Controlling Display-o-Tron with NodeJS

The Display-o-Tron 3000 and Display-o-Tron HAT can be used to add visual feedback to processes running on your Raspberry Pi. This is super-handy for headless servers or other cases where the Pi doesn't have a monitor connected to it.

This tutorial covers everything from the start to finish, including the installation of Raspbian, the recently released Node.js v4.x.x and Joris Vervuurt's JVSDisplayOTron module.

After the installation steps, we'll create a simple Node.js app that controls the Display-o-Tron's display and backlight.

Cool, right? Let's get started!

Step One: Installing Raspbian

1. Download the latest version of the Raspbian image:

   The latest version can be download via the following URL: https://downloads.raspberrypi.org/raspbian_latest

2. Install the downloaded image as described in the installation guide:

   For Linux users: https://www.raspberrypi.org/documentation/installation/installing-images/linux.md

   For Mac OS users: https://www.raspberrypi.org/documentation/installation/installing-images/mac.md

   For Windows users: https://www.raspberrypi.org/documentation/installation/installing-images/windows.md

3. Setup Raspbian:

   1. Insert the SD card into the Raspberry Pi

      Make sure to connect it to the internet using an Ethernet cable. Power up the Pi, give it some time to boot and lookup it's IP address using Angry IP Scanner or your preferred method.

      1. SSH into the Pi. The default password is `raspberry`.

      On your computer, enter the following command (replace the IP address with the IP address of your Pi): `ssh pi@192.168.x.x`

      2. It is now time to run the raspi-config tool to setup Raspbian:

      ```
      sudo raspi-config
      ```

      You'll want to at expand the filesystem, change the user password and set the correct timezone. Of course, you may change other settings like the memory split as well.

      When you exit raspi-config, you will be prompted to reboot the Pi. 4.

4. Update Raspbian:

   1. After the Pi has rebooted, SSH back into it using your newly set user password. Next, you'll want to update all installed packages:

   ```
   2. sudo apt-get update && sudo apt-get upgrade
   ```

   3. After the updates have been installed, you may want to reboot the Pi to make sure everything is running properly:

   ```
   4. sudo reboot
   ```

Step two: installing Node.js and npm

Make sure you have GCC 4.8 or layer

SSH back into the Pi and check the installed default version of GCC using the following command:

```
sudo gcc --version
```

In order to compile Node.js v4.x.x, GCC 4.8 or later is required. If the currently installed version is 4.8 or later, you're good to go. Otherwise, you'll need to install a newer version of GCC:

1. Using nano, open the sources.list file:

```
2. sudo nano /etc/apt/sources.list
```

   Replace "wheezy" with "jessie".

   Press Control + X, then press Y and confirm by pressing Enter.

3. Update the package list:

```
4. sudo apt-get update
```

5. Install GCC 4.8 or later (in this example, we'll install GCC 4.9):

```
6. sudo apt-get install gcc-4.9 g++-4.9
```

7. Using nano, open the sources.list file:

```
8. sudo nano /etc/apt/sources.list
```

   Replace jessie with wheezy.

   Press Control + X, then press Y and confirm by pressing Enter.

9. Update the package list:

   sudo apt-get update

Download & Compile Node 4.0.0

On your Pi, enter the following commands one-by-one.

You may want to update the version numbers (at the time of writing v4.0.0 is the most recent):

```
wget https://nodejs.org/dist/v4.0.0/node-v4.0.0.tar.gz

tar -xzf node-v4.0.0.tar.gz

cd node-v4.0.0

CC=gcc-4.9 CXX=g++-4.9 ./configure

CC=gcc-4.9 CXX=g++-4.9 make
```

You'll need to be patient. The compilation process takes quite some time. In my case - on a Raspberry Pi 2B - it took around two and a half hours. 😓 Once the compilation has finished...

Install Node.js and npm:

```
sudo make install
```

Update npm to the latest version

```
sudo npm install npm@latest -g
```

Hooray! You should now have a fully functional installation of Node.js v4.x.x and npm!

Step three: controlling a Display-O-Tron from Node.js!

1. Before you can control a Display-O-Tron from Node.js (or Python), a number of prerequisites need to be installed. This is super-easy using Pimoroni's installation script:

```
2. curl get.pimoroni.com/dot3k | bash
```

3. After all pre-requisites are installed, reboot your Pi:

```
4. sudo reboot
```

5. SSH back into the Pi and create a folder in the root of your Pi, that will contain all Node.js apps: `cd ~ && mkdir node_apps && cd node_apps`

6. Create a new folder that will contain the example Node.js app:

```
mkdir jvsdisplayotron_example && cd jvsdisplayotron_example
```

8. Create a new npm package.json file:

```
npm init
```

Enter the information as follows:

```
name: (jvsdisplayotron_example)
version: (1.0.0)
description: JVSDisplayOTron Example
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC) MIT
```

Press Enter and confirm by pressing Enter again.

10. Create the entry point file of the app:

```
touch index.js
```

12. Install the JVSDisplayOTron module and save it as a dependency of the app:

```
npm install jvsdisplayotron --save
```

Note that you'll probably see the following warnings:

```
npm WARN package.json jvsdisplayotron_example@1.0.0 No repository field.
npm WARN package.json jvsdisplayotron_example@1.0.0 No README data
```

You may ignore these warnings for now. If you're writing a module that you plan to publish to npm, make sure to add all required fields prior to publishing.

See https://docs.npmjs.com/files/package.json for more information.

14. Using your preferred editor, open the index.js file that has been created in the folder of the app:

Start by loading the JVSDisplayOTron module:

```
// Load the JVSDisplayOTron module.
var JVSDisplayOTron = require('jvsdisplayotron');
```

Next, we need to initialize the Display-O-Tron. If you want to control a Display-O-Tron 3000, use the following line of code:

```
// Initialize the Display-O-Tron 3000.
var dot3k = new JVSDisplayOTron.DOT3k();
```

If you want to control a Display-O-Tron HAT, use the following line of code:

```
// Initialize the Display-o-Tron HAT.
var dothat = new JVSDisplayOTron.DOTHAT();
```

Note that for this tutorial, I'm using a Display-O-Tron HAT.

We'll start by setting the contrast of the display to a better-readable value:

```
// Set the display contrast to a better-readable value.
dothat.lcd.setContrast(45);
```

Next, we'll write the text 'Hi from Node.js!' on the display:

```
// Write 'Hi from Node.js!' on the display.
dothat.lcd.write('HI from Node.js!');
```

Without support from Pimoroni and Kiwi Electronics, the JVSDisplayOTron module wouldn't exist. So... why not thank them by writing something on the display?:

```
// Write 'Thx to @pimoroni & @kiielectro' on the display, starting on the second line.

// Note that the text automatically wraps to the next line of the display.

dothat.lcd.setCursorPosition(0, 1);

dothat.lcd.write('Thx to @pimoroni & @kiwielectro');
```

Finally, we'll add some nice backlight colors:

```
// Add some nice backlight colors.

dothat.backlight.setLeftToRGB(255,0,0);

dothat.backlight.setLeftToRGB(0,255,0);

dothat.backlight.setLeftToRGB(0,0,255);
```

If no subsequent calls to JVSDisplayOTron are made, it is best to kill the Python process used by JVSDisplayOTron. The Boolean argument specifies whether the Display-O-Tron should be reset.

In this example, the Display-O-Tron will maintain it's state:

```
// To reduce resource usage, kill the JVSDisplayOTron process if no subsequent calls are made.

dothat.kill(false);
```

Save the index.js file and run the following command on the Pi:

```
cd ~/node_apps/jvsdisplayotron_example && node index.js
```

That's it! You should now see the Display-O-Tron come alive!

The JavaScript interfaces for controlling the Display-O-Tron 3000 and Display-O-Tron HAT are quite similar, but they're not identical.

Full documentation can be found here: https://github.com/jorisvervuurt/JVSDisplayOTron/tree/master/documentation

Code examples can be found here: https://github.com/jorisvervuurt/JVSDisplayOTron/tree/master/examples