**Interação com Hardware usando JavaScript**

[***RAFAEL SPECHT DA SILVA***](https://braziljs.org/blog/author/rafael-specht-da-silva/)

***Vídeo:*** [***https://www.youtube.com/watch?v=3qzi0VqCFqo&feature=youtu.be***](https://www.youtube.com/watch?v=3qzi0VqCFqo&feature=youtu.be)

**É isso mesmo que você leu!**

Já ouvimos falar que o JavaScript “roda em tudo”, certo? Existem diversos exemplos que provam que tal afirmação está correta, e hoje irei mostrar um deles. Inclusive, se contássemos o que anda acontecendo atualmente, faríamos alguém dos anos 90 rir muito. Primeiramente, existem frameworks e plataformas diferentes:

O [***CYLON.JS***](http://cylonjs.com/), por exemplo, suporta 43 plataformas desde Arduino, passando por alguns drones até Leap Motion.

Já o [***TESSEL.JS***](https://tessel.io/), é uma placa que roda Node.js e tem módulos plugáveis que permitem conexão WiFi, câmera, 3G e etc.

A diferença entre os dois acima é: No Cylon.js, rodamos o código em um computador que irá enviar dados via uma interface para a plataforma (vamos usar Arduino como exemplo para facilitar). Uma interface bem comum e barata é o cabo USB! Logo, iniciamos um processo Node.js que envia comandos ou recebe informações para o Arduino. Se paramos o processo ou removermos o cabo, encerramos essa comunicação. Isso porque o software estava rodando no computador e o Arduino não funciona stand-alone.

Já no Tessel, a implementação vai no chip da placa, eliminando a necessidade de uma conexão constante. Quem está acostumado a usar a linguagem C para fazer isso pode achar estranho, mas seria difícil ter todo o ambiente dentro do microprocessador. Nesse caso, o computador é usado para gravar e enviar o programa que escrevemos para a placa via USB. Uma vez que o cabo é removido, o código já está salvo e pode rodar sem ter um computador conectado.

No artigo “[***JAVASCRIPT BEYOND THE WEB***](http://www.sitepoint.com/javascript-beyond-web/)“, o autor fala sobre o JavaScript fora da web e apresenta diferentes plataformas e soluções.

Como temos várias opções para brincar, talvez seja difícil escolher uma para começar. Nesse post eu usarei um framework bem popular, com uma API clara e que oferece suporte a várias plataformas: o [***JOHNNY FIVE***](http://johnny-five.io/)!

Baseado no protocolo **Firmata**, o Johnny Five suporta plataformas como Arduino, Intel Galileo, Raspberry Pi (isso mesmo) e [***MUITO MAIS***](http://johnny-five.io/#platform-support). Dependendo da plataforma escolhida, a comunicação com um computador pode ou não ser necessária. Os exemplos demonstrados aqui usam um **Arduino Uno**!

***INSTALAÇÃO***

Teoricamente, basta instalar o módulo para que a mágica aconteça:

npm install johnny-five

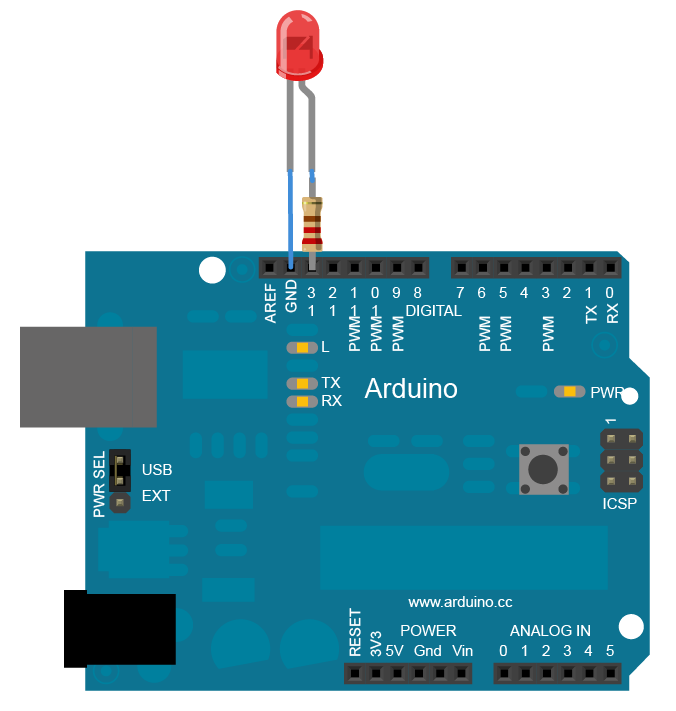
mas algumas dependências que lidam com compilação de módulos nativos de baixo nível, como o serial-port, podem causar um pouco de dor de cabeça dependendo do sistema operacional utilizado ou em versões mais novas do *Node.js*.

***O “HELLO WORLD” DO HARDWARE***

Como nem sempre a plataforma que estamos usando tem um monitor, o “Hello World” costuma ser um led piscando! Então vamos seguir a tradição e começar por ele. Para isso, vamos precisar de:

* O Arduino Uno (e o cabo de conexão USB)
* Um led
* Um resistor de 220 ohm

Na própria [***DOCUMENTAÇÃO DO JOHNNY FIVE***](http://johnny-five.io/api/led/) o exemplo não mostra esse resistor, mas para evitar que ele queime, seremos prudentes e o adicionaremos ao circuito como na imagem.



Se você nunca mexeu com circuitos eletrônicos fique tranquilo, pois o resistor funciona independente da maneira como é ligado (não tem um lado certo). Já o led tem que ser ligado de forma que a corrente elétrica circule de forma que aquela “seta” em seu interior fique apontando do pino onde está o resistor para o pino “gnd”.

Para permitir que o *Node.js* envie os comandos para o Arduino você deve fazer upload do protocolo Firmata para o microcontrolador:

1. Baixe a [***IDE DO ARDUINO***](https://www.arduino.cc/en/main/software)
2. Plugue o Arduino na USB do computador
3. Abra a IDE, depois abra o arquivo no caminho File > Examples > Firmata > StandardFirmata
4. Clique em upload!

Se não houve qualquer erro no upload, você conseguiu gravar o protocolo no microcontrolador e estamos prontos para usar o nosso amigo JavaScript! Vamos criar um arquivo blink.js na pasta onde instalamos o módulo Johnny Five:

// Como o johnny five é um módulo Node.js,

// ele é carregado como qualquer outro

var five = require('johnny-five');

// Instanciamos uma placa, que neste caso

// será a do Arduino que se comunicará com o computador

var board = new five.Board();

// O evento de ready é disparado quando a comunicação

// é estabelecida entre o processo Node.js e o Arduino

board.on('ready', function() { // Instanciamos um led no pino 13 var led = new five.Led(13);

// Chamamos o método blink do led que recebe

// a duração da fase que piscará em milissegundos

led.blink(500);

});

Basta rodarmos o script no terminal:

node blink.js

E pronto! Temos nosso “Hello World”. Podemos ver o exemplo rodando no vídeo abaixo:

***“SOCORRO, TEM UM LED NO MEU SERVIDOR”***

Agora que sabemos como fazer um led piscar e considerando que temos um computador que é responsável pelo processo *Node* que envia os comandos, podemos imaginar todas as aplicações que já escrevemos se comunicando com hardware!

Neste próximo exemplo vamos criar um servidor http que liga e desliga um led. Vamos usar a mesma configuração de hardware. A ideia é que através de uma url, o servidor mude o estado do led e assim toda vez que ele for acessado ligue ou desligue. Ou seja, com poucas linhas vamos conseguir acessar o hardware pela rede. Talvez isso explique porque algumas empresas estão usando JavaScript para seus dispositivos e frameworks ligados a essa tal “Internet das Coisas”.

// Vamos carregar os módulos johnny five e http var five = require('johnny-five'); var http = require('http');

// Instanciaremos uma placa

var board = new five.Board();

// isReady vai ser verdadeira quando a placa

// disparar o evento 'ready'

var isReady = false;

// isOn guarda o estado do led para sabermos

// se está aceso ou apagado

var isOn = false;

// Vamos criar uma variável global para o led

// dessa forma podemos acessá-lo de forma global

// sem precisar de muita complexidade

var led;

// quando a placa se conecta e está pronta...

board.on('ready', function() {

// instanciamos um led no pino 13

led = new five.Led(13);

// certificamos que o led estará desligado

led.off();

// setamos a variável que usamos para

// saber o estado atual da placa

isReady = true;

});

// Vamos criar um servidor http extremamente simples

// que escuta a porta 3000

http.createServer(function (req, res) {

// Em toda a requisição checamos a url

// acessada verificando se ela foi feita

// para a raiz do servidor...

if (req.url == '/') {

// chamamos a função que muda o estado do led toggleLed();

// encerramos a requisição com a variável isOn

// informando se o led está aceso ou não

// a concatenação com uma string se dá porque

// o método end precisa de uma string ou um buffer

res.end(isOn + '');

} else {

// caso a requisição não seja para a raiz

// encerramos a conexão sem fazer mais nada

res.end();

}

}).listen(3000);

console.log('listening at 3000');

// Função responsável por ligar e desligar o led

function toggleLed () {

// Se a placa não estiver pronta

// a execução não prossegue

if (!isReady) { return; }

// Se o led estiver ligado...

if (isOn) {

// o método off é chamado, desligando-o

led.off();

// a variável recebe false

isOn = false;

} else {

// se o led estiver desligado, o método on

// é chamado, ligando-o

led.on();

// a variável recebe true

isOn = true;

} }

Veja este exemplo rodando no vídeo abaixo:

Isso pode parecer banal e fácil, mas essa é a beleza de tudo isso. Usar conhecimentos prévios para criar um servidor e poder acender coisas na sua casa! Esse post foi introdutório e espero poder abordar mais sobre hardware aqui no portal BrazilJS. A documentação do Johnny Five contém muitos exemplos legais e espero que eles inspirem a descobrir mais usos. Imagine tocar uma sirene sempre que houver um erro no build ou piscar uma luz do escritório quando um merge para a branch master for feito. Espero ter mostrado algo novo para alguns e ficarei feliz em ajudar com qualquer dúvida 🙂

Até a próxima!