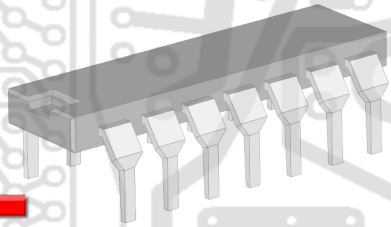
Prototyping the Internet of Things with JavaScript



Tessel

Espruino

Charalampos Doukas

v 1.0

Prototyping the Internet of Things with JavaScript

Charalampos Doukas

This book is for sale at http://leanpub.com/iot-javascript

This version was published on 2015-02-17

Contents

Foreword

The Internet of Things (IoT) is creating an exciting new world of opportunity for inventors, researchers and entrepreneurs as well as hardware & software developers. The IoT is gaining a great deal of momentum as the concept of a connected home becomes increasingly attractive. Development of prototypes and concepts are now also much easier to create, allowing small companies and entrepreneurs to bring their ideas to life without a huge capital outlay. Online platforms that support IoT data collection are presenting new and exciting opportunities to move this area of technology forward at a pace never seen before. As this open source hardware has made embedded development easier and more affordable, a great number of people are now interested in learning how the IoT works and building their own connected devices.

JavaScript (JS) is the most popular programming language in open source repositories[1] with the server-side, Node.js, enabling fast prototyping of web services and back-end applications. There are already numerous embedded platforms available that allow users to program directly in JS and support connectivity with the Internet. In addition, the most popular open source libraries that implement IoT protocols (MQTT, CoAP, WebSockets) are already available for JS. *This is the first time that developers can use one language to code on devices and write server applications.* And with code-less graphical environments such as Node-RED[2] prototyping an IoT application has never been easier.

Keep in mind that the projects presented in the book make use of hardware including the Tessel WiFi[3] board, the RaspberryPi[4] and a number of sensors and actuators.

In addition, this book is not for novice programmers. It assumes at least an average level of programming skill as well as some basic knowledge of JS, as well as a good understanding of computer networks (Client/Server, IP networking) and Internet protocols (HTTP mainly). There is no requirement for specific electronics or hardware skills, although being familiar with I/O pins, digital and analog peripherals will help.

I hope that by the end of this book you will have learned new things about the IoT world and will have fun prototyping your IoT projects with JS.

Thank you for purchasing *Prototyping the Internet of Things with JavaScript*!

Charalampos

Trento, Italy

February 2015

---

[1] http://redmonk.com/sogrady/2015/01/14/language-rankings-1-15/

[2] http://nodered.org/

[3] http://tessel.io

[4] http://www.raspberrypi.org/

Part 1 JavaScript for the Internet of Things

The Internet of Things

There has been a much hype and talk over the last few of years surrounding the Internet of Things (IoT). We have seen many of startups trying to sell (sometimes with great success - see the SmartThings[5]) connected objects (lights, thermostats, gateways and the like), hardware platforms or services (data collection and visualisation, device management, etc.) for connected objects.

This chapter will provide a generic introduction of the Internet of Things and present you with the fundamentals of an IoT application.

So what is the Internet of Things?

I am sure you can find at least a dozen definitions explain exactly what is: ¡the Internet of Things¢. For the purpose of clarity, this book sees the IoT as a collection of technologies and notions that enables developers to connect objects with online services.

These would include, but not be limited to:

- Embedded systems and especially Microcontrollers (MCUs), System-on-Chip devices (SoCs) and networking modules
- Wireless communication networks (WiFi, Bluetooth, ZigBee, ZWave, RF, etc.) and Protocols (MQTT, CoAP, WebSockets, etc.)

Notions, including:

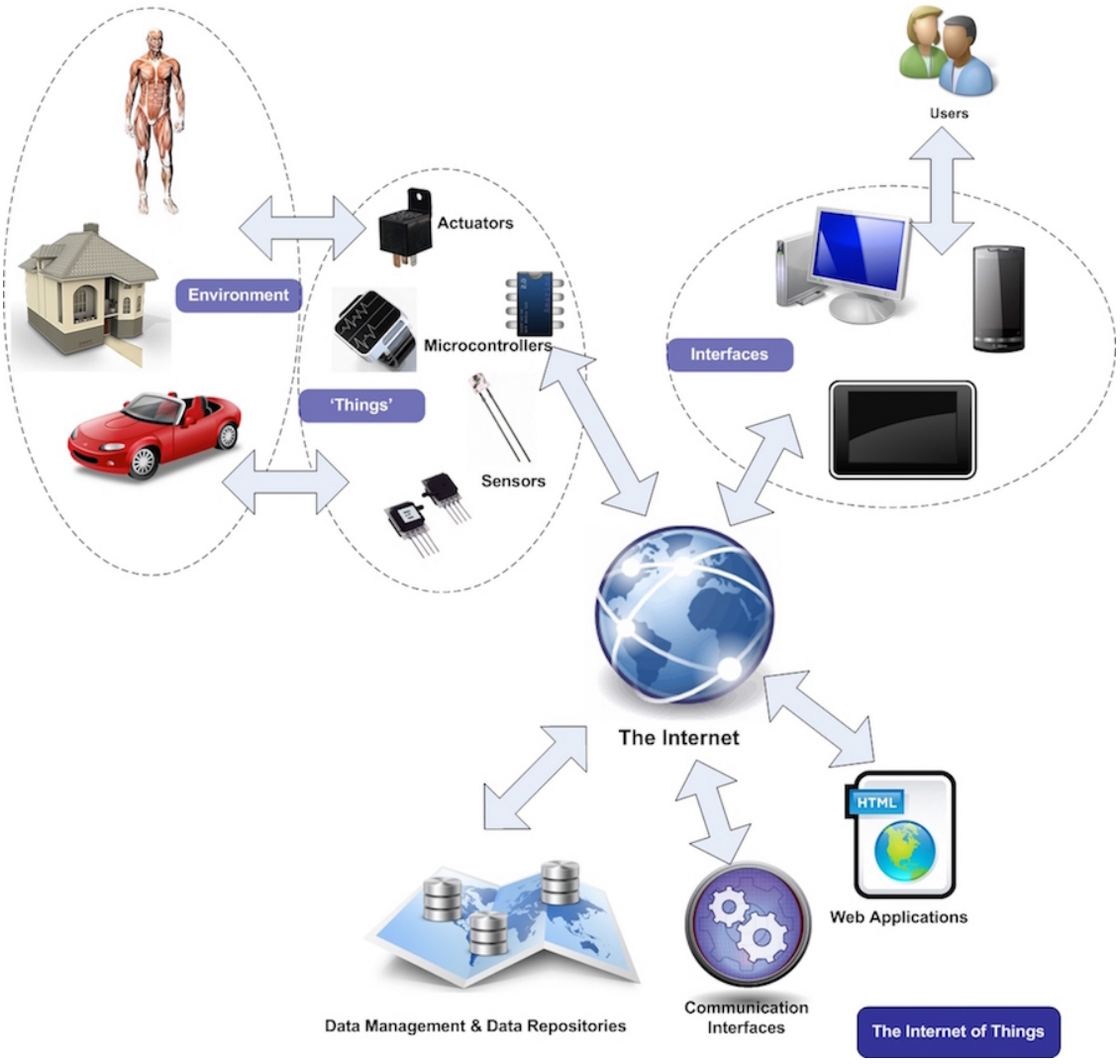
- Sensing the user context, transmitting that information and processing it
- Acting remotely on devices
- Adding sensing and acting capabilities to legacy devices (think of your
- Processing the device data, making smart decisions on behalf of the user and/or alerting them based on sepcific conditions

Connecting devices with online services makes sense for:

- Adding intelligence to ¡dumb¢ devices
- Combining device information with information from online services (such as social networks, weather channels, user profiles, etc.)

[5] http://www.smartthings.com/

- Enabling remote control



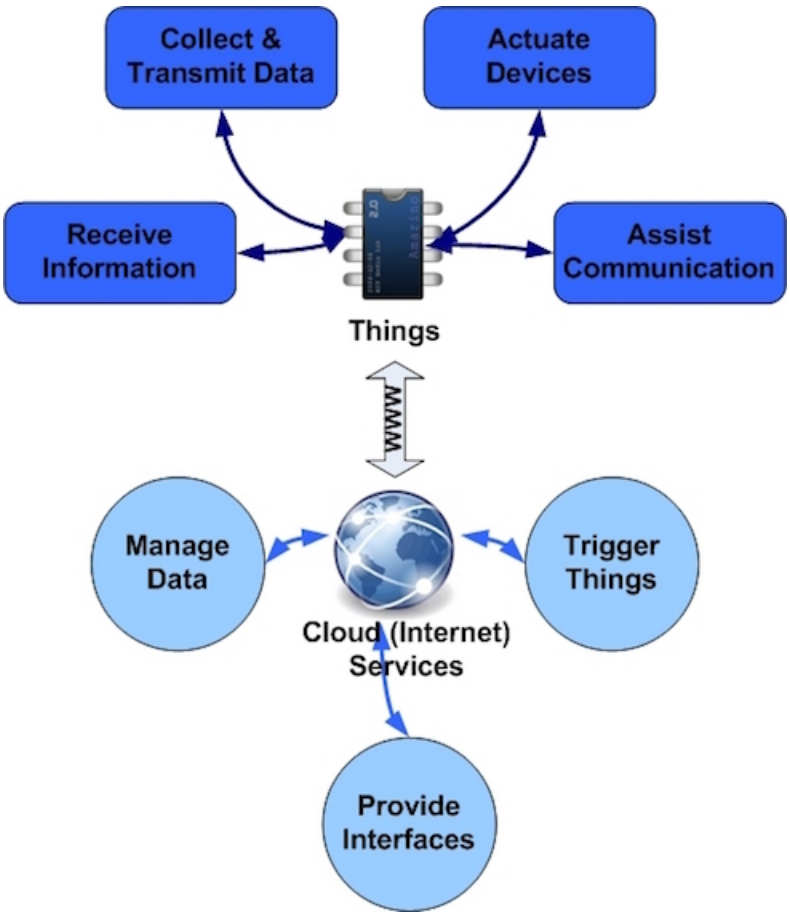
The Internet of Things

Basic components of the IoT

What do you need to build an Internet of Things solution? The answer to this question might be obvious to some of you, but let¢s take it down to the very basic elements of an IoT application:

- At least one physical device that has some sensing or actuating capabilities. This means that the device features a microcontroller and sensors and/or actuators.
- At least one communication module for that device that allows it to interact with other devices or online services.

- A gateway. There is no ubiquitous Internet in the air; your mobile phone connects to a gateway (base station) for voice or data connection over 3G/4G or a WiFi access point. In the same way, an IoT device needs a gateway to connect to the Internet. No matter what the communication technology is. Wired (Ethernet, KNX, etc.) or wireless (WiFi, ZigBee, Bluetooth), there is always a device part of a larger IP network that enables communication with the Internet. Most times you don¢t need to build yourself a gateway, but there are cases you might have to (if you are using for example Bluetooth or RF communication modules). This is covered in a later chapter in one of the case studies.
- At least one online application that interacts with that physical device. You may call it online service, cloud-based application, server application, back-end service, etc. This will be the software that runs on a centralised (or not) infrastructure and interacts with the connected device by collecting data, processing it and interfacing with the user.
- Users. There is no point in building an IoT project if there is no one to use it. This one goes especially for those who predict the millions, billions, zillions of connected devices that will be in use by 2020. *Finding users (thus good use cases) is not a topic covered by this book.*



Microcontrollers and Cloud within the IoT

Open Source Hardware for the IoT

Open Source Hardware (OSHW) is one of the main reasons IoT has exploded in popularity and availability in recent years. Making the Arduino[6] microcontroller platform available to the mainstream through simplifying the coding process and allowing people to build their own hardware version has two major impacts: a) prices went down and devices become much more affordable, b) companies started building and providing more hardware platforms to connect sensors and actuators with the Internet

The most popular OSHW for embedded development and IoT would be:

- Arduino family[7] that can be used either in combination with shields that provide networking connectivity or as boards with integrated networking abilities (e.g., the Arduino Yun[8])
- Spark Core[9], an ARM-based microcontroller with integrated WiFi connectivity that can be programmed in code similar to Arduino, over the Web.
- Pinoccio[10] boards, a wireless, web-ready microcontroller with WiFi, LiPo battery, & built-in radio. Suitable also for creating Mesh-networks[11].
- openPicus FlyportPro[12] modules that come with integrated WiFi, GPRS or Ethernet connectivity. Feature many IoT capabilities, like remote programming, secure communication, internal web server and low power modes.
- Tessel[13] a microcontroller with built-in WiFi that runs JavaScript and is Node-compatible. Tessel has a special place in this book; its own chapter about programming embedded devices with JavaScript.
- Libelium Waspmote[14] which is an open source wireless sensor platform especially focused on the implementation of low consumption modes to allow the sensor nodes (“motes”) to be completely autonomous and battery powered.
- XinoRF[15] an Arduino UNO R3 compatible electronics development board with an onboard 2-wayCiseco SRF data radio, which supports over-the-air programming.
- RFduino[16] a small, low cost, Arduino compatible, wireless enabled microcontroller.
- Espruino[17], currently the most affordable way to start with microcontrollers and JavaScript. This comes with a Web IDE for programming it without the need of special software and can be connected to the Internet using WiFi and other additional modules. In the next Chapter there is an overview of the Espruino platform.

[6] http://www.arduino.cc
[7] http://arduino.cc/en/Main/Products
[8] http://arduino.cc/en/Main/ArduinoBoardYun
[9] https://www.spark.io/
[10] http://pinocc.io/
[11] http://en.wikipedia.org/wiki/Mesh_networking
[12] http://www.openpicus.com/site/products
[13] https://www.tessel.io/
[14] http://www.cooking-hacks.com/index.php/documentation/tutorials/waspmote
[15] http://shop.ciseco.co.uk/xinorf-100-arduino-uno-r3-based-dev-board-with-radio-transciever/
[16] http://www.rfduino.com/
[17] http://www.espruino.com/

While there are many more embedded development boards, this list includes those that are more ¡IoT-ready¢ by supporting wireless communication modules and easier to implement. You can refer to this excellent resource[18] from Postscapes for more information on DIY embedded platforms.

Open Source Software for the IoT

When we talk about IoT and software, it can be software that runs a) on devices (connected objects or gateways) and b) back-end services. For both cases there is a great deal of open source software that you can rely on for prototyping your IoT application.

Summary

So far you have learned a few things about the Internet of Things concept, the main building blocks of an IoT project and open source hardware & software you can use to build your own projects. In the next chapter we dive into the world of JavaScript and see how it is connected with the IoT!

[18] http://postscapes.com/internet-of-things-hardware

JavaScript for the Internet of Things

After the introduction on the Internet of Things it is time to get into the main concept of this book: JavaScript and IoT. This chapter will introduce you to development boards such as the Espruino and the Tessel, which can be programmed directly in JS.

Why JavaScript ?

You may like JavaScript (and Node.js[19]) for its simplicity and the fast prototyping it offers and you may agree that Node can be used for building great server-side apps, but you may also wonder how JavaScript can be suitable IoT, since an IoT project requires coding both on devices and on back-end applications.

Here are a few arguments about why using JavaScript for prototyping and IoT application (both hardware and software) is the best approach to creating Iot devices:

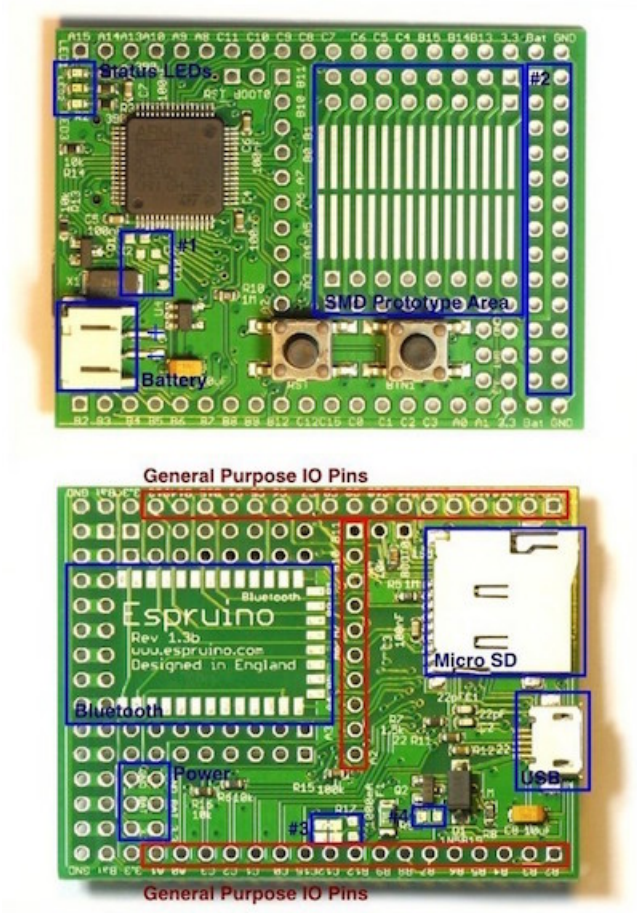
- IoT is event driven: A sensor needs to send some data based on some condition met (e.g., a timer expired or a threshold reached), or a user requests data, or some actuation needs to take place based on a similar condition. JavaScript is event-driven programming. You can build your application following the same principle as your physical device interacts with its context.
- JavaScript is the language of the Internet.. Web applications are built through JS (now also on the server-side), while cross-platform tools leverage on JS and allow developers to code once in JS and develop mobile applications for various platforms.
- You can use the same coding environment, libraries and style both for the devices and the back-end development. This is particularly useful when you are a beginner in the embedded world and/or you want to quickly prototype connected projects to test your ideas.
- There is huge community support with libraries and online resources for interacting with sensors and actuators, managing data, deploying your code into the cloud, etc. All in JavaScript.
- JavaScript is a mature and open language. There are no restrictions for special IDEs or coding environments in place, while frameworks and recent enhancements allow developers to deliver structured and efficient code.
- There is a great visual environment (Node-RED[20] built in Node, that you can use to quickly and easily create IoT workflows with devices communicating with online services, webpages, databases and more.
- There are already hardware platforms you can program directly using JavaScript. Explanations on both Espruino and the Tessel can be found in the following sections.

[19] http://en.wikipedia.org/wiki/Node.js

[20] http://www.nodered.org

Meet the Espruino board

The Espruino[21] is probably the first programmable microcontroller board available to the mainstream, that is programmed directly in JavaScript. Also, it is probably the most affordable option to begin with embedded JavaScript, as a typical Arduino board is set at an appealing price point.



The Espruino board - annotated. ***Image courtesy: Espruino***

The Espruino board comes with some numerous attractive features:

- Small (smaller than an Arduino Uno)
- 32-bit 72MHz ARM Cortex M3 CPU, 256KB of Flash memory, 48KB of RAM
- Micro USB connector
- Input Voltage range between 3.6v - 15v and a battery connector
- Built-in SD card connector
- 44 GPIO Pins (26 PWM Pins, 16 ADC Pins, 3 USARTs, 2 SPI, 2 I2C and 2 DACs)*

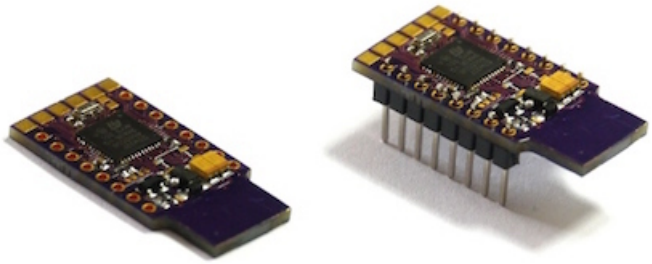
[21] http://www.espruino.com/

*This refers to the type of hardware devices (sensors, actuators and interfaces such as LCD screens as well as communication modules) you can connect to it. If you look at the modules[22] webpage of Espruino, you will see various sensors (accelerometers, ambient sensors, keyboards, motors and servo motors) and wireless modules (WiFI, RF, etc.).

One of the most attractive features of Espruino is the fact that programming the board can be done directly from your browser (through a Chrome plugin) so there is no requirement to install specific software. In addition, the Espruino acts as an interpreter, therefore your code does not need to be completed before it is uploaded allowing you to amend and alter the code as you go.

The Espruino Pico

Th Espruino Pico is a smaller version of the Espruino board; around the same size of a USB key.



The Espruino Pico. ***Image courtesy: Espruino***

It has similar features to the main board (though less GPIO pins available) and it is ideal (size-wise and power consumption-wise) for projects including 'wearables' or where size is critical. It also comes with integrated HTTP client and server and supports IoT protocols such as MQTT (more on that in later chapters).

At the time of writing, the Espruino Pico is under production and is scheduled to launch in May 2015.

Both the Espruino board and Espruino Pico, as well as the Espruino software are Open Source.

You will see examples of coding on the Espruino in the following chapters of this book. For more information on the Espruino you can visit the official website[23].

Meet the Tessel WiFi board

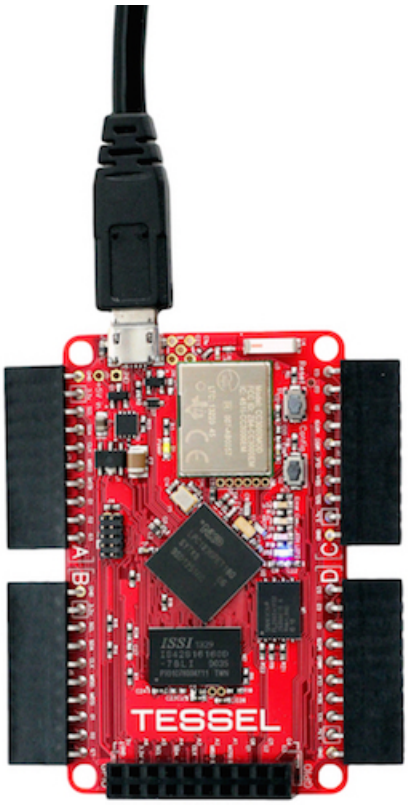
The Tessel[24] board is a microcontroller platform that can also be programmed in JavaScript. The main differences from Espruino - and its greatest features – is that it supports Node.js (at least

[22] http://www.espruino.com/Modules

[23] http://www.espruino.com/

[24] https://tessel.io/

10,000 Node packages are supported) and it comes with a WiFi module integrated. It could could be said that it is the first IoT-oriented hardware development platform for JavaScript.



The Tessel WiFi board. ***Image courtesy: TechnicalMachine***

Programming the Tessel soon feels as if you are writing Node.js code on your computer for communicating with the Internet. The only difference is that you use functions to control physical devices using the I/O pins of the board. As you might have expected, you need Node.js installed to program and configure the Tessel board.

The board's main features:

- 180mhz ARM Cortex-M3 LPC1830
- 32mb SDRAM
- 32mb Flash
- TI CC3000 WiFi radio
- 20-pin GPIO bank for general prototyping
- Micro USB or battery power

The Tessel also comes with its own hardware modules for sensing or controlling the physical world. While you can use I/Os pins that support all kinds of hardware communication you would need

(digital, analog, SPI, I2C and serial), the Tessel modules (camera, accelerometer, ambient sensing, servo controlling, GPRS, Bluetooth, etc.) that can be more easily plugged and come with their own software libraries, so integration becomes easier, especially for non-hardware experts.

More on the Tessel board will be presented in the following chapters.

JavaScript & IoT: What else is out there?

Software

Many frameworks and components have been recently developed that simplify the communication and management of devices.

The following list is inspired from Postscapes[25]:

- The Thing System[26]: A set of software components and network protocols for discovering and using IoT devices.
- noduino[27]: A simple and flexible JavaScript and Node.js Framework for accessing basic Arduino controls from Web Applications using HTML5, Socket.IO and Node.js.
- DeviceJS[28]: DeviceJS is a JavaScript based development platform for reacting to sensors and controlling devices. It's built on top of Node.js and a real-time JSON database. A DeviceJS application can run on a single device, or across many devices in different locations.
- duino[29]: An Arduino framework for Node.js.
- resin.io[30]: A great framework built on top of Node.js for managing devices (status and code) remotely as easy as using it.
- Johny-Five[31]: An Open Source, Firmata Protocol-based, IoT and Robotics programming framework for many devices and platforms (Arduino (all models), Electric Imp, Beagle Bone, Intel Galileo & Edison, etc.).
- Cylon.js[32]: A popular JavaScript framework for robotics, physical computing, and the Internet of Things. Supports platforms including the Arduino, the Beaglebone, the SparkCore in addition to drones, robots and many other devices.

[25] http://postscapes.com/javascript-and-the-internet-of-things
[26] http://thethingsystem.com/
[27] http://semu.github.io/noduino/
[28] http://devicejs.org/
[29] https://github.com/ecto/duino
[30] https://resin.io/
[31] https://github.com/rwaldron/johnny-five
[32] http://cylonjs.com/

Hardware

In addition to the Espruino and Tessel boards, there are (and will be continuously developed) boards that support programming directly in JavaScript.

- Kinoma Create[33]: JavaScript-powered construction kit for prototyping IoT applications. Comes with WiFi and Bluetooth integrated, as well as a color capacitive touchscreen and supports various sensors and actuators.

Summary

Using JavaScript for prototyping IoT applications is clearly a good way to go. So, jump into the next chapter for some hands on advice on how to use JS to develop you IoT device.

[33] http://kinoma.com/