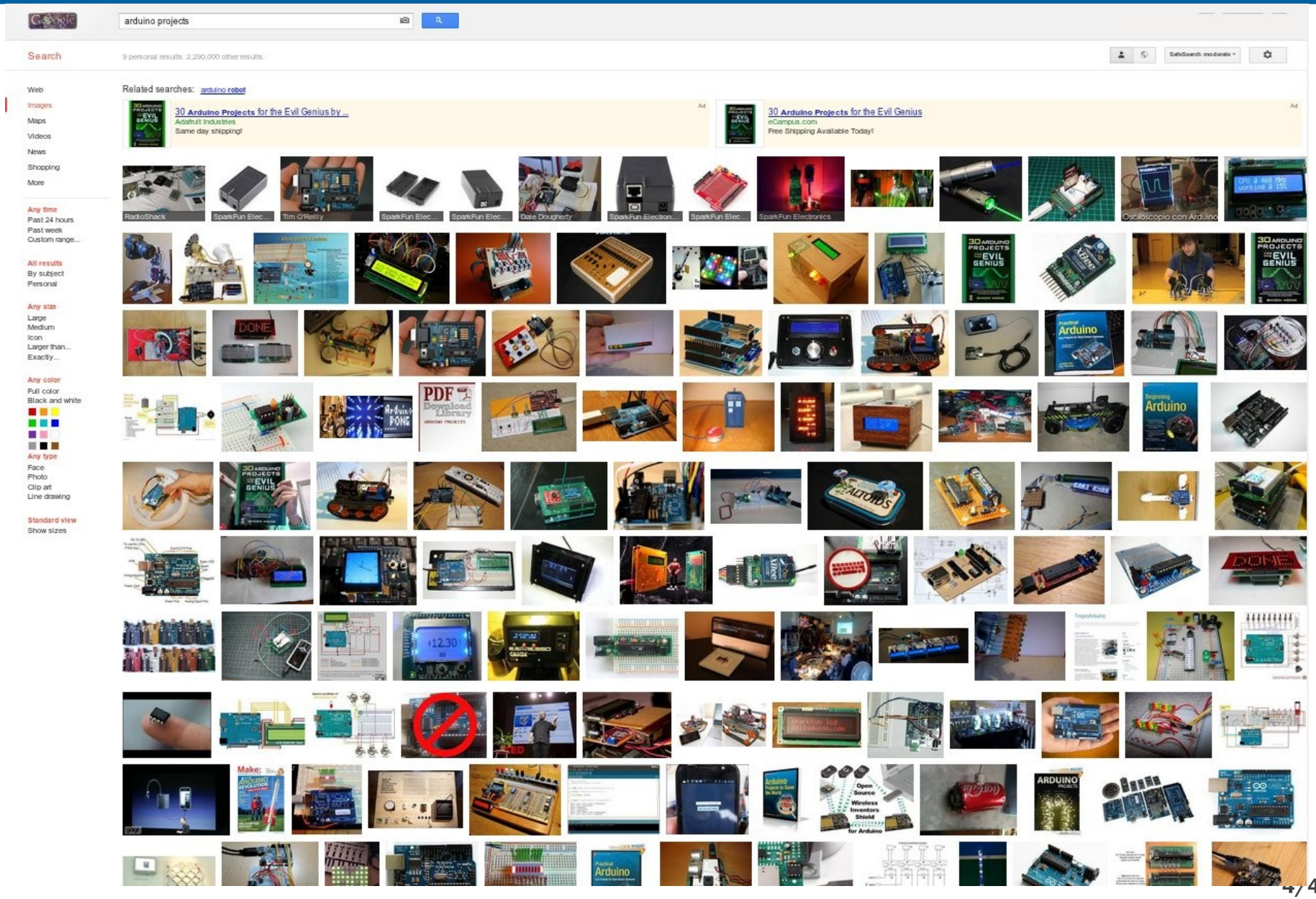


Let's create interactive applications
by organizing billions of connected devices!

IoT with OSS/OSHW



IoT HW devices



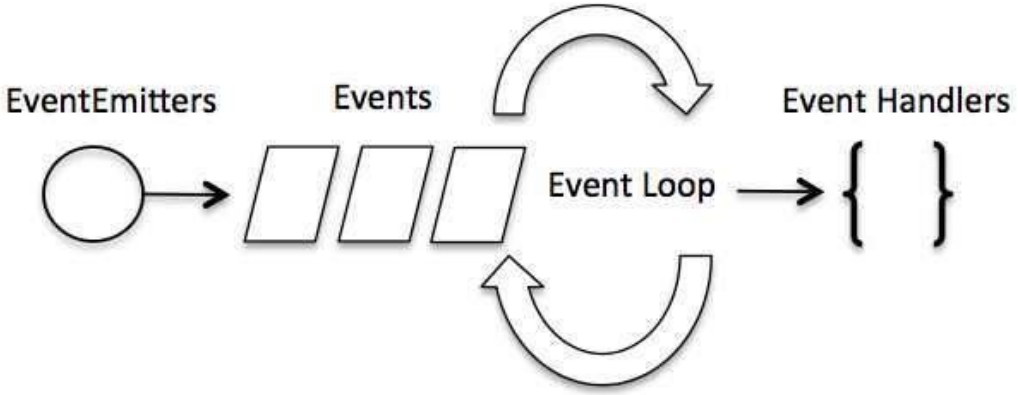
Web Technologies for IoT

- Existing Web technologies provide simple and well-known way for creating interactive applications
- Device can be integrated to the Web with REST API
- JavaScript can be used for application development
  - Application that is running on device
  - Application that organizes devices
- Devices can be identified by its unique URI

There is a need of Web Platform for IoT!

Why JavaScript in IoT ?

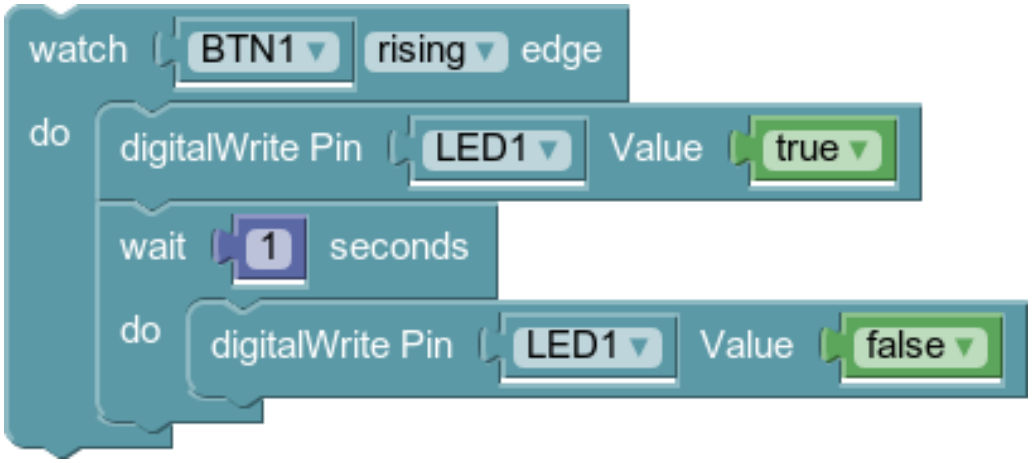
- Programming model well-suited for embedded devices
  - Single threaded programming model
- Event-driven asynchronous programming model
  - Supports asynchronous function calls and I/O
  - Asynchronous calls are useful for event-driven sensor handling



- Community driven module repository
  - Over a hundred thousand packages available via the npm package manager
  - Modules can include native components

JavaScript for IoT

```
setWatch(function(){
    digitalWrite(LED1,1);
    setTimeout(function(){
        digitalWrite(LED1,0);
    },1000);
},BTN1,{edge:"rising"});
```



Everything that can be written in JavaScript“

”.will eventually be written in JavaScript

)Atwood's law(

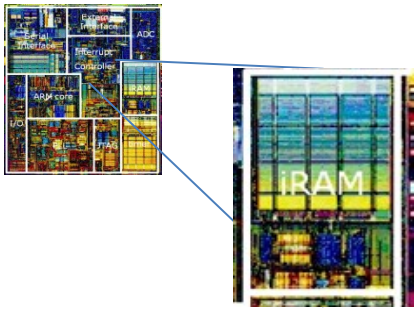
Node.js on IoT devices?

IoT Hardware	Memory spec	IoT Hardware	Memory spec
 **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 : 33KB (Flash:32KB, EEPROM:1KB)
 **Arduino Uno**	**-RAM : 2KB** -ROM : 33KB (Flash:32KB, EEPROM:1KB)	 **mbed – LPC1768**	-RAM : 32KB -ROM : 512KB
 **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

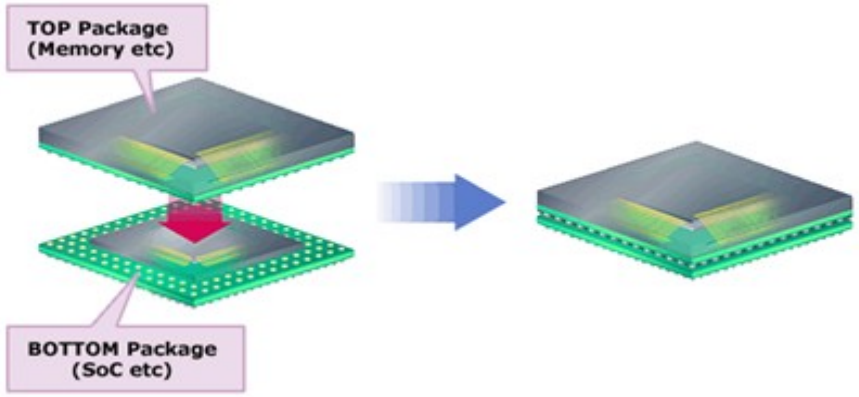
http://postscapes.com/internet-of-things-hardware#resources

Memory placement

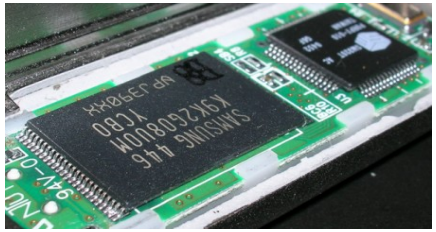
- SoC (System on chips)



- POP (Package on package)



- On Board



- External

Memory concerns…

Name of Product / Board		ARTIK 1	ARTIK 5	ARTIK 10	OpenMote	Arduino Uno	Arduino Due	Raspberry Pi I	Raspberry Pi II
Processor	Name	MIPS32	Cortex-A7	Cortex-A15/Cortex-A7	TI CC2538	ATmega328	AT91SAM3X8E	BCM2835	Cortex-A7
	Architecture	MIPS32 DualCore	ARM Cortex-A7 DualCore	ARM Cortex-A15 QuadCore + ARM Cortex-A7 QuadCore	ARM Cortex-M3	8bit AVR	ARM Cortex-M3	ARM1176JF-S	ARM Cortex-A7 QuadCore
	Clock Speed	250Mhz + 80Mhz	1Ghz	4@1.3Ghz + 4@1.0Ghz	32Mhz	20Mhz	84Mhz	700Mhz	900Mhz
SoC Built-in / POP	RAM	1MB SRAM	512MB LPDDR3	2GB LPDDR3	32KB SRAM	2KB SRAM	96KB SRAM		
	Flash		4GB eMMC		512K NOR	96KB NOR	512KB NOR		
On Board	RAM							256MB DDR2	1GB DDR3
	Flash	4MB NAND Flash		16GB eMMC					
External	Flash		SDCARD	SDCARD				SDCARD	4GB eMMC

JS engine for low memory devices

Subset of JavaScript

- Language leveling based on
  - manual customization
  - target device specification
- **No new syntax** constructs like in
  - TypeScript
  - CoffeeScript
  - WearScript
- Support of built-ins

Access to **peripherals**

- Define **Common Peripherals API**
  - Sensors
  - Actuators
- Vendor chooses what to implement

Requirements:

- Cortex M3/M4/M4F
- Ability to run in a few KB of memory
- **On-device** compilation and execution
- Optional **over-the-air updater**

JerryScript Project Goals

Develop ECMA-262 5.1 compliant small footprint engine:

- Memory consumption can be limited to the specified value
  - this will limit engine's capability to handle memory consuming applications
- Language features can be limited
  - based on chosen memory limit
  - by manual tuning
- Parser should provide full support of ECMA-262 and produce correct byte-code for interpretation
  - optionally develop set of byte-code optimizations
- Define ECMA-262 5.1 subset for Micro Controllers
- Define prototype of Common-API for interaction with peripherals

How we provide a small overhead

JerryScript is a pure interpreter
- No overhead for storing compiled code
- High-level byte-code that simplifies implementation

Representation of JS objects optimized for size
- Designed to save memory, not for performance

Parser doesn't store AST
- It produces byte-code directly from the source code

Parser process code line-by-line
- It doesn't store full program in memory

Target segment for memory

Done by 2014, Planned in 2015 and 2016

Segment	Memory Footprint	Binary Size	Configuration
Pico	8KB~16KB	~100KB	Compact JerryScript + Subset Profile
Nano	16KB~64KB	~200KB	JerryScript + Full Profile
Micro	64KB~256KB		JerryScript + Bytecode Optimization
Light	256KB~	~300KB	JerryScript with JIT
Full	8MB~	10MB	V8

Development Environment

- Host OS
  - Linux
    - 32bit/64bit Ubuntu 14.04 LTS versions.
  - Mac OSX
- Development Tools
  - Cmake
  - Valgrind
  - STM32F4 BSP
- Considering cloud based development environment, in the future

- Target OS
  - RTOS
    - NuttX as 1st reference
  - Tizen and other embedded linux based OS
- Reference H/W
  - STM32F4
    - STM32F4 Discovery board with BB(Base Board)
  - OpenMote
  - ARM mbed platform (NUCLEO-F411RE, Freescale FRDM-K64F)

To do …

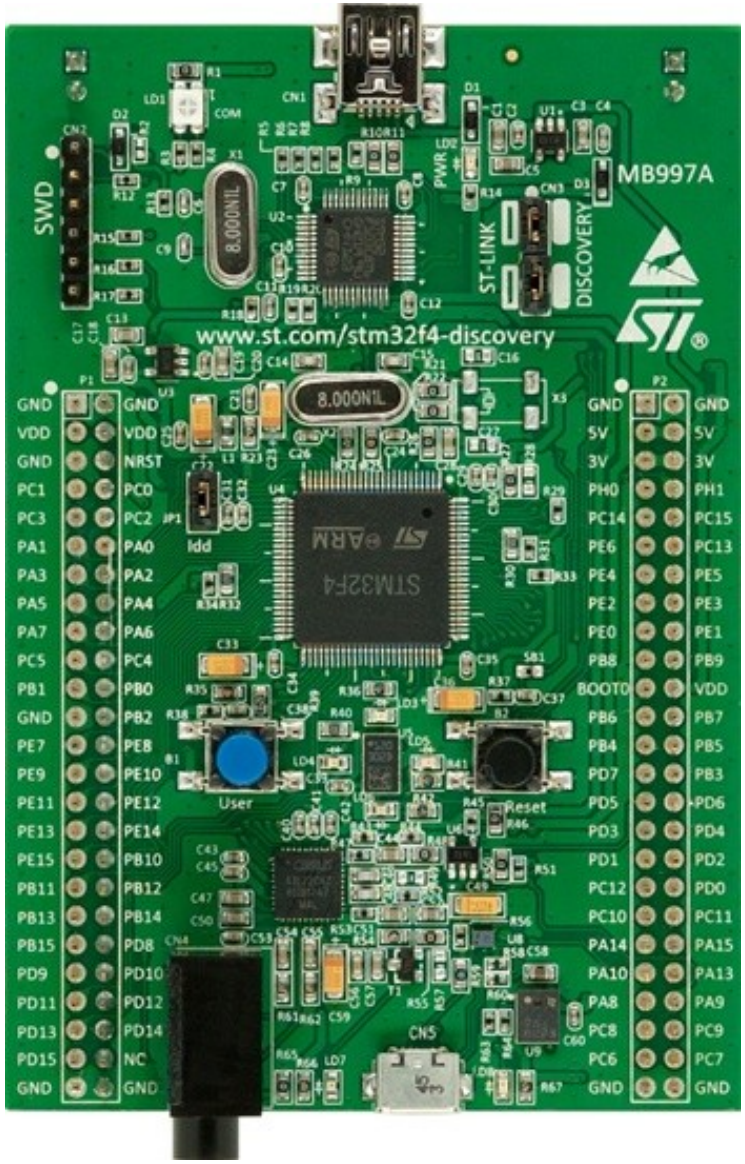
- 2015 (under development)
  - ECMA-262 v5.1 full profile support, and v6.0 if possible
    - built-in Objects
  - Implement embedding API for IoT.js
    - Multiple context.
  - Runtime memory footprint optimization
    - GC optimization
  - Byte-code Optimizer for performance

- 2016 (planned)
  - Implement JIT(to enable inline cache) for performance.
    - Expect 10 times faster engine.

JavaScript Subset Profile

- Do not support eval(), same as Function or new Function.
  - eval() requires run time compilation because source text might not appear in program or can be generated at run time.
- Do not support of “with” Operator
  - With makes access to named references inefficient, because the scope for such access cannot be compiled until run time.
- Do not support addition, deletion or assignment to the properties of built-in objects.
  - This limitation is needed to support a more efficient implementation based on static compilation of built-in objects without risking that objects are mutated on shadowed by dynamically added properties.
- Do not support Unicode.
- No Global object, no Arguments object.
- Switch Number type representation from 64bit to 32/16/8bit.
  - double → float, Cortex-M4F has only float support
- Immutable strings.
  - No mutation or re-creation
- Limit recursion level
  - Configurable

Development Board



- STM32F4 Discovery
- 168 Mhz
- 64KB CCM + 128KB RAM
  core coupled memory
- 1MB Flash
- Able to run Espruino
  - to evaluate and compare

Native Code vs. Interpretation

Native

Execution time: **4ms**

```
volatile int count = 10000;
volatile int i;
volatile float x = 7;
volatile float y = 3;

volatile float tmp1 __unused;
volatile float tmp2 __unused;
volatile float tmp3 __unused;
volatile float tmp4 __unused;

for (i = 0; i < count; i++)  {
  tmp1 = x * x;
  tmp2 = y * y;
  tmp3 = tmp1 * tmp1;
  tmp4 = tmp2 * tmp2;
}
```

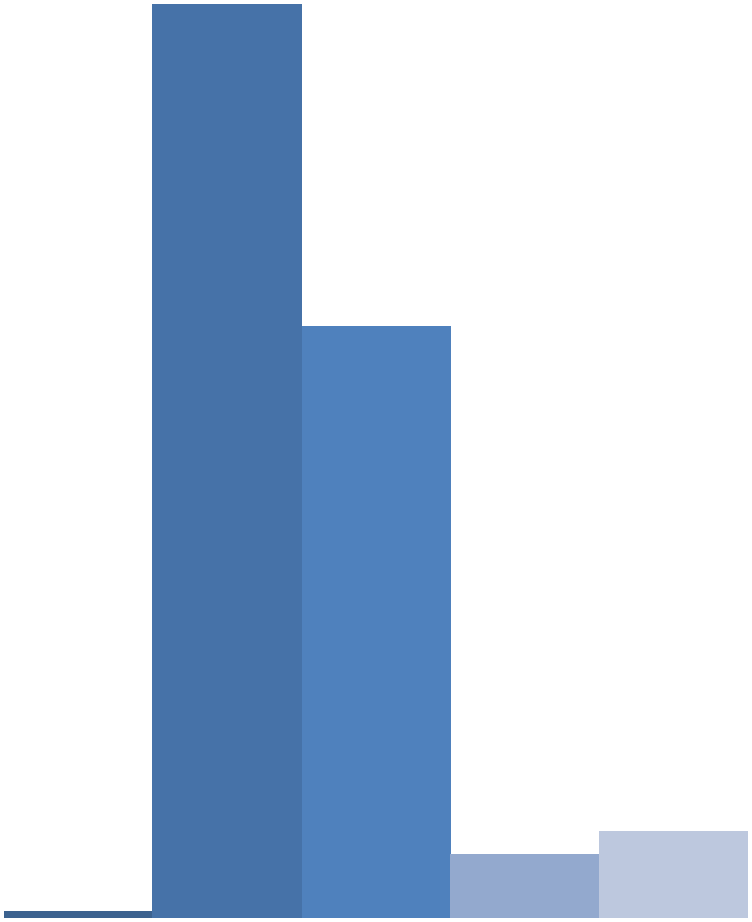
JerryScript

Execution time: **4870ms**

```
var count = 10000;
var x = 7;
var y = 3;

var tmp1;
var tmp2;
var tmp3;
var tmp4;

for (var i = 0; i < count; i++)
{
  tmp1 = x * x;
  tmp2 = y * y;
  tmp3 = tmp1 * tmp1;
  tmp4 = tmp2 * tmp2;
}
```

Host Performance Comparison

Memory Consumption

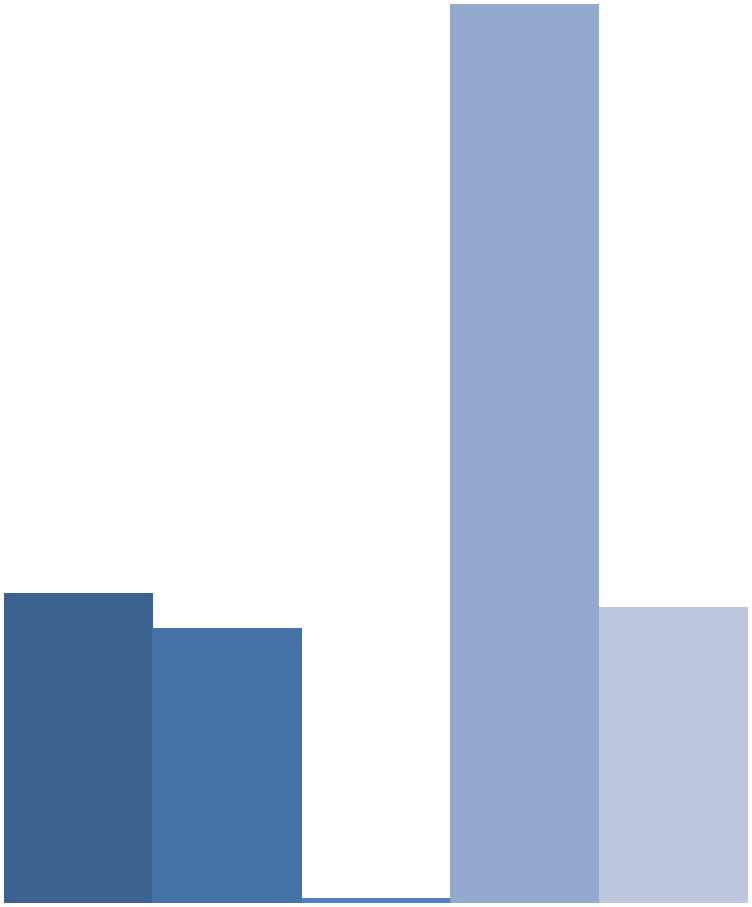


```
var count = 10000000;
var x = 7;
var y = 3;

var tmp1;
var tmp2;
var tmp3;
var tmp4;

for (var i = 0; i < count; i++)
{
  tmp1 = x * x;
  tmp2 = y * y;
  tmp3 = tmp1 * tmp1;
  tmp4 = tmp2 * tmp2;
}
```

x86_64 GNU/Linux

Host Performance Comparison

Execution Time



```
var count = 10000000;
var x = 7;
var y = 3;

var tmp1;
var tmp2;
var tmp3;
var tmp4;

for (var i = 0; i < count; i++)
{
  tmp1 = x * x;
  tmp2 = y * y;
  tmp3 = tmp1 * tmp1;
  tmp4 = tmp2 * tmp2;
}
```

x86_64 GNU/Linux

Target Performance Comparison

Benchmark	Memory (Kb)					Performance (sec)		
	JerryScript			Duktape	V8	JerryScript (Full)	Duktape	V8
	(Full)	(Compact Profile)	(Minimal Footprint)					
SunSpider v1.0.2	964	940	836	2,268	24,840	41	8	3
ubench	300	300	264	1,880	20,428	151	106	4
Global Sum	1,264	1,240	1,100	4,148	45,268	192	115	7
	1.15	1.13	**1**	**3.8**	**41.2**	28.4	16.9	1

```
V8    b2ed25304e203bbd22d6b09db575980f6aecf30a
Jerry abc2b55297eff13538fca2440d127ba79cdcc8b3
duk   cd2c19761b07e8d675f2079328fbfc9dd9c3b83a
```

* Measured on arndale board, on arm x32 Linux

Target Performance Comparison

Benchmark	Memory (Kb)					Performance (sec)		
./tests/ubench/*	JerryScript			Duktape	V8	JerryScript (Full)	Duktape	V8
	(Full)	(Compact Profile)	(Minimal Footprint)					
function-closure.js	32	32	28	504	1,900	1.90	5.73	0.24
function-correct-args.js	36	36	32	172	2,196	33.80	9.73	0.43
function-empty.js	32	32	28	172	2,196	11.93	12.83	0.45
function-excess-args.js	36	36	32	172	2,272	24.25	10.37	0.47
function-missing-args.js	32	32	28	172	2,516	25.37	8.63	0.50
function-sum.js	36	36	32	172	2,324	18.14	8.64	0.38
loop-empty-resolve.js	32	32	28	172	2,624	1.86	2.18	0.22
loop-empty.js	32	32	28	172	2,184	16.23	19.17	0.56
loop-sum.js	32	32	28	172	2,216	17.43	28.76	0.65
Sum	300	300	264	1,880	20,428	151	106	4
	1.1	1.1	**1.0**	**7.1**	**77.4**	38.9	27.3	1

* Measured on arndale board, on arm x32 linux

Target Performance Comparison

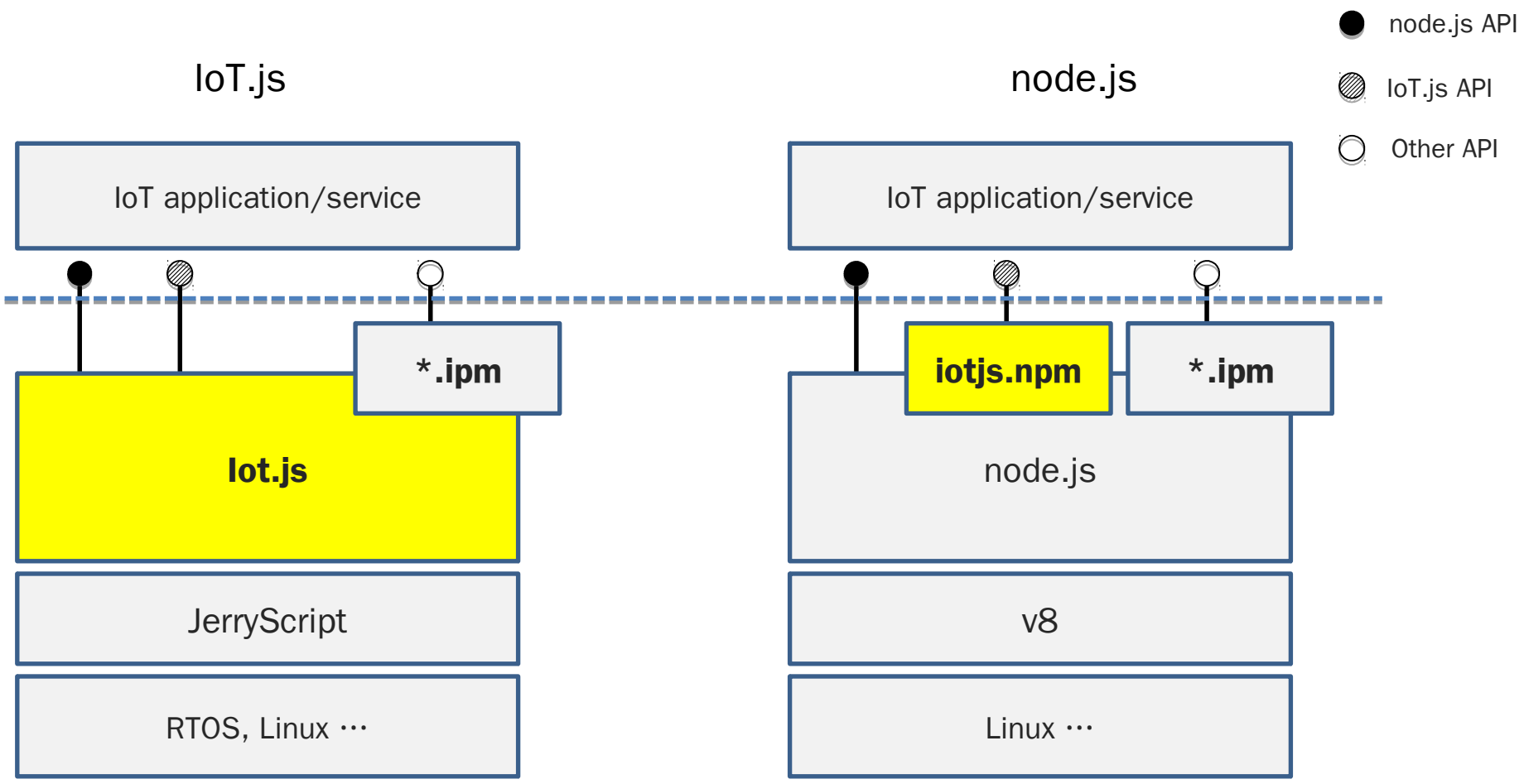
Benchmark	Memory (Kb)					Performance (sec)		
./tests/sunspider-1.0.2/*	JerryScript (Full)	JerryScript (Compact Profile)	JerryScript (Minimal Footprint)	Duktape	V8	JerryScript (Full)	Duktape	V8
3d-cube.js	172	148	128	224	1,788	2.59	0.38	0.26
access-binary-trees.js	96	100	88	240	1,880	1.36	0.36	0.22
access-fannkuch.js	64	64	52	176	2,164	6.07	0.65	0.23
access-nbody.js	84	84	72	184	2,064	2.79	0.49	0.23
bitops-3bit-bits-in-byte.js	44	44	36	172	2,068	2.24	0.19	0.22
bitops-bits-in-byte.js	40	40	32	172	2,340	3.18	0.29	0.23
bitops-bitwise-and.js	32	32	28	172	1,892	2.01	2.33	0.26
controlflow-recursive.js	208	204	208	216	2,228	1.92	0.37	0.22
math-cordic.js	52	52	48	176	2,164	3.38	0.80	0.23
math-partial-sums.js	48	48	44	172	1,864	1.55	0.96	0.26
math-spectral-norm.js	56	56	44	176	2,296	1.84	0.32	0.24
string-fasta.js	68	68	56	188	2,092	12.07	1.35	0.28
Sum	964	940	836	2,268	24,840	41	8	3
	1.2	1.1	**1.0**	**2.7**	**29.7**	14.2	2.9	1

* Measured on arndale board, on arm x32 linux

IoT.js

- Node.js compatible,
  - with JerryScript engine
  - for light-weight devices and small things of IoT
  - Supporting RTOS
- Providing portable module system
  - for feature expansion
  - On demand installation support for limited storage on device
  - Framework for downloadable application support

node.js vs. IoT.js

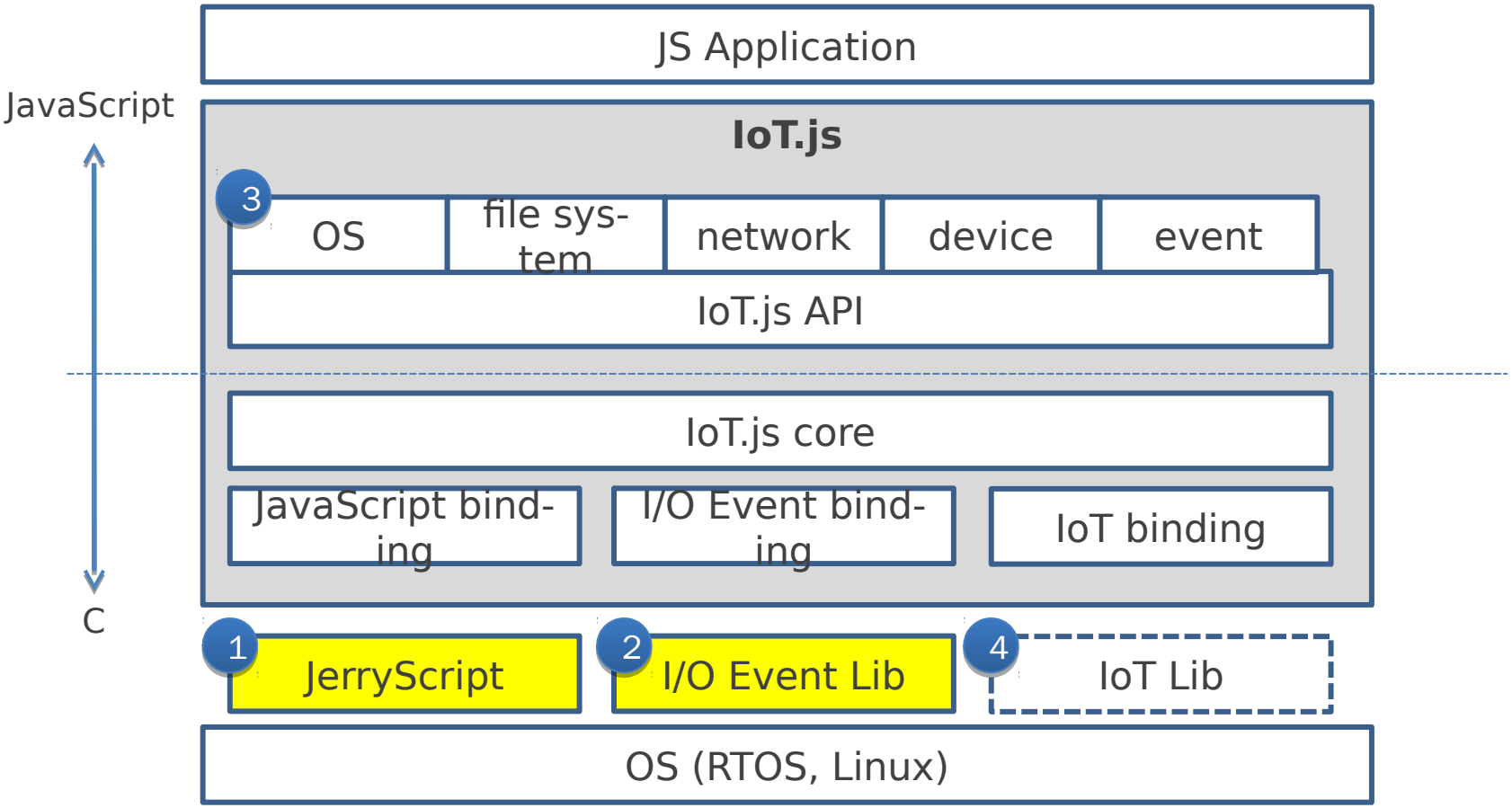


Goal of IoT.js

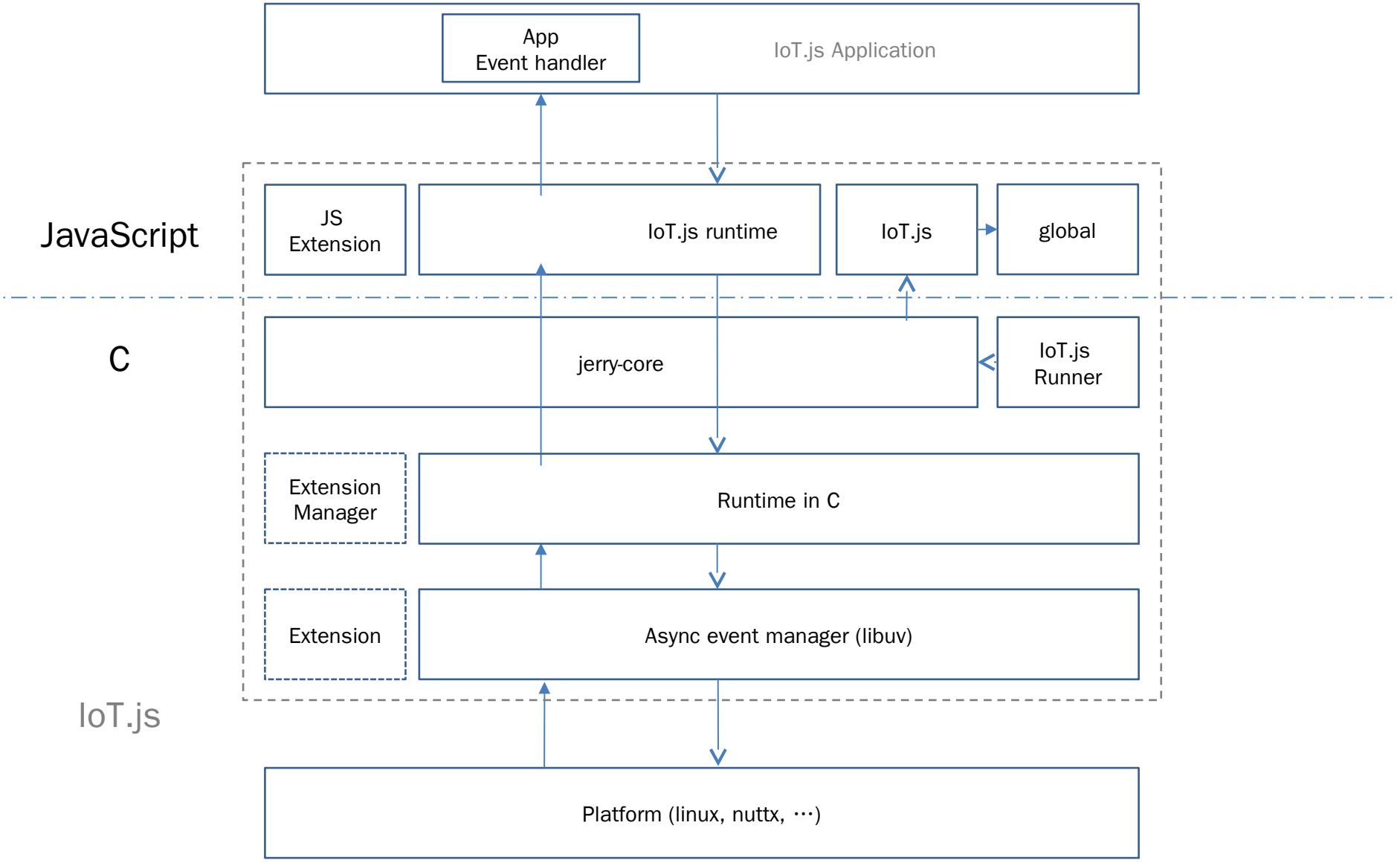
- Very small in size compared to node.js
  - Roughly less than 500KB vs 15MB
- Backward compatible with node.js applications
  - Support basic node.js API
  - Support npm like module system even on RTOS environment
- Open Innovation
  - Launching pure open source development project

Architecture of IoT.js

1 JavaScript 2 I/O event library + 3 lightweight framework + 4 IoT library



Architecture of IoT.js



npm compatiblity

- 1st, JavaScript + node.js basic API (subset) only npm.
  - Guarantee only backward compatibility in very small target segmen t. (Pico, Nano, Micro)
  - Try to achieve full compatibility in relatively big target segment (Light and over)
- 2nd, native npm → limited, under consideration
  - Too many problems.
    - Most RTOS not support dynamic module loading.
    - Need compilation when installation, but not suitable in most of our targ et environment
      - Consider binary distribution → too complex to manage
      - Host device assisted npm installation (cross-compilation of npm during instal l process)

Future Plan

- Being an standard IoT application platform
  - Integration for OIC IoTivity and others
  - Basic API & service logic implementation with JavaScript
  - Plan to participate in W3C Web of Things Interest Group (http://www.w3.org/WoT/IG/)
- Multiple segment for different devices
  - As for RAM amount:
    - pico: 8K~16K,
    - nano: 16K~64K
    - micro: 64K~256K,
    - light: 256K~16M
    - Big ones left for node.js
- Platform for most Samsung Electronics Appliances
  - Tizen RT, SmartThings

What others are doing?

- AllJoyn.js
  - AJTCL(AllJoyn Thin Core Library) JavaScript wrapper with duktape
  - AllJonyn npm* also exist for node.js but Object and methods differs that from AllJoyn.js

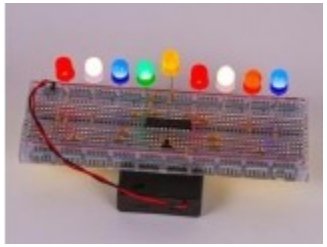
- deviceJS
  - IoT application framework built on node.js
  - Used by WigWag (http://ww.wigwag.com)

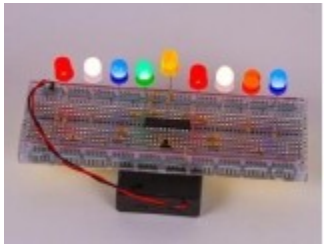
* https://www.npmjs.com/package/alljoyn

Demo Environments

NuttX on STM32F4

Linux on Raspberry Pi2





Demo Environments

- **Scenario**
  - Run LED blinking demo led.js on RPi2 with node.js
  - Again run same led.js on STM32F4 board with iot.js

SFJSE Wi-Fi demo

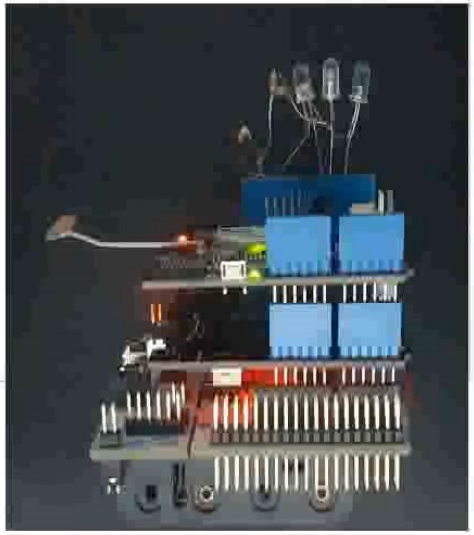
106.109.9.218:2000

```
for (var i = 0; i < 20; i++)
{
  LEDToggle (2);
  if (i % 2 == 0)
  {
    LEDToggle (3);
  }
  if (i % 5 == 0)
  {
    LEDToggle (1);
  }

  wait (100);
}
```

Submit

```
var toggle = [1, 2, 3, 1, 1, 1, 2, 3, 1];

for (var i = 0; i < toggle.length; i++)
{
  LEDToggle (toggle [i]);
  wait (200);
}
```

Submit



All LEDs on

All LEDs off

All LEDs toggle

All LEDs blink

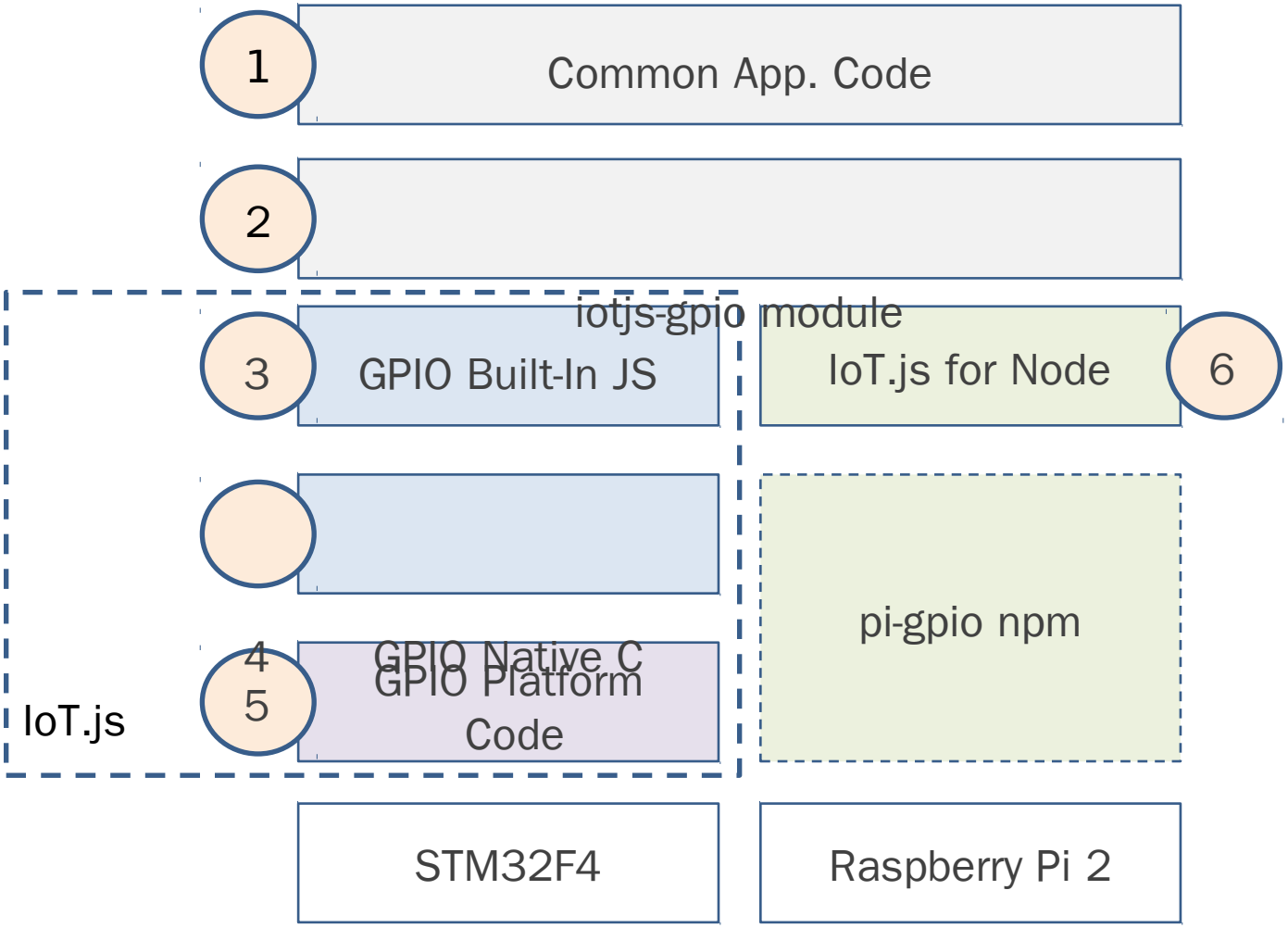
LED cosine

LED sine

Red LED random

All LEDs random

Demo IoT.js


1
Common App. Code
2
iotjs-gpio module
3
GPIO Built-In JS
IoT.js for Node
6
pi-gpio npm
4
GPIO Native C
5
GPIO Platform Code
IoT.js
STM32F4
Raspberry Pi 2

Sample Code

- Common function “run()” for both iot.js and node.js

1

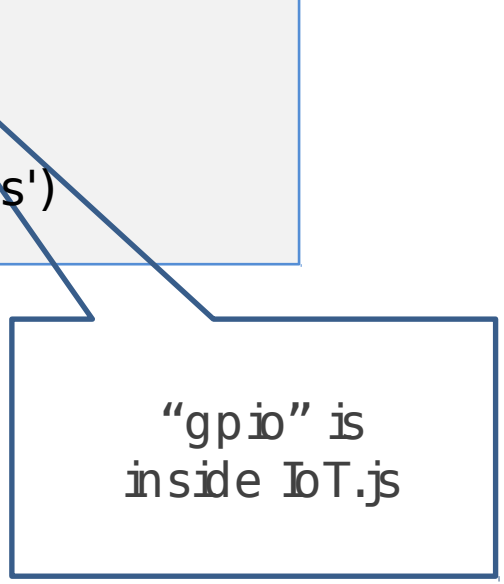
```
var gpio = require("iotjs-gpio")


function gpio_run() {
  var on = 1;
  var idx = 0;


  console.log("start blinking...");
  intervalId = setInterval(function() {
    var portpin = gpiocfg.map(gpio_pout[idx]);
    var err = gpio.write(portpin, on);
    idx = idx + 1;
    if (idx >= gpio_ocnt) {
      idx = 0;
      on = (on + 1) % 2;
    }
  }, 100);
```

Sample Code

- iotjs-gpio module to switch iot.js or node.js

2

```
if (process.iotjs) {
   module.exports = require('gpio');
}
else {
   module.exports = require('./node-gpio.js')
}
```



Sample Code

- Built-in gpio.js that calls native gpioctl

3

```
var gpioctl = process.binding(process.binding.gpioctl)


GPIO.write = function(portpin, val, callback) {
  var err = gpioctl.writepin(portpin, val);


  if (util.isFunction(callback)) {
    process.nextTick(function() {
      callback(err);
    });
  }
  else {
    return err;
  }
};
```

Sample Code

- Native gpioctl code; Binding JS "writepin()" to C "WritePin()"

4

```
JHANDLER_FUNCTION(WritePin, handler) {
   JObject* jgpioctl = handler.GetThis();
   GpioControl* gpioctrl =
                        GpioControl::FromJGpioCtl(*jgpioctl);


   uint32_t portpin = handler.GetArg(0)->GetInt64();
   uint8_t data = (uint8_t)handler.GetArg(1)->GetInt32();
   int err = gpioctrl->WritePin(portpin, data);
   ...
}


JObject* InitGpioCtl() {
   ...
   jgpioctl = new JObject();
   jgpioctl->SetMethod(JSCT("writepin"), WritePin);
   ...
```

Sample Code

- Target platform dependent code; for NuttX/STM32F4

5

```
int GpioControlInst::WritePin(uint32_t portpin,
                              uint8_t data) {
  if (_fd > 0) {
    struct gpioioctl_write_s wdata;
    wdata.port = portpin;
    wdata.data = data;
    return ioctl(_fd, GPIOIOCTL_WRITE,
                 (long unsigned int)&wdata);
  }
  return IOTJS_GPIO_NOTINITED;
}
```

Sample Code

- for node.js, for gpio, use “pi-gpio” module

```
6   var pi_gpio = require('pi-gpio');


    GPIO.write = function(port, val, callback) {
      pi_gpio.write(port, val, (callback || noop));
    };
```

Enjoy with us

www.jerryscript.net
www.iotjs.net