
IBM i Meets IoT

Michael Dawson

Node.js Lead for Red Hat and IBM

Jesse Gorzinski

Business Architect

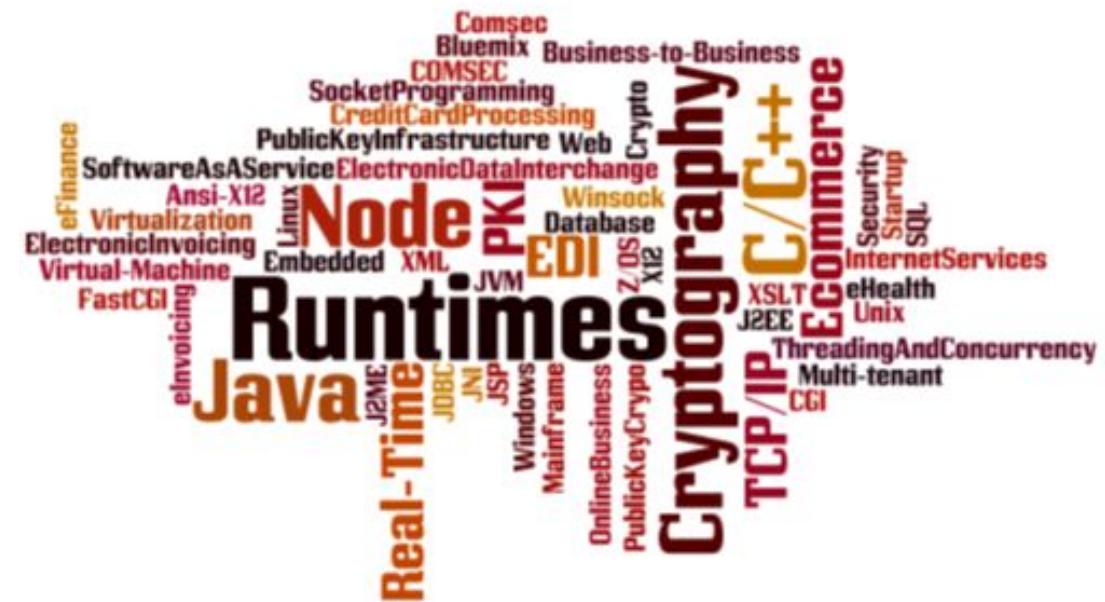
Agenda Key:

About Michael Dawson

Node.js lead for Red Hat and IBM



- Active Node.js community member
 - Collaborator
 - Node.js Technical Steering Committee TSC Chair
 - Community Committee member
 - Working group(s) member/leadership
- Active OpenJS Foundation member
 - Voting Cross Project Council Member
 - Node.js Community Director 2019-2021

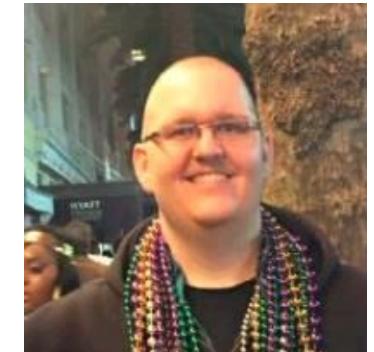


- Twitter: @mhdawson1
- GitHub: @mhdawson
- LinkedIn: <https://www.linkedin.com/in/michael-dawson-6051282>

About Jesse

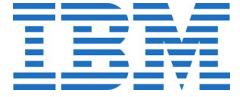
Business Architect of Open Source on i

- Leader of development teams
- Owns IBM iOSS strategy



- Twitter: @IBMJesseG
- GitHub: @ThePrez
- Linkedin: <https://www.linkedin.com/in/ibmjesseg>

Agenda



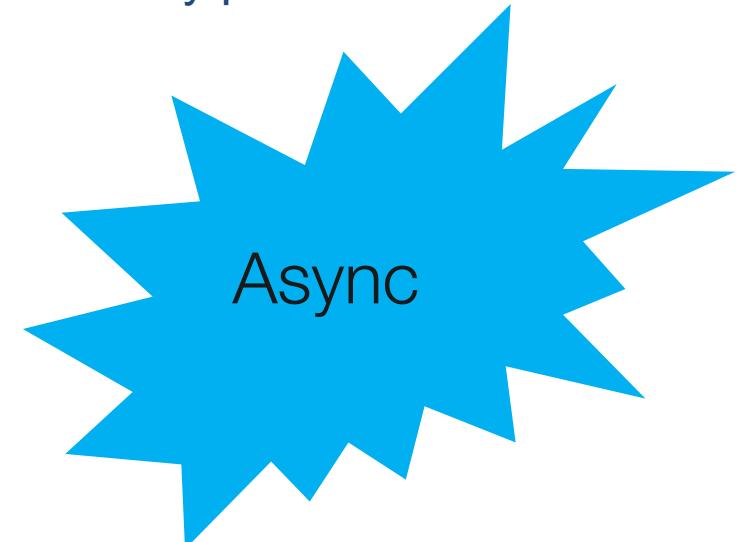
- Intro to IoT and MQTT
- Using MQTT with Node.js and IBM i
- Let's Look at some devices
- Anatomy of a simple MQTT Light and Temperature Sensor
- Consuming MQTT data on IBM i
- Leveraging the Cloud

- Internet of Things (IoT)
 - network of physically connected devices (things)
 - devices provide data
 - devices can be controlled
 - https://en.wikipedia.org/wiki/Internet_of_Things

MQTT Introduction

- MQTT (MQ Telemetry Transport)
 - lightweight publish/subscribe
 - small footprint
 - low bandwidth (minimum size is 2 bytes)
 - From <http://mqtt.org/>

“MQTT is a machine-to-machine (M2M)/"Internet of Things" connectivity protocol”



MQTT - History

- Invented in 1999
 - Andy Standford-Clark(IBM)
 - Arlen Nipper (Eurotech)
- IBM and Eurotech Donated MQTT to Eclipse Paho project in 2013 - <https://www.eclipse.org/paho/>
 - open-source client implementations
- Mosquitto broker also moved into the Eclipse foundation in 2013 -
<https://projects.eclipse.org/projects/technology.mosquitto>
 - open-source broker
- Version 3.11 is an OASIS standard
- ISO standard as of 2016 (IOS/IC 20922)
- Version v5.0 official as OASIS standard as of April 2019
https://en.wikipedia.org/wiki/Comparison_of_MQTT_implementations – which clients/brokers provide support

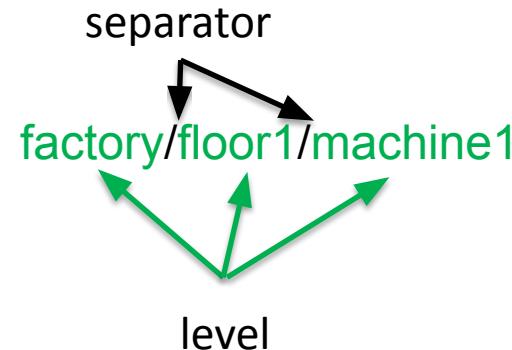
MQTT – Key terminology

- Client
 - Publishers and subscribers
 - Paho
 - mqtt.js (MQTT 5 support is experimental)
- Broker
 - Mosquitto
 - ActiveMQ
- Topic
 - Shared id to subscribe or publish on
- Message
 - Free form text
- QoS
 - Quality of Service (0-2)

MQTT Topics



- **Topics** are one or more **levels** separated by the topic level **separator**



- **Restrictions**
 - Must be at least one character
 - Case sensitive

- **Wildcards**

+ Matches one level

factory/+machine1 factory/floor1/machine1 (yes) factory/floor1/room1/machine1 (no)

matches multiple levels

only allowed at end

factory/# factory/floor1/machine1 (yes) factory/floor1/room1/machine1 (yes)

- 3 Levels
 - 0 – At most once
 - 1 – At least once
 - 2 – Exactly once
- Downgrade of QoS
 - Uses QoS of receiver, so downgrade may occur if sending used higher level
- More overhead for each level
- 0 is generally the default

Why IoT with IBM i?

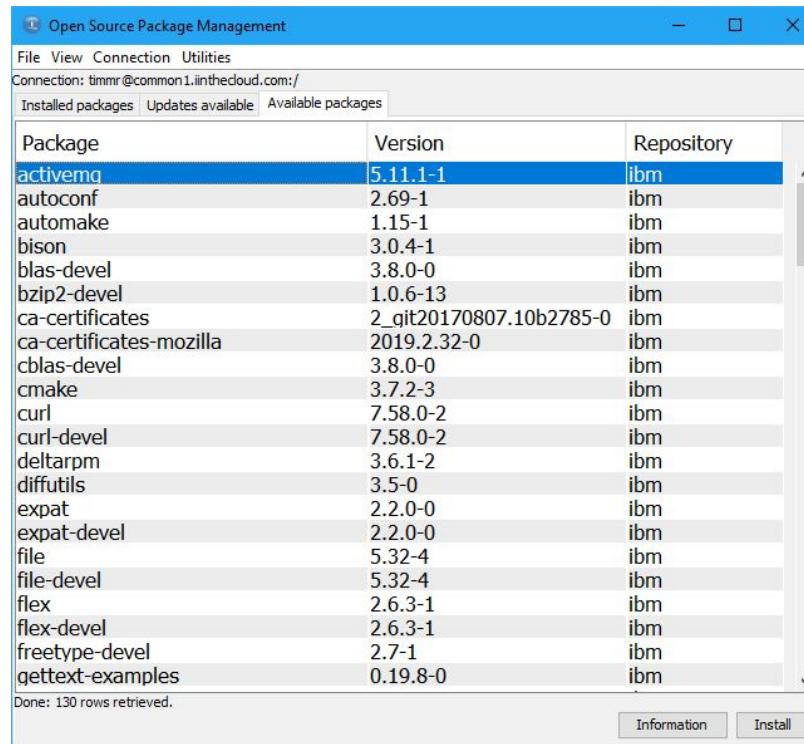


<https://www.ibm.com/it-infrastructure/us-en/resources/power/ibm-i-customer-stories/#/kj-trucking/>

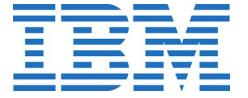


Using MQTT with IBM i

- ActiveMQ Install

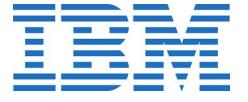


The screenshot shows the 'Package Installation' window. It starts with a message 'Setting up Install Process' followed by 'Resolving Dependencies'. It then lists the transaction steps: '--> Running transaction check', '--> Package activemq.noarch 0:5.11.1-1 will be installed', and '--> Finished Dependency Resolution'. Below this, it shows the 'Dependencies Resolved' section. The next section, 'Installing:', lists the package 'activemq' with version '5.11.1-1', architecture 'noarch', repository 'ibm', and size '41 M'. The 'Transaction Summary' section shows 'Install 1 Package'. It then details the download and installation process: 'Total download size: 41 M', 'Installed size: 47 M', 'Is this ok [Y/N]: y', 'Downloading Packages: activemq-5.11.1-1.ibmi7.2.noarch.rpm | 41 MB 00:08', 'Running Transaction Check', 'Running Transaction Test', 'Transaction Test Succeeded', 'Running Transaction', and 'Installing : activemq-5.11.1-1.noarch [#####] 1/1'.



Using MQTT with IBM i

- Start ActiveMQ
`activemq start`
- Configure
 - <http://activemq.apache.org/mqtt.html>
 - Default config in - /QOpenSys/pkg/lib/activemq/conf/
- Support
 - <http://ibm.biz/ibmi-oss-support>



MQTT install

npm install mqtt



Simple Client (client-local.js)

```
const fs = require('fs');
const path = require('path');
const mqtt = require('mqtt');

// setup mqtt
let mqttOptions;
mqttOptions = {
    key: fs.readFileSync(path.join(__dirname, 'mqttclient', '/client.key')),
    cert: fs.readFileSync(path.join(__dirname, 'mqttclient', '/client.cert')),
    ca: fs.readFileSync(path.join(__dirname, 'mqttclient', '/ca.cert')),
    clientId: 'simple-client',
    checkServerIdentity: function() { return undefined },
    rejectUnauthorized: false,
    username: '',
    password: ''
}

const mqttClient = mqtt.connect('mqtts:common1.iinthecloud.com:8883', mqttOptions);
mqttClient.on('connect', () => {
    console.log('connected');
    mqttClient.subscribe('onibmi/topic');
    mqttClient.on('message', (topic, message) => {
        console.log('message received topic (' + topic + ') message (' + message.toString() + ')');
    });
});
```



Simple Publisher (publisher-local.js)

```
const fs = require('fs');
const path = require('path');
const mqtt = require('mqtt');

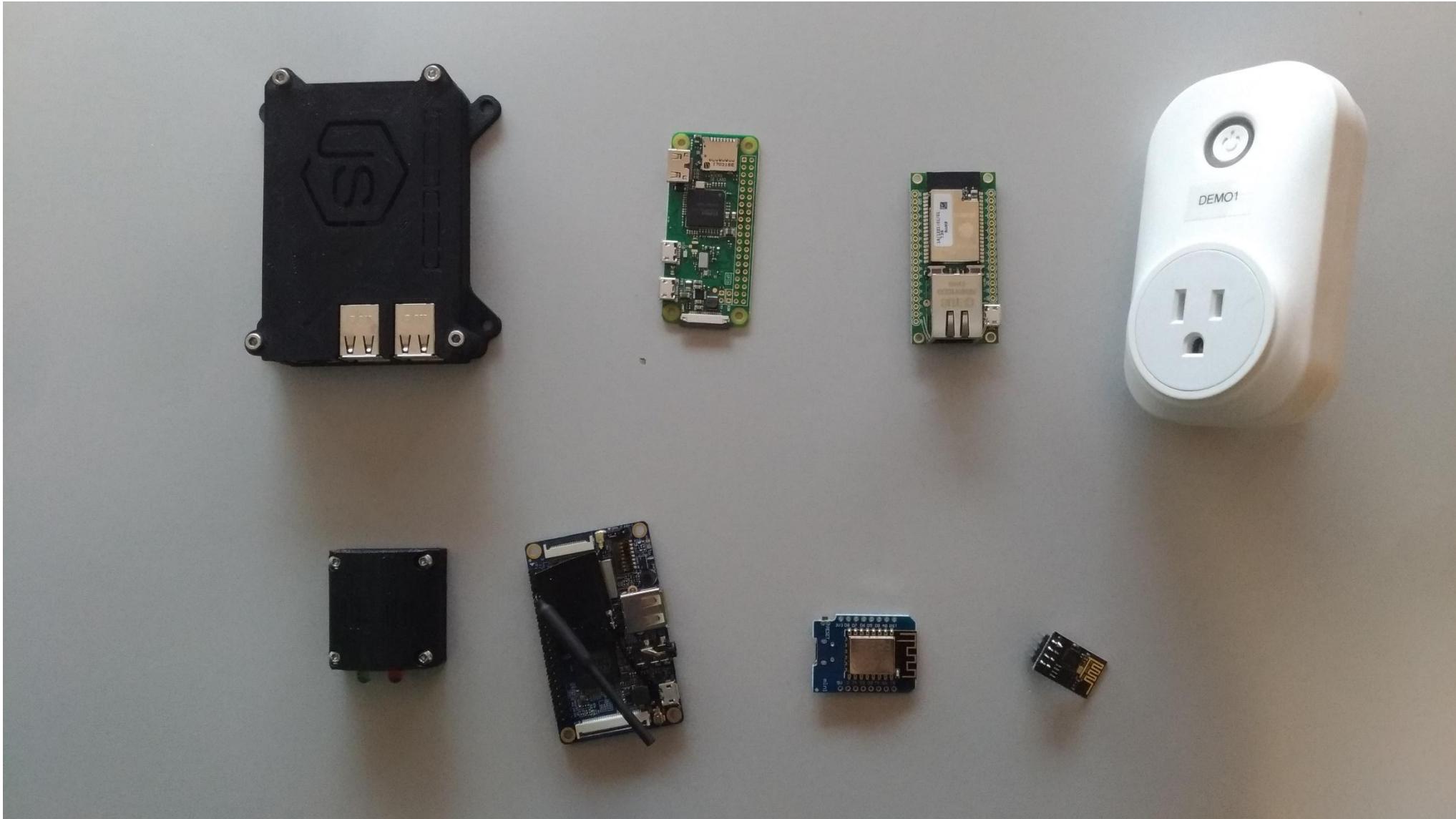
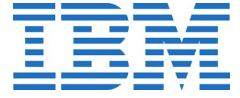
// setup mqtt
let mqttOptions;
mqttOptions = {
    key: fs.readFileSync(path.join(__dirname, 'mqttclient', '/client.key')),
    cert: fs.readFileSync(path.join(__dirname, 'mqttclient', '/client.cert')),
    ca: fs.readFileSync(path.join(__dirname, 'mqttclient', '/ca.cert')),
    clientId: 'simple publish',
    checkServerIdentity: function() { return undefined },
    rejectUnauthorized: false,
    username: '',
    password: ''
}

const mqttClient = mqtt.connect('mqtts:common1.iinthecloud.com:8883', mqttOptions);
mqttClient.on('connect', () => {
    console.log('connected');
    setInterval(() => {
        console.log('publishing');
        mqttClient.publish('onibmi/topic', 'hello world');
    }, 10000 );
});
```

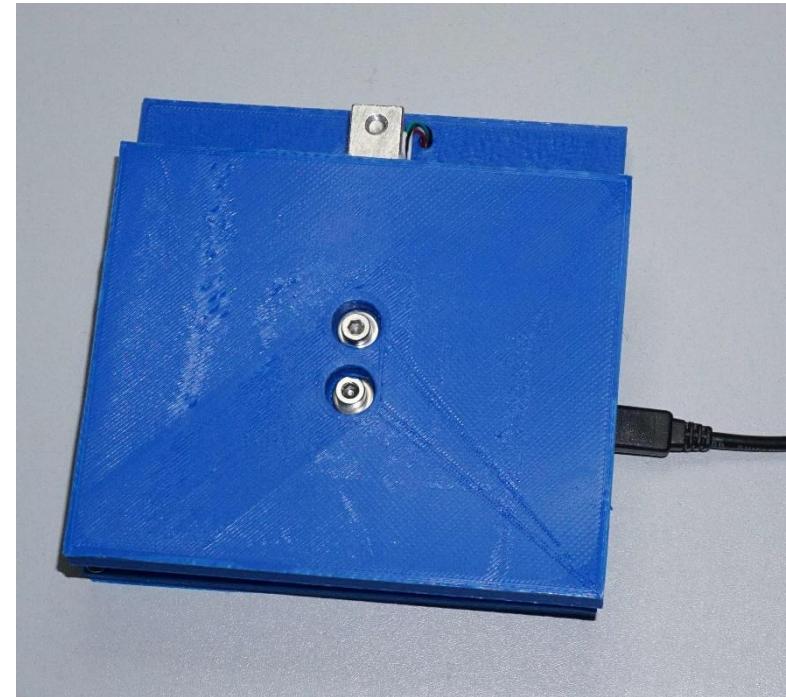
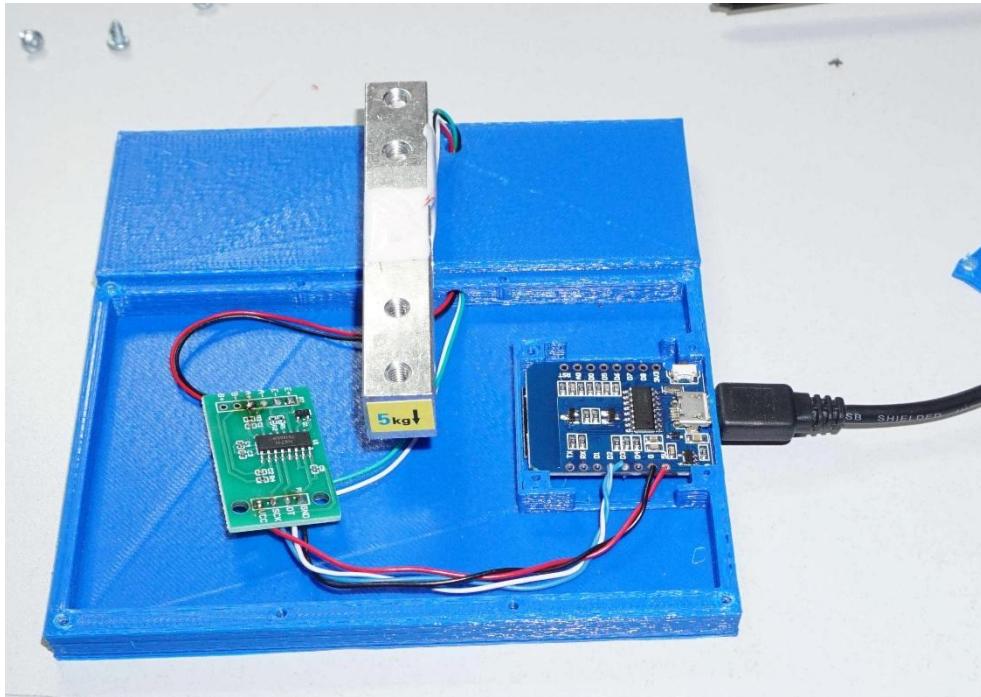
```
driveway@Common1:~/michael/mqttsamples$ node  
publish-local.js  
connected  
publishing  
publishing  
publishing  
publishing  
publishing  
publishing  
publishing  
publishing  
publishing  
publishing
```

```
driveway@Common1:~/michael/mqttsamples$ node client-local.js  
connected  
message received topic (onibmi/topic) message (hello world)  
message received topic (onibmi/topic) message (hello world)
```

Lets look at some Devices



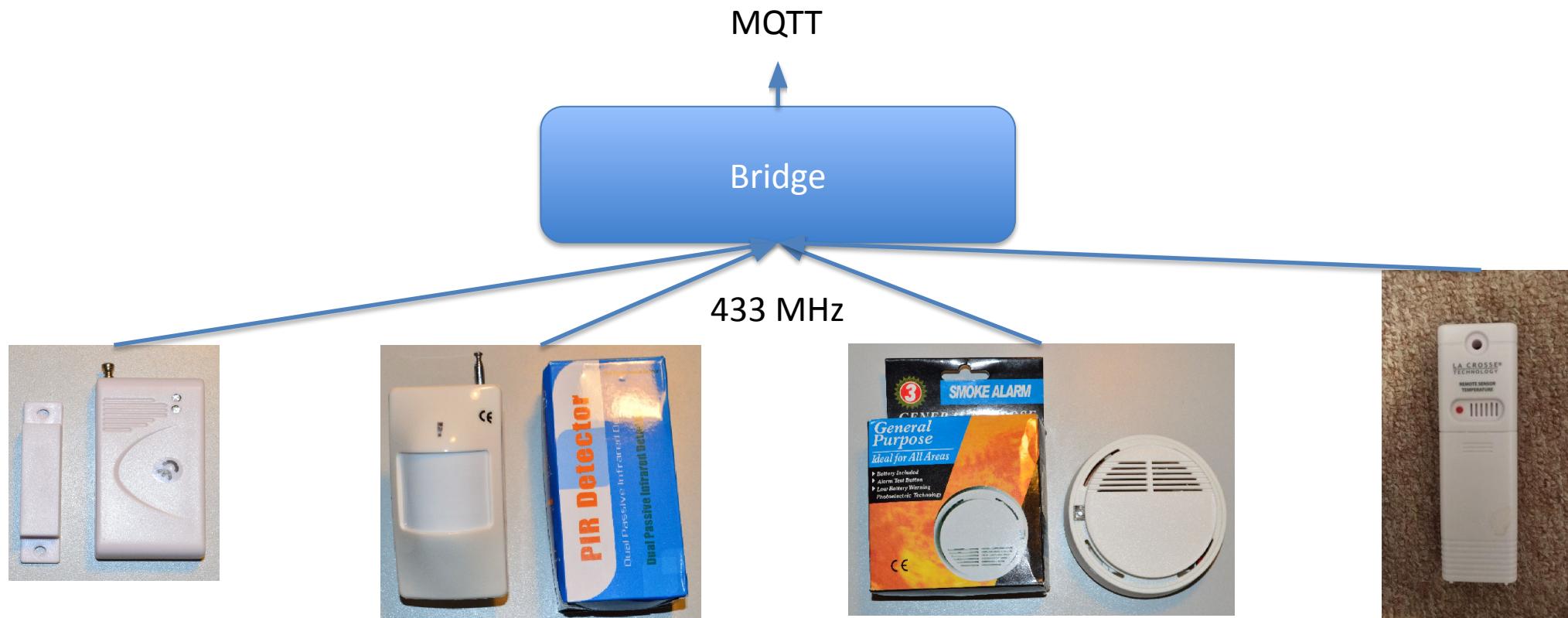
Lets look at some Devices



Other/Existing Devices

- Common approach is gateway or bridge
- As an example 433MHz to MQTT bridge

<https://github.com/mhdawson/arduino-esp8266/tree/master/Mqtt433Bridge>

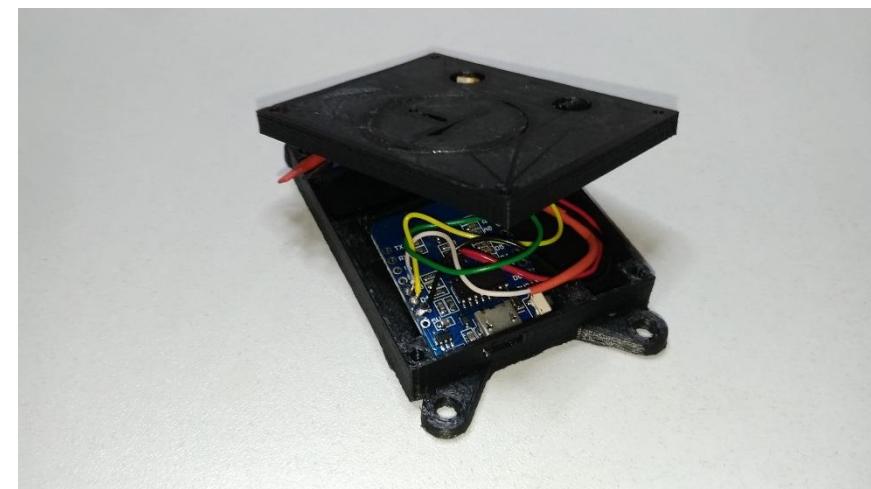
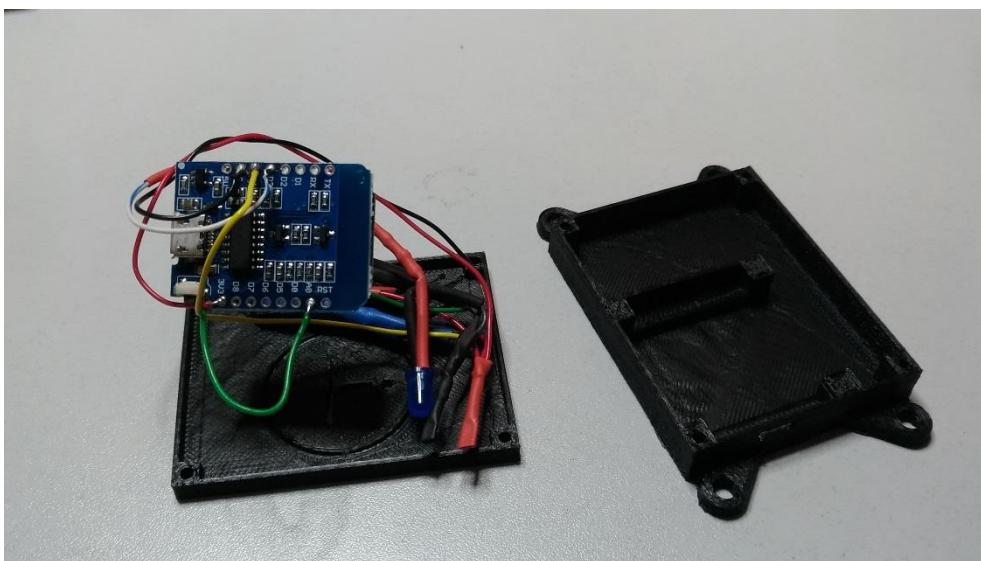


Anatomy of a Simple Device

- Maybe your business grows plants?
- **Temperature** and **Light** intensity through the day might be interesting?

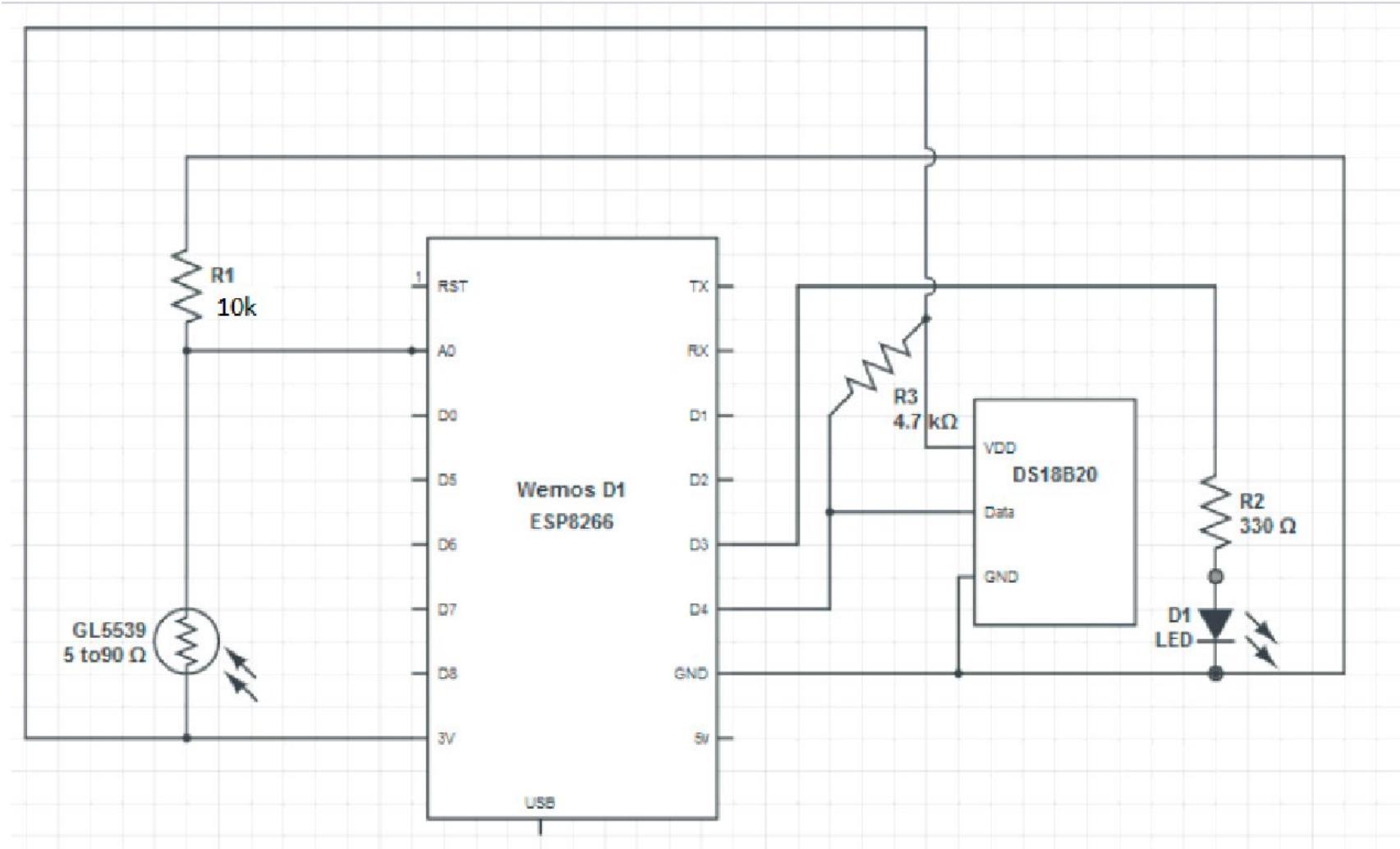
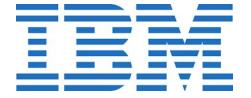


Anatomy of a Simple Device – Temp and Light Sensor



<https://github.com/mhdawson/arduino-esp8266/tree/master/TempAndLightSensor>

Anatomy of a Simple Device



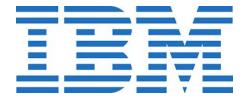
Anatomy of a Simple Device



```
5 #ifndef __SENSOR_CONFIG_H__
6 #define __SENSOR_CONFIG_H__
7 #define LIGHT_TOPIC "factory/1/light"
8 #define TEMP_TOPIC "factory/1/temp"
9 #define LED_TOPIC "factory/1/led"
10#endif

1 // wireless setup
2 #ifndef __WIRELESS_H__
3 #define __WIRELESS_H__
4 const char *ssid = "XXXXXXXXXXXX"; // cannot be longer than 32 characters!
5 const char *pass = "XXXXXXXXXX"; //
6 const char* mqttServerString = "XX.XX.XX.XX";
7 const uint16_t mqttServerPort = 8883;
8
9#define USE_CERTS
10unsigned char client_cert[] PROGMEM = {
11    0x30, 0x82, 0x03, 0x92, 0x30, 0x82, 0x02, 0x7a, 0x02, 0x09, 0x00, 0xcb,
12    .....
13};
14 unsigned int client_cert_len = YYYY;
15
16
17 unsigned char client_key[] PROGMEM = {
18    0x30, 0x82, 0x04, 0xa5, 0x02, 0x01, 0x00, 0x02, 0x82, 0x01, 0x01, 0x00,
19    .....
20};
21 unsigned int client_key_len = ZZZZ;
22#endif
```

Anatomy of a Simple Device



```
5 #include <Arduino.h>
6 #include <ESP8266WiFi.h>
7 #include <WiFiClientSecure.h>
8 #include <PubSubClient.h>
9 #include <OneWire.h>
10 #include <DallasTemperature.h>
11
12 // device specifics
13 #include "WirelessConfig.h"
14 #include "SensorConfig.h"
15
16 #define TRANSMIT_INTERVAL_SECONDS 60
17 #define MILLIS_IN_SECOND 1000
18 #define LOOP_DELAY 100
19
20 #define LED_PIN D3
21 #define LED_BLINK_TIME_SECONDS 2
22
23 #define MAX_MESSAGE_SIZE 100
24
25 #define LIGHT_PIN A0
26
27 #define DS18B20_PIN D4 // don't use D0 or D2 as can interfere with boot
28 OneWire ds(DS18B20_PIN);
29 DallasTemperature tempSensors(&ds);
```

RQ

Anatomy of a Simple Device

```
31  bool ledOn = true;
32  void toggleLED() {
33      if (ledOn) {
34          ledOn = false;
35          digitalWrite(LED_PIN, LOW);
36      } else {
37          ledOn = true;
38          digitalWrite(LED_PIN, HIGH);
39      }
40  }
41
42  void callback(char* topic, uint8_t* message, unsigned int length) {
43      if (strncmp((const char*)message,"on", strlen("on")) == 0) {
44          digitalWrite(LED_PIN, HIGH);
45          ledOn = true;
46      } else {
47          digitalWrite(LED_PIN, LOW);
48          ledOn = false;
49      }
50  };

```

Anatomy of a Simple Device

```
52  WiFiGenericClass wifi;
53
54 #ifdef USE_CERTS
55 // if certs are used the following must be defined in WirelessConfig.h
56 // unsigned char client_cert[] PROGMEM = {bytes in DER format};
57 // unsigned int client_cert_len = 918;
58 // unsigned char client_key[] PROGMEM = {bytes in DER format};
59 // unsigned int client_key_len = 1193;
60 //
61 // conversion can be done using
62 // openssl x509 -in cert -out client.cert -outform DER
63 // openssl rsa -in key -out client.key -outform DER
64 // and then using xxd to generate the required array and lengths
65 // see https://nofurtherquestions.wordpress.com/2016/03/14/making-an-esp8266-web-accessible/
66 // for more detailed info
67 WiFiClientSecure wclient;
68 #else
69 WiFiClient wclient;
70#endif
71
72 PubSubClient client(mqttServerString, mqttServerPort, callback, wclient);
73
74 int counter = 0;
75 char macAddress[] = "00:00:00:00:00:00";
```

Runs when MQTT message received



Anatomy of a Simple Device

```
77 void setup() {
78     delay(1000);
79
80     // Setup console
81     Serial1.begin(115200);
82     delay(10);
83     Serial.println();
84     Serial.println("Started");
85
86     pinMode(LED_PIN, OUTPUT);
87     digitalWrite(LED_PIN, HIGH);
88
89 #ifdef USE_CERTS
90     wclient.setCertificate_P(client_cert, client_cert_len);
91     wclient.setPrivateKey_P(client_key, client_key_len);
92 #endif
93
94     // turn off the Access Point as we are not using it
95     wifi.mode(WIFI_STA);
96     WiFi.begin(ssid, pass);
97
98     // first reading always seems to be wrong, read it early and
99     // throw it away
100    tempSensors.requestTemperatures();
101
102    // get the mac address to be used as a unique id for connecting to the mqtt server
103    byte macRaw[6];
104    WiFi.macAddress(macRaw);
105    sprintf(macAddress,
106            "%02.2X:%02.2X:%02.2X:%02.2X:%02.2X:%02.2X",
107            macRaw[0],
108            macRaw[1],
109            macRaw[2],
110            macRaw[3],
111            macRaw[4],
112            macRaw[5]);
113 }
```

Anatomy of a Simple Device

```
115 void loop() {  
116     client.loop();  
117     delay(LOOP_DELAY);  
118  
119     // make sure we are good for wifi  
120     if (WiFi.status() != WL_CONNECTED) {  
121         Serial.print("Connecting to ");  
122         Serial.println(ssid);  
123         WiFi.reconnect();  
124  
125         if (WiFi.waitForConnectResult() != WL_CONNECTED) {  
126             Serial.println("Failed to reconnect WIFI");  
127             Serial.println(WiFi.waitForConnectResult());  
128             delay(1000);  
129             return;  
130         }  
131     }  
132  
133  
134     if (!client.connected()) {  
135         if (client.connect(macAddress)) { ← Make sure ID is Unique !  
136             Serial.println("mqtt connected:");  
137             Serial.println(macAddress);  
138             Serial.println("\n");  
139             client.subscribe(LED_TOPIC); ← Subscribe to topics of interest  
140         }  
141     }  
}
```

Anatomy of a Simple Device

```
143     counter++;
144     if (counter == (TRANSMIT_INTERVAL_SECONDS * (MILLIS_IN_SECOND/LOOP_DELAY))) {
145         Serial.println("Sending");
146
147         // don't send out temperature too often as we'll get
148         // incorrect values if we sample too often
149         char tempMessage[MAX_MESSAGE_SIZE];
150         char floatBuffer[10];
151         tempSensors.requestTemperatures();
152         float currentTemp = tempSensors.getTempCByIndex(0);
153         sprintf(tempMessage, MAX_MESSAGE_SIZE, "0, 0 - temp: %s",
154                 dtostrf(currentTemp, 4, 2, floatBuffer));
155         client.publish(TEMP_TOPIC, tempMessage);
156
157         char lightMessage[MAX_MESSAGE_SIZE];
158         int lightValue = analogRead(LIGHT_PIN);
159         sprintf(lightMessage, MAX_MESSAGE_SIZE, "0, 0 - light: %d", lightValue);
160         client.publish(LIGHT_TOPIC, lightMessage);
161
162         toggleLED();
163         counter = 0;
164     } else if (counter == (LED_BLINK_TIME_SECONDS * (MILLIS_IN_SECOND/LOOP_DELAY))) {
165         toggleLED();
166     }
167 }
```

Publish data

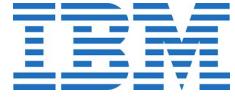
Anatomy of Simple Device

- Don't like C++? Can use JavaScript as well
 - <https://github.com/mhdawson/espruino-stuff/blob/master/SmartPlug.js>

```
54 client.on('publish', function(message) {  
55   console.log(message);  
56   if (message.topic === (devicePrefix + '/power')) {  
57     if (message.message === 'on') {  
58       powerState = 1;  
59     } else if (message.message === 'off') {  
60       powerState = 0;  
61     }  
62     digitalWrite(powerPin, powerState);  
63     console.log('Power state:' + powerState);  
64   } else if (message.topic === (devicePrefix + '/led')) {  
65     clearLedFlashTimer();  
66     if (message.message === 'on') {  
67       ledState = 1;  
68     } else if (message.message === 'off') {  
69       ledState = 0;  
70     } else if (message.message.substr(0, 'flash'.length) === 'flash') {  
71       try {  
72         timeout = message.message.split(':')[1];  
73         startFlashTimer(timeout);  
74       } catch (err) {  
75         console.log(err);  
76       }  
77     }  
78     digitalWrite(ledPin, (ledState + 1) % 2);  
79     console.log('Led state:' + ledState);  
80   } else if (message.topic === (devicePrefix + '/query_state')) {  
81     client.publish(devicePrefix + '/state/power', powerState);  
82     client.publish(devicePrefix + '/state/led', ledState);  
83   }  
84 });
```



Consuming Sensor Data on IBM i (client-sensor.js)



```
const fs = require('fs');
const path = require('path');
const mqtt = require('mqtt');

// setup mqtt
let mqttOptions;
mqttOptions = {
    key: fs.readFileSync(path.join(__dirname, 'mqttclient', '/client.key')),
    cert: fs.readFileSync(path.join(__dirname, 'mqttclient', '/client.cert')),
    ca: fs.readFileSync(path.join(__dirname, 'mqttclient', '/ca.cert')),
    clientId: 'simple-client',
    checkServerIdentity: function() { return undefined },
    rejectUnauthorized: false,
    username: '',
    password: ''
}

const mqttClient = mqtt.connect('mqtts:common1.iinthecloud.com:8883', mqttOptions);
mqttClient.on('connect', () => {
    console.log('connected');
    mqttClient.subscribe('factory/1/light');
    mqttClient.subscribe('factory/1/temp');
    mqttClient.on('message', (topic, message) => {
        console.log('message received topic (' + topic + ') message (' + message.toString() + ')');
    });
});
```

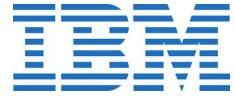


Controlling Sensor on IBMi (control.js)

```
const fs = require('fs');
const path = require('path');
const mqtt = require('mqtt');

// setup mqtt
let mqttOptions;
mqttOptions = {
    key: fs.readFileSync(path.join(__dirname, 'mqttclient', '/client.key')),
    cert: fs.readFileSync(path.join(__dirname, 'mqttclient', '/client.cert')),
    ca: fs.readFileSync(path.join(__dirname, 'mqttclient', '/ca.cert')),
    clientId: 'control client',
    checkServerIdentity: function() { return undefined },
    rejectUnauthorized: false,
    username: '',
    password: ''
}

const mqttClient = mqtt.connect('mqtts:common1.iinthecloud.com:8883', mqttOptions);
mqttClient.on('connect', () => {
    console.log('connected');
    mqttClient.publish('factory/1/led', process.argv[2], () => {
        setTimeout( () => {
            process.exit(0);
        }, 2000);
    });
});
```



Consuming Sensor Data on IBM i – Store to DB

```
const { DBPool } = require('idb-pconnector');
const pool = new DBPool();

async function setupDb() {
  try {
    await pool.prepareExecute('CREATE SCHEMA JESSEGIOT');
  }catch(err) {
    if(err.stack.includes('SQLSTATE=42710')) {
      console.log('schema already exists');
    } else {
      console.log('error: '+err.stack);
    }
  }
  try {
    await pool.prepareExecute(`CREATE OR REPLACE TABLE JESSEGIOT.IOT_RECORDS (
      DEVICE VARCHAR(80) ALLOCATE (10) CCSID 1208 NOT NORMALIZED NOT NULL NOT HIDDEN,
      SENSORVALUE DECIMAL(7, 2) NOT NULL NOT HIDDEN,
      SENSORTIME TIMESTAMP(6) GENERATED ALWAYS FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP
      NOT NULL NOT HIDDEN
    )
    NOT VOLATILE UNIT ANY KEEP IN MEMORY NO`);

  }catch(err) {
    console.log('error: '+err.stack);
  }
  console.log('Database setup complete!');
}
```



Consuming Sensor Data on IBM i – Store to DB

```
const mqttClient = mqtt.connect('mqtts:common1.iinthecloud.com:8883', mqttOptions); mqttClient.on('connect', () => {
  console.log('connected');
  mqttClient.subscribe('factory/1/light');
  mqttClient.subscribe('factory/1/temp');
  mqttClient.on('message', (topic, message) => {
    let value = message.toString().replace(/\.*:/g, '').replace(/[^\d.]+/g, '');
    pool.prepareExecute('insert into JESSEGIOT.IOT_RECORDS(device, sensorvalue) values(?, ?)', [topic, value]);
    console.log('message received topic (' + topic + ') message (' + message.toString() + ')');
  });
});
```

Consuming Sensor Data on IBM i – It's here

The screenshot shows the 'Run SQL Scripts' interface in the IBM i SQL Scripts tool. The title bar indicates the script is named 'iot.sql' and is running on 'common1.iinthecloud.com(lhost)'. The menu bar includes File, Edit, View, Run, VisualExplain, Monitor, Options, Connection, Tools, and Help. Below the menu is a toolbar with various icons. The main area contains two SQL queries:

```
1 -- description: Temperature
2 SELECT *
3   FROM JESSEGIOT.IOT_RECORDS
4  WHERE DEVICE LIKE '%temp'
5  ORDER BY SENSORTIME DESC
6  LIMIT 50;
7
8 -- description: Light
9 SELECT *
10  FROM JESSEGIOT.IOT_RECORDS
11 WHERE DEVICE LIKE '%light'
12 ORDER BY SENSORTIME DESC
13 LIMIT 50;
14
```

Below the queries is a table displaying the retrieved sensor data:

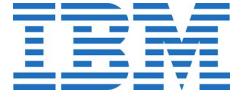
DEVICE	SENSORVALUE	SENSORTIME
factory/1/temp	24.75	2019-04-12 16:07:05.510507
factory/1/temp	24.75	2019-04-12 16:06:35.337323
factory/1/temp	24.75	2019-04-12 16:06:05.127732
factory/1/temp	24.75	2019-04-12 16:05:34.939690
factory/1/temp	24.75	2019-04-12 16:05:04.754581
factory/1/temp	24.75	2019-04-12 16:04:34.556798
factory/1/temp	24.75	2019-04-12 16:04:04.355322

At the bottom of the table, it says 'Done: 43 rows retrieved.'

Below the table are tabs for 'Temperature', 'Light', 'Messages', and 'Global Variables and Special Registers'. The 'Temperature' tab is selected.

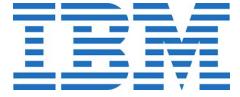
The status bar at the bottom indicates the connection details: 'Connected to relational database lhost on common1.iinthecloud.com as TIMMR - 060877/QUSER/QZDASOINIT using JDBC configuration 'Common'

Demo

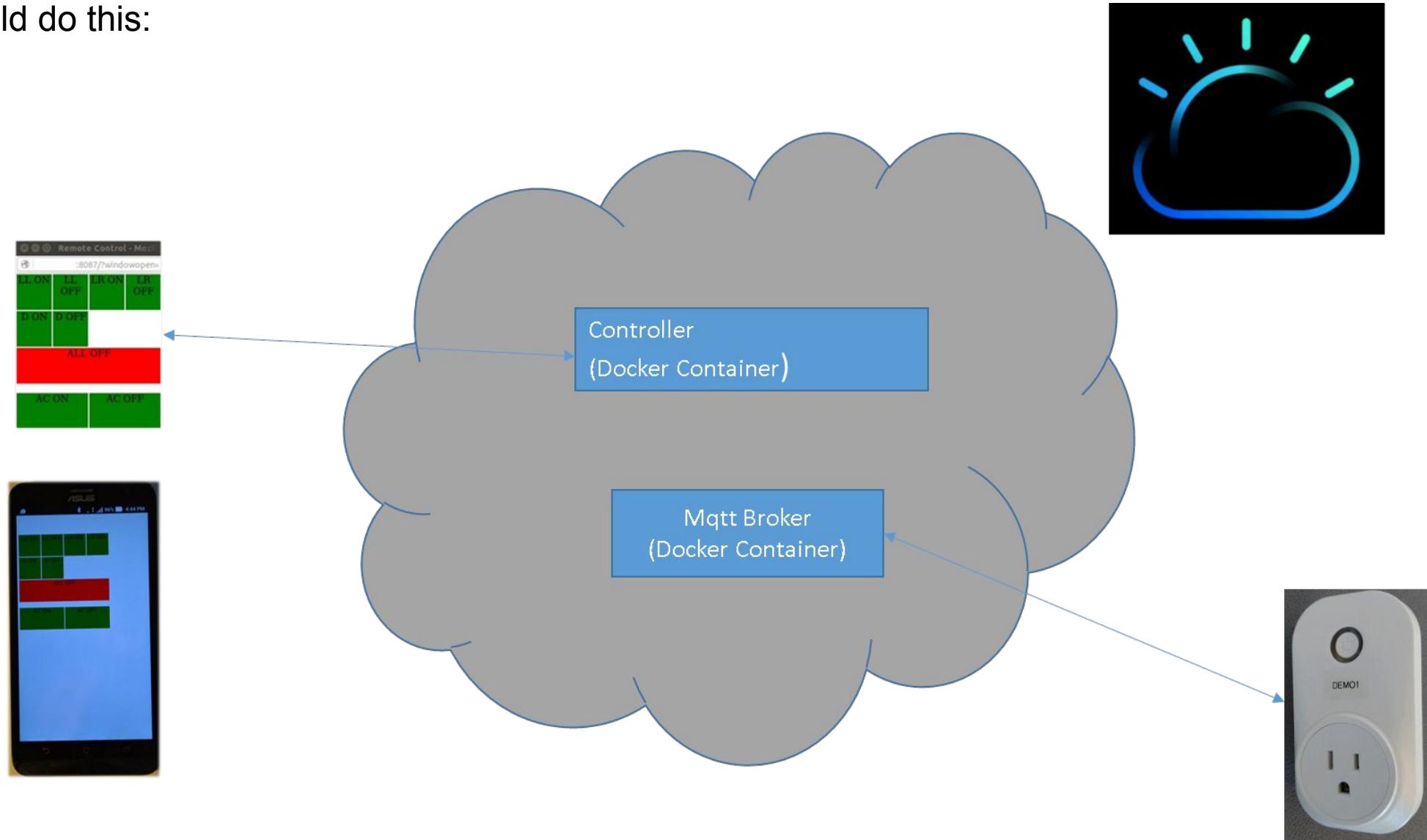


- Live Device data flow

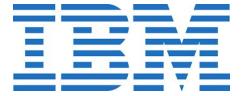
Leveraging the Cloud



- Could do this:



Leveraging the Cloud – Even Better



- Cloud Based Service
 - Don't worry about infrastructure
 - Get started fast
 - Easy visualization

Internet of Things



Internet of Things Platform

Lite • IBM

This service is the hub of all things IBM IoT, it is where you can set up and manage your connected devices so that your apps can access their live a...



AT&T Flow Designer

Third Party

Design, Build and Deploy IoT Solutions in Minutes



AT&T IoT Data Plans

Third Party

Launch your IoT product fast with IoT data plans



Bosch IoT Rollouts

Third Party

Rollout software and firmware updates to devices



UnificationEngine

Third Party

Intelligent IoT messaging for all H2M communications.

Leveraging the Cloud

```
diff --git a/TempAndLightSensor/SensorConfig.h b/TempAndLightSensor/SensorConfig.h
index 4459876..b33e08d 100644
--- a/TempAndLightSensor/SensorConfig.h
+++ b/TempAndLightSensor/SensorConfig.h
@@ -4,7 +4,7 @@
#ifndef __SENSOR_CONFIG_H__
#define __SENSOR_CONFIG_H__
-#define LIGHT_TOPIC "factory/1/light"
-#define TEMP_TOPIC "factory/1/temp"
-#define LED_TOPIC "factory/1/led"
+#define LIGHT_TOPIC "iot-2/evt/light/fmt/json"
+#define TEMP_TOPIC "iot-2/evt/temp/fmt/json"
+#define LED_TOPIC "iot-2/cmd/led/fmt/txt"
#endif
diff --git a/TempAndLightSensor/TempAndLightSensor.ino b/TempAndLightSensor/TempAndLightSensor.ino
index 8acbbab..3185175 100644
--- a/TempAndLightSensor/TempAndLightSensor.ino
+++ b/TempAndLightSensor/TempAndLightSensor.ino
@@ -13,7 +13,7 @@
#include "WirelessConfig.h"
#include "SensorConfig.h"

#define TRANSMIT_INTERVAL_SECONDS 60
#define TRANSMIT_INTERVAL_SECONDS 30
#define MILLIS_IN_SECOND 1000
#define LOOP_DELAY 100

@@ -132,11 +132,12 @@
void loop() {

    if (!client.connected()) {
        if (client.connect(macAddress)) {
+       if (client.connect("dal3kr9:TempAndLightSensor:device2", MQTT_USERNAME, MQTT_PASSWORD)) {
            Serial.println("mqtt connected:");
            Serial.println(macAddress);
            Serial.println("\n");
            client.subscribe(LED_TOPIC);
+       } else {
+           Serial.println("Failed to connect to mqtt server\n");
        }
    }

@@ -150,13 +151,13 @@
void loop() {
    char floatBuffer[10];
    tempSensors.requestTemperatures();
    float currentTemp = tempSensors.getTempCByIndex(0);
-   sprintf(tempMessage, MAX_MESSAGE_SIZE, "0, 0 - temp: %s",
+   sprintf(tempMessage, MAX_MESSAGE_SIZE, "{ \"temp\": %s }",
           dtostrf(currentTemp, 4, 2, floatBuffer));
    client.publish(TEMP_TOPIC, tempMessage);

    char lightMessage[MAX_MESSAGE_SIZE];
    int lightValue = analogRead(LIGHT_PIN);
-   sprintf(lightMessage, MAX_MESSAGE_SIZE, "0, 0 - light: %d", lightValue);
+   sprintf(lightMessage, MAX_MESSAGE_SIZE, "{ \"light\": %d }", lightValue);
    client.publish(LIGHT_TOPIC, lightMessage);

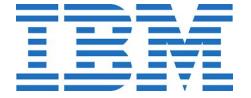
    toggleLED();
}
```

https://cloud.ibm.com/docs/services/IoT/reference/security?topic=iot-platform-connect_devices_apps_gw#connect_devices_apps_gw

```
-#define LIGHT_TOPIC "factory/1/light"
-#define TEMP_TOPIC "factory/1/temp"
-#define LED_TOPIC "factory/1/led"
+#define LIGHT_TOPIC "iot-2/evt/light/fmt/json"
+#define TEMP_TOPIC "iot-2/evt/temp/fmt/json"
+#define LED_TOPIC "iot-2/cmd/led/fmt/txt"

-    sprintf(lightMessage, MAX_MESSAGE_SIZE, "0,
0 - light: %d", lightValue);
+    sprintf(lightMessage, MAX_MESSAGE_SIZE, "{\"
\"light\": %d }", lightValue);
```

Leveraging the Cloud



IBM Watson IoT Platform

Browse Action Device Types

This table shows a summary of all devices that have been added. It can be filtered, organized, and searched on using different criteria. To get started, you can add devices by using the Add Device button, or by using API.

	Device ID	Device Type	Class ID	Date Added
device1	TempAndLightSensor	Device	Apr 5, 2019 6:46 PM	
device2	TempAndLightSensor	Device	Apr 5, 2019 6:41 PM	

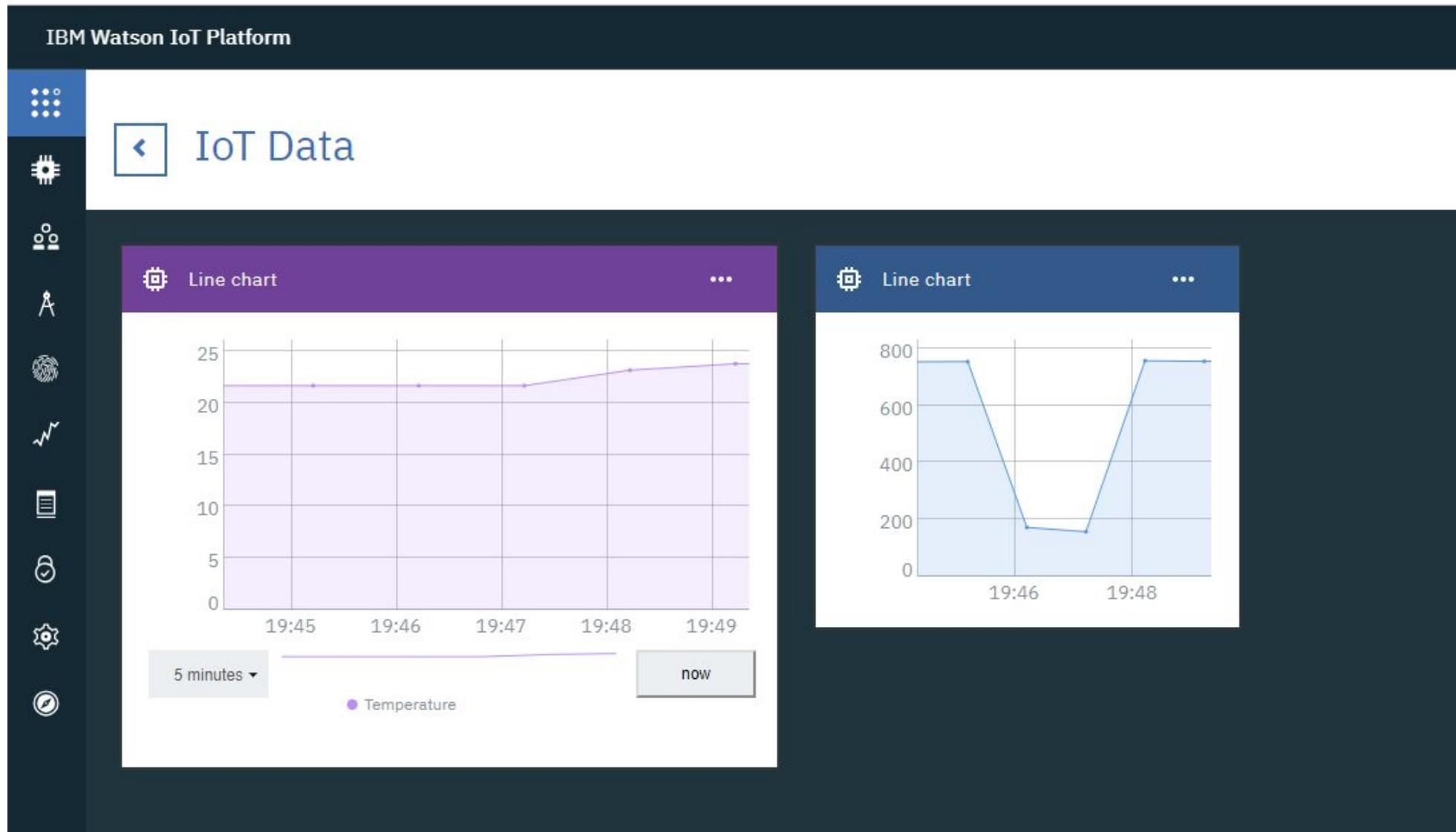
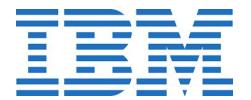
2 results

Identity Device Information Recent Events State Logs

Showing Raw Data | The recent events listed show the live stream of data that is coming and going from this device.

Event	Value	Format	Last Received
light	{"light":756}	json	15 minutes ago
temp	{"temp":21.81}	json	15 minutes ago
light	{"light":757}	json	16 minutes ago
temp	{"temp":21.75}	json	16 minutes ago
light	{"light":760}	json	17 minutes ago

Leveraging the Cloud





Consuming Data on IBMi (client-ibmcloud.js)

```
const fs = require('fs');
const path = require('path');
const mqtt = require('mqtt');

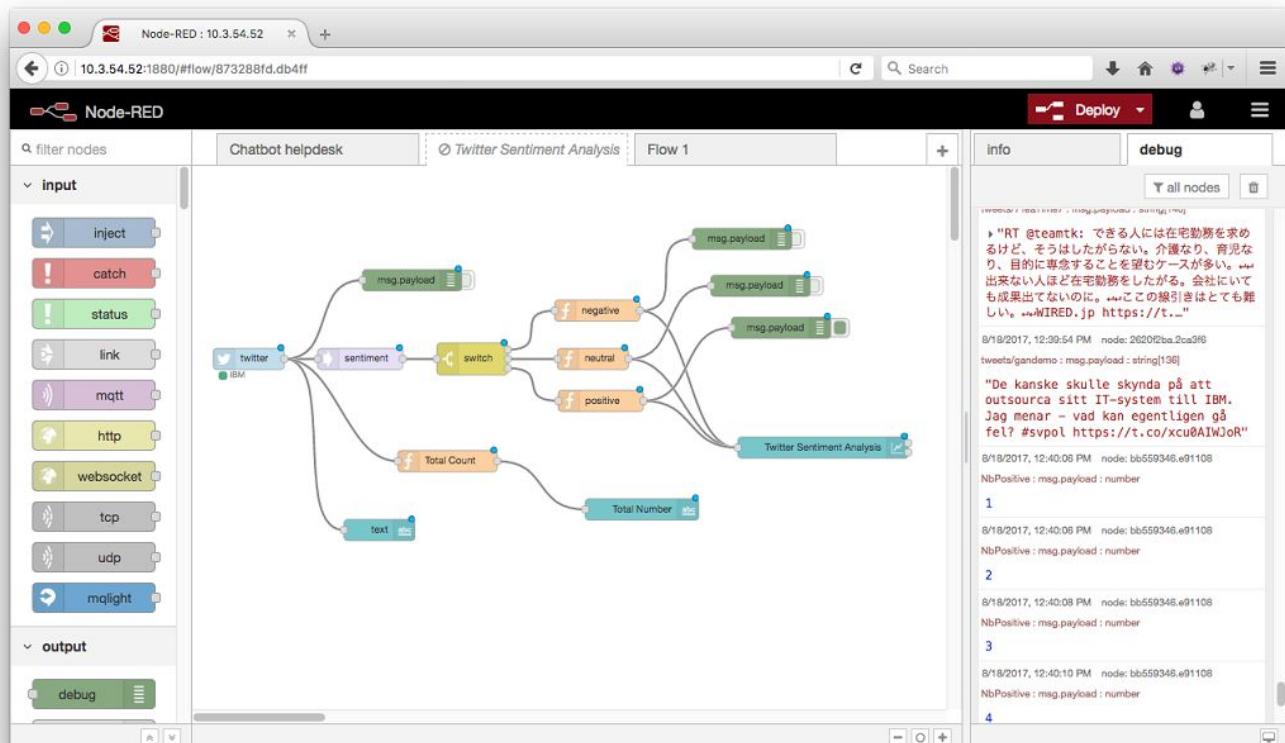
// setup mqtt
let mqttOptions;
mqttOptions = {
    key: fs.readFileSync(path.join(__dirname, 'mqttclient', '/client.key')),
    cert: fs.readFileSync(path.join(__dirname, 'mqttclient', '/client.cert')),
    ca: fs.readFileSync(path.join(__dirname, 'mqttclient', '/ca.cert')),
    clientId: 'A:XXXXXX:XXXXXXXXXX',
    checkServerIdentity: function() { return undefined },
    rejectUnauthorized: false,
    username: 'a-XXXXXX-XXXXXXXXXX',
    password: 'XXXXXXXXXXXXXXXXXX'
}

const mqttClient = mqtt.connect('mqtts:XXXXXX.messaging.internetofthings.ibmcloud.com:8883', mqttOptions);
mqttClient.on('connect', () => {
    console.log('connected');
    mqttClient.subscribe('iot-2/type/+id/+evt/+fmt/+');
    mqttClient.on('message', (topic, message) => {
        console.log('message received topic (' + topic + ') message (' + message.toString() + ')');
    });
});
```

Node-RED

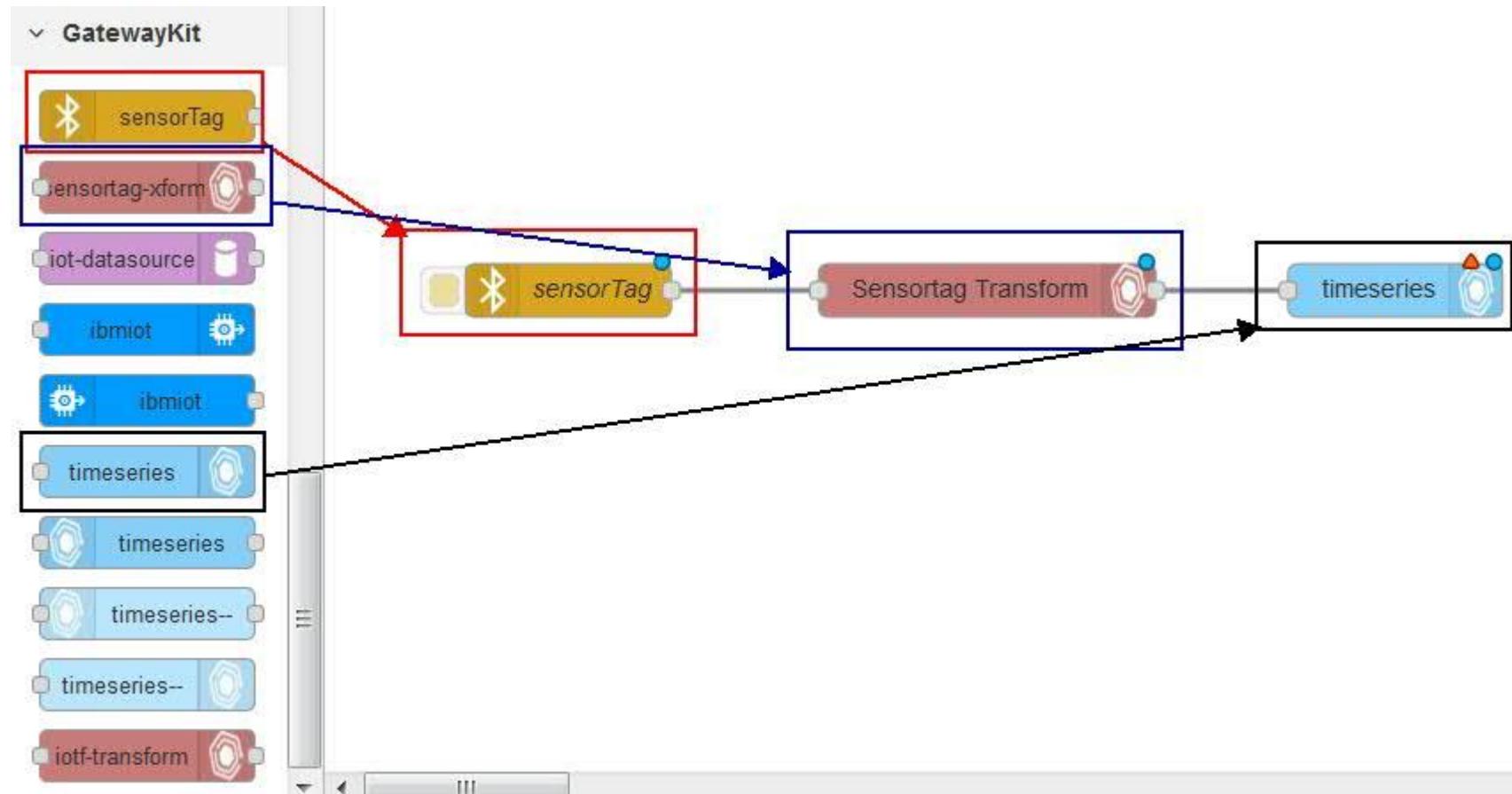


- Created by IBM
- Low-code graphical way to write flow-based programs
- <https://developer.ibm.com/tutorials/i-running-node-red/>
- Nodes can be any building block that can receive and send messages. Examples include:
 - Db2 for i queries
 - IoT devices
 - Web pages
 - Web APIs
 - Cloud services
 - Dashboards



Node-RED for IoT

- https://www.ibm.com/developerworks/community/blogs/cee6c09c-a315-4b04-ad14-57d6a60fa8bb/entry/Creating_A_Node_red_Flow_for_IoT?lang=en



Come to tomorrow's session!

The screenshot shows an IBM i terminal window titled 'A - COMMON1.IINTHECLOUD.COM'. The window has a toolbar at the top with icons for file operations like Open, Save, Print, and Help. Below the toolbar is a menu bar with File, Edit, View, Communication, Actions, Window, and Help. A tab bar shows two tabs: 'A - COMMON1.IINTHECLOUD.C...' and 'B - IDEVPHP.IDEV CLOUD.COM'. The main area is a terminal window titled 'Display Messages' with the following content:

```
System: COMMON1
Program . . . : *DSPMSG
Library . . . :
Delivery . . . : *NOTIFY

Queue . . . : DRIVEWAY
Library . . . : QUSRSYS
Severity . . . : 00

Type reply (if required), press Enter.

- From . . . : DRIVEWAY      03/02/21  15:18:36
  0, 0 - light: 84
- From . . . : DRIVEWAY      03/02/21  15:18:36
  0, 0 - temp: 25.25
- From . . . : DRIVEWAY      03/02/21  15:19:07
  0, 0 - light: 84
- From . . . : DRIVEWAY      03/02/21  15:19:07
  0, 0 - temp: 25.25
- From . . . : DRIVEWAY      03/02/21  15:19:37
  0, 0 - light: 84
- From . . . : DRIVEWAY      03/02/21  15:19:37
  0, 0 - temp: 25.31
- From . . . : DRIVEWAY      03/02/21  15:19:37
  0, 0 - light: 82

Bottom
```

At the bottom of the terminal window, there are function key definitions:

- F3=Exit
- F11=Remove a message
- F12=Cancel
- F13=Remove all
- F16=Remove all except unanswered
- F24=More keys

The status bar at the bottom right shows the date and time: 08/001.

```
context.addRoutes(new RouteBuilder() {
    @Override
    public void configure() {
        from("paho:factory/1/light?brokerUrl=ssl://localhost")
            .to("jt400://driveway:xxxxxx@localhost/qsys.lib/QUSRSYS.lib/DRIVEWAY.MSGQ?guiAvailable=false");
    }
});
context.addRoutes(new RouteBuilder() {
    @Override
    public void configure() {
        from("paho:factory/1/temp?brokerUrl=ssl://localhost")
            .to("jt400://driveway:xxxxxx@localhost/qsys.lib/QUSRSYS.lib/DRIVEWAY.MSGQ?&guiAvailable=false");
    }
});
```

Summary



- Intro to IoT and MQTT
- Using MQTT with Node.js and IBM i
- Lets Look at some devices
- Anatomy of a simple MQTT Light and Temperature Sensor
- Consuming MQTT data on IBMi
- Leveraging the Cloud
- Using Node-Red
- Apache Camel (learn more in the next session)



The screenshot shows a web browser window with the title bar "Apache ActiveMQ". The address bar contains "common1.iinthecloud.com:8161". The main content area displays the Apache ActiveMQ homepage. On the left, there's a large "ActiveMQ" logo with a feather graphic. On the right, there's a logo for "The Apache Software Foundation" with the URL "http://www.apache.org". Below the logo, a "Support" link is visible. In the center, a "Welcome to the Apache ActiveMQ!" message is displayed, followed by a list of actions: "What do you want to do next?", "Manage ActiveMQ broker", "See some Web demos (demos not included in default configuration)", and "Copyright 2005-2013 The Apache Software Foundation.". To the right of the welcome message, there's a sidebar titled "Useful Links" with links to "Documentation", "FAQ", "Downloads", and "Forums". At the bottom, it says "Graphic Design By Hiram".

The screenshot shows the ActiveMQ Administration Console interface. At the top, there's a navigation bar with links: Home, Queues, Topics, Subscribers, Connections, Network, Scheduled, and Send. On the right side, there's a sidebar with sections for Queue Views (Graph, XML), Topic Views (XML), Subscribers Views (XML), and Useful Links.

Topics

Name ↑	Number Of Consumers	Messages Enqueued	Messages Dequeued	Operations
ActiveMQ.Advisory.Connection	0	3	0	Send To Active Subscribers Active Producers Delete
ActiveMQ.Advisory.Consumer.Topic.factory.1.light	0	2	0	Send To Active Subscribers Active Producers Delete

The screenshot shows a web browser window displaying the Apache ActiveMQ Admin Topics page. The URL is `common1.iinthecloud.com:8161/admin/topics.jsp`. The page lists five topics:

Topic	Producers	Subscribers	Active Producers	Active Subscribers	Actions
ActiveMQ.Advisory.MasterBroker	0	1	0	1	Send To Active Subscribers Active Producers Delete
ActiveMQ.Advisory.Topic	0	3	0	3	Send To Active Subscribers Active Producers Delete
factory.1.light	0	0	0	0	Send To Active Subscribers Active Producers Delete
factory.1.temp	0	0	0	0	Send To Active Subscribers Active Producers Delete
onibmi.topic	0	36	0	36	Send To Active Subscribers Active Producers Delete

Active Durable Topic Subscribers										
Client ID	Subscription Name	Connection ID	Destination	Selector	Pending Queue Size	Dispatched Queue Size	Dispatched Counter	Enqueue Counter	Dequeue Counter	Operations
Offline Durable Topic Subscribers										
Client ID	Subscription Name	Connection ID	Destination	Selector	Pending Queue Size	Dispatched Queue Size	Dispatched Counter	Enqueue Counter	Dequeue Counter	Operations
Active Non-Durable Topic Subscribers										
Client ID	Subscription Name	Connection ID	Destination	Selector	Pending Queue Size	Dispatched Queue Size	Dispatched Counter	Enqueue Counter	Dequeue Counter	Operations
simple-...	ID:COMM...	factory...			0	0	0	0	0	
simple-...	ID:COMM...	factory...			0	0	0	0	0	

File Edit View History Bookmarks Tools Help

localhost : Connections x +

common1.iinthecloud.com:8161/admin/connections.jsp 2009 ... Search

ActiveMQ™

The Apache Software Foundation <http://www.apache.org/>

Home | Queues | Topics | Subscribers | Connections | Network | Scheduled | Send Support

Connections

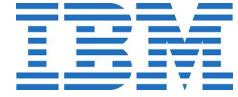
Connector mqtt+nio

Name ↑	Remote Address	Active	Slow
simple publish	tcp://127.0.0.1:51379	true	false
simple-client	tcp://172.29.5.242:51368	true	false

Network Connectors

- Queue Views
 - Graph
 - XML
- Topic Views
 - XML
- Subscribers Views
 - XML
- Useful Links

The END!



Special notices

This document was developed for IBM offerings in the United States as of the date of publication. IBM may not make these offerings available in other countries, and the information is subject to change without notice. Consult your local IBM business contact for information on the IBM offerings available in your area.

Information in this document concerning non-IBM products was obtained from the suppliers of these products or other public sources. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. Send license inquiries, in writing, to IBM Director of Licensing, IBM Corporation, New Castle Drive, Armonk, NY 10504-1785 USA.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

The information contained in this document has not been submitted to any formal IBM test and is provided "AS IS" with no warranties or guarantees either expressed or implied.

All examples cited or described in this document are presented as illustrations of the manner in which some IBM products can be used and the results that may be achieved. Actual environmental costs and performance characteristics will vary depending on individual client configurations and conditions.

IBM Global Financing offerings are provided through IBM Credit Corporation in the United States and other IBM subsidiaries and divisions worldwide to qualified commercial and government clients. Rates are based on a client's credit rating, financing terms, offering type, equipment type and options, and may vary by country. Other restrictions may apply. Rates and offerings are subject to change, extension or withdrawal without notice.

IBM is not responsible for printing errors in this document that result in pricing or information inaccuracies.

All prices shown are IBM's United States suggested list prices and are subject to change without notice; reseller prices may vary.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

Any performance data contained in this document was determined in a controlled environment. Actual results may vary significantly and are dependent on many factors including system hardware configuration and software design and configuration. Some measurements quoted in this document may have been made on development-level systems. There is no guarantee these measurements will be the same on generally-available systems. Some measurements quoted in this document may have been estimated through extrapolation. Users of this document should verify the applicable data for their specific environment.

Revised September 26, 2006



Special notices (cont.)

IBM, the IBM logo, ibm.com AIX, AIX (logo), AIX 5L, AIX 6 (logo), AS/400, BladeCenter, Blue Gene, ClusterProven, Db2, ESCON, i5/OS, i5/OS (logo), IBM Business Partner (logo), IntelliStation, LoadLeveler, Lotus, Lotus Notes, Notes, Operating System/400, OS/400, PartnerLink, PartnerWorld, PowerPC, pSeries, Rational, RISC System/6000, RS/6000, THINK, Tivoli, Tivoli (logo), Tivoli Management Environment, WebSphere, xSeries, z/OS, zSeries, Active Memory, Balanced Warehouse, CacheFlow, Cool Blue, IBM Systems Director VMControl, pureScale, TurboCore, Chiphopper, Cloudscape, Db2 Universal Database, DS4000, DS6000, DS8000, EnergyScale, Enterprise Workload Manager, General Parallel File System, , GPFS, HACMP, HACMP/6000, HASM, IBM Systems Director Active Energy Manager, iSeries, Micro-Partitioning, POWER, PowerExecutive, PowerVM, PowerVM (logo), PowerHA, Power Architecture, Power Everywhere, Power Family, POWER Hypervisor, Power Systems, Power Systems (logo), Power Systems Software, Power Systems Software (logo), POWER2, POWER3, POWER4, POWER4+, POWER5, POWER5+, POWER6, POWER6+, POWER7, System i, System p, System p5, System Storage, System z, TME 10, Workload Partitions Manager and X-Architecture are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries.

A full list of U.S. trademarks owned by IBM may be found at: <http://www.ibm.com/legal/copytrade.shtml>.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

AltiVec is a trademark of Freescale Semiconductor, Inc.

AMD Opteron is a trademark of Advanced Micro Devices, Inc.

InfiniBand, InfiniBand Trade Association and the InfiniBand design marks are trademarks and/or service marks of the InfiniBand Trade Association.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Java, JavaScript and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries or both.

Microsoft, Windows and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries or both.

NetBench is a registered trademark of Ziff Davis Media in the United States, other countries or both.

SPECint, SPECfp, SPECjbb, SPECweb, SPECjAppServer, SPEC OMP, SPECviewperf, SPECapc, SPEChpc, SPECjvm, SPECmail, SPECimap and SPECcsfs are trademarks of the Standard Performance Evaluation Corp (SPEC).

The Power Architecture and Power.org wordmarks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

TPC-C and TPC-H are trademarks of the Transaction Performance Processing Council (TPPC).

UNIX is a registered trademark of The Open Group in the United States, other countries or both.

•Node.js is an official trademark of Joyent. IBM SDK for Node.js is not formally related to or endorsed by the official Joyent Node.js open source or commercial project..

•“TWITTER, TWEET, RETWEET and the Twitter logo are trademarks of Twitter, Inc. or its affiliates.”

Other company, product and service names may be trademarks or service marks of others.

Revised December 2, 2010