Mathematica-Node.js combination I/O on the RaspberryPi Zero

Node.jsのrpioによる外部機器の制御

Peripherals control using Node.js rpio

kobayashikorio@gmail.com
2018/2/28

MathematicaからNode.jsを使う理由

- **Node.js**はブラウザなしに単独で実行できる

  (1) V8 JIT により機械語レベルで高速に実行される

  (2) Node.jsはサーバーサイドのJavaScript実行環境である

- **Node.js**は**I/O**を非同期でかつ高速に実行できる

  (1) I/Oを実行しつつ、HTTPリクエストを受け付けることができる

  (2) Node.jsのI/OはMathematicaのI/Oより高速に実行できる

- **Node.js**は**node-rpio**を通じて、**GPIO, I2C, SPI**を制御できる

- **Mathematica**から**HTTP**関数等を通じて**Node.js**を制御することができる

RaspberryPiを使う理由

- 安価な**RaspberryPi**上で、高速にコードを実行できる可能性がある
- 各種コードを、システムクラッシュのリスクを考慮せずに、試用することができる
- **RaspberryPi, *Mathematica*, Node.js,** 各種外部機器の組合わせたシステムを、安価にかつ高速に実行できる可能性がある

Node.jsのインストール

- インストール手順

```
pi@raspberrypi:~ $ sudo apt install -y nodejs npm
pi@raspberrypi:~ $ sudo npm cache clean
pi@raspberrypi:~ $ sudo npm install n -g
pi@raspberrypi:~ $ sudo n stable
pi@raspberrypi:~ $ sudo npm update npm -g
pi@raspberrypi:~ $ node -v
v4.8.2
pi@raspberrypi:~ $ npm -v
5.7.1
```

- **rpio**のインストール

```
pi@raspberrypi:~ $ sudo npm install rpio
```

MathematicaからNode.jsにアクセスする方法

HTTP関数を通じてアクセスする手順

- **Node.js**で**web server**を動かす
- **Mathematica**の**HTTP**関数から**Node.js**にアクセスする
- **Mathematica**から**Node.js**にパラメータを送信し結果を受け取る

Mathematicaからパラメータを、サーバーに送り、サーバー側がこれを使うサンプル・プログラム

- サーバー側の**js**

```
var http = require (' http');
var url = require (' url');
var server = http.createServer ();

server.on (' request', function (req, res) {
var obj = url.parse (req.url, true); //parsing url parameter
    console.log (obj.query);
    res.writeHead (200, {' Content - Type' : ' text/plain'});
    res.write (' hello world');
    res.end ();});
server.listen (3000);
```

- **Mathematica**のプログラム
- 実行手順

ExternalEvaluateを通じてアクセスする

- **ExternalEvaluate**から**Node.js**にアクセスする

Node.js samples

- **sample 1: “hello” output**

  Node.jsの 1 行プログラムである

```
$ cat progam.js
 console.log(“Hello”)
$ node program.js
$ Hello
```

- **sample 2: GPIO pin on/off**

  RaspberryPi Zeroのピンを操作してLEDを点滅させるプログラム

```
var rpio = require(‘rpio’);
var LED_PIN_1 = 13; // GPIO27
 rpio.open(LED_PIN_1, rpio.OUTPUT, rpio.LOW);

for (var i = 0; i < 100; i++ ) {
   rpio.write(LED_PIN_1, rpio.HIGH);
   rpio.msleep(100);
   rpio.write(LED_PIN_1, rpio.LOW);
   rpio.msleep(100);
}
```

- **sample 3: asyncronous operation**

  Node.jsの非同期動作のサンプル・プログラム

```
var EventEmitter = require(‘events’).EventEmitter;

function asyncFunc() {
  var ev = new EventEmitter;
  console.log(‘in asyncFunc’);
  setTimeout(function () {
    ev.emit(‘done’, ‘foo’, ‘bar’);
  }, 1000);
  return ev;
}

var async = asyncFunc();
async.on(‘done’, function(arg1, arg2) {
  console.log(arg1, arg2);
})
var http = require(‘http’);
var url = require(‘url’);
var server = http.createServer();

server.on(‘request’, function(req, res) {

   var obj = url.parse(req.url,true);
```

```
    console.log(obj.query);

    res.writeHead(200, {‘Content-Type’ : ‘text/plain’});
    res.write(‘hello world’);
    res.end();
});

server.listen(3000);
```

■ **sample 4: LED flicker**

Node.jsとMathematicaを結合してLEDを点滅間隔をコントロールするプログラム

■ 実行手順

1. Node.js プログラムをコンソールから開始する（プログラム名が led.js であれば、$ node led.js）
2. *Mathematica* プログラムを実行する
3. URLExecute[req, {“val” -> 1000, “opt” -> “good”}] のval の値を変更すると、LEDの周期が変更される

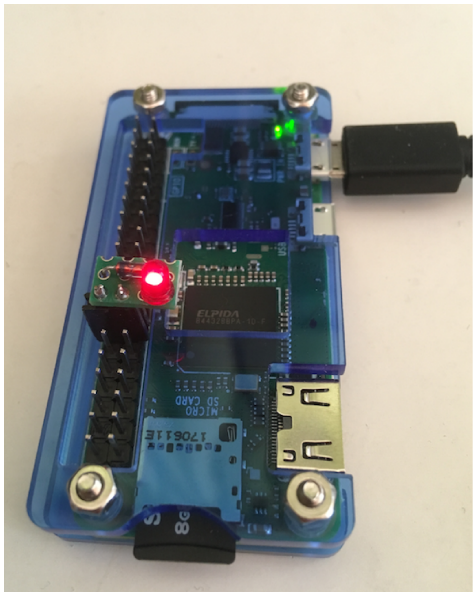
■ **Mathematica**

```
req = HTTPRequest[“localhost:3000”]
URLRead[req]
URLExecute[req, {“val” -> 1000, “opt” -> “good”}]
```

■ **Node.js**

```
//
var http = require(‘http’);
var url = require(‘url’);
var rpio = require(‘rpio’);

var Led = 13; // GPIO2
rpio.open(Led, rpio.OUTPUT, rpio.LOW);
var period=100;
var isOn=false;
var blink;
//
timer = {
    start:function(period) {
    blink= setInterval( function() {
    if(isOn) {
    rpio.open(Led, rpio.OUTPUT, rpio.LOW);
    isOn = false;
    }
    else {
    rpio.open(Led, rpio.OUTPUT, rpio.HIGH);
    isOn = true;
    }
   }, period);
   return blink;
   },

   change:function(newPeriod) {
   clearInterval(blink);

    blink= setInterval( function() {
    if(isOn) {
    rpio.open(Led, rpio.OUTPUT, rpio.LOW);
    isOn = false;
```

```
    }
    else {
    rpio.open(Led, rpio.OUTPUT, rpio.HIGH);
    isOn = true;
    }
   },newPeriod);
   return blink
  }
};
//
var blink=timer.start(period);

var server = http.createServer();
server.on('request', function(req, res) {
    var obj = url.parse(req.url,true).query;

    period=Number(obj.val);// get value of val parameter
    console.log(period);// val value
    var blink=timer.change(period);

    res.writeHead(200, {'Content-Type' : 'text/plain'});
    res.write('period changed');
    res.end();
});
server.listen(3000);
```



Node.jsによりLED点滅させておいて、Mathematica からその周期を任意のタイミングで変更する

reference

1. https://nodejs.org/api/http.html -> Node.js v9.6.1 Documentation
2. https://developer.mozilla.org/ja/docs/Web/JavaScript -> JavaScript reference in Japanese
3. node-rpio can control i2c -> https://github.com/jperkin/node-rpio
4. async -> https://github.com/caolan/async#queue
5. Async Functions -> https://tc39.github.io/ecmascript-asyncawait/
6. Pythonじゃなくてもok！ Raspberry Pi + JavaScript で「Lチカ」から「明るさ制御」 -> https://www.indetail.co.jp/blog/171002/
7. Raspberry Pi に Node.js をインストールする（nvm利用）-> https://make.kosakalab.com/make/electronic-work/nodejs_raspi/
8. ラズパイで手軽に作る！ドアロックセンサーの実現【第一回 AWS IoTの設定】-> https://www.indetail.co.jp/blog/161209/
9. node.jsでwebサーバを作ってみる-> https://qiita.com/ritukiii/items/7f28554369d63eb373c3
10. Node.jsでRaspberryPiのGPIOを良しなにする方法 -> https://qiita.com/koki_cheese/items/a4555d6ec3a32273cf36
11. callback -> JavaScript中級者への道【5. コールバック関数】 -> https://qiita.com/matsuby/items/3f635943f25e520b7c20
12. async/awaitをつかった非同期処理の直列実行 -> http://kitak.hatenablog.jp/entry/2016/06/17/101402
13. JavaScript Promiseの本 -> https://azu.github.io/promises-book/#introduction
14. JavaScriptでLEDを制御しよう -> https://ics.media/entry/10547/2
15. setInterval(function(),time) change time on runtime -> https://stackoverflow.com/questions/10576106/set-intervalfunction-time-change-time-on-runtime
16. Node書くならEventEmitterについて知っとくべし -> https://qiita.com/yuku_t/items/d69d3a2c7dafa7d04e87
17. spawn -> http://muddydixon.hatenablog.com/entry/20110708/1310134554