Embedded Engineer Take-Home Challenge

The core features of our scientific IoT platform are the ability for researchers to easily record, organize, and share their instrument data. For this exercise, we ask that you implement two small components that demonstrate a repeating pattern of asynchronous communication between a hardware and a driver.

User Story

As a user, I'd like to have an interactive model illustrating how asynchronous serial communication works between a serial device and a driver, so that I can better understand what the driver does and how it works.

Concepts

The communication between instruments and our platform happens in various ways, such as a serial port communication, file-based exchange or network protocols. All methods have in common that a particular communication protocol needs to be implemented in our platform, to interpret the instrument communication.

The model should consist of two processes that communicate asynchronously (e.g. by writing to files, using IPC or means of your choice). Either process should write/read formatted ASCII, parse the result and print ongoing activity and intermediate state in a readable way to the user. It might be helpful to artificially delay communication to demonstrate what is happening.

MVP Task

Please develop two simple Node.js applications which satisfy the above user story.

Consider one part to be a device driver and performing a query command, and the other part to be a virtual instrument which simulates its state and gives a response.

Example

The following example, might be used as a starting point for the demonstrated communication.

The documentation excerpt below is from an analytical balance that is used to determine the weight of a sample. As analytical balances need to measure very small differences in weight they are extremely sensitive to motion. While the display of the balance will always show the current weight the balance also has routine to measure a so called *stable weight*. The stable weight is calculated internally over a small period of time allowing the balance to account for minimal fluctuations of the sensor data.

A measurement of the stable weight can also be triggered via the serial interface of the instrument. For that purpose an ASCII based text command is sent to the balance via a USB or RS232 port. This is the documentation for the command:

S – Send stable weight value

Command	**S**	Send the current stable net weight value.
Response	**S␣S␣WeightValue␣Unit**	Current stable weight value in unit actually set under unit 1.
	S␣I	Command not executable (balance is currently executing another command, e.g. taring, or timeout as stability was not reached).
	S␣+	Balance in overload range.
	S␣–	Balance in underload range.

Example

Command	**S**	Send a stable weight value.
Response	**S␣S␣␣␣␣␣␣100.00␣g**	The current, stable weight value is 100.00 g.

Comments

- The duration of the timeout depends on the balance type.
- To send the stable weight value in actually displayed unit, see 'SU' command in level 2
- The draft shield closes with this command, when the "Door function" is set on "Automatic". It opens after sending a stable weight.

As an example, the output of one ***potential*** implementation could look like this:

Expected Results

Output Process 2 (Driver)

```
$ Driver online
$ Sending Command: Send stable weight value
$ Sending: "S\n"
$ ...
```

Output Process 1 (Device)

```
$ Device online
$ Listening.
$ Received: [0x53] is "S"
$ Listening.
$ Received: [0x53, 0x0A] is "S\n"
$ Command recognized.
$ ...
```

Acceptance Criteria

This task is purposefully open-ended - you are free to come up with your own solution that satisfies the provided user story.

Please code the solution in JavaScript (ES6) using the Node.js framework.

You can get extra points for backing your application with unit tests and enhancing the driver process with a CLI to actively send commands instead of a fixed procedure.

Important:
We understand that not everyone can dedicate the same amount of time to this challenge - it is up to you to determine the amount of time you spend on the exercise. So that the reviewer understands how you are defining the scope of work, please clearly indicate your own “Definition of Done” for the task in a README file along with any other pertinent information.

Regardless of how far you take the solution towards completion, please assume you are writing production code because we will be reviewing as such.

Your solution should clearly communicate your development style, abilities, and approach to problem solving.

There is no time limit for the exercise, but once you receive the exercise please let us know when we should expect a response. Upon completion, please push your solution to GitHub and send us a link.

Let us know if you have any questions, and we look forward to seeing your approach.

Good Luck!