Wiring the Internet of Things with Node-RED
Hello,

I’m Nick O’Leary

IBM Developer Advocate
Node-RED Project Lead
@knolleary
Node-RED
Flow-based programming for the Internet of Things

- Browser-based editor
- Node.js runtime
- Extendable palette
2013: Created by IBM Emerging Technology Services and open-sourced later that year.

2015: Added to the default Raspberry Pi image.

2016: Became a founding project of the JS Foundation.

2018: Hit 1,000,000 installs
Flow-based Programming

• Invented by J. Paul Morrison at IBM in the early 1970’s

• A network of asynchronous processes communicating by means of streams of structured data chunks

• Each process is a black box – it doesn’t know what has come before it, or what comes after it; it just acts on the data it receives and passes the result on
Why Node-RED?

- Editing words
- Manipulating numbers
- Arranging ideas
topic: "weather/uk"
payload: "sunny!"
payload: "Jedi are totally amazing!"

sentiment: { score: 4 }
module.exports = function(RED) {
    "use strict";
    function RandomNode(n) {
        RED.nodes.createNode(this, n);
        this.low = Number(n.low || 1);
        this.high = Number(n.high || 10);
        this.inte = n.inte || false;
        this.property = n.property || "payload"
        var node = this;
        this.on("input", function(msg) {
            var value;
"keywords": [ "node-red", "random" ],
"node-red" : {
    "nodes" : {
        "random": "random.js"
    }
},
package.json

```
"keywords": [ "node-red", "random" ],
"node-red": {
  "nodes": {
    "random": "random.js"
  }
},
```

From the edge to the cloud

Easily-installed on the default Raspberry Pi image, Node-RED can be used out of the box to begin creating IoT applications.

Available in the IBM Cloud catalog as a Quick Start application, it takes moments to create cloud applications that combine services from across the platform.

Easily installed on other cloud platforms.
Embedded in commercial devices

MultiConnect® Conduit™

MobiusGateway
MobiusFlow

OPTO 22 SNAP PAC Industrial Controller

Node-RED
Tooling in the Cloud

HITACHI
LUMADA

SIEMENS
MindSphere
Visual Flow Creator

sense tecnic

Node-RED
Particle

IoT RULES ENGINE beta

Get to market in a fraction of the time with a drag-and-drop IoT application builder

Node-RED
• Bluetooth Low Energy
• Espruino JavaScript interpreter pre-installed
• nRF52832 SoC - 64MHz ARM Cortex M4, 64kB RAM, 512kB Flash
• 8 x 0.1” GPIO
• 9 x SMD GPIO
• Silicone cover with tactile button
• MAG3110 Magnetometer
• IR Transmitter
• Built in thermometer, light and battery level sensors
• Red, Green and Blue LEDs
• NFC tag programmable from JavaScript
• Pin capable of capacitive sensing

Raspberry Pi + SenseHAT

• 8×8 RGB LED matrix
• Five-button joystick
• Gyroscope
• Accelerometer
• Magnetometer
• Temperature
• Barometric pressure
• Humidity
At this point of the talk, I switch to a demo.

The demo walks through the basics of Node-RED using a Pi and SenseHAT.

It shows how the sensor data from the SenseHAT can be used to control the display on the SenseHAT.

It also shows connecting to the device to the IBM Watson IoT Platform to share its data.
Node-RED Projects Feature

- A project is a set of files that represent a complete, redistributable Node-RED application.

- The Runtime runs a single project at any one time, but the Editor provides an easy way to switch between projects.

- A project exists locally on disk, but can be linked to a remote git repository
Puck.js

- Espruino JavaScript interpreter
- Bluetooth Low Energy
- nRF52832 SoC - 64MHz ARM Cortex M4, 64kB RAM, 512kB Flash
- 8 x 0.1" GPIO
- Silicone cover with tactile button
- MAG3110 Magnetometer
- IR Transmitter
- Built in thermometer, light and battery level sensors
- Red, Green and Blue LEDs
- NFC tag programmable from JavaScript
- Pin capable of capacitive sensing
A simple example of using a Puck.js button as a remote control.

Node-RED
At this point of the talk, I switch to the next demo.

I cloned the existing Node-RED project from my GitHub repo.

This included setting up the Puck.js BLE device and the code on the Pi to detect when the Puck’s button was pressed.

The flow then controlled the colour of the SenseHAT to match the colour of the Puck passed on the number of clicks.
IBM Cloud

{ 
  temperature: 30,
  humidity: 61,
  pressure: 1010
}

{ }
IBM Cloud

{ temperature: 30,
humidity: 61,
pressure: 1010
}
IBM Cloud

{ temperature: 30, humidity: 61, pressure: 1010 }

{ temperature: 30, humidity: 61, pressure: 1010 }

{ temperature: 30, humidity: 61, pressure: 1010 }

git push resin master
{  
  "name": "node-red-resin-demo",
  "description": "A Node-RED Project",
  "version": "0.0.1",
  "dependencies": {
    "node-red": "0.19.*",
    "node-red-contrib-ibm-watson-iot": "0.2.8",
    "node-red-node-pi-sense-hat": ">=0.0.18",
    "puck-remote": "github:knolleary/puck-remote"
  },
  "scripts": {
    "start": "node --max-old-space-size=160 ./node_modules/node-red/red.js --userDir . --settings ./settings.js flows.json"
  },
  "node-red": {
    "settings": {
      "flowFile": "flows.json",
      "credentialsFile": "flows_cred.json"
    }
  }
}
The last part of the demo was showing how pushing the changes to a git repo on Resin.io would then push the app out to the other Raspberry Pis on stage.
module.exports = {
    verbose: true,
    uiPort: 80,
    credentialSecret: process.env.NODE_RED_CREDENTIAL_SECRET
};
FROM resin/raspberrypi3-node:8-slim

RUN apt-get update && apt-get install -yq \
    python3=3.4.2-2 sense-hat raspberrypi-bootloader i2c-tools build-essential \
    libssl-dev libffi-dev libyaml-dev python3-dev python3-pip python-rpi.gpio git &
    pip3 install sense-hat rtimulib pillow

WORKDIR /usr/src/app

COPY package.json package.json

RUN JOBS=MAX npm install --unsafe-perm && npm cache clean --force && rm -rf /tmp/*

RUN apt-get remove build-essential libssl-dev libffi-dev libyaml-dev \
    python3-dev python3-pip git \
    && apt-get autoremove && apt-get clean && rm -rf /var/lib/apt/lists/*

COPY . ./

ENV INITSYSTEM on

CMD ["npm", "start"]
# Node-RED Demo Dashboard

This image shows the dashboard for a Node-RED demo application. The dashboard is accessed via the URL `https://dashboard.resin.io/apps/1265406/devices`. The application is named `NodeREDDemo`.

## Devices Overview

The dashboard displays the status, name, last seen, UUID, OS version, OS variant, supervisor version, IP address, and release for each device.

### Device Details

<table>
<thead>
<tr>
<th>Status</th>
<th>Name</th>
<th>Last Seen</th>
<th>UUID</th>
<th>OS Version</th>
<th>OS Variant</th>
<th>Supervisor Version</th>
<th>IP Address</th>
<th>Release</th>
</tr>
</thead>
<tbody>
<tr>
<td>Online</td>
<td>white-snow</td>
<td>Currently online (for 42 minutes)</td>
<td>5b6d63</td>
<td>Resin OS 2.15.1+rev2</td>
<td>development</td>
<td>7.16.6</td>
<td>9.69.252.157</td>
<td>6eceb98</td>
</tr>
<tr>
<td>Online</td>
<td>divine-pine</td>
<td>Currently online (for an hour)</td>
<td>4f916f</td>
<td>Resin OS 2.15.1+rev2</td>
<td>development</td>
<td>7.16.6</td>
<td>9.246.41.179</td>
<td>6eceb98</td>
</tr>
</tbody>
</table>

The application's status is 100% with the version `6eceb98`.

---

35
This timeline proposes how the features will be staged across releases. The precise timings of these releases is not yet known.

The 0.21 release may split into 2 or more sub releases prior to 1.0.
Feature: Split Editor/Runtime Packaging

- Split the packaging of node-red into multiple packages.

- The existing node-red package will become a meta-package that pulls them together to deliver Node-RED as it is used today.

- The internal modules will be published under the @node-red organization on npm.
Feature: Split Editor/Runtime Packaging

- `@node-red/editor-api` - provides an Express application that can be used to serve the Node-RED editor.

- `@node-red/editor-client` - provides all of the client-side resources of the Node-RED editor application.

- `@node-red/nodes` - provides all of the core Node-RED nodes.

- `@node-red/registry` - provides the node registry, responsible for discovering and managing the node modules available to the Node-RED runtime.

- `@node-red/runtime` - provides the core flow engine of Node-RED. It is the main entry point for the runtime.

- `@node-red/util` - provides common utilities shared by the Node-RED components, including logging and i18n.

- `node-red` - the existing package that pulls the above packages together and delivers exactly the same experience as it does today.
Thank You

https://nodered.org

@knolleary
https://medium.com/@knolleary