serialport public

Welcome your robotic javascript overlords.
Better yet, program them!

```
eeeee eeeee eeeee eeee        e   eeeee
8   8 8   88 8   8 8          8   8   "
8e  8 8    8 8e  8 8eee       8e  8eeee
88  8 8    8 88  8 88      e   88    88
88  8 8eee8 88ee8 88ee 88 8ee88 8ee88

eeeee eeee eeeee  e  eeeee e     eeeee eeee
8   " 8    8   8  8  8   8 8     8   8 8  8
8eeee 8eee 8eee8e 8e 8eee8 8e    8eee8 8
   88 88   88   8 88 88  8 88    88    8
8ee88 88ee 88   8 88 88  8 88eee 88    8eee
```


For all discussions, designs, and clarifications, we recommend you join our Gitter Chat room: **https://gitter.im/voodootikigod/node-serialport**

Version: 1.6.1 - Released March 5, 2015

Imagine a world where you can write JavaScript to control blenders, lights, security systems, or even robots. Yes, I said robots. That world is here and now with node-serialport. It provides a very simple interface to the low level serial port code necessary to program **Arduino** chipsets, **X10** wireless communications, or even the rising **Z-Wave** and **Zigbee** standards. The physical world is your oyster with this goodie. For a full break down of why we made this, please read **NodeBots - The Rise of JS Robotics**.

Robots, you say?

This library is admittedly a base level toolkit for building amazing things with real world (including robots). Here are a couple of those amazing things that leverage node-serialport:

- **Johnny-Five** - JavaScript IoT and Robotics programming framework.
- **Cylon.js** - JavaScript Robotics, By Your Command.
- **node-l8smartlight** (**source**) A node library to control the L8 Smartlight via Bluetooth or USB port
- **firmata** Talk natively to Arduino using the firmata protocol.
- **tmpad source** - a DIY midi pad using infrared, arduino, and nodejs. **Video**
- **duino** - A higher level framework for working with Arduinos in node.js.
- **Arduino Drinking Game Extravaganza** - AKA "The Russian" a hexidecimal drinking game for geeks by Uxebu presented at JSConf EU 2011.
- **Arduino controlling popcorn.js** - Controlling a popcorn.js video with an Arduino kit.
- **Robotic JavaScript** - The first live presentation of the node-serialport code set as presented at JSConf EU 2010.
- **devicestack** - This module helps you to represent a device and its protocol.
- **reflecta** A communication protocol that combines Arduino Libraries and NodeJS into an integrated system.
- **rc4pt-node** - Control Popcorntime with an Infrared receiver and Arduino.

For getting started with node-serialport, we recommend you begin with the following articles:

- **Arduino Node Security Sensor Hacking** - A great all around "how do I use this" article.
- **NodeBots - The Rise of JS Robotics** - A survey article of why one would want to program robots in JS.
- **Johnny-Five Getting Started Guide** - Introduction to using the

Dependents

gps-stream, ccxmleventemitter, node-buspirate, sp-socket, nodebotsdaymed13, breakout-server, s2serial, linux-lock-pi, dynanode, scop, motorola-ssi, gps, cubelets, iot-stack, arduino, sensordrone-bt, serialtft, irobot, espruino-cli, hkavr, arduinode, spheron, orangutan-instance, mindset-js-binary-parser, brickpi-coffeescript, teleinfo, node-espruino, fake-xbee-module, hackerchat, svd-xbee, microadsb, red-gate-coffee, ev3-nodejs-bt, rbn-arduino, nmea-gps-logger, elroy-pebble-driver, elroy-sphero-driver, zetta-pebble-driver, zetta-zigbee-photosensor-driver, noduino, zigbee, audica-radio, habithings, serial-port-stream, zetta-sphero-driver, uwcenterstack-hapticcontroller, skynet-hub-serial, xbee-promise, tanita-scale, eventduino, and 189 more

high level Johnny-Five library (awesome).

How To Use

Using node-serialport is pretty easy because it is pretty basic. It provides you with the building block to make great things, it is not a complete solution - just a cog in the (world domination) machine.

Special Notes

- Support for Node.js version 0.8.x has been removed. Version 1.4.0 is the last version that supported node.js version 0.8.x.
- Currently support for Node.js version 0.11.x is dealing with an issue in the latest version of v. 0.11.13. We have confirmed things are fine with 0.11.10 and earlier, but not 0.11.11+.

Good luck.

To Install

For most "standard" use cases (node v0.10.x on mac, linux, windows on a x86 or x64 processor), node-serialport will install nice and easy with a simple

```
npm install serialport
```

We are using **node-pre-gyp** to compile and post binaries of the library for most common use cases (linux, mac, windows on standard processor platforms). If you are on a special case, node-serialport will work, but it will compile the binary when you install. Follow the instructions below for how that works.

Installation Special Cases

This assumes you have everything on your system necessary to compile ANY native module for Node.js. This may not be the case, though, so please ensure the following are true for your system before filing an issue about "Does not install". For all operatings systems, please ensure you have Python 2.x installed AND not 3.0,

node-gyp (what we use to compile) requires Python 2.x.

Windows:

- Windows 7 or Windows 8.1 are supported.
- Install **Visual Studio Express 2013 for Windows Desktop**.
- If you are hacking on an Arduino, be sure to install **the drivers**.
- Install **node.js 0.10.x** matching the bitness (32 or 64) of your operating system.
- Install **Python 2.7.6** matching the bitness of your operating system. For any questions, please refer to their **FAQ**. Default settings are perfect.
- Open the 'Visual Studio Command Prompt' and add Python to the path.

Mac OS X:

Ensure that you have at a minimum the xCode Command Line Tools installed appropriate for your system configuration. If you recently upgraded the OS, it probably removed your installation of Command Line Tools, please verify before submitting a ticket.

Desktop (Debian/Ubuntu) Linux:

You know what you need for you system, basically your appropriate analog of build-essential. Keep rocking! Ubuntu renamed the `node` binary `nodejs` which can cause problems building `node-serialport`. The fix is simple, install the **nodejs-legacy package** that symlinks `/usr/bin/nodejs => /usr/bin/node` or install the more up to date nodejs package from **Chris Lea's PPA**.

```
# Ubuntu node
sudo apt-get install nodejs nodejs-legacy

# Or Chris Lea's PPA Node (more up to date)
sudo add-apt-repository ppa:chris-lea/node.js
sudo apt-get update
sudo apt-get install nodejs

sudo apt-get install build-essential
npm install serialport
```

Raspberry Pi Linux:

- Starting with a a vanilla New Out of the Box Software (NOOBS) Raspbian image (currently tested: 5/25/2013)
- Log into your Raspberry Pi through whatever means works best and ensure you are on a terminal prompt for the remaining steps. This could be local or through an SSH (or a serial connection if you like).
- Issue the following commands to ensure you are up to date:

```
sudo apt-get update
sudo apt-get upgrade -y
```

- Download and install node.js:

```
wget http://nodejs.org/dist/v0.10.16/node-
tar xvfz node-v0.10.16-linux-arm-pi.tar.gz
sudo mv node-v0.10.16-linux-arm-pi /opt/no
```

- Set up your paths correctly:

```
echo 'export PATH="$PATH:/opt/node/bin"' >
source ~/.bashrc
```

- Install using npm, note this will take a while as it is actually compiling code and that ARM processor is getting a workout.

```
npm install serialport
```

To Use

Opening a serial port:

```
var SerialPort = require("serialport").Serial
var serialPort = new SerialPort("/dev/tty-usb
  baudrate: 57600
});
```

When opening a serial port, you can specify (in this order).

1. Path to Serial Port - required.
2. Options - optional and described below.

The options object allows you to pass named options to the serial port during initialization. The valid attributes for the options object are the following:

- baudrate: Baud Rate, defaults to 9600. Should be one of: 115200, 57600, 38400, 19200, 9600, 4800, 2400, 1800, 1200, 600, 300, 200, 150, 134, 110, 75, or 50. Custom rates as allowed by hardware is supported.
- databits: Data Bits, defaults to 8. Must be one of: 8, 7, 6, or 5.
- stopbits: Stop Bits, defaults to 1. Must be one of: 1 or 2.
- parity: Parity, defaults to 'none'. Must be one of: 'none', 'even', 'mark', 'odd', 'space'
- buffersize: Size of read buffer, defaults to 255. Must be an integer value.
- parser: The parser engine to use with read data, defaults to rawPacket strategy which just emits the raw buffer as a "data" event. Can be any function that accepts EventEmitter as first parameter and the raw buffer as the second parameter.

Note, we have added support for either all lowercase OR camelcase of the options (thanks @jagautier), use whichever style you prefer.

open event

You MUST wait for the open event to be emitted before reading/writing to the serial port. The open happens asynchronously so installing 'data' listeners and writing before the open event might result in... nothing at all.

Assuming you are connected to a serial console, you would for example:

```
serialPort.on("open", function () {
  console.log('open');
  serialPort.on('data', function(data) {
    console.log('data received: ' + data);
  });
```

```
  serialPort.write("ls\n", function(err, resu
    console.log('err ' + err);
    console.log('results ' + results);
  });
});
```

You can also call the open function, in this case instanciate the serialport with an additional flag.

```
var SerialPort = require("serialport").Serial
var serialPort = new SerialPort("/dev/tty-usb
  baudrate: 57600
}, false); // this is the openImmediately fla

serialPort.open(function (error) {
  if ( error ) {
    console.log('failed to open: '+error);
  } else {
    console.log('open');
    serialPort.on('data', function(data) {
      console.log('data received: ' + data);
    });
    serialPort.write("ls\n", function(err, re
      console.log('err ' + err);
      console.log('results ' + results);
    });
  }
});
```

List Ports

You can also list the ports along with some metadata as well.

```
var serialPort = require("serialport");
serialPort.list(function (err, ports) {
  ports.forEach(function(port) {
    console.log(port.comName);
    console.log(port.pnpId);
    console.log(port.manufacturer);
```

```
  });
});
```

Parsers

Out of the box, node-serialport provides two parsers one that simply emits the raw buffer as a data event and the other which provides familiar "readline" style parsing. To use the readline parser, you must provide a delimiter as such:

```
var serialport = require("serialport");
var SerialPort = serialport.SerialPort; // lo

var sp = new SerialPort("/dev/tty-usbserial1"
  parser: serialport.parsers.readline("\n")
});
```

To use the raw parser, you just provide the function definition (or leave undefined):

```
var serialport = require("serialport");
var SerialPort = serialport.SerialPort; // lo

var sp = new SerialPort("/dev/tty-usbserial1"
  parser: serialport.parsers.raw
});
```

You can get updates of new data from the Serial Port as follows:

```
serialPort.on("data", function (data) {
  sys.puts("here: "+data);
});
```

You can write to the serial port by sending a string or buffer to the write method as follows:

```
serialPort.write("OMG IT WORKS\r");
```

Enjoy and do cool things with this code.

Reference Guide

Methods

SerialPort (path, options, openImmediately, callback)

Create a new serial port on `path`.

path

The system path of the serial port to open. For example, `/dev/tty` on Mac/Linux or `COM1` on Windows.

options (optional)

Port configuration options.

- `baudRate`
- `dataBits`
- `stopBits`
- `parity`
- `rtscts`
- `xon`
- `xoff`
- `xany`
- `flowControl`
- `bufferSize`
- `parser`
- `encoding`
- `dataCallback`
- `disconnectedCallback`
- `platformOptions` - sets platform specific options, see below.

Unix Platform Options

An object with the following properties:

- `vmin` (default: 1) - see `man termios`
- `vtime` (default: 0) - see `man termios`

openImmediately (optional)

Attempts to open a connection to the serial port on `process.nextTick`. The default is `true`. Set to `false` to manually call `open()` at a later time, but note you'll need to use factory error listener in the case of constructor errors.

callback (optional)

Called when a connection has been opened. The callback should be a function that looks like: `function (error) { ... }`

.open (callback)

Opens a connection to the given serial port.

callback (optional)

Called when a connection has been opened. NOTE: Will NOT be called if openImmediately is set to false as open will not be performed. The callback should be a function that looks like: `function (error) { ... }`

.isOpen()

Returns `true` if the port is open.

.write (buffer, callback)

Writes data to the given serial port.

buffer

The `buffer` parameter accepts a `Buffer` object, or a type that is accepted by the `Buffer` constructor (ex. an array of bytes or a string).

callback (optional)

Called once the write operation returns. The callback should be a function that looks like: `function (error) { ... }` *Note: The write operation is non-blocking. When it returns, data may still have not actually been written to the serial port. See `drain()`.*

.pause ()

Pauses an open connection.

.resume ()

Resumes a paused connection.

.flush (callback)

Flushes data received but not read. See `tcflush()` for Mac/Linux and `FlushFileBuffers` for Windows.

callback (optional)

Called once the flush operation returns. The callback should be a function that looks like: `function (error) { ... }`

.drain (callback)

Waits until all output data has been transmitted to the serial port. See `tcdrain()` for more information.

callback (optional)

Called once the drain operation returns. The callback should be a function that looks like: `function (error) { ... }`

Example

Writes `data` and waits until it has finish transmitting to the target serial port before calling the callback.

```
function writeAndDrain (data, callback) {
  sp.write(data, function () {
    sp.drain(callback);
  });
}
```

.close (callback)

Closes an open connection.

callback (optional)

Called once a connection is closed. Closing a connection will also remove all event listeners. The callback should be a function that looks like: `function (error) { ... }`

Events

.on('open', callback)

.on('data', callback)

.on('close', callback)

.on('error', callback)

Credit

A special thanks to **The Hybrid Group** for helping create and hosting the precompiled binaries for OSX, Linux, & Windows. This makes installing node-serialport much easier and much simpler for individuals using those platforms. Thank you.

You Need Help

Documentation

Support / Contact Us

Registry Status

Website Issues

CLI Issues

Security

About npm

About npm, Inc

Jobs

npm private modules

Blog

Twitter

GitHub

Legal Stuff

Code of Conduct

Package Name Disputes

npm License

Privacy Policy

Reporting Abuse

Trademark Policy