MQTT and Node-RED and the Internet of Things

Kirk Carlson
Retired – long time Unix/Linux user
Kirk.Carlson@att.net
github.com/kirkcarlson

MQTT

- Message Queueing and Telemetry Transport
- Invented at by Andy Stanford-Clark (IBM) and Arlen Nipper (Cirrus Link) in 1999.
- A light weight protocol for telemetry and control
- While queuing is historically significant, it isn’t necessary for the Internet of Things

MQTT Node Types

- Client
  - Is an end point that produces or consumes messages
- Broker
  - Acts as a hub to accept and distribute messages
  - Each client typically connects to one broker
  - A broker can connect to many clients

MQTT Operations

- Uses a Publish-Subscribe or Pub-Sub model
- Data producers publish data to a broker
- Data consumers subscribe to data from a broker
- There is some overhead to establish and release the connections to a broker so connections are normally long lived.

MQTT Common Operations

- CONNECT
- CONNACK
- PUBLISH
- SUBSCRIBE
- SUBACK

MQTT Less Common Operations

- PUBACK, PUBREC, PUBREL, PUBCOMP
- UNSUBSCRIBE
- UNSUBACK
- PINGREQ
- PINGRESP/PING
- DISCONNECT

MQTT PUB Encoding Basics

- Topic (UTF-8 string, no wild cards)
- Payload (string, integer, JSON, etc.)

- Quality of Service--QoS (0, 1, or 2)
- Retention flag (yes, no)
- Duplicate (0 initial transmission, 1 repeat)

MQTT SUB Encoding Basics

- Payload is list of topics, may use wild cards
  - ‘+’ is a wild card to select a single level
  - ‘#’ is a wild card to select one or more levels
  - A wild card is the only character allowed on a level
  - A ‘#’ must be the first or last level
- Payload must be UTF-8 encoded string
- Requested QoS (0, 1, or 2) for subsequent broker publishes to client

MQTT topic

- Just a UTF-8 string of characters
  - *except ‘+’ and ‘#’*
- Can be free form, although local rules facilitate wild card use
- Current practice uses slashes to delimit levels of a hierarchy to structure the data

Topic Examples

- nodename/temperature/unit/value
- nodename/humidity/unit/value
- nodename/heartbeat
- #/value gives all values
- nodename/# gives everything from node
- +/temperature/# gives all temperatures from all nodes
- #/heartbeat give heartbeat from all nodes

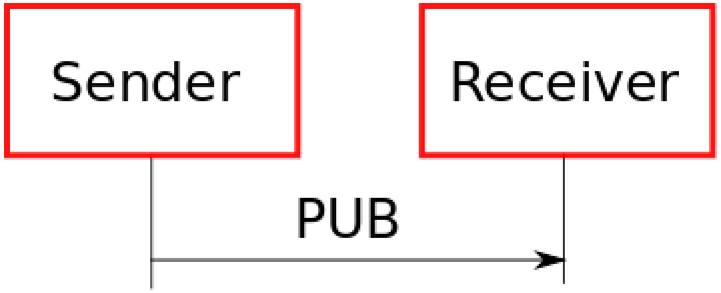
MQTT Payload

- Variable length
- Format is application specific
- Strings and integers are easy
- A string can be JSON for more complex data

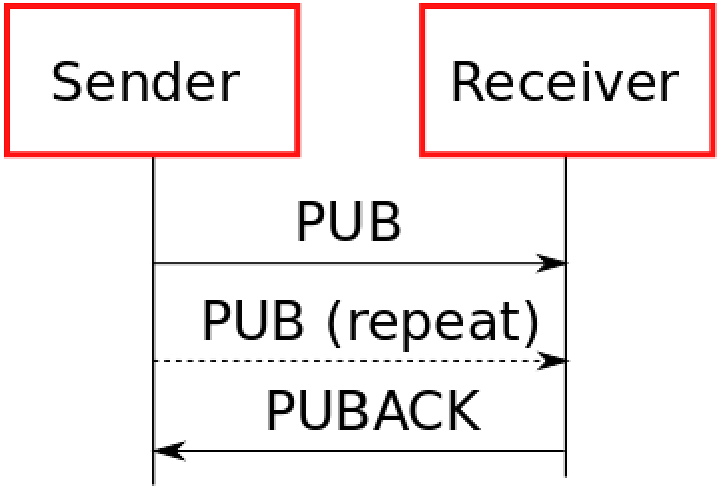
Quality of Service – QoS

- QoS 0: At most once or “fire and forget”
- QoS 1: At least once
  - Message is sent until message ID ACKed
  - Message may be duplicated
- Qo2 2: Exactly once
  - Message is sent until message ID is RECed
  - ID is kept by receiver until ID is RELed
  - ID is kept by sender until ID is COMPed

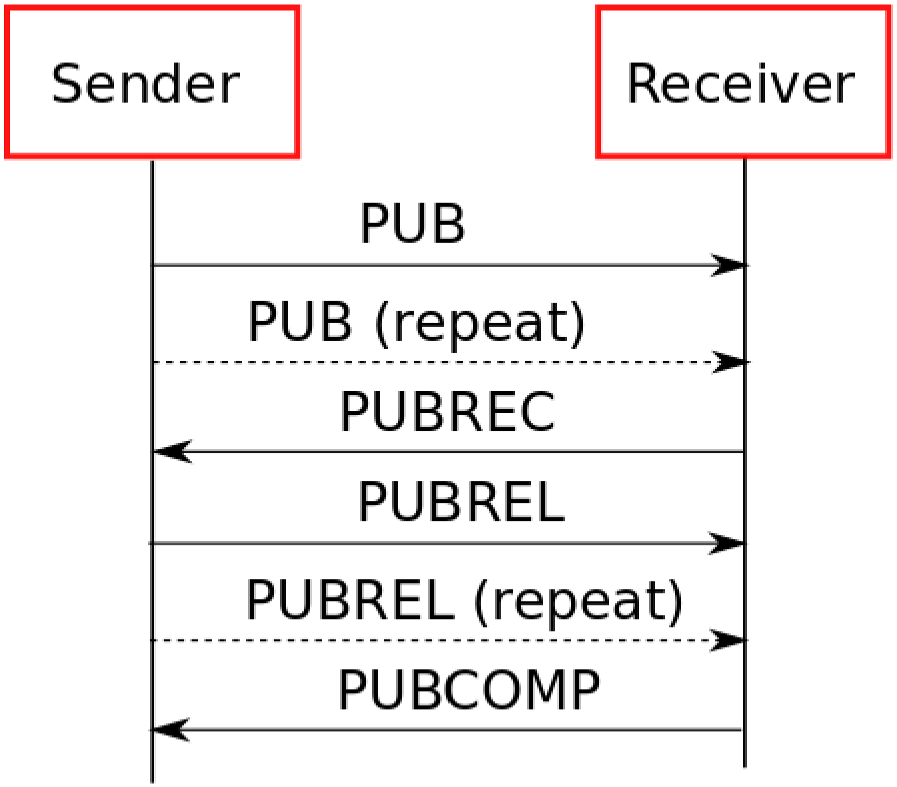
QoS 0 Message Sequence



QoS 1 Message Sequence



QoS 2 Message Sequence



Node-RED

- “They” call it an event wiring tool
- It is based on nodes and connections
- Focused on message flows
  - When a message arrives, do something with it

Implementation

- Node-RED has been part of Raspbian for some time
- Built with JavaScript on Node.js
- It is accessed with a browser
- Easy install procedure for Debian and Ubuntu

Using Node-RED

- It is extended with node npm modules
  - Raspberry Pi GPIO
  - Node-RED-Dashboard   ***MUST HAVE***
- Flows, the user source code, may be shared as JSON which carries the graphic information, attributes, and interconnections

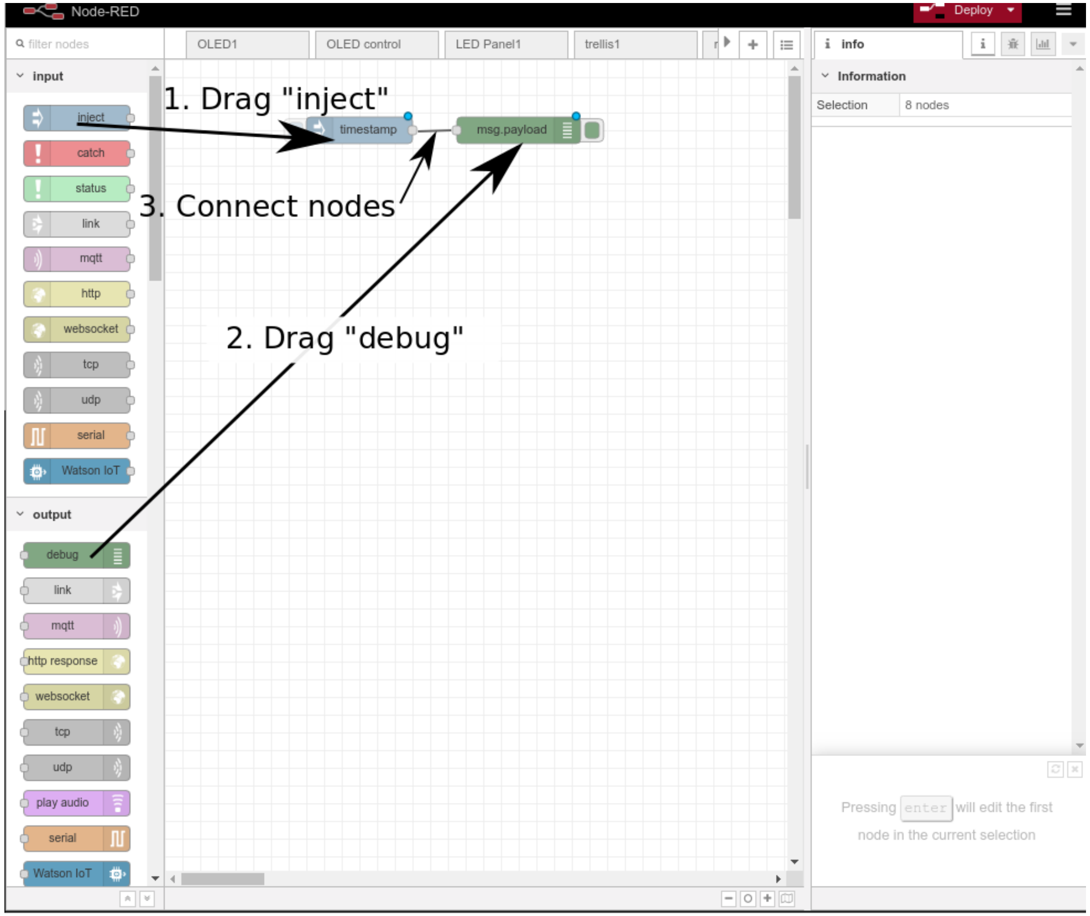
What is a Node?

- Something that manipulates a message
- Could be an MQTT endpoint
  - publisher or subscriber
- Could be a function to transform messages
- Could be just about any event producer
- Could be just about any event consumer

Simplest Node-RED Demo

- Two Nodes
  - An injection node
  - A debugging node
    - Just prints what it receives on the debugging window
- Configure
  - Change injection node topic to “message/first”
  - Change injection node payload to “Hello World”
- Deploy – compile into JSON and JavaScript

Add nodes and connect

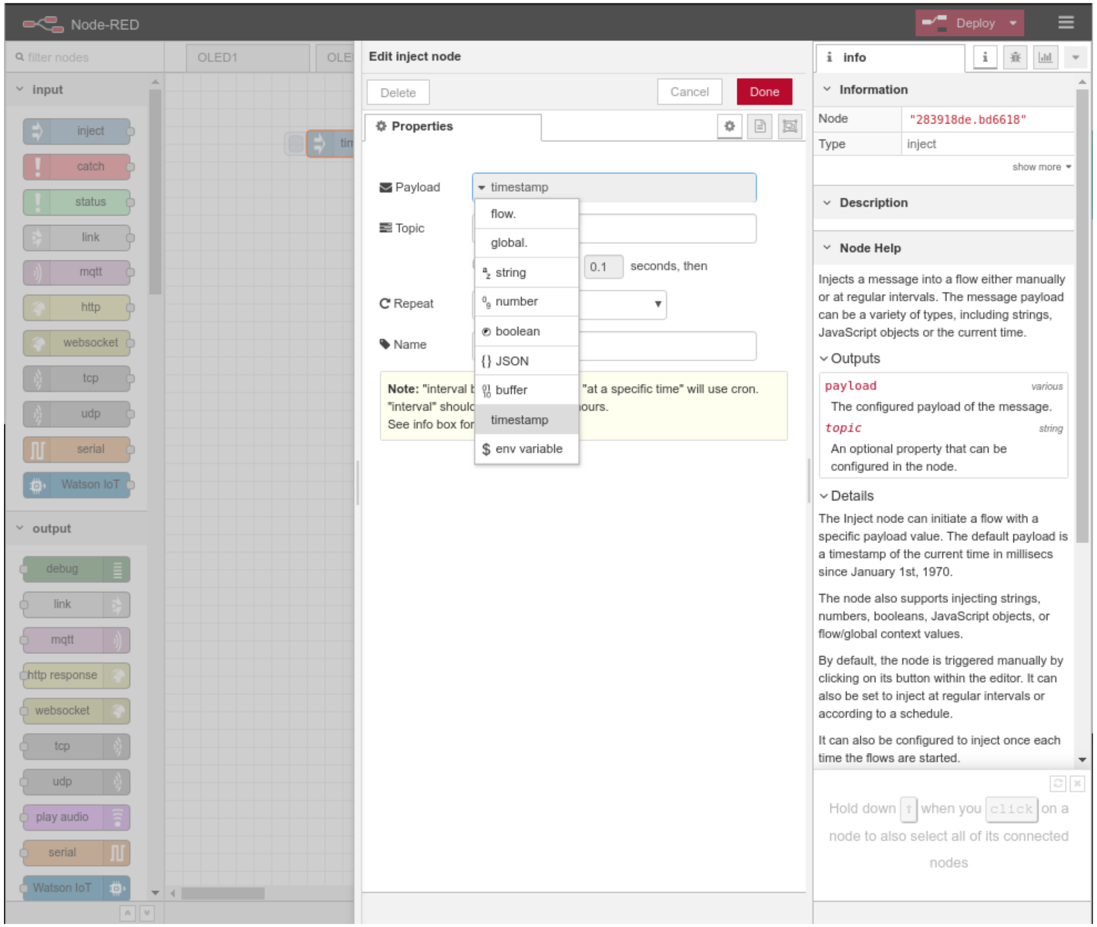


1. Drag "inject"

2. Drag "debug"

3. Connect nodes

Configure Inject



Payload: timestamp

Topic

0.1 seconds, then

Note: "interval b... "at a specific time" will use cron. "interval" shoul... ours. See info box for

Node	"283918de.bd6618"
Type	inject

Injects a message into a flow either manually or at regular intervals. The message payload can be a variety of types, including strings, JavaScript objects or the current time.

Outputs

`payload` *various*
The configured payload of the message.

`topic` *string*
An optional property that can be configured in the node.

Details

The Inject node can initiate a flow with a specific payload value. The default payload is a timestamp of the current time in millisecs since January 1st, 1970.

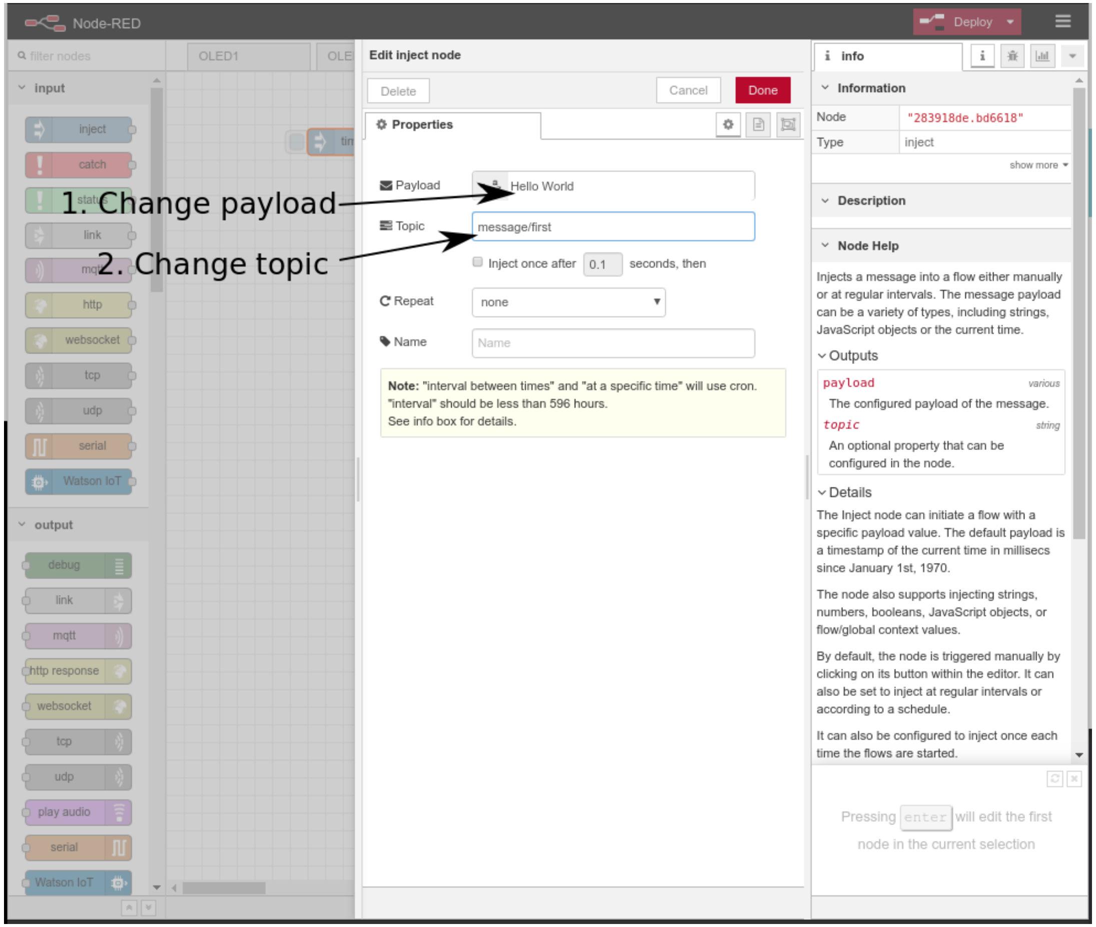
The node also supports injecting strings, numbers, booleans, JavaScript objects, or flow/global context values.

By default, the node is triggered manually by clicking on its button within the editor. It can also be set to inject at regular intervals or according to a schedule.

It can also be configured to inject once each time the flows are started.

Hold down ⇧ when you `click` on a node to also select all of its connected nodes

Configure inject (continued)



1. Change payload

2. Change topic

Edit inject node

Delete Cancel Done

Properties

Payload: Hello World

Topic: message/first

□ Inject once after 0.1 seconds, then

Repeat: none

Name: Name

Note: "interval between times" and "at a specific time" will use cron.
"interval" should be less than 596 hours.
See info box for details.

Information

Node	"283918de.bd6618"
Type	inject

show more

Description

Node Help

Injects a message into a flow either manually or at regular intervals. The message payload can be a variety of types, including strings, JavaScript objects or the current time.

Outputs

payload *various*
The configured payload of the message.

topic *string*
An optional property that can be configured in the node.

Details

The Inject node can initiate a flow with a specific payload value. The default payload is a timestamp of the current time in millisecs since January 1st, 1970.

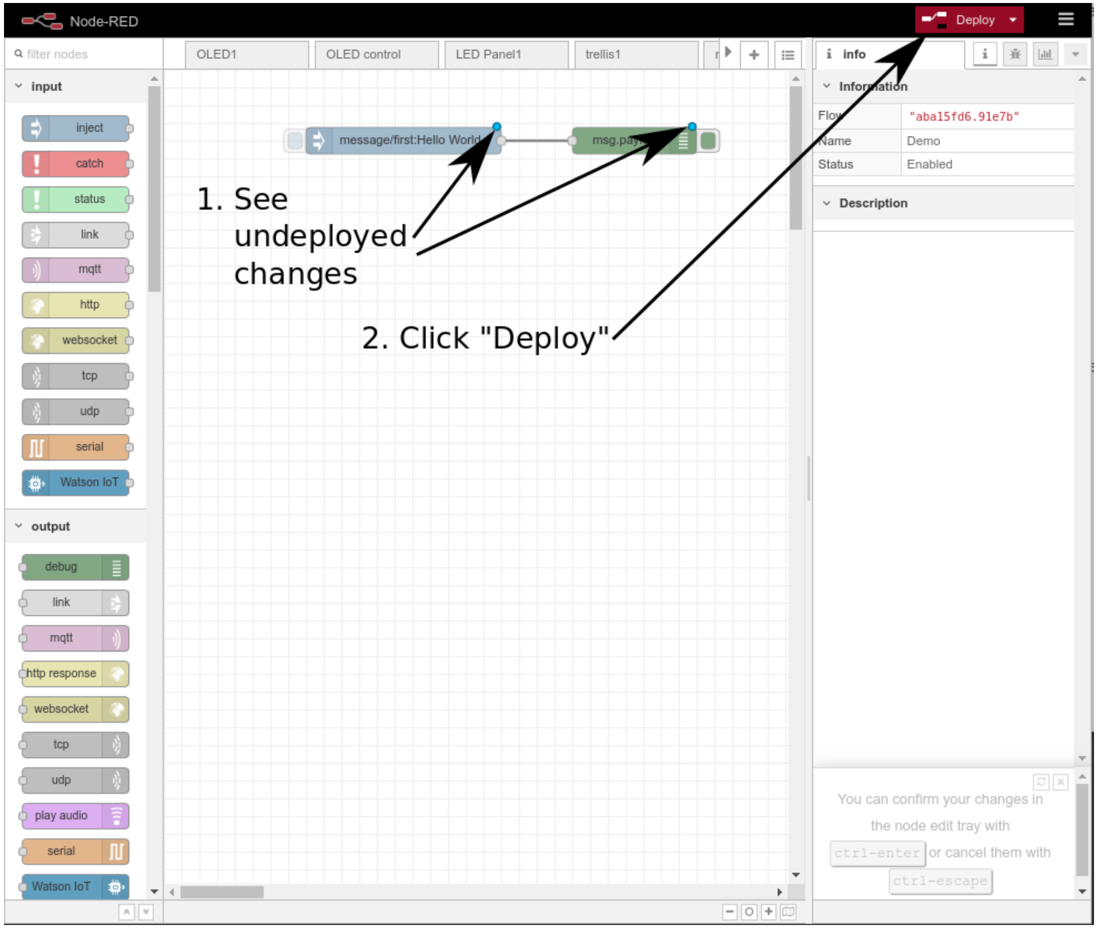
The node also supports injecting strings, numbers, booleans, JavaScript objects, or flow/global context values.

By default, the node is triggered manually by clicking on its button within the editor. It can also be set to inject at regular intervals or according to a schedule.

It can also be configured to inject once each time the flows are started.

Pressing enter will edit the first node in the current selection

Deploy



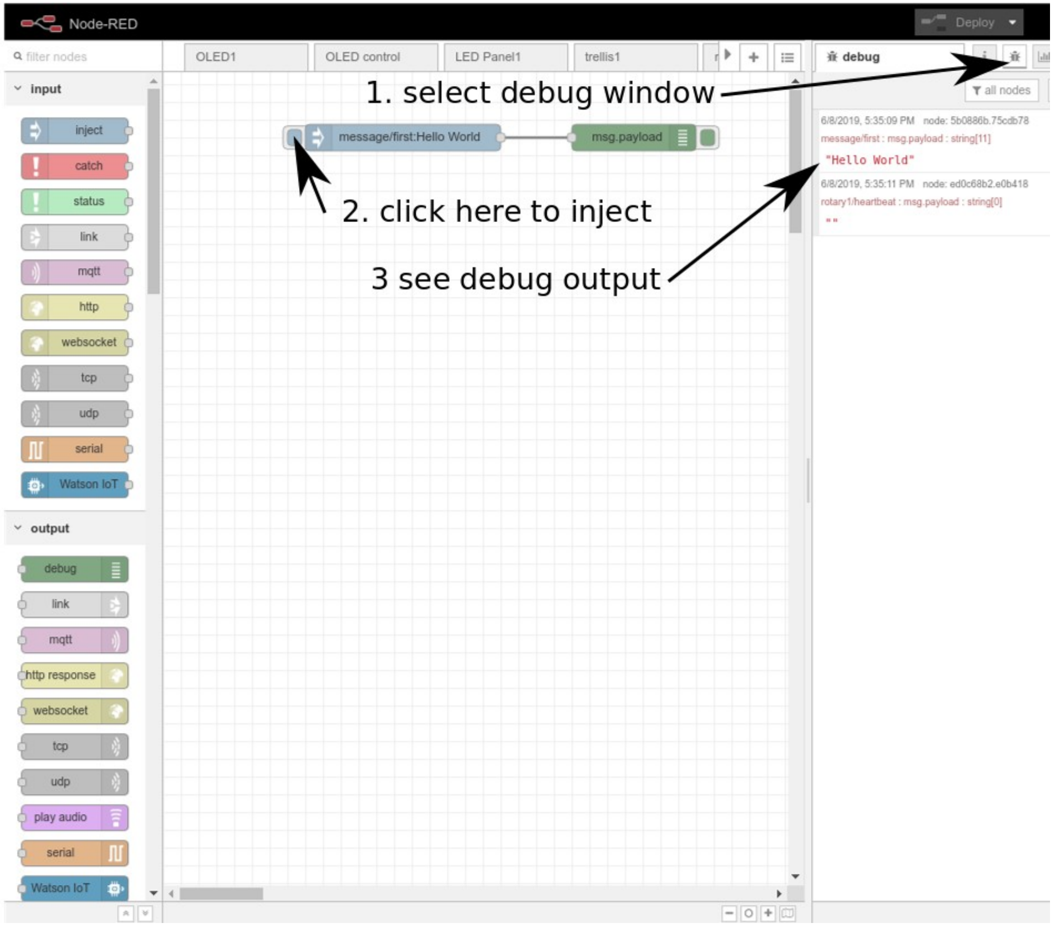
1. See undeployed changes

2. Click "Deploy"

Flow	"aba15fd6.91e7b"
Name	Demo
Status	Enabled

You can confirm your changes in the node edit tray with `ctrl-enter` or cancel them with `ctrl-escape`

Run it



1. select debug window

2. click here to inject

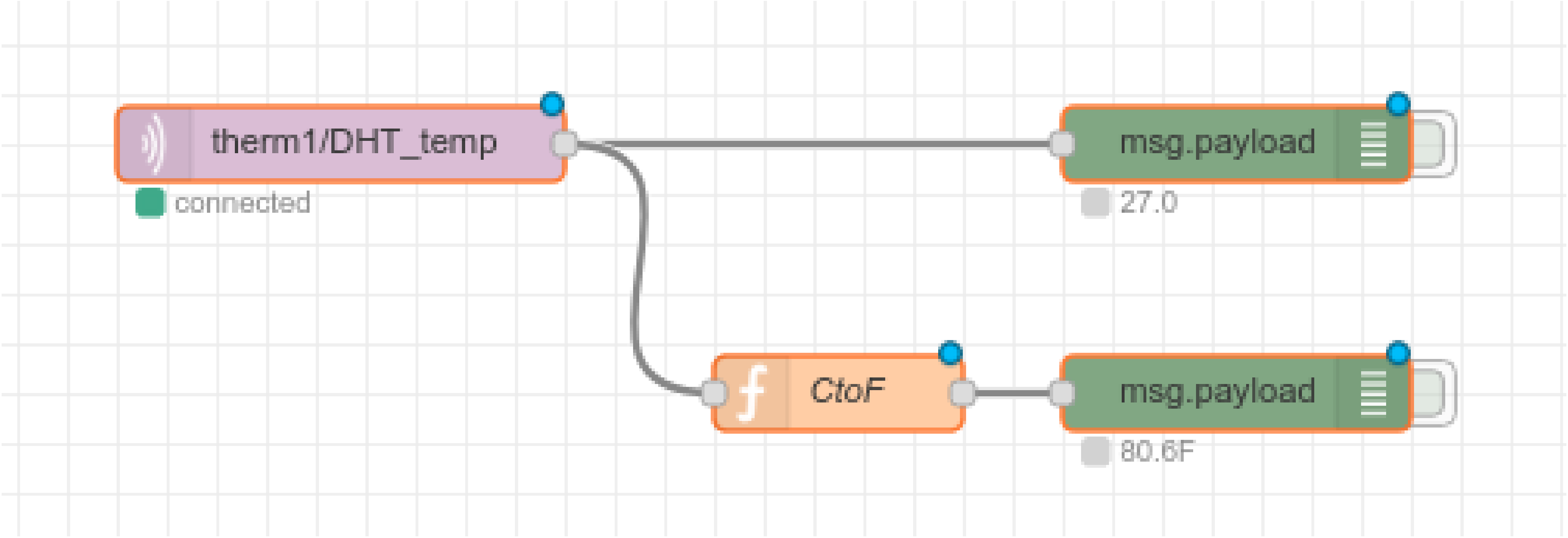
3 see debug output

```
6/8/2019, 5:35:09 PM   node: 5b0886b.75cdb78
message/first : msg.payload : string[11]
"Hello World"
6/8/2019, 5:35:11 PM   node: ed0c68b2.e0b418
rotary1/heartbeat : msg.payload : string[0]
""
```

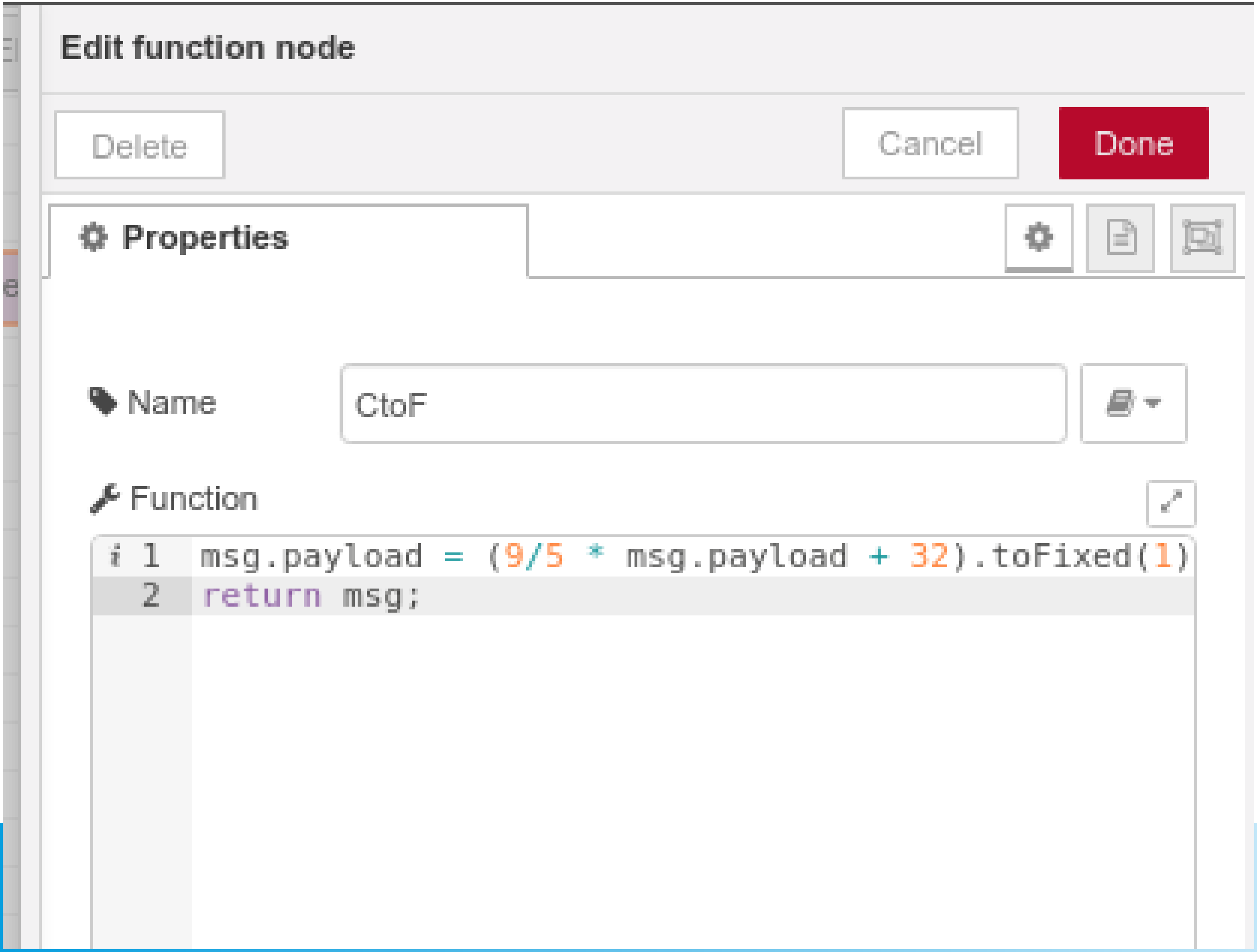
Simple Function Demo

- Subscribe to an MQTT message from a temperature sensor
- Send the payload to a MQTT display device
- Change the units from Celcius to Farhenheit

Function demo flow



Simple Function



Name: CtoF

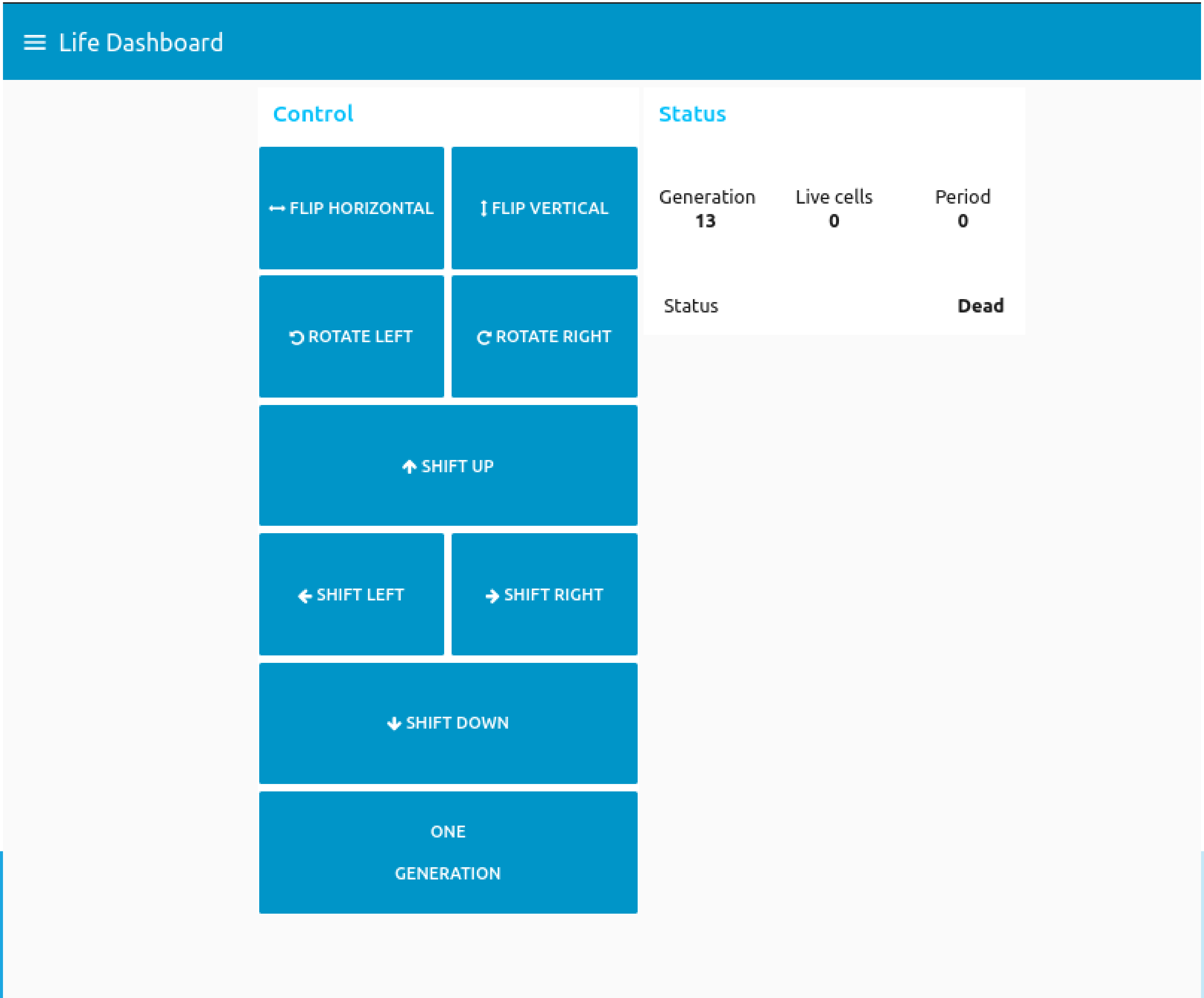
Function

```
msg.payload = (9/5 * msg.payload + 32).toFixed(1)
return msg;
```

Dashboards

- Dashboards allow control access with browser
  - Make a button, slider, text input, color selector
  - Make a text display, gauge, chart
  - Make a noise
- Repeat and get creative

Dashboard Example for Desktop



Life Dashboard

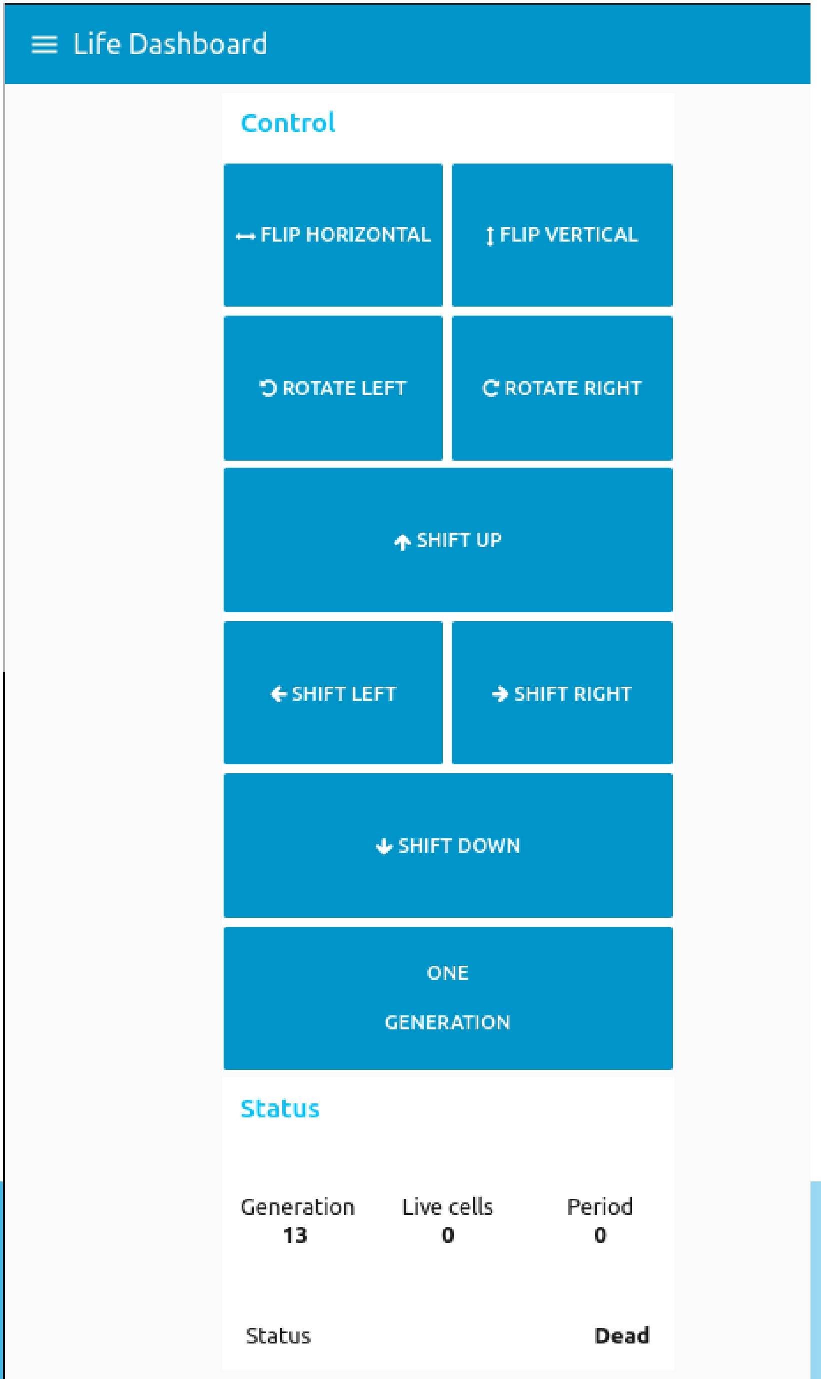
Control

Status

Generation	Live cells	Period
13	0	0

Status **Dead**

Dashboard Example for Mobile



Generation 13

Live cells 0

Period 0

Status Dead

MQTT Demo

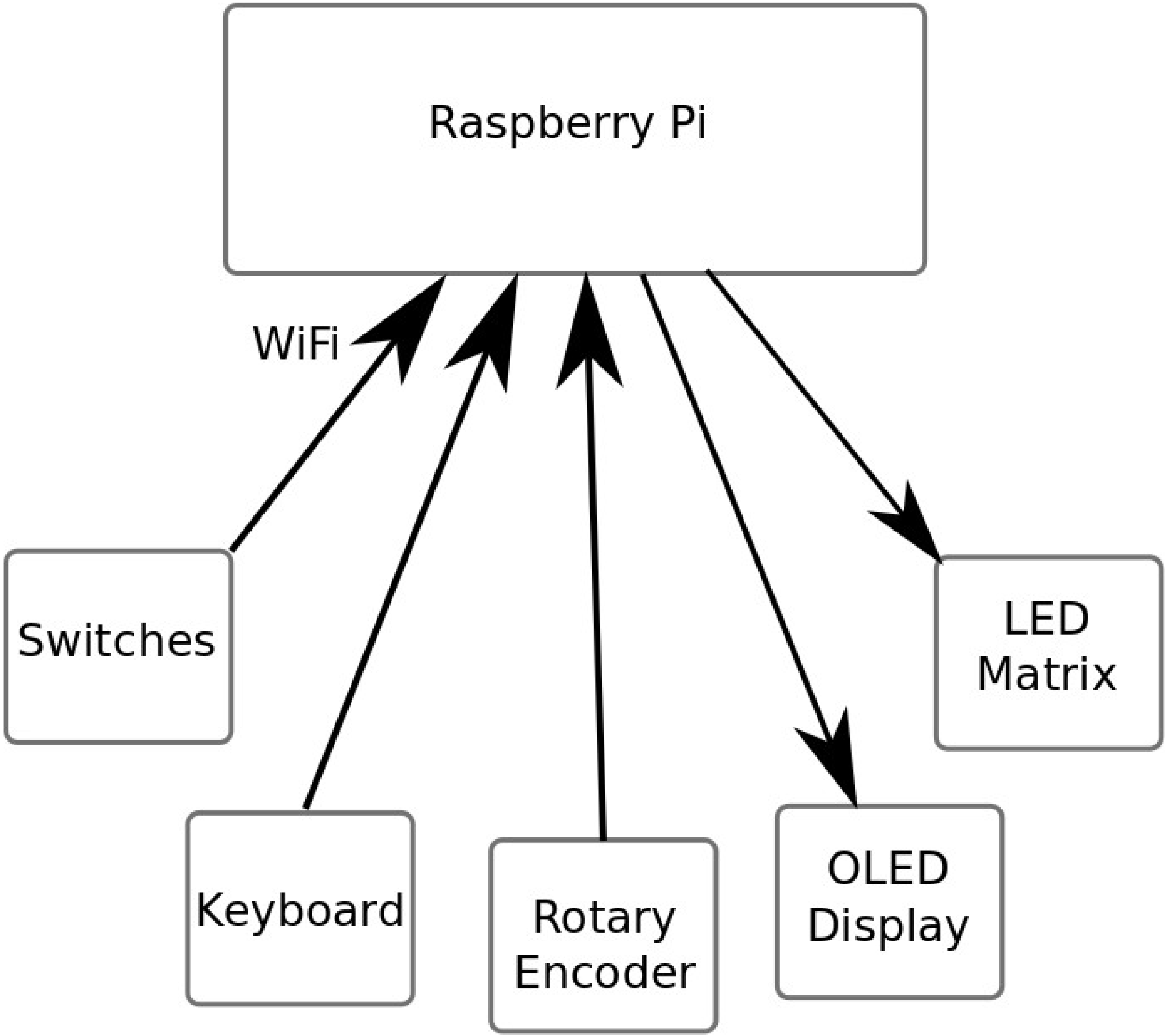
- Basic setup is for a STEM fair
- Flashing lights to attract the kids
- Interactivity to hold attention for a while
- Talk about computers, software, messaging, hardware, whatever is of interest
- Activity is based around Conway’s Game of Life

Conway’s Game of Life

- Count neighboring live cells (0-8)
- Alive and 0 or 1 live cells: die from isolation
- Alive and 2 or 3 live cells: live for another day
- Alive and 4 or more live cells: die from crowding
- Dead and exactly three live cells: become live

- Great waste of computer resources in 70’s

Basic Setup of Demo



MQTT Demo (continued)

- Devices are generic and fairly dumb
  - *Using Particle Photons with particle.io disabled*
- Node-RED is used to connect devices
- Functionality implemented in Node-RED
- Dashboard created to allow remote control
  - Smart phone
  - Desktop browser

Raspberry Pi is Hub

- It is a WiFi access point
- It serves as the router and DHCP
- It is the MQTT broker
- It is the Node-RED server
  - Provides most of the functionality
- It is the dashboard server for a GUI

MQTT Security

- Default is ease of use, NOT security
- “Can” use TLS as part of TCP transport
- “Can” use password authentication
- Lots of vulnerabilities beyond this protection

MQTT Presence Detection

- Manually register nodes in Node-RED
- Do not accept requests from unknown nodes
  - Watch global subscriptions (“#/door”, “+/window”)
- Add a watchdog timer and heartbeat to nodes
  - Detect failing nodes (power, connection, ...)
- Report missing or failing nodes
- (MQTT connection protects link, not end-to-end)

Use Cases

- Collect and display sensor data
- Security system
  - Secure interfaces
  - Intrusion detection
  - Attack detection
  - Redundancy
    - power, comms, sensors, alarms, control
  - External monitor and reporting

Attack Surfaces

- Device to broker
  - WiFi denial of service
  - WiFi jamming
  - WiFi hijacking
  - Node impersonation
  - MQTT denial of service
  - MQTT jamming

Attack Surfaces (continued)

- Broker to Node-RED or vise versa
  - Denial of service
  - Impersonation (wrong nodes on authorized link)
  - Impersonation (publish commands on inbound links)

Attack Surfaces (continued)

- Broker to device
  - Impersonation (unauthorized commands)
  - Impersonation (wrongful acknowledge)
  - Denial of service (WiFi and MQTT)
  - Jamming (WiFi and MQTT)

Attack Surfaces (continued)

- Within Raspberry Pi
  - Intercept with false routes
  - Administer Node-RED
  - Inspect or replace Node-RED files
  - Inspect or replace node authorization files?
  - Inject messages to Node-RED
  - Subscribe to Node-RED feeds

Attack Surfaces (continued)

- Power
  - Power to the node devices
  - Power to the Raspberry Pi
  - Power to any required communication equipment
  - Primary power – grid?
  - Backup power – battery, generator, solar
  - Duration of backup

Attack Surfaces (continued)

- External Communication Links
  - Physically cut wires for phone, SDL, cable
  - Illegally jam cell phone
  - Illegally hijack cell phone
  - Denial of service on recipient
  - Denial of service on sender
  - Prior fake false negatives masking true positive
  - Improper redundancy provisioning

Attack Surfaces (continued)

- Node-RED-Dashboard
  - No native authentication
  - Either add something: password, auth token, ???
  - OR consider carefully what you expose

Attack Conclusion

- Wireless in general, WiFi in particular, sucks
- Wired solutions take more time
- Wired solutions are more secure
- Encryption secures link data
- Bi-directional authentication prevents hijacking and impersonation
- Need to separate Pi users and permissions