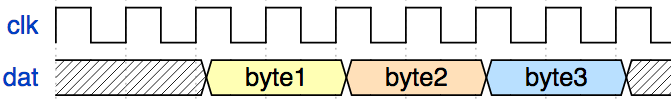
**Serial communication with NodeJS**

[arduino](http://thinkingonthinking.com/tags/arduino" \o "View posts tagged with \"arduino\")

Mar 2015

How can you talk to hardware peripherals, lights or robots from your laptop?

For many cases, the answer is: [Serial communication](http://en.wikipedia.org/wiki/Serial_communication). With serial communication, you can send bits and bytes “in series” over a wire. This often looks as in the figure below.



While you will always find a wire for data, the clock is optional in some communication schemes. The clock can be included for accurate timing of data signals. But in other communication schemes, the clock is left out. From operating system perspective, a nice overview is [here](https://www.freebsd.org/doc/en/articles/serial-uart/). This is possible when timing is defined on the sending and receiving ends, for example when data transmission should happen on 57600 bits per second.

Besides timing, serial communication is about configuration of wires. Data travels in one or multiple directions, usually on wires. But serial communication concepts often apply to wire-*less* communication too.

There are different forms of serial communication such as [RS-232](http://en.wikipedia.org/wiki/RS-232) or [USB](http://en.wikipedia.org/wiki/USB). In both cases, the core building block in a micro chip is an [UART](http://en.wikipedia.org/wiki/Universal_asynchronous_receiver/transmitter) (or “universal asynchronous receiver and transmitter”). This digital logic can be configured to different data formats and transmission speeds.

On your computer, an operating system connects to the UART with *a driver*. The driver takes care of inputs and outputs, and configuration of the signals (or [ioctl](http://en.wikipedia.org/wiki/Ioctl) in computing language). This means, if you want to talk to your Arduino from your laptop with NodeJS, your JavaScript actually talks to the driver for the serial port.

Sounds complicated? Maybe some code examples clear things up.

**The serialport library**

JavaScript and hardware - why would you combine both? There are some good arguments about [using JavaScript with hardware](http://www.voodootikigod.com/nodebots-the-rise-of-js-robotics/).

The physical world is accessible for JavaScript projects thanks to the [serialport module](https://www.npmjs.com/package/serialport) by [Chris Williams](https://github.com/voodootikigod).

Let’s create a fresh JavaScript project and add this module to it:

$ npm install --save-dev serialport

After the module has been downloaded, you are ready to start some serial communication fun.

**Opening a port**

Before any communication can happen, you must *open* the serial port. This is the start of every program. In the file “dump\_usb.js”, you write:

var serialport = require("serialport");

var SerialPort = serialport.SerialPort;

var serialPort = new SerialPort("/dev/cu.usbmodem14131", {

baudrate: 9600,

parser: serialport.parsers.readline("\n")

});

You must pass the port name, something like “COM1” on Windows or “/dev/ttyS0” on POSIX platforms. In case of my Arduino, it is “/dev/cu.usbmodem14131”.

Next, you must configure the port, mainly the transmission speed for sending and receiving bits. (Serial port configuration can also include things as flow control type, parity, stop bits and character size - this is out of scope for most Arduino projects).

For Arduino projects, a good choice for transmission speed is 9600 baud, because you can still watch incoming bits “live”. If there is a speed mismatch between your computer and your Arduino, you’ll see weird output.

As last argument for opening the port, you better set a *parser* for the incoming data. The default readline parser is nice to get line breaks out of the way.

**Prepare the Arduino**

Once the serial port is open, we have access to its data stream. The easiest thing to do is dumping data from the stream to the terminal.

Let’s have the Arduino send some data over with the [Serial library](http://arduino.cc/en/reference/serial). The following Arduino sketch “counter.ino” increases a counter, and sends the value over:

setup() {

Serial.begin(9600);

}

int i=0;

void loop() {

Serial.println(i++);

delay(100); // poll every 100ms

}

The last “delay” gives the laptop some time to process the data. Providing a delay is a good idea when you want to watch a data stream with human eyes instead of a CPU’s.

**Reading from the port**

To read bits from the port, you must add some more lines to the “dump\_usb.js” script:

var serialport = require("serialport");

var SerialPort = serialport.SerialPort;

var serialPort = new SerialPort("/dev/cu.usbmodem14131", {

baudrate: 9600,

parser: serialport.parsers.readline("\n")

});

serialPort.on("open", function () {

console.log('open');

serialPort.on('data', function(data) {

console.log(data);

});

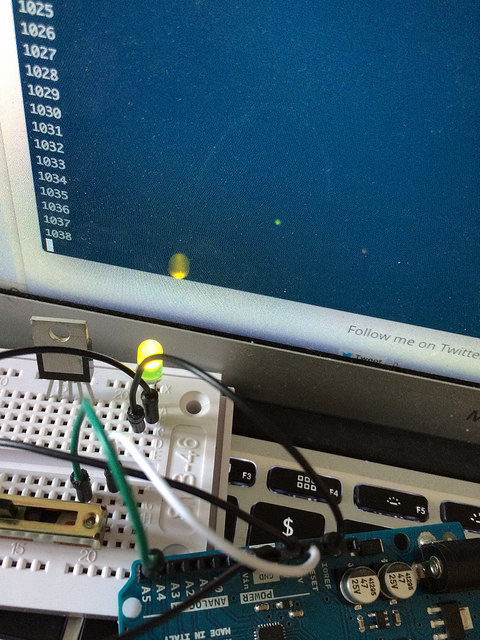
});

The last part is important: We listen to the *open* event from the driver. Then, we subscribe to its *data* events. The received bits are displayed to the screen with *console.log(data)*.

Let’s run this script:

$ node dump\_usb.js

If everything works out, you’ll see some numbers from the Arduino scrolling down on the screen. With a bit more advanced Arduino breadboard setup (see the screenshot), you can easily measure the room temperature or the position of a potentiometer.

[](https://www.flickr.com/photos/pmulder99/16623712038)

**Conclusion**

This post gave you some basic concepts for communication with an embedded device with NodeJS. Not too much happened at this stage. Just numbers scrolling down the screen. Things will get more fun in a future blog post, where we’ll connect the serial port to the browser by using a simple “polling” mechansism. This will enable us to write some simple web interfaces for hardware.

*Note: This post is the first post of a new series about JavaScript for embedded devices. This will be the topic of my new book. You can track its progress on the*[*book website*](http://embeddednodejs.com/chapters.html)*. Any feedback, suggestions and criticism is highly appreciated.*

[Leave me feedback](https://github.com/mulderp/mulderp.github.com/issues/new)

Follow me on Twitter [here.](http://twitter.com/mulpat)