

Method 1: Communicating Using Web Sockets

Web Socket is a protocol which provides a full duplex(multiway) communication that is it allows communication in both directions simultaneously. **ws** is a simple to use, blazing fast, and thoroughly tested WebSocket client and server implementation.

There are 3 basic ws functions:

- `ws.onopen` : emitted when connected
- `ws.send` : sending a send event to websocket server
- `ws.onmessage` : event emitted when receiving message

Chalk is a library that provides a simple and easy to use interface for applying ANSI colours and styles to your command-line output. The Node.js module to help us accomplish custom formatting of messages

Installing

1. WebSocket module

```
npm install ws
```

2. Chalk module

```
npm install chalk
```

API Docs

1. [/doc/ws.md](#) for Node.js-like documentation of ws classes and utility functions.
2. [/chalk](#) For Chalk Module and its specifications.

Run Code (Folder: Web_Sockets)

1. Open 2 Terminals
2. Navigate to the folder where the `demo_server.js` and `demo_client.js` files are saved.
3. On one terminal run the following command (run Server file first):

node demo_server.js

4. On second terminal run the client file using the following command:

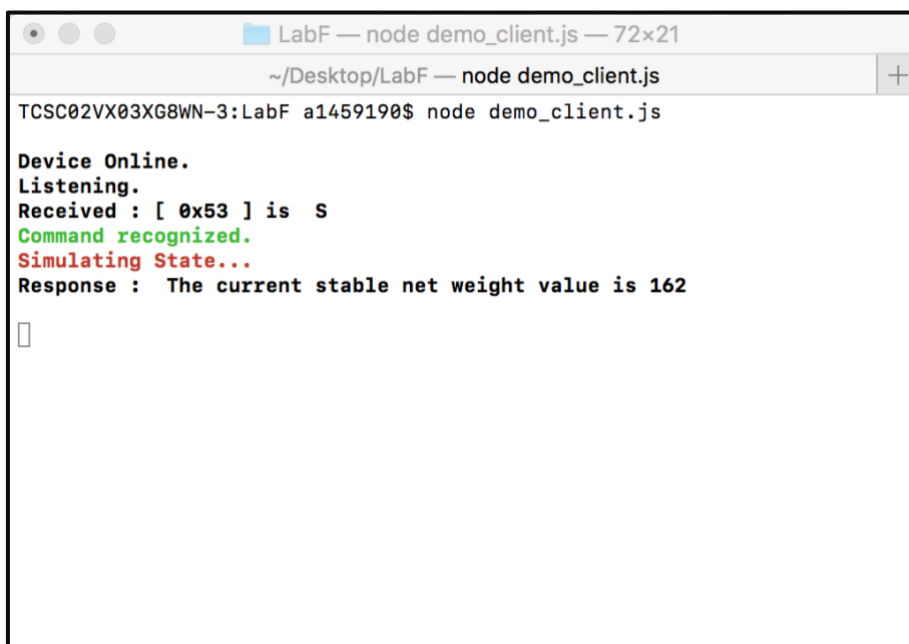
node demo_client.js

5. Terminal Output for the Server:

A terminal window titled "LabF — node demo_server.js — 72x22" with a path bar showing "~/Desktop/LabF — node demo_server.js". The output shows the server starting, receiving a command, and sending a response.

```
TCSC02VX03XG8WN-3:LabF a1459190$  
TCSC02VX03XG8WN-3:LabF a1459190$ node demo_server.js  
  
Driver Online.  
Sending Command : Send the current stable net weight value  
Sending : " S "  
  
█
```

6. Terminal Output for the Client:

A terminal window titled "LabF — node demo_client.js — 72x21" with a path bar showing "~/Desktop/LabF — node demo_client.js". The output shows the client starting, listening, receiving a command, and sending a response.

```
TCSC02VX03XG8WN-3:LabF a1459190$ node demo_client.js  
  
Device Online.  
Listening.  
Received : [ 0x53 ] is S  
Command recognized.  
Simulating State...  
Response : The current stable net weight value is 162  
  
█
```

Method 2: Communicating Using Node-IPC

Node-IPC a nodejs module for local and remote Inter Process Communication with full support for Linux, Mac and Windows. It also supports all forms of socket communication from low level unix and windows sockets to UDP and secure TLS and TCP sockets.

Chalk is a library that provides a simple and easy to use interface for applying ANSI colours and styles to your command-line output. The Node.js module to help us accomplish custom formatting of messages

Installing

3. WebSocket module

```
npm install node-ipc
```

4. Chalk module

```
npm install chalk
```

API Docs

3. [/node-ipc](#) for Node.js-like documentation of ws classes and utility functions.
4. [/chalk](#) For Chalk Module and its specifications.

Run Code (Folder : IPC)

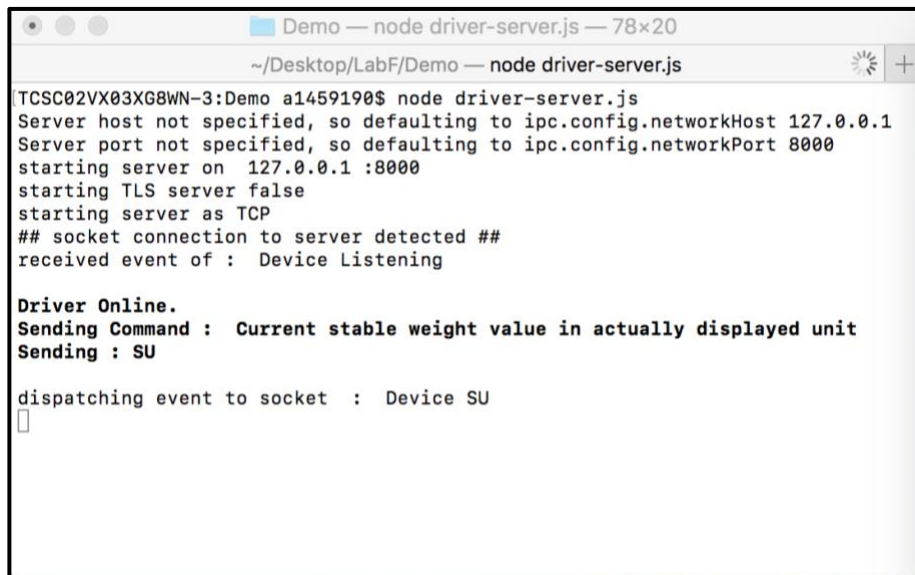
7. Open 2 Terminals
8. Navigate to the folder where the driver-server.js and device-client.js files are saved.
9. On one terminal run the following command (run Server file first):

```
node driver-server.js
```

10. On second terminal run the client file using the following command:

```
node device-client.js
```

11. Terminal Output for the Server:

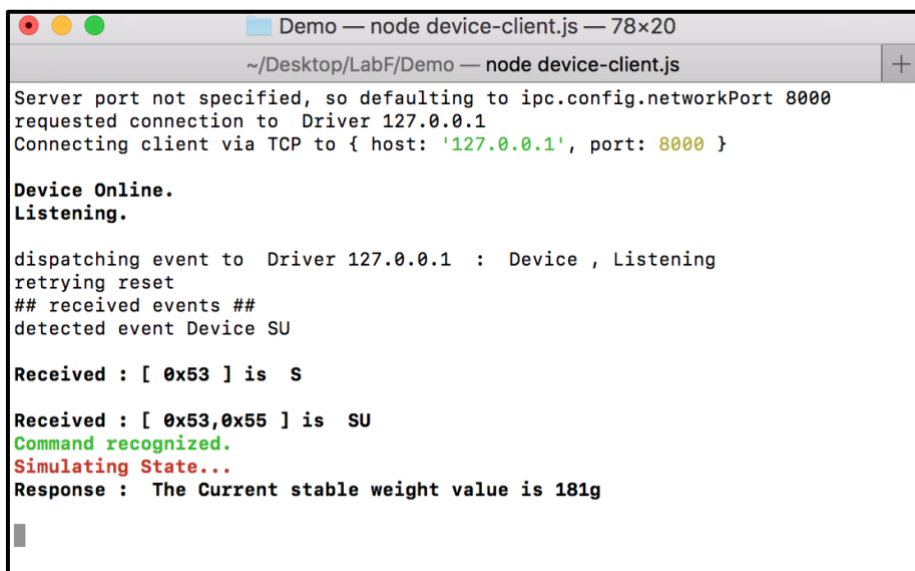
A terminal window titled "Demo — node driver-server.js — 78x20" with a path bar showing "~/Desktop/LabF/Demo — node driver-server.js". The output shows the server starting on 127.0.0.1:8000, detecting a socket connection, and sending commands to a device. The device responds with "SU".

```
TCSC02VX03XG8WN-3:Demo a1459190$ node driver-server.js
Server host not specified, so defaulting to ipc.config.networkHost 127.0.0.1
Server port not specified, so defaulting to ipc.config.networkPort 8000
starting server on 127.0.0.1 :8000
starting TLS server false
starting server as TCP
## socket connection to server detected ##
received event of : Device Listening

Driver Online.
Sending Command : Current stable weight value in actually displayed unit
Sending : SU

dispatching event to socket : Device SU
█
```

12. Terminal Output for the Client:

A terminal window titled "Demo — node device-client.js — 78x20" with a path bar showing "~/Desktop/LabF/Demo — node device-client.js". The output shows the client connecting to the server, receiving events, and sending commands. The server responds with "SU" and "The Current stable weight value is 181g".

```
Server port not specified, so defaulting to ipc.config.networkPort 8000
requested connection to Driver 127.0.0.1
Connecting client via TCP to { host: '127.0.0.1', port: 8000 }

Device Online.
Listening.

dispatching event to Driver 127.0.0.1 : Device , Listening
retrying reset
## received events ##
detected event Device SU

Received : [ 0x53 ] is S
Received : [ 0x53,0x55 ] is SU
Command recognized.
Simulating State...
Response : The Current stable weight value is 181g
█
```