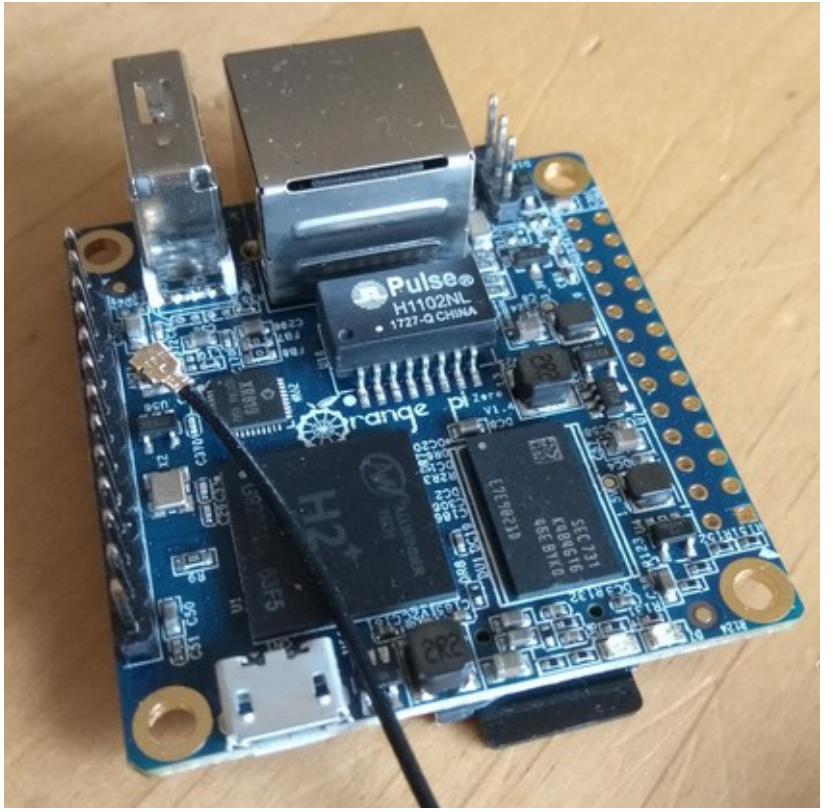
Node.js
on
(Raspberry) Pi

A crash course

Martin Berka

Raspberry Pi

- Open-source single-board computer with peripherals.
- RAM and processing power of a basic laptop.
- Hard drive is an SD card.
  - Also like a laptop – eMMC storage.



Orange Pi, photo by Andrew Berka

What can it (3B+) do?

- WiFi + Ethernet port
- Bluetooth
- 4 USB ports
- HDMI monitor output
- TRS audio jack
- 40 bare pins, inc. 5-volt/ground + 26 General Purpose I/O pins that can send/receive signals

All the things!

0: Equipment

- A Pi with WiFi (3, 4, or Zero-W) and whatever peripherals you plan on using. $10 (minimal) - $35.
- Good micro-SD card (Class 10 / 80MB/s or better, or your Pi will be slow). Sandisk is standard, 8GB Transcend Premium works. $7.50+.
- Good USB power supply, micro-USB for most Pi models. 3A recommended; laptop USB port good enough.

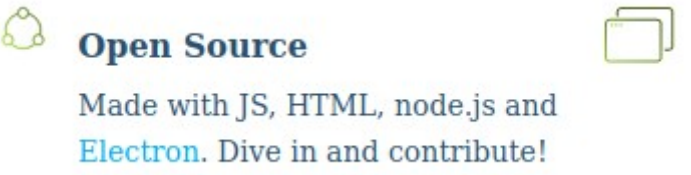
1: Get a Raspbian system image

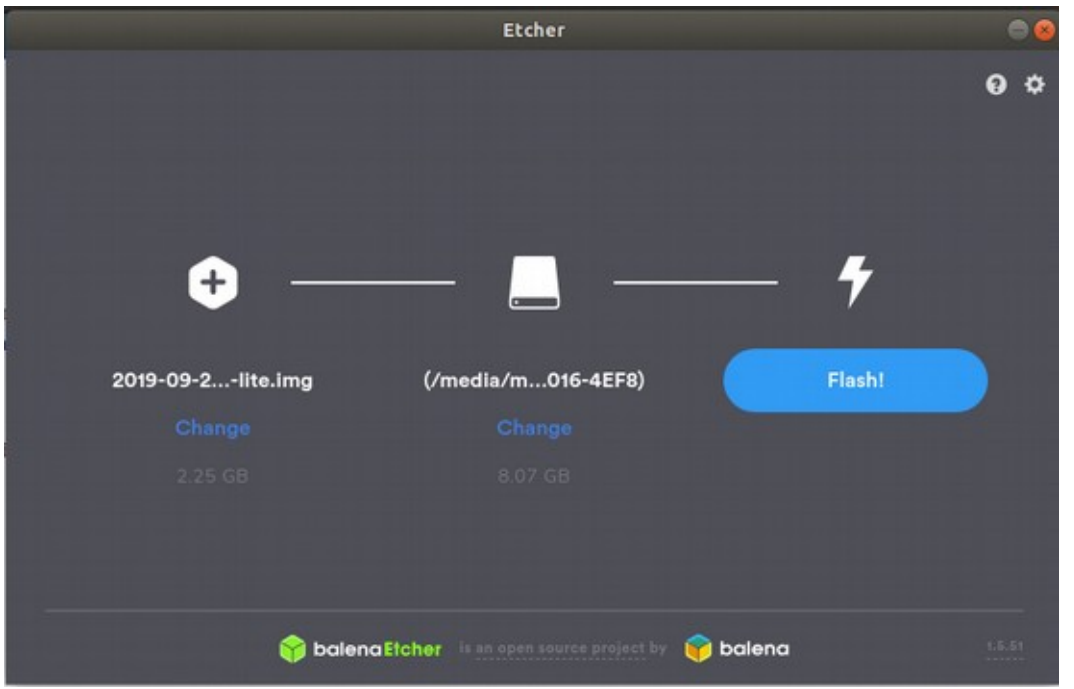
- For headless: **Comitup**, https://davesteele.github.io/comitup/
- https://www.raspberrypi.org/downloads/raspbian/ only if you want to use a keyboard+monitor every time you connect to a new WiFi network.
- Regular to use the GUI, Lite for headless / command line only (server).

2: Flash the micro-SD card

Easy: **Balena Etcher**
https://www.balena.io/etcher/

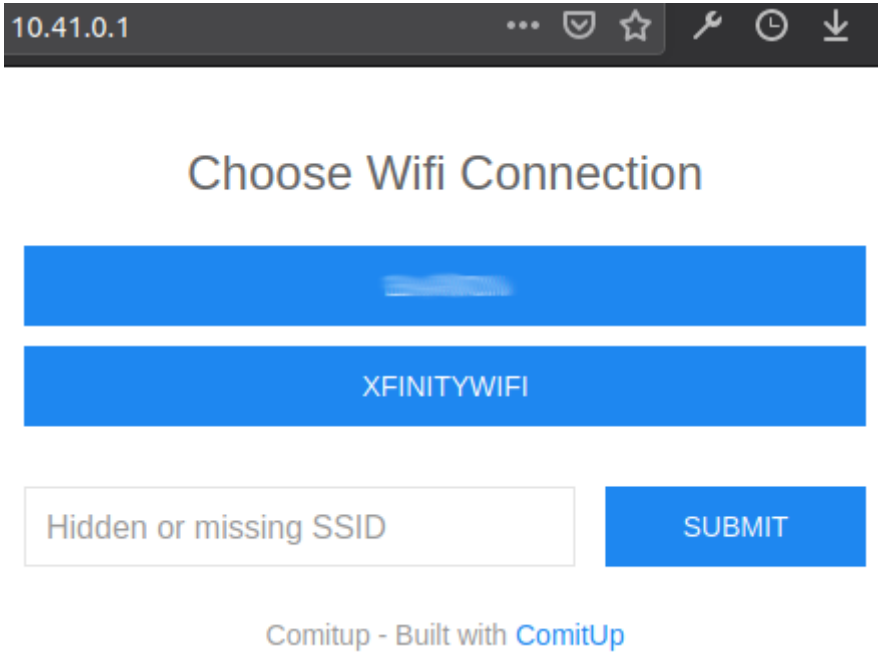
- Open, select the zip file (no unzip required)
- Insert SD card (auto-detected)
- Flash!, wait several minutes. Remove SD card when done.
- Insert card into Pi board and connect Pi to USB power.





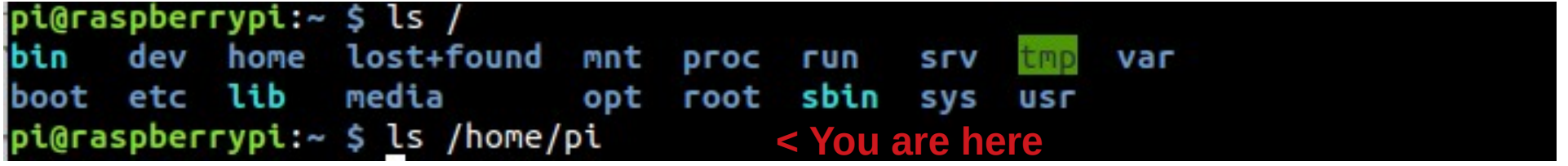
3. Connect Pi to WiFi

- Look for a network like comitup-## (e.g. comitup-82); connect to it.
- Go to the network sign-in page / 10.41.0.1
- Select network name, then enter password and confirm.
- Network should go away.



4. Sign into the Pi

- `ssh pi@raspberrypi.local` (or the Pi's local IP address). Password is raspberry.
  - `passwd` to change this. Good people set good passwords.



5. Install Node.js

From NodeSource:

```
curl -sL https://deb.nodesource.com/setup_12.x | bash -
sudo apt-get install -y nodejs
```

You can now enter/run Node.js with `nodejs` **or** `sudo node`. ctrl+d to exit Node (or twice to exit ssh).

```
pi@raspberrypi:~ $ nodejs
> const whatWeAreDoing = "PDXNode";
undefined
> for (let i = 0; i < 5; i++) console.log(whatWeAreDoing);
PDXNode
PDXNode
PDXNode
PDXNode
PDXNode
```

Demo

- Connect to the Alchemy Code Lab network.
- Go to raspberrypi.local
- Pi is running a modified version of the Express Hello World.

Playing Audio

- Easiest: mpg123 external library.

```
sudo apt-get install mpg123 -y
npm i child_process
```

- In Node:

```
import { exec } from 'child_process';
exec(`mpg123 ${__dirname}/someFile.mp3`);
```

Talking to USB devices

- Full control: use Johnny-Five, http://johnny-five.io/, to control all Arduino I/O and decision-making with Node.js.
  - We did this for NodeBots Day.

- Direct serial: exchange text/binary with an independent Arduino, power meter, etc.

Serial read/write example

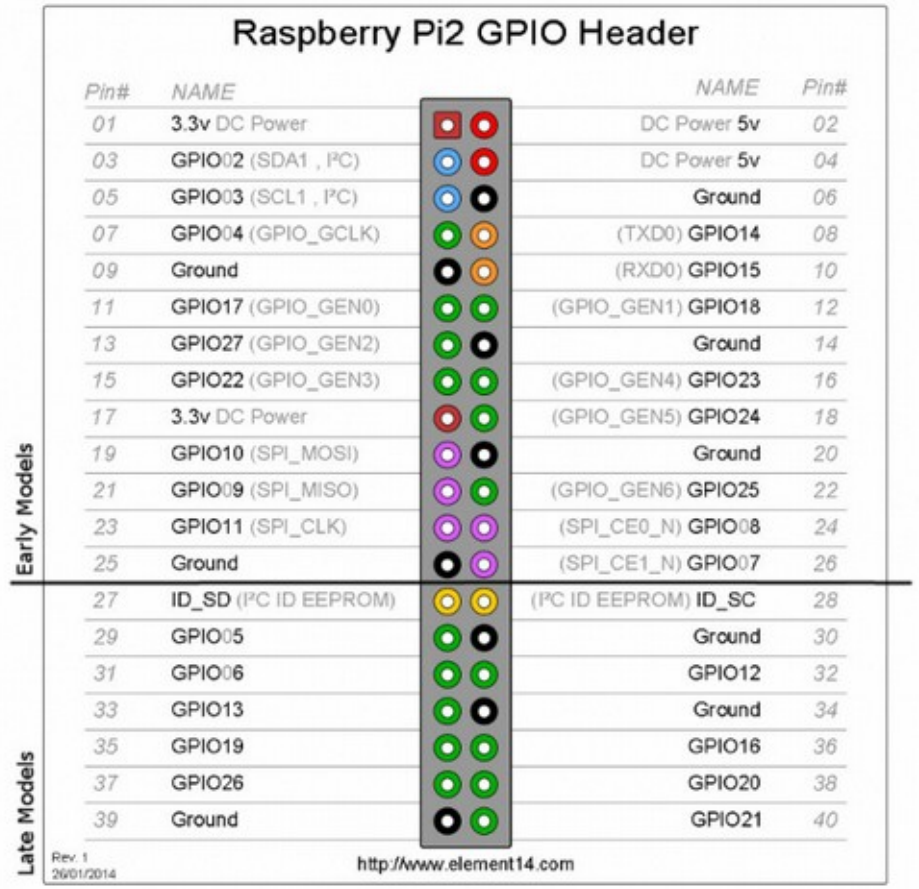
- `npm i –save serialport`
- Node.js:

```
const SerialPort = require('serialport');
const sp = new SerialPort('/dev/ttyUSB0');
sp.on('data', (message) => { // Message is a Buffer
  console.log(`Got ${message.toString()}`);
});
sp.write('PiDX Node');
```

- (connect USB device before running).

I/O pins

- GPIO = either output a high/low voltage, or read one
- RX/TX = serial communication like a USB port
- SPI = communicate with other devices



Sparkfun Tutorial: Raspberry gPIo, https://learn.sparkfun.com/tutorials/raspberry-gpio/gpio-pinout, Creative Commons BY-SA

I/O Applications

- Directly control anything that does not draw much current (< 16 mA).
- Switch on/off relays controlling power to larger devices.
- Communicate digitally via 1-wire or other protocols.

I/O Pin example

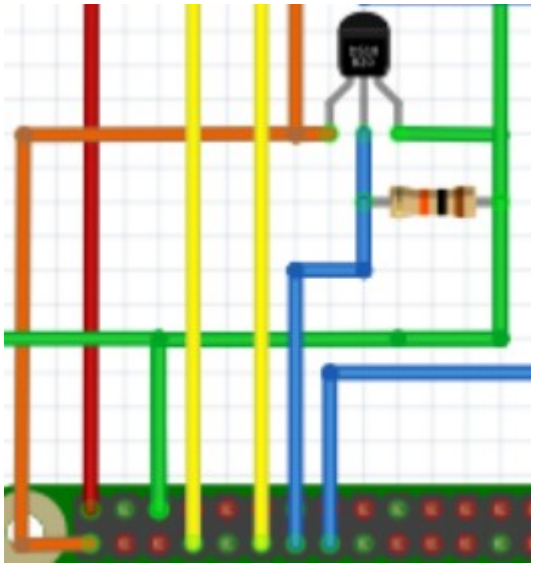
```
const { Gpio } = require('onoff');
const pin = new Gpio(4, 'out'); // output, GPIO4
pin.writeSync(0); // Set low
let toggle = 0;

setInterval(() => {
   toggle = 1 - toggle;
  pin.writeSync(toggle); // Set high
}, 1000);
```

Digital (1-Wire) example

For DS18B20 temperature sensor. In /boot/config.txt:

- dtoverlay=w1-gpio,gpiopin=4,pullup=on
- Node:

```
const promisify = require('util.promisify-all');
// Interfaces with temp sensor through 1-Wire
const ds18b20 = promisify(require('ds18b20'));
const sensors = await ds18b20.sensors(); // Get list
if (ids.length) {
  const temp = await ds18b20.temperature(sensors[0]);
  console.log(`T: ${temp} C`);
}
```



Forever and Ever

Same ways to keep Node running:

- Docker: safer, good for servers.

  ```
  docker pull arm32v6/node
  ```

- Forever.js: same auto-restart, easier access to hardware.
- Etc.

Getting around firewalls

- raspberrypi.local only exists on the local network.
- Exposing the Pi to the internet requires opening a hole in the firewall and setting up network address translation.
  - There are easier ways to run a server. Not recommended for IoT.
- MQTT (MQ Telemetry Transport) if you need to respond to messages from a server. MQTT.js: https://github.com/mqttjs/MQTT.js
- Reverse SSH tunnel to get into the Pi remotely.

Summary

- Minimal; big peripherals removed, ports present.
- Requires special steps to set up without screen/keyboard.
- Can control various ports with Node.js.
- Caveats around architecture and security.

Thank you!

Questions!

Martin Berka, martin@mber.dev

- Checkmark icon from the Noun Project, by Hrbon
- Node.js logo, Raspberry Pi image and Raspbian downloads image are from the linked sites of those open-source projects.
- All other unlabeled image are own works.
-