LoRa & Node-RED Proof-of-Concept: from Multitech xDot to Telit deviceWISE

The following are notes for a successful proof-of-concept where LoRa RF technology and Node-RED were used to transmit fictitious sensor data from a Raspberry Pi 3 (RPi) to a Telit deviceWISE backend application.

Figure 1 below illustrates the flow:

- A minicom terminal emulation session is configured on the RPi, is connected to the Multitech xDot LoRa device USB port and is used to send AT commands to the xDot
- After the xDot has joined the network, AT-SEND commands are used to send JSON *publish-property* commands
- Matching network ID, network passphrase and selected frequency sub-band between xDot and Multitech MultiConnect Conduit gateway permit two-way LoRa communications
- Node-RED running on the gateway is used to transmit the JSON *publish-property* command strings containing fictitious sensor values to the Telit MQTT endpoint

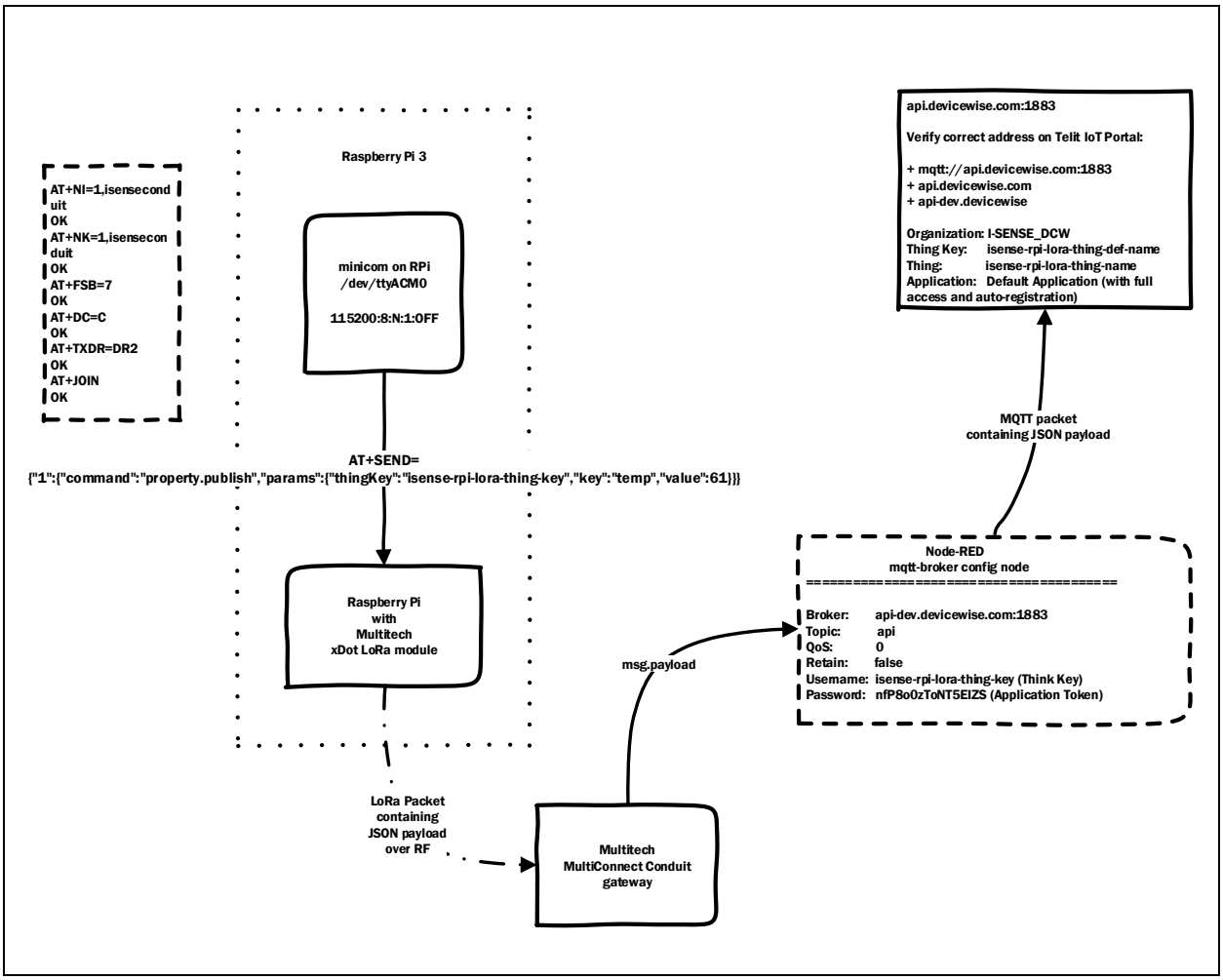


Figure 1 - proof-of-concept topology (xDot -> Conduit -> Node-RED -> deviceWISE)

xDot LoRa mCard (Accessory Card)

XXX

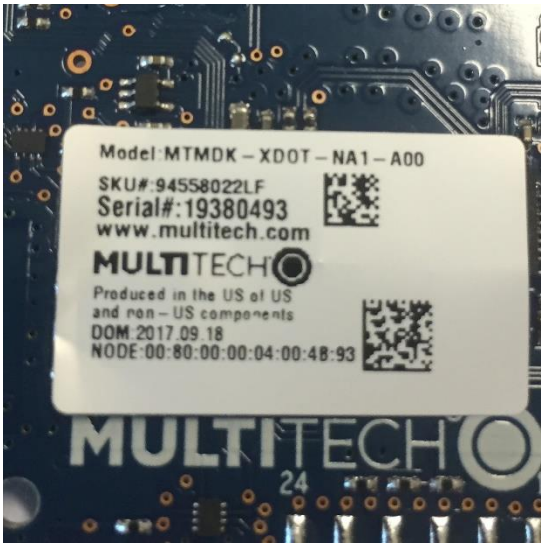


Figure 2 - Multitech xDot LoRa Module (chip nameplate on left, xDot board nameplate on right)

Windows 10

- Plug into COM port
    - One COM port for AT commands (XXX) and the other for DEBUG (mbed XXX)
- Configure as Baud=115200, Data bits=8, Parity=N, Stop bits=1, , Flow Control=OFF

Lunix (Rapsbian)

XXX

- Serial USB ports are /dev/ttyACMx (AT command port) and /dev/ttyACMy (DEBUG port), where x and y are integers and 0 <= x < y
-

How to make xDot join a network (connect to Multitech Conduit)

```
>AT+NI=1,isenseconduit          // set network name to 'isenseconduit'
>OK
>AT+NK=1, isenseconduit         // set network passphrase to 'isenseconduit'
>OK
>AT+FSB=7                       // set frequency sub-band to match Conduit value in LoRa config
>OK
>AT+DC=C                        // ensure Class C communication (constant RECEIVE)
>OK
>AT+TXDR=DR2                    // change transmission data rate to accommodate 129 characters
>OK
>AT+JOIN                        // join the network (connect to Conduit)
>OK
>AT+SEND=hello world            // to verify working (check Node-RED debug console)
```

How to configure xDot for serial mode

Serial Mode

Configure the device to wake periodically or on interrupt, wait for data on serial port, send data out, and go back to sleep. Refer to *Chapter 5, Power Management* for more information.

- Configure the device to wake up after 10 seconds of sleep and send data from the serial port:

```
AT+WM=0

OK
AT+WI=10

OK
AT+WD=100

OK
AT+WTO=20

OK
AT+SMODE=1

OK
AT&W
```



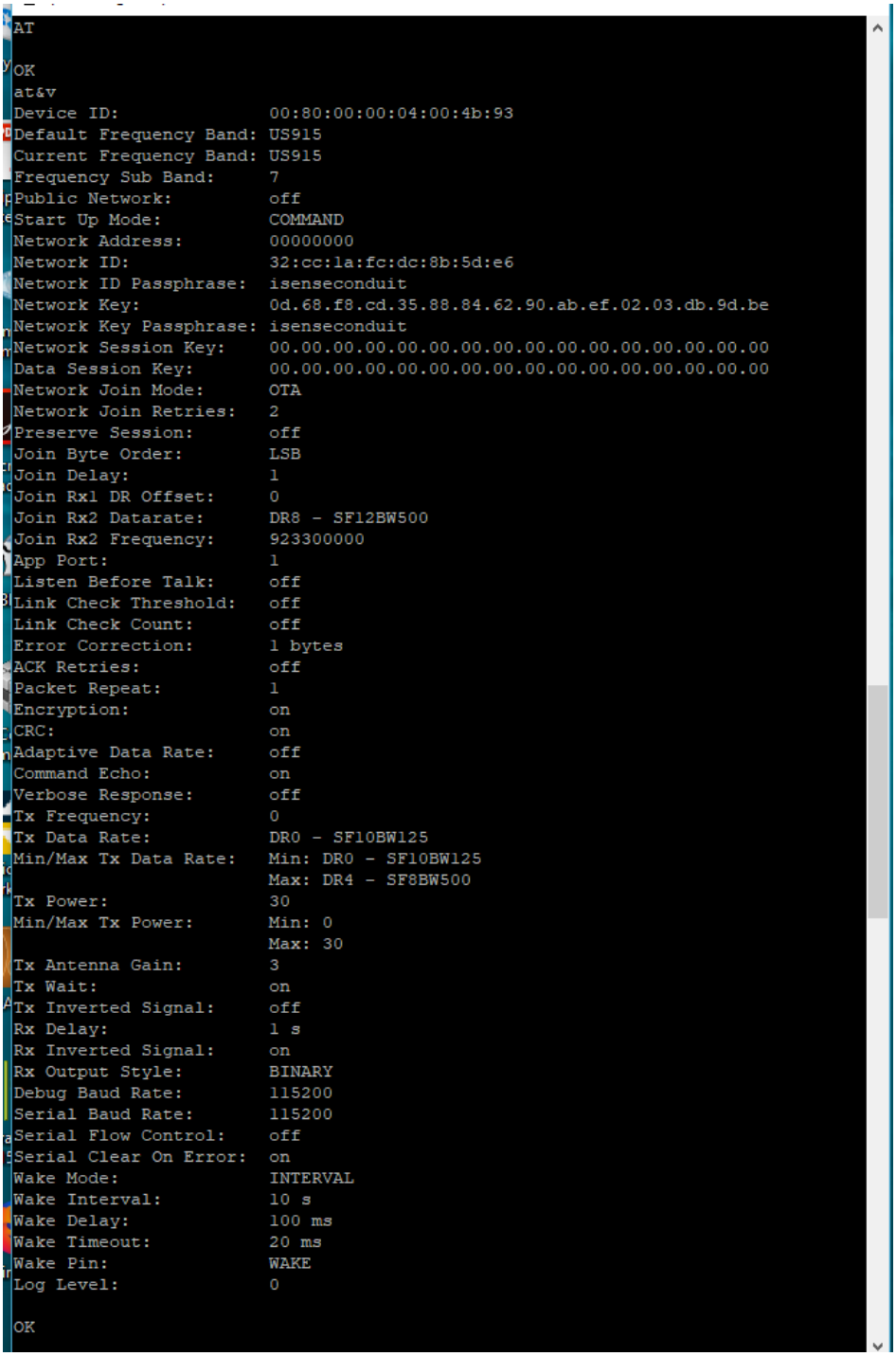
```
OK
ATZ

OK
```

Device resets into Serial Mode.

Settings

AT&V command



```
AT

OK
at&v
Device ID:              00:80:00:00:04:00:4b:93
Default Frequency Band: US915
Current Frequency Band: US915
Frequency Sub Band:     7
Public Network:         off
Start Up Mode:          COMMAND
Network Address:        00000000
Network ID:             32:cc:1a:fc:dc:8b:5d:e6
Network ID Passphrase:  isenseconduit
Network Key:            0d.68.f8.cd.35.88.84.62.90.ab.ef.02.03.db.9d.be
Network Key Passphrase: isenseconduit
Network Session Key:    00.00.00.00.00.00.00.00.00.00.00.00.00.00.00.00
Data Session Key:       00.00.00.00.00.00.00.00.00.00.00.00.00.00.00.00
Network Join Mode:      OTA
Network Join Retries:   2
Preserve Session:       off
Join Byte Order:        LSB
Join Delay:             1
Join Rx1 DR Offset:     0
Join Rx2 Datarate:      DR8 - SF12BW500
Join Rx2 Frequency:     923300000
App Port:               1
Listen Before Talk:     off
Link Check Threshold:   off
Link Check Count:       off
Error Correction:       1 bytes
ACK Retries:            off
Packet Repeat:          1
Encryption:             on
CRC:                    on
Adaptive Data Rate:     off
Command Echo:           on
Verbose Response:       off
Tx Frequency:           0
Tx Data Rate:           DR0 - SF10BW125
Min/Max Tx Data Rate:   Min: DR0 - SF10BW125
                        Max: DR4 - SF8BW500
Tx Power:               30
Min/Max Tx Power:       Min: 0
                        Max: 30
Tx Antenna Gain:        3
Tx Wait:                on
Tx Inverted Signal:     off
Rx Delay:               1 s
Rx Inverted Signal:     on
Rx Output Style:        BINARY
Debug Baud Rate:        115200
Serial Baud Rate:       115200
Serial Flow Control:    off
Serial Clear On Error:  on
Wake Mode:              INTERVAL
Wake Interval:          10 s
Wake Delay:             100 ms
Wake Timeout:           20 ms
Wake Pin:               WAKE
Log Level:              0

OK
```

Multitech LoRa Gateway (MultiConnect Conduit)

- Can be configured as:
  - Packet Forwarder or
  - LoRa Network Server
- Page: https://www.multitech.com/models/94557247LF
- data sheet: https://www.multitech.com/documents/publications/data-sheets/86002170.pdf

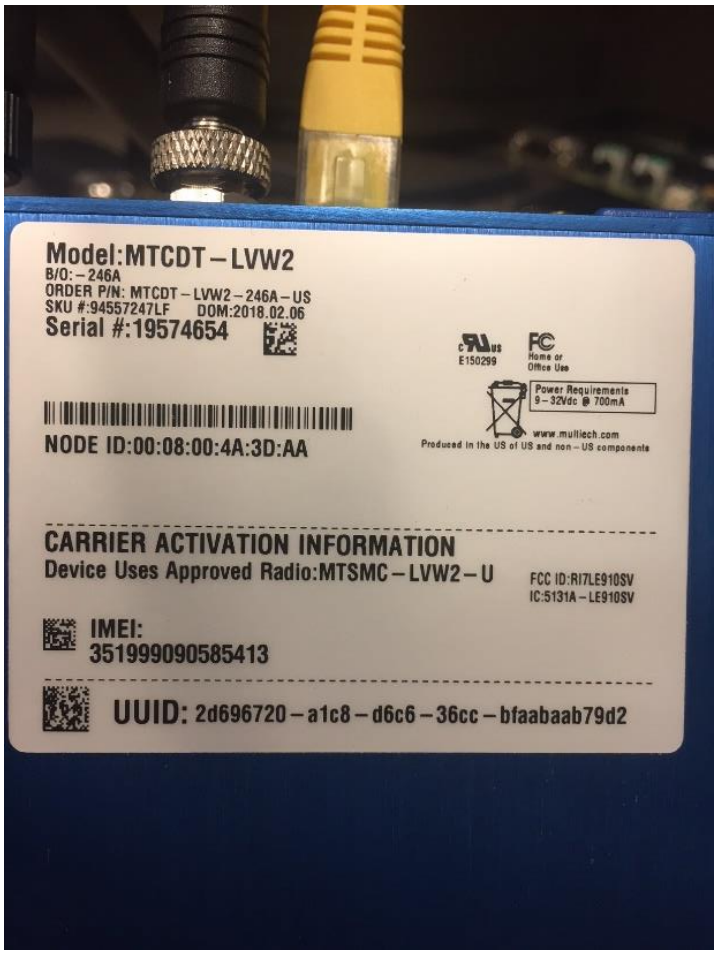


Figure 3- Multitech Multiconnect Conduit

- Current DHCP assigned IP address=10.15.13.100
- There's also an outdoor version of the Conduit with an IP67 rating for protection against dust and water immersion up to 1 meter (https://www.multitech.com/brands/multiconnect-conduit-ip67)

IP67 MultiConnect Conduit Gateway

Figure 4 - IP67 version of Conduit gateway

MTAC-MFSER-DTE Serial mCard (Accessory Card)

This is …

```
# mts-io-sysfs show ap2/serial-mode
loopback
# mts-io-sysfs store ap2/serial-mode rs232
#
#
# // could be ap2 or ap1
```

Figure 5 - Change serial mode from loopback to rs232

Campbell Scientific CH200 12V Voltage Regulator

The Campbell Scientific CH200 monitors and charges the 12V battery that supplies the Multitech MultiConnect Conduit gateway. This applies only to the MTCDT Series, since the MTCDTIP Series (IP67 Base Station) utilizes 48vdc power over Ethernet.

If the Node-RED software discovers a serial port and the CH200, it issues the RD_STAT> to the CH200 every hour to obtain status information about the battery and regulator. Since our current IoT portal (Telit deviceWISE) only supports numeric Thing property values, the two string values are converted to numbers (XXX). An example property.batch command that is published on deviceWISE follows:


```
{
  "cmd": {
    "command": "property.batch",
    "params": {
      "data": [
        {
          "key": "mt-conduit-diag-12v-charging-regulator-batt-voltage",
          "value": 12.789
        },
        {
          "key": "mt-conduit-diag-12v-charging-regulator-batt-current",
          "value": -0.2988
        },
        {
          "key": "mt-conduit-diag-12v-charging-regulator-load-current",
          "value": 0.3009
        },
        {
          "key": "mt-conduit-diag-12v-charging-regulator-charge-input-voltage",
          "value": 0.036
        },
        {
          "key": "mt-conduit-diag-12v-charging-regulator-charge-input-current",
          "value": 0
        },
        {
          "key": "mt-conduit-diag-12v-charging-regulator-temp",
          "value": 24.31
        },
        {
          "key": "mt-conduit-diag-12v-charging-regulator-charge-state",
          "value": 12
        },
        {
          "key": "mt-conduit-diag-12v-charging-regulator-charge-source",
          "value": 21
        },
        {
          "key": "mt-conduit-diag-12v-charging-regulator-check-battery",
          "value": 0
        }
      ],
      "thingKey": "19574654",
      "ts": "2018-06-15T16:54:39.728Z"
    }
```


Figure 6 - Campbell Scientific CH200 12V charging regulator data

The string values for Charge State and Charge Source are assigned numeric values.

Parameter	String Value	Numeric Value
Charge State	REGULATOR FAULT	11
	NO CHARGE	12
	CURRENT LIMITED	13
	CYCLE CHARGING	14
	FLOAT CHARGING	15
	BATTERY TEST	16
Charge Source	NONE	21
	SOLAR	22
	CONTINUOUS	23

Table 1 - CH200 numeric codes for string values

Figure 7: IP67 -2

LoRa Tuning & Research

- Radio propagation (https://www.youtube.com/watch?v=T3dGLqZrjIQ&feature=youtu.be)
    - Wireless modulation technology
    - Physical layer for long0range communications
    - Low bandwidth
    - Low battery
    - License-free ISM bands
    - Link budget
    - LoRaWAN
        - Defines speeds and bandwidths
        - B-directional communication
        - Mobility
        - Localization
        - Communications protocol like IP that utilizes LoRa physical layer
        - Data rates: 300 bps to 5500 bps
        - Bandwidth vs. range:
            - RFID: low range low bandwidth
            - BT: lower range, higher bandwidth
            - WIFI: much higher bandwidth, much lower range
        - Pick 2 Rule (difficult to get all 3 – laws of physics)
            - Distance, speed, power
            - LoRa picks long distance and low power
            - Speed is sacrificed
    - Link budget: Tx(P=14dBm)->Tx power->connector loss->antenna gain->free space path loss and fading->antenna gain-connector loss->Rx input power (hopefully enough for receiver sensitivity; P=-100dBm)
    - 3 factors: Tx power – loss – receiver sensitivity
    - Free space path loss:
        - depends on distance squared
        - a sphere that gets bigger
        - energy is the surface
        - *double the distance: 6dB path loss*
        - dependency on wavelength (or frequency): because of antenna
            - higher frequency requires smaller antenna

        - FSPL = 20 log(d) + 20log(f)-147.55
        - FSPL delta (LoRa – 2.4Ghz) = 9dB
        - FSPL: falls with square of distance
        -
    - Receiver sensititvity
        - Minimum detectable signal
            - Thermal noise defines the noise floor
            - SNR: how much above floor
            - NF: receiver noise figure in dB
            - S(dBm)=-174+10log(BW)+NF+SNR
            - Scales with log of bandwidth (better than hi-bandwidth WiFi)
            - Where sensitivity line touches received power determine distance travelled
- Example LoRa link budget
    - Tx power: 14 dBm (EU) [25 mW]
    - BW: 125 kHz [10*log(125000)=51]
    - SNR: -20 (for SF12)
    - Noise Figure: 6dB
    - Sensitivity: -174 + 51 + 6 - 20 = -137 dBm
    - Link budget = 14 + 137 = 151 dB
    - In free space: FSPL = 150dB at 800km!
- Multipath fading:
    - Wireless reach: loss of signal from three forms of attenuation: Obstruction, Diffraction, Reflection (inverts signal wave and cancels with normal wave)
    - Reflection problem in urban environment
- Structural attenuation:
    - Attenuation of materials: Glass 1 dB, lumber 3dB, brick 4dB, concrete (102mm) 12dB, concrete block 23 dB, …
    - One concrete wall = 4 times the distance in FS
- Fresnel zone (reflection on Earth, phase shift -> attenuation)
    - F1: loss, F2:gain, F3: loss
    - F1 = 8.656(d/f)
    - Any object with F1 will degrade – not just line of sight!
    - F1 should be 60% clear of obstacles (80% better)
- Fading Margin

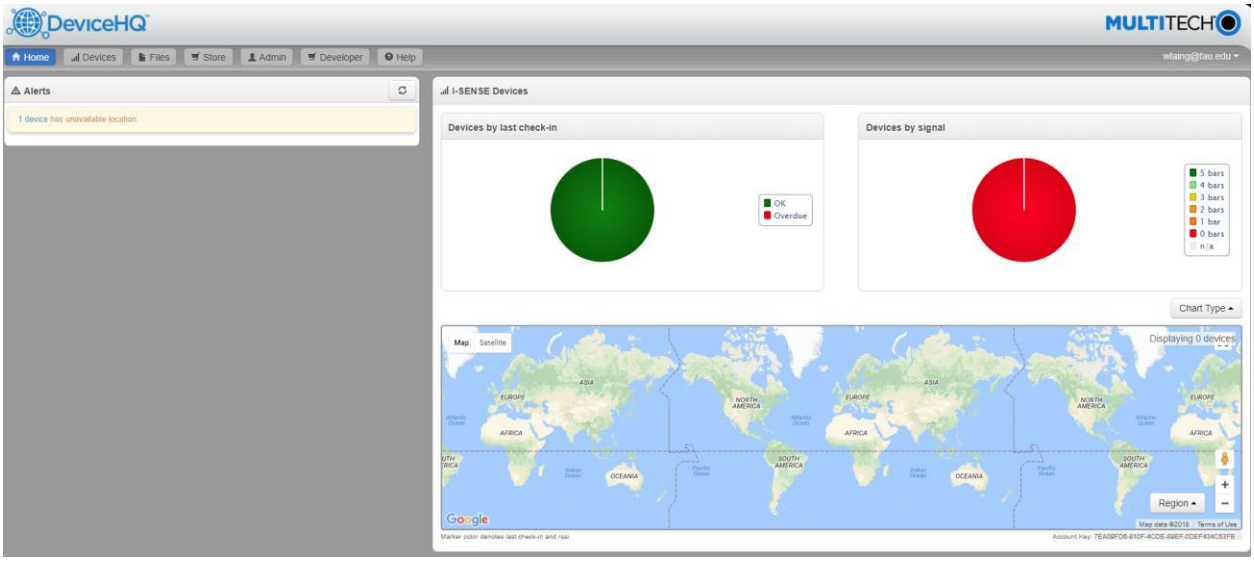
- Antennas
  - Omni-directional required (horizontal plane towards horizon vs isotropic point source)
  - Higher gain means narrower beam (extra dB in desired direction)
  - USA: max gain of +6dBi
  - Not all wideband antennae work (tuned to frequency)
  - Low SWR value
  - dBi = dBd + 2.15
- Range Summary
  - Increase height of gateway
  - Outdoors better than indoors
  - Minimize cable length gateway-antenna
  - Good connectors (N-type)
  - Good cable (LMR-400 or equivalent-> 1.5dB per 10m)
  - Avoid competing sources (GSM/UMTS)
  -
- Lora modulation
  - Uses spread spectrum technique: a variation of chirp spread spectrum
  - Increase reach (range) by increasing energy per bit:
    - ^ Transmit power (EU 868MHz: 14dBm=25mW)
    - v Modulation rate (spreading factors change modulation rate)
- LoraWAN scalability
-

MT LoRA Demo Issues/Questions

- https://10.15.13.100/help/index.htm#t=Configuring_your_device/LoRa_Network_Server_Configuration.htm
    - No Enabled button
    - No Public button
- Changed Setup->LoRa-Settings->Tx Power (dBm): 3 // was 26
- Changed Administration->Remote Management ->Account Key: 7EA09FD6-910F-4CDE-89EF-0DEF434C53FB
- Changed Administration->Remote Management ->Enabled ☑
- Installed aws-sdk
- Installed aws-iot-sdk-javascript
- 14 March 2018: save configuration to XXX
- Installed new version of MQTT-broker (npm i node-red-contrib-mqtt-broker), but it failed. Apparently, it requires more recent versions of mosca and npm on the Conduit AEP.
-

Multitech DeviceHQ

- http://www.multitech.net/developer/wp-content/uploads/2015/08/DeviceHQ_Dev_User_Guide.pdf
- Deployment guide: http://www.multitech.net/developer/wp-content/uploads/2017/12/S000640.pdf
- Does not appear to allow the uploading and management of device/sensor data
- Facilitates the management and deployment of Node-RED apps



- can set notifications by email or text for various events like missed check-ins or low cellular signal
-

deviceWISE

- login to deviceWISE Application Execution Platform AEP (https://open.devicewise.com/) – formerly IoT Portal
- create a Developer->Thing definition->New thing definition
    - key
    - name
    - Auto def attributes: √
    - Auto def properties: √
- create a Thing->New thing
    - Thing definition-> select the Thing definition created above
    - Key:
    - Name:
- create a Developer->Applications->New Application
    - Name:
    - Auto Registration Thing definition ID-> select Thing Definition of Thing that might auto-register
    - Org Admin: √
    -

```
{
"1" :
    {
      "command" : "property.publish",
      "params" :
                  {
                    "thingKey" : "isense-rpi-lora-thing-key",
                    "key" : "temp",
                     "value" : 87
                   }
     }
}
```

Figure 8 - Example JSON property.publish command to publish a single Thing property

```
AT+SEND={"1":{"command":"property.publish","params" :{"thingKey" : "isense-rpi-lora-thing-key","key" : "sum","value" : 1024}}}
```

Figure 9 - single line AT-command format for copying and pasting into terminal window

```
{
"1" :
    {
      "command" : "property.publish",
      "params" :
                  {
                    "thingKey" : "isense-rpi-lora-thing-key",
                    "key" : "temp",
                     "value" : 87
                   }
     }
}
```

Figure 10 - Example JSON property.batch command to publish multiple Thing properties

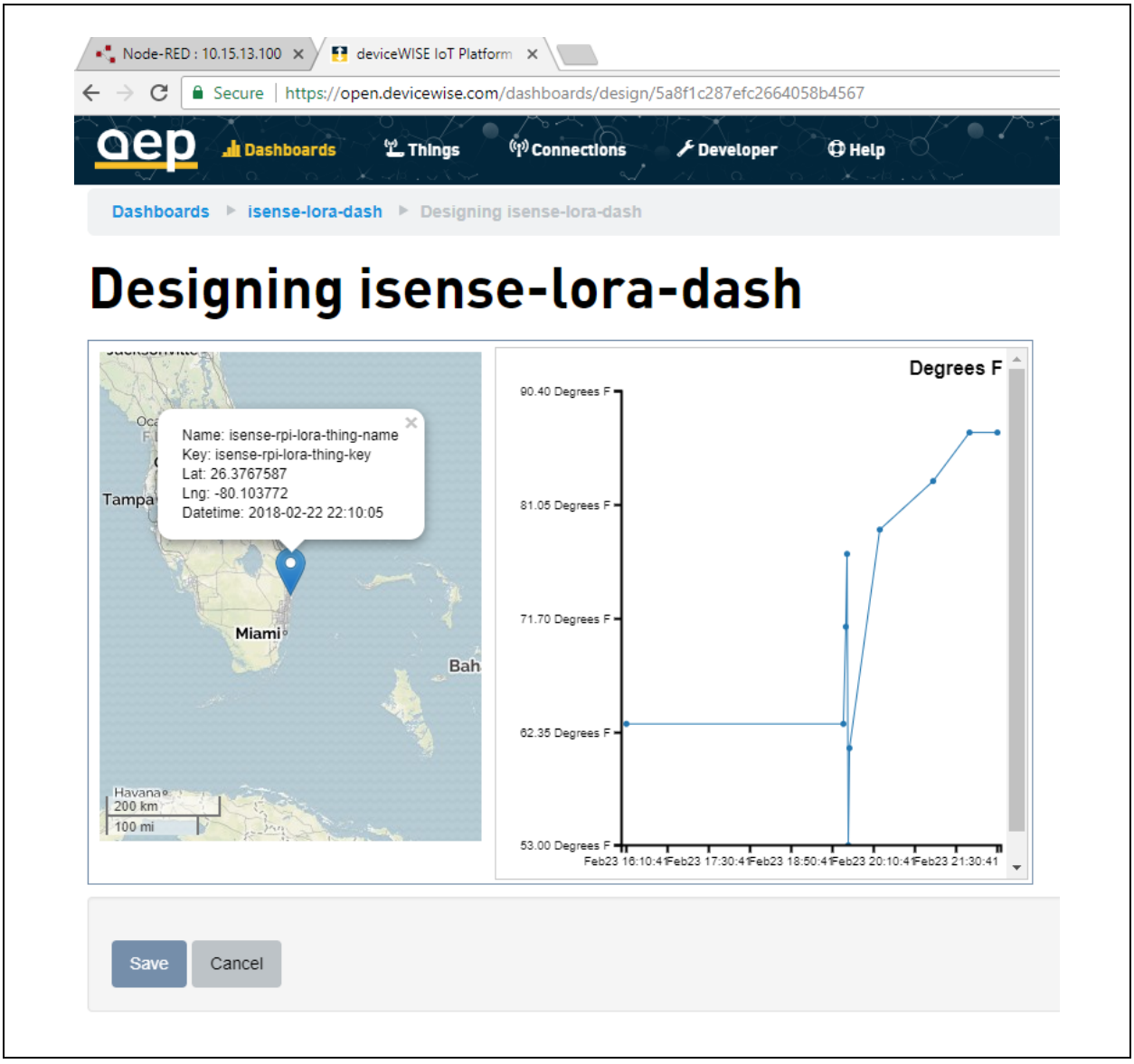
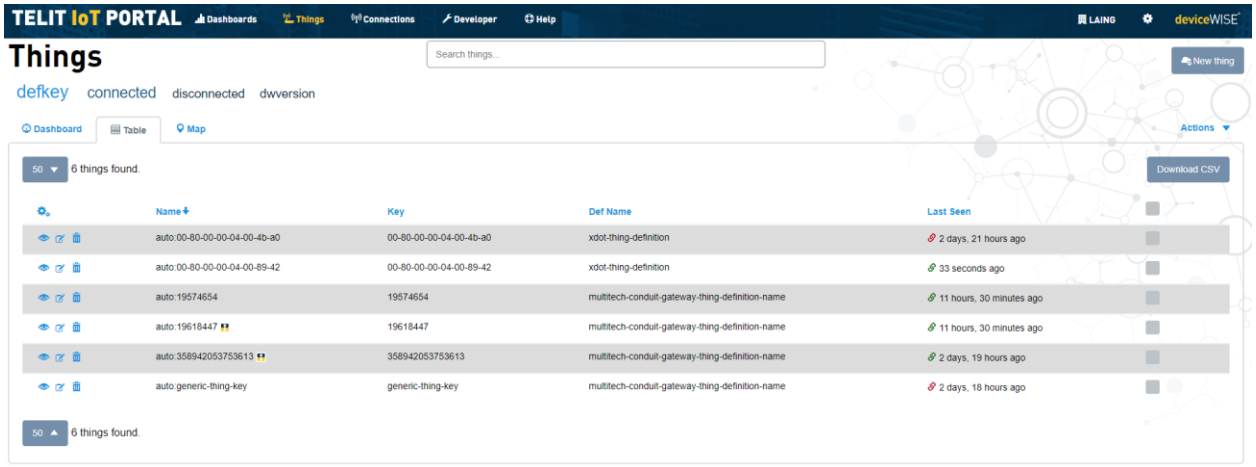


Figure 11 - deviceWISE dashboard showing Thing property graph

deviceWISE WorkBench

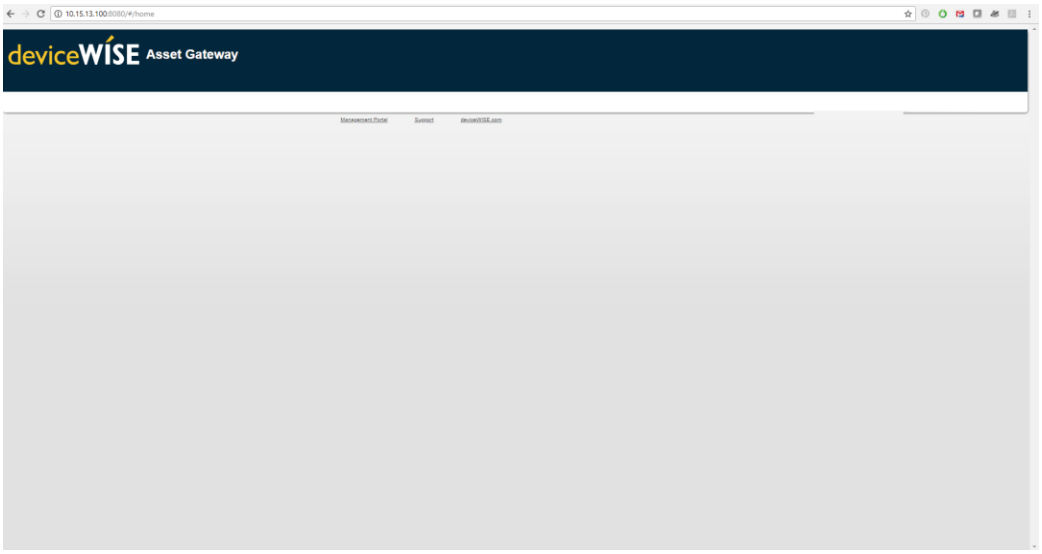
- Log into Developer portal
- Look at Things



-
    - Why does one gateway Things have two Workbench icons? Workbench is not running.
    - Both gateways are *connected* according to the Portal,
        - Only one has Workbench icon. Why is that?
        - Node-RED shows both MQTT nodes as connected to api-dev.devicewise.com
            - 19618447 used as both ThingKey and ClientID (AppId)
            - 19574654 used as both ThingKey and ClientID (AppId)
    - Why did that 19618447-gateway auto-register twice, once using IMEI (auto: 358942053753613) and another time using serial number (auto:19618447)?
        - I deleted the auto: 358942053753613, but it auto-registered again as IMEI
    - How do we know if Thing is connected via MQTT?
        - xDot end-device never really connected, right, just bound to gateway?
        - Why does one xDot show as connected and one as disconnected (length of time until *Last Seen*?)
- Launch Workbench
    - Scan TR50
        - Shows auto: 19618447
            - Does not show auto: 358942053753613 (same gateway)
            - Does not show auto: 19574654 (different gateway) but Portal says connected and Node-RED MQTT shows connected
                - Is it possible Asset Gateway software not running in gateway? Probable is running, because same device is discoverable by using Network Scan and IP
        - Try to connect, but got error: *19618447: Connection with node could not be established*
    - Scan Network using 10.15.13.151
        - Shows IP67-19618447

        - Info: *There are no licenses loaded*
    - Scan Network using 10.15.13.100
        - Shows *NEW NODE*
        - Alerts-> Info-> There are no licenses loaded
        - Generated a license and added in Administration->Licenses->New
        - Used TR50->TR50 Connection Management-> Appl Token: 3IRO*** (isense-generic-workbench-gateway-application)
        - aaa

-
-



\

- Start up Node-RED on Multitech MultiConnect Conduit gateway
  - Login to gateway at https://10.15.13.100
  - Apps->Launch Node-RED (right-click to launch in new window)
  - Login to Node-RED using same credentials as gateway
  - Click on 'debug' tab to see messages sent from xDot
- Log into RPi and Access xDot
  - SSH into RPi
  - Start minicom (minicom –s /dev/ttyACM0)
  - Configure serial port setup 115200:8:N:1 (Serial port setup menu item)
  - Serial port setup->A Serial Device: /dev/ttyACM0
  - Exit (not Exit from minicom)
- At the RPi command prompt:
  - // Verify connectivity to xDot by typing 'AT' and receiving 'OK' back
  - >AT
  - >OK
  - >AT+NI=1,isenseconduit // set network name to 'isenseconduit'
  - >OK
  - >AT+NK=1, isenseconduit // set network passphrase to 'isenseconduit'
  - >OK
  - >AT+FSB=7 // set freq sub-band to match value in Conduit LoRa config
  - >OK
  - >AT+DC=C // ensure Class C communication (constant RECEIVE)
  - >OK
  - >AT+TXDR=DR2 // change transmission data rate to accommodate 129 characters
  - >OK
  - >AT+JOIN // join the network (connect to Conduit)
  - >OK
  - >AT+SEND=hello world // to verify connectivity to Node-RED, check debug tab
  - >OK
  - AT+ACK=1 to be notified when a message is not received // optional
  - >OK

DC Water

XXX

Software Components

Software components associated with the DC Water proof-of-concept are represented by green color in Figure 12 below.

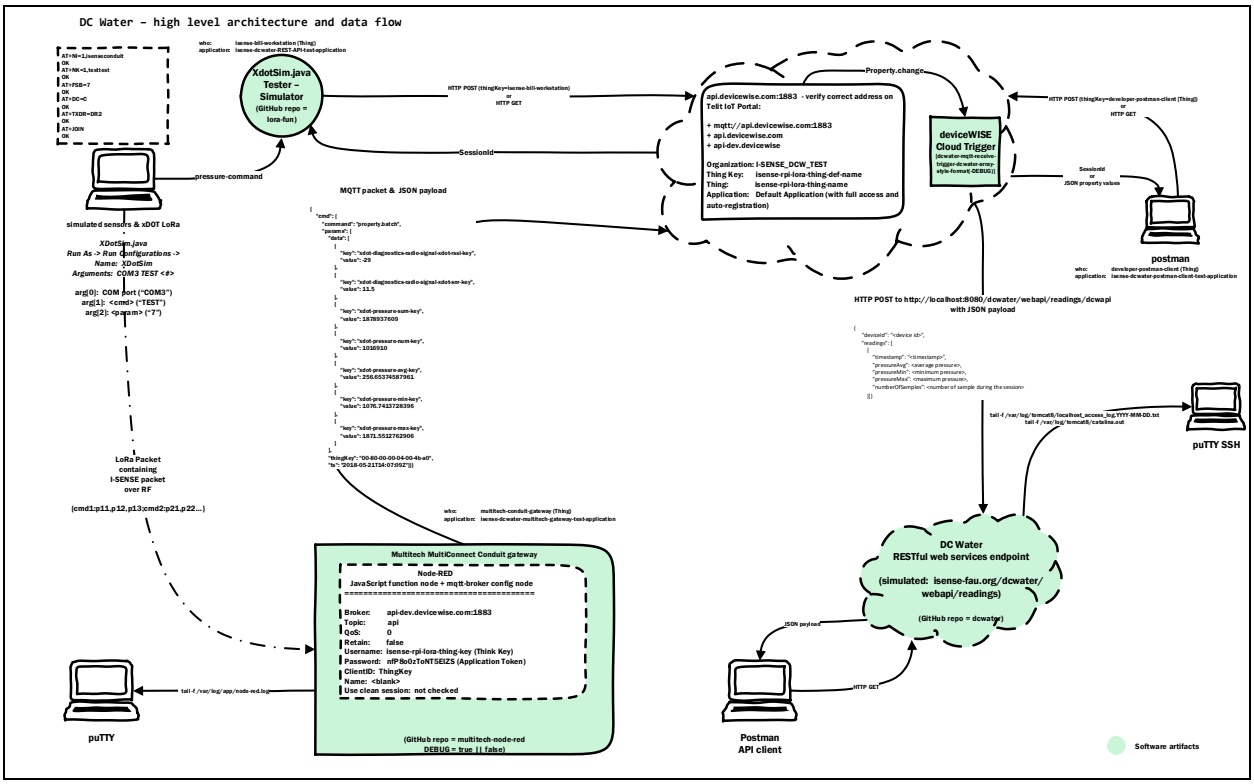


Figure 12 - DC Water test and deployment architecture

A summary of the GitHub repositories and their contents follow:

- lora-fun: simulation and test environment (Java)
    - can be used to configure the xDot by running the XDotSim run configuration in Eclipse with the arguments “<COM-PORT>”, “string” and “<AT-COMMAND>”
    - can be used to simulate xDot data packets to the gateway via LoRa by running the XDotSim run configuration in Eclipse with the arguments “<COM-PORT>”, “TEST” and “<NUMBER-OF-PACKETS>”
- multitech-node-red: the gateway code (Node-RED and JavaScript)
- dcwater: the REST API (Java)

REST API

There are several ways to send pressure readings to DC Water using REST APIs. The HTTP POST operation can be initiated from either the gateway or from the deviceWISE IoT portal.

Initiating the HTTP POST from a gateway can be achieved using the Node-RED graphical programming language. Node-RED on the gateway is already being used to parse incoming xDot messages, compose well-formed deviceWISE MQTT commands and publish these commands to the devceiWISE IoT portal using MQTT over wired Ethernet or cellular. Node-RED offers an http-request node to achieve the POST.

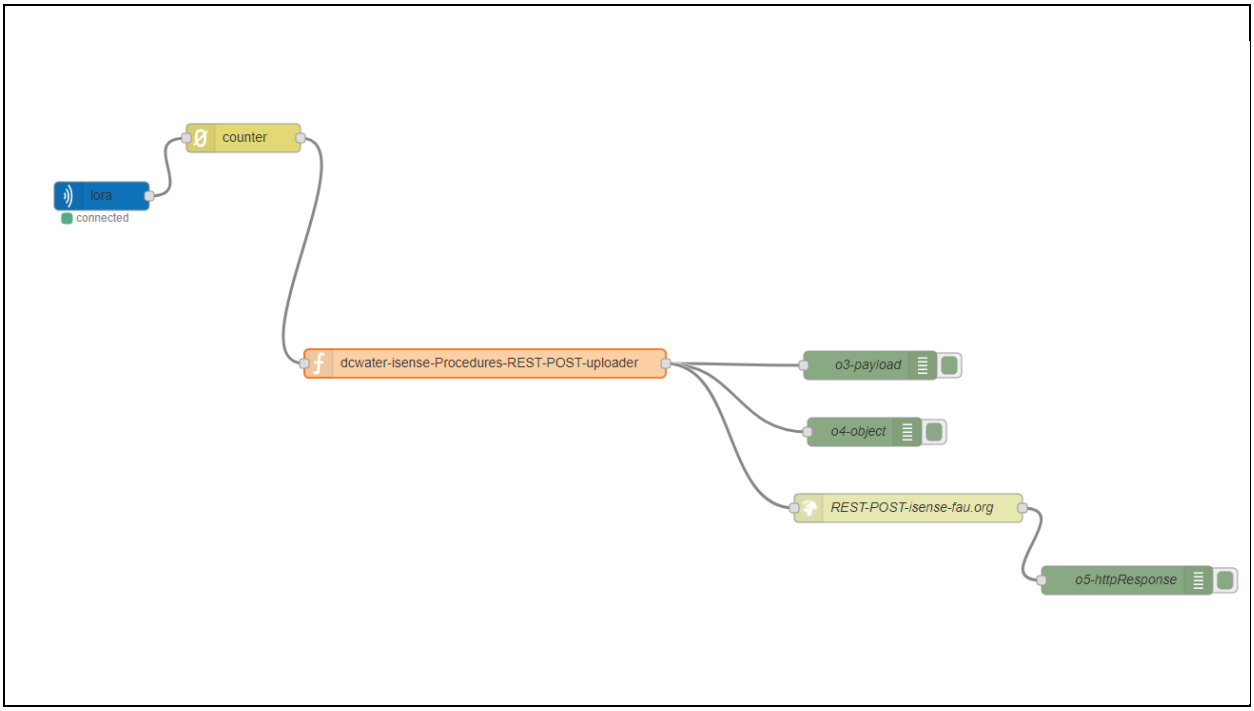


Figure 13 – Node-RED program example

The HTTP POST can also be initiated from the deviceWISE IoT portal using Cloud Triggers and the http.client node (Figure 14). However, at this time, it does not seem possible to capture all the pressure readings associated with one property.batch command accurately and repeatedly. Current methods (ex., the property.current action) return pressure reading values from more than one property.batch command. Pressure reading values from the property.current action could be from property.batch commands that arrived before or after the triggering event. This means that published values might be missed while others might be duplicated.

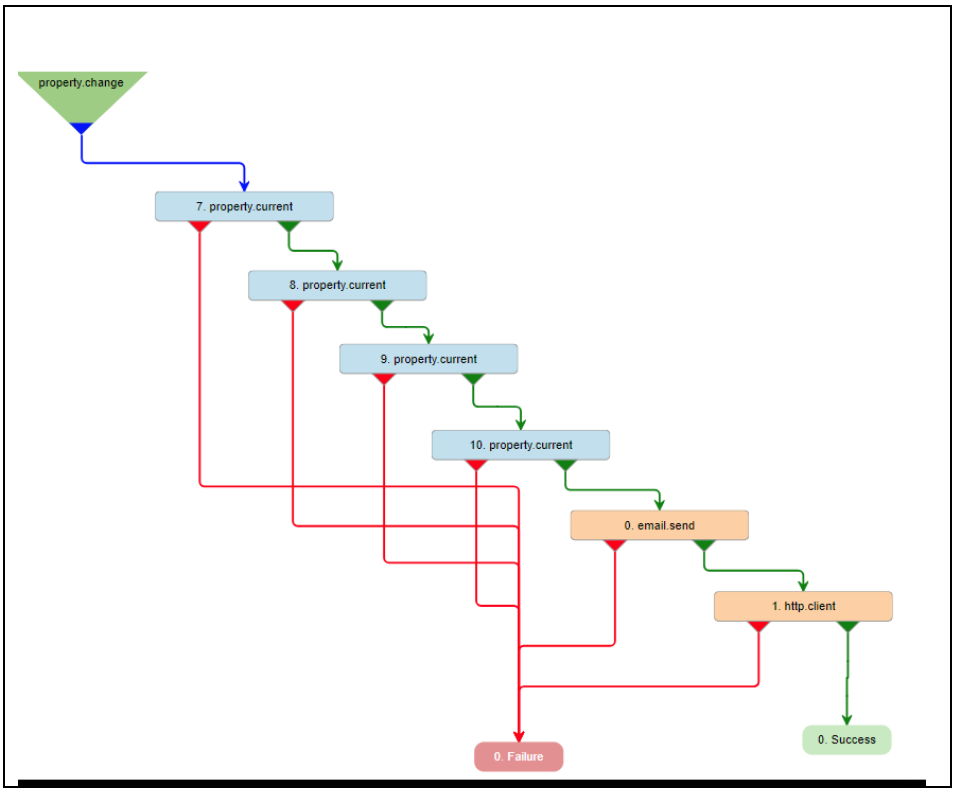


Figure 14 - deviceWISE Cloud Trigger example

property.change Cloud Trigger event to capture property.batch commands

This first approach features two API calls to the deviceWISE IoT portal and triggering a portal event based on one of the pressure properties changing values. The gateways generate two API calls for each data packet received from an xDot. One of the deviceWISE API calls is to bind the xDot to the gateway (thing.bind), since the gateway is the device that's connected to the IoT portal. The other deviceWISE API call is to publish all the property values (property.batch) sent by the xDot in the data packet.

- property.change event for one of the pressure properties
- four property.current actions to get the other pressure properties
- email action for debugging
- HTTP client to POST JSON using the DC Water REST API

The problem with this approach: it's not possible to guarantee that the trigger-generated set of pressure values are from the same property.batch. Therefore, some of the pressure values might be duplicates from previous property.batch commands. And some of the pressure values might never be sent to the REST API.

- [NO] Gerhard Lapp at Telit recommended not specifying timestamps in each of the property.current actions, but that did not fix the problem.

- [NO] Gerhard Lapp at Telit also recommended using the event timestamp instead of the property timestamp. This did not fix the problem either. The trigger event can be 10 seconds later than the timestamp of the property change. So it's possible to get property values that are newer than the value that triggered the event!
- [NO] I experimented with eliminating the fractional seconds in timestamps in the gateway JavaScript code, but that did not fix the problem.
- [ ] Another solution would be to POST one property value per REST API call to deviceWISE, and, as Gerhard said, "try to convince the other site to group the received values".
- [N/A] borrow from the next approach: substitute property.publish for property.batch? Gerhard clarified in a later email that we was referring to Asset Gateway here.
- [ ] add 10 second delay (Adrienne Clark) to the property.current action

property.change Cloud Trigger event to capture property.publish commands

This approach is a variation of the previous approach that requires:

- changing the JavaScript code on the gateway to publish MQTT property.publish commands instead of MQTT property.batch commands
- moving from property.current actions to mqtt.receive actions and parsing the raw property.publish JSON
    - [YES] was able to demonstrate the creation of a second MQTT broker node connecting to deviceWISE IoT portal using a new application (app-name) token and publishing to the dcwater topic
        - Can get the raw JSON
        - Need to explore ways to extract pressure values
- [ ] for some reason, when you trigger on an mqtt.receive event, configuring a property.current action now seems to require both a thing keys and a property key. We cannot write a separate trigger every time a new gateway is deployed.
- [ ] Why does Gephard recommend changing from property.batch to property.publish and also change from property.current actions to mqtt.receive actions in the cloud trigger?

This approach was recommended by Gerhard Lapp at Telit to address the shortcomings of the previous approach. It would alter how we get data from the gateway to the deviceWISE IoT portal

- [-] It requires changing JavaScript code on the gateway, and DC Water already has a gateway. If needed, we can update the DC Water's software using Multitech's DeviceHQ.
- [-] And it requires an additional API call to Telit for each packet received from the xDot

deviceWISE IoT Portal as an MQTT Relay

This approach merely uses the deviceWISE IoT portal as an MQTT broker. We would need to develop an MQTT client elsewhere and subscribe to the desired topic. In that case, the

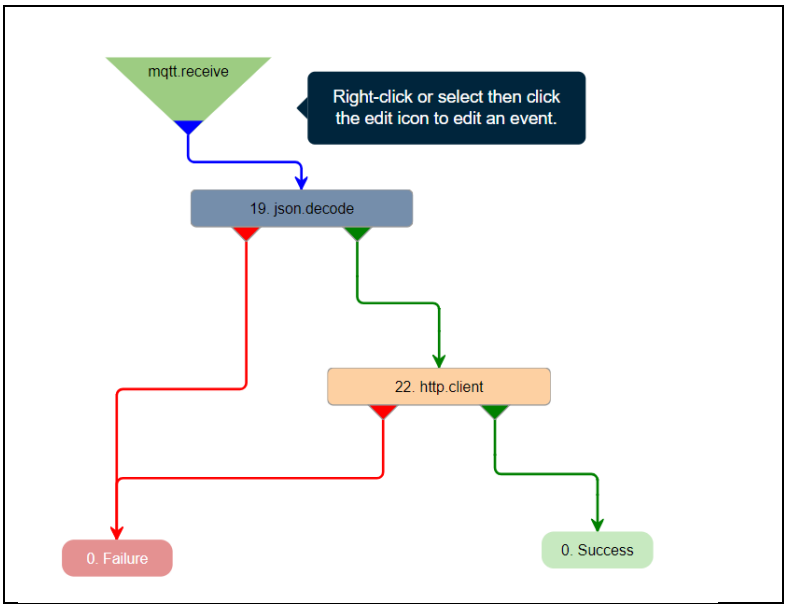


Figure 15 - deviceWISE Cloud Trigger to POST using REST API

DC Water subscribes to deviceWISE Topic

We could publish two times, one publish to topic=api to record all our values on the deviceWISE portal, and publish a second time to topic=dcwater. DC Water could subscribe directly to the MQTT broker at deviceWISE.

- DC Water would need to become a deviceWISE customer
- This increase cost by adding one API call per pressure reading (from 2 to 3)

Priorities

1. Have Telit fix the property.current Cloud Trigger action (we can discuss what they probably need to do for this one)
2. Get DC Water to agree to receive pressure reading values one-at-a-time over the REST API. This means we would POST five times to send the pressure readings: sum, number, minimum, maximum and average (only two of sum, number and average need to be sent)
3. We can POST directly from gateway?
4. How about a better API for DC Water – like JAVA classes?
5. Just have DC Water subscribe to a deviceWISE topic directly. We could publish twice: once for all our data to be stored on the deviceWISE IoT Portal, and the second time for DC Water to subscribe to only pressure readings (extra API call)

AWS

- Installed an AWS MQTT IoT library for Node-RED on 3/9/2018
    - https://flows.nodered.org/node/node-red-contrib-aws-iot-hub
    - Provide three nodes
        - aws-mqtt-out (to publish to a topic)
        - aws-mqtt-in (to publish to a topic)
        - aws-thing
- https://www.npmjs.com/package/aws-iot-device-sdk
    - npm install aws-iot-device-sdk
- node-RED log file: /var/log/app/node-red.log
-

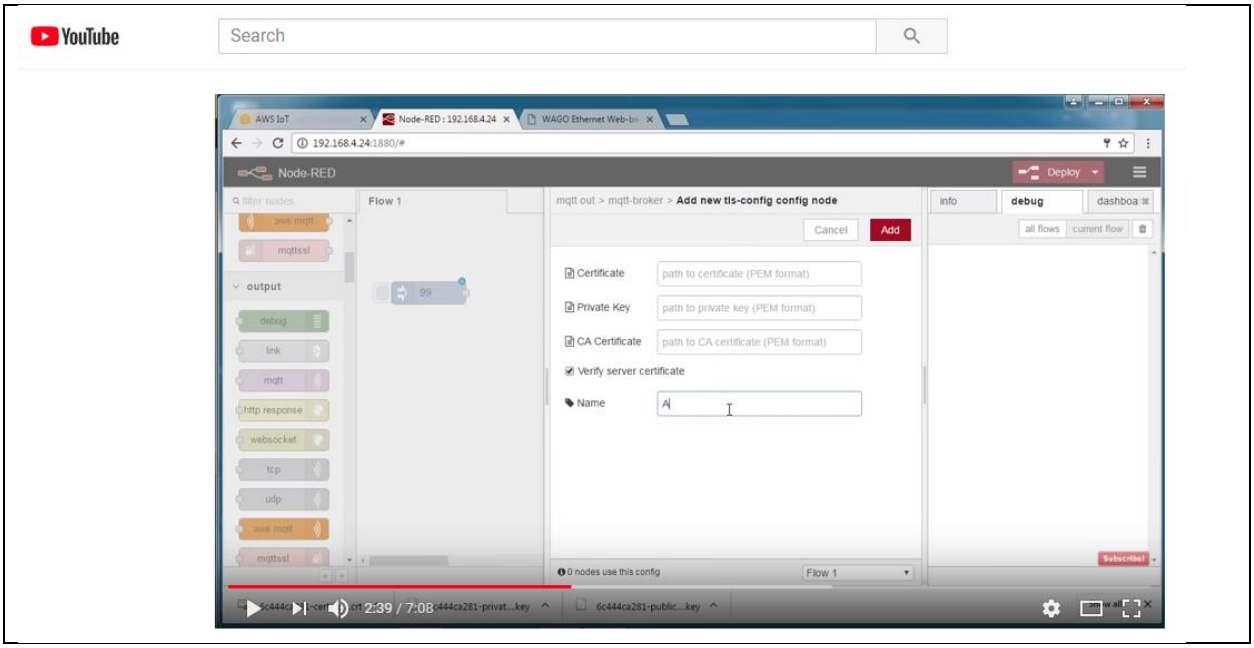
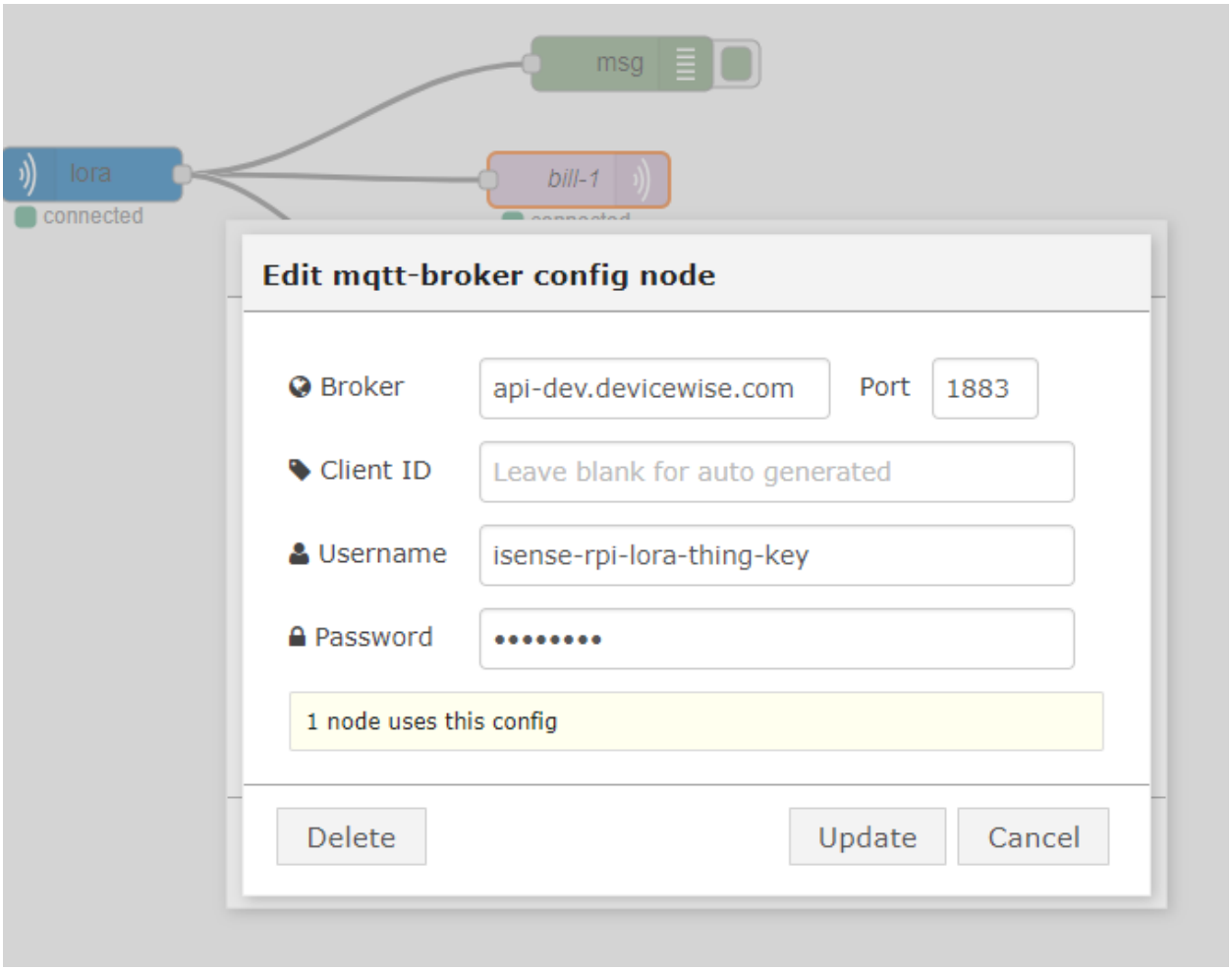


Figure 16 - Node-RED features currently not supported by Conduit AEP firmware 1.4.5


msg
lora
connected
bill-1
Edit mqtt-broker config node
Broker
api-dev.devicewise.com
Port
1883
Client ID
Leave blank for auto generated
Username
isense-rpi-lora-thing-key
Password
••••••••
1 node uses this config
Delete
Update
Cancel

Testing

deviceWISE (dW)

- use REST API for POST to get SessionId
- use REST API for GET and dW *property.current* command
    - https://help.devicewise.com/display/ARG/property.current

Issues/Questions/Risks

	Item	Pri	Status/Resolution
1	Sometimes the Conduit gateway does not accept new packets, especially after overnight. Packets are at least not showing up in the Node-RED debug window.	H	
2	Thing Identity becomes locked intermittently on deviceWISE IoT AEP portal – required manual unlocking to accept additional property.publish or property.batch value updates.	H	
3	Re-evaluate I-SENSE LoRa packet requirements for data, commands and diagnostics.	M	
4	Investigate the Multitech mlinux library for xDot	L	
5	Think about LoRa emulator/simulator	L	
6	Why does IP67 Conduit have a LoRa node-id in addition to its node-id? The indoor Conduit has only one node-id.		
7	deviceWISE: maybe there should be a separate Application and Thingkey for the Conduit gateway, since the property.batch command will contain the Thingkey for the intended Thing?		
8	CH200 using mac address but others using serial number!	H	
9	CH20o every other trigger produces non-parseable results	M	

Gear Summary

Name	Current IP	Model	B/O	Serial #	Node ID	IMEI	Network	Passphrase	FSB
Blue Conduit	10.15.17.71 10.15.13.100	MTCDT-LVW2	246A	19574654	00:08:00:4A:3D:AA	351999090585413	isensecond uit	testtest	7
White Conduit -1 (IP67)	10.15.17.119 10.15.13.96	MTCDTIP-LAT1	266A-915	19580877	00:08:00:4A:3E:93	358942053755279	ip67conduit	19580877	3
White Conduit -2 (IP67)	10.15.12.123 10.15.13.151	MTCDTIP-LAT1	266A-915	19618447	00:08:00:4A:43:9F	358942053753613	MTCDT-19618447	19618447	1
xDOT (Jiannan)	N/A	MTXDOT-UFL		19368204	00:80:00:00:04:00:4B:A0				
xDOT (Bill)	N/A	MTMDK-XDOT-NA1	A00	19380493	00:80:00:00:04:00:4B:93				
xDOT	N/A	MTXDOT-NA1	A00	19578811	00:80:00:00:04:00:89:4A				
xDOT	N/A				00:80:00:00:04:00:89:4B				
xDOT	N/A				00:80:00:00:04:00:89:48				
xDOT	N/A				00:80:00:00:04:00:89:45				