OpenJS World 2021

Virtual Experience | June 2, 2021

OpenJS World 2021

Virtual Experience | June 2, 2021

Internet of Things (IoT) with Node.js: Both Practical and Fun!

Michael Dawson (Red Hat)
Node.js Lead for
Red Hat and IBM

Jesse Gorzinski (IBM)
Business Architect
IBM

Michael Dawson

- Node.js Lead for Red Hat and IBM

- Active Node.js community member
  - Collaborator
  - Node.js Technical Steering Committee TSC
  - 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

Jesse Gorzinski

Business Architect of Open Source on i
- –Leader of development teams
- –Owns IBM i open source strategy

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

Agenda

- Intro to IoT and MQTT
- Using MQTT with Node.js
- Let's Look at some devices
- Anatomy of a simple MQTT Light and Temperature Sensor
- MQTT and Devices in Action!
- Leveraging the Cloud
- Reactive Applications/Kafka

IoT Introduction

- 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/

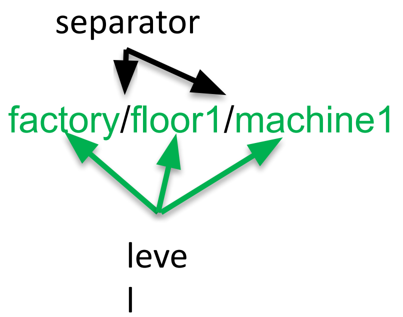
“MQTT is a machine-to-machine (M2M)/"Internet of Things" connectivity protocol”

MQTT - 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**

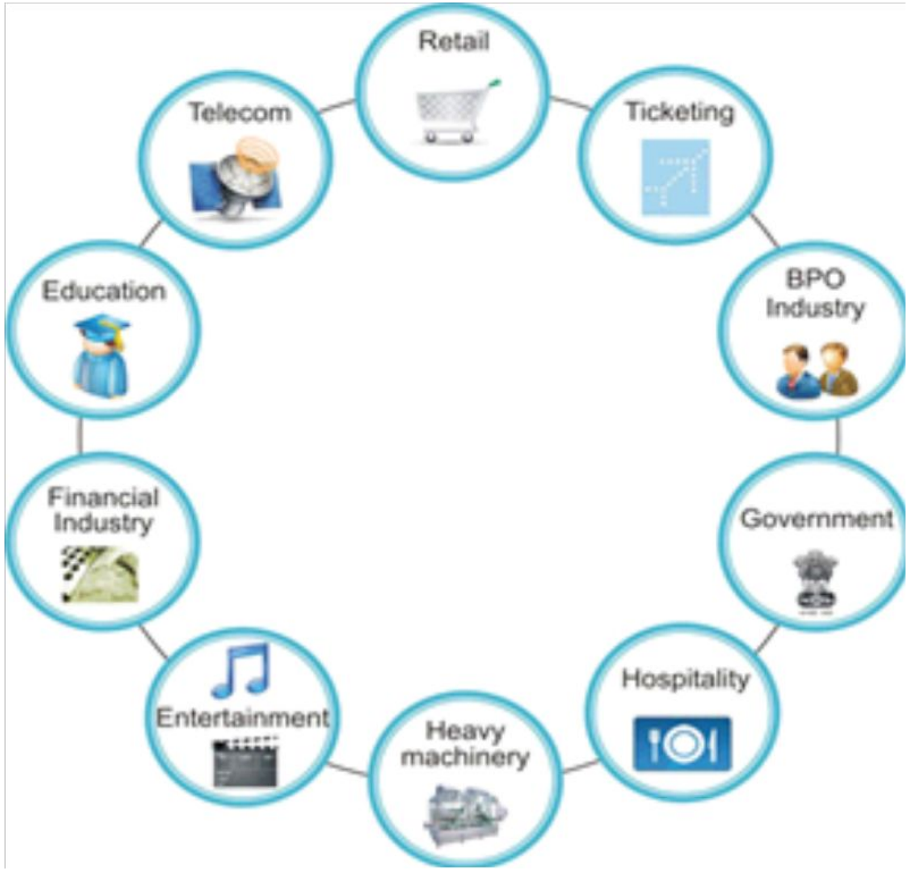
+ Matches one level
    factory/+/machine1   factory/floor1/machine1 (yes) factory/floor1/room1/machine1 (no)

 # matches multiple levels
   only allowed at end
   factory/#   factory/floor1/machine1 (yes) factory/floor1/room1/machine1 (yes)

MQTT Quality of Service (QoS)

- 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 ?


Retail
Ticketing
BPO Industry
Government
Hospitality
Heavy machinery
Entertainment
Financial Industry
Education
Telecom

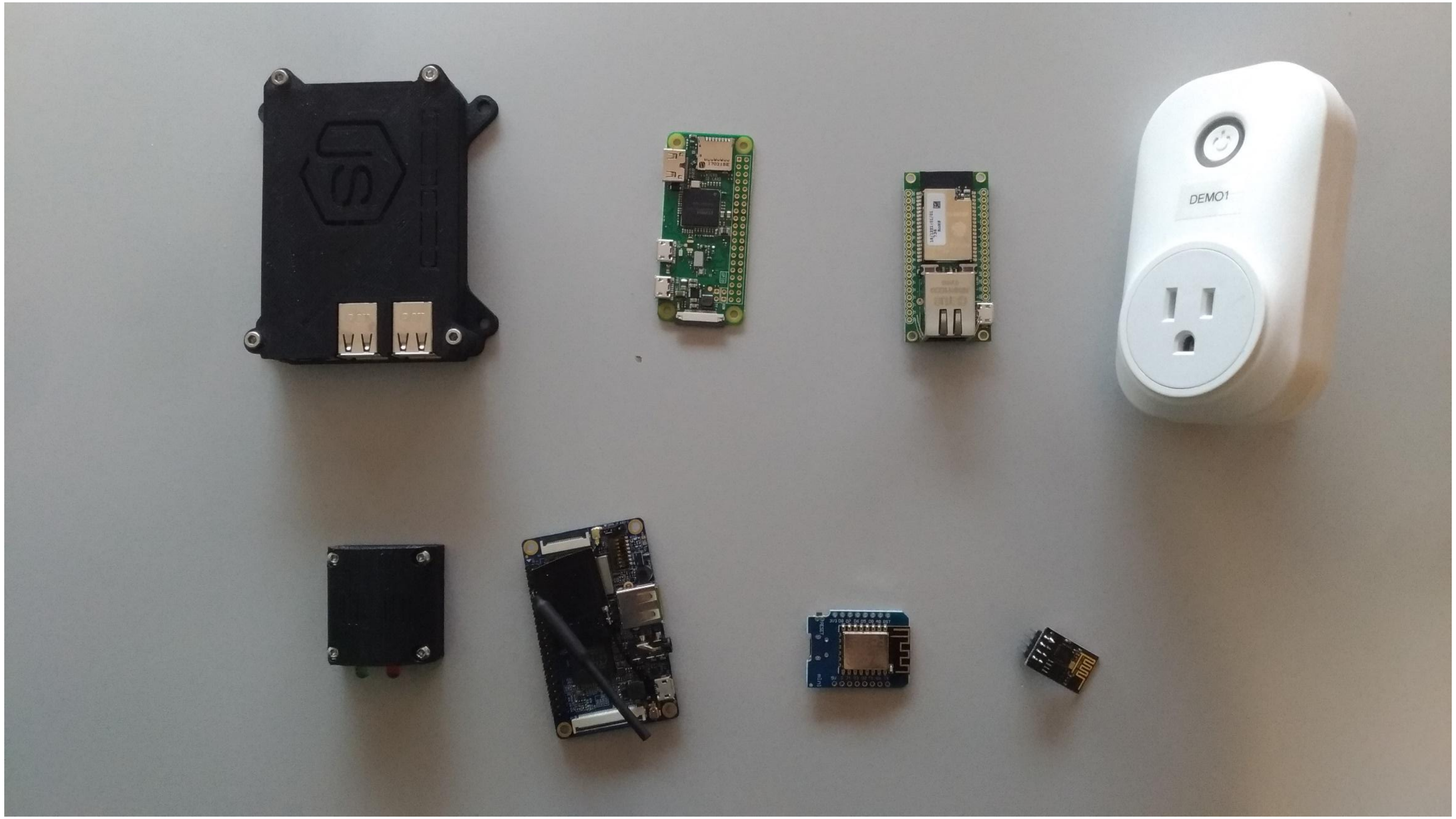
Simple Publisher

```
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 );
});
```

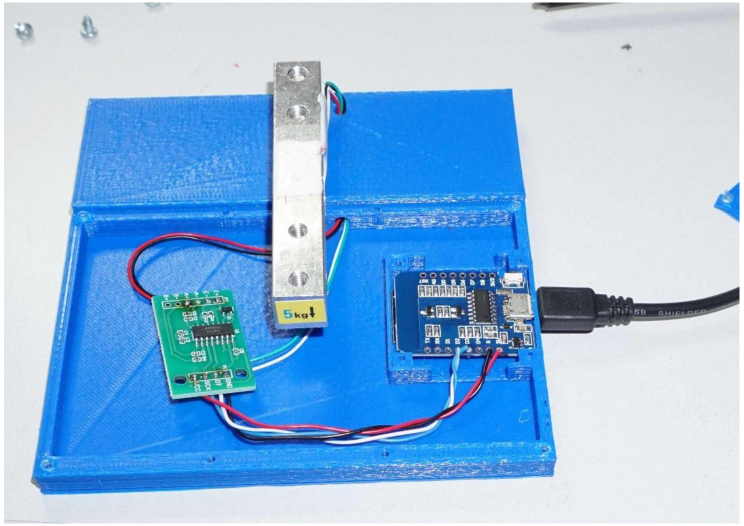
Simple Client

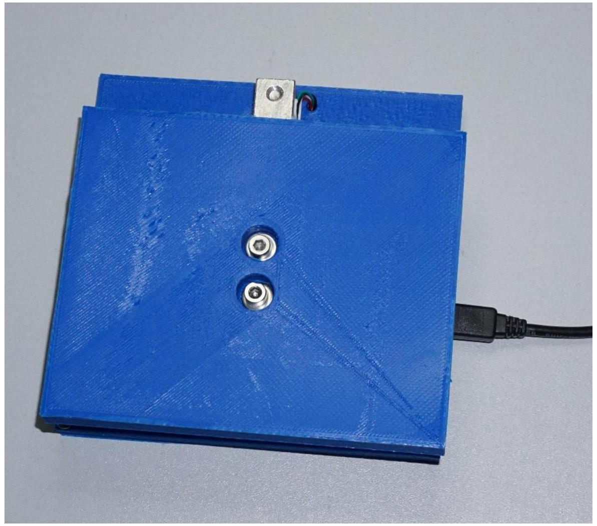
```
const fs = require('fs');
const path = require('path');
const mqtt = require('mqtt');

// 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() + ')');
  });
});
```

Let’s Look at some Devices



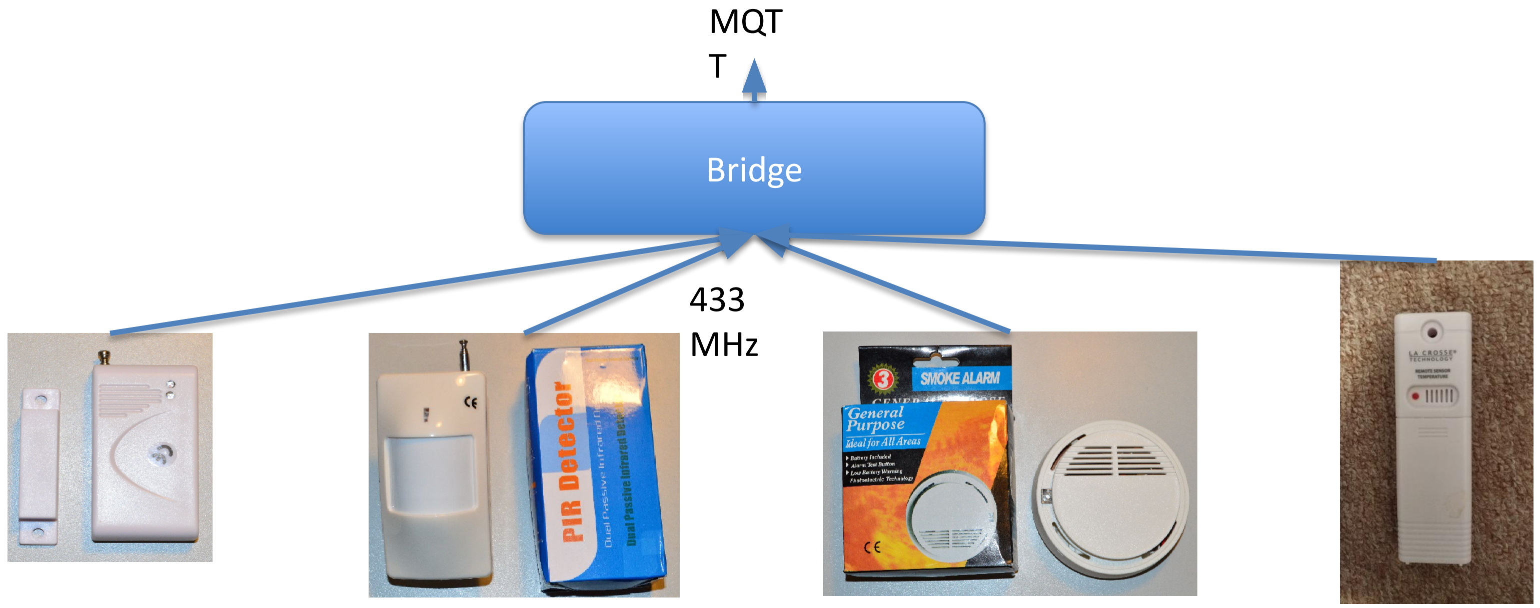
Let’s Look at some Devices





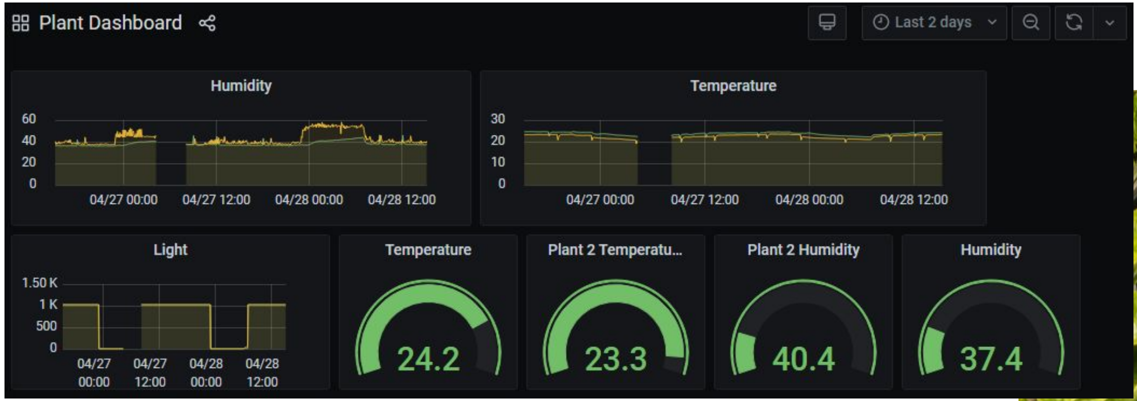
What about Existing Devices?

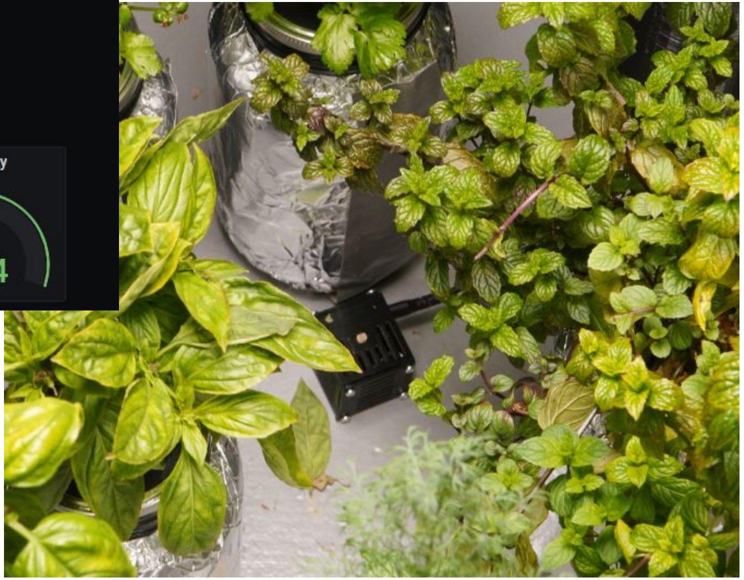
- Common approach is gateway or bridge
- As an example 433MHz to MQTT bridge
  https://github.com/mhdawson/arduino-esp8266/tree/master/Mqtt433Bridge



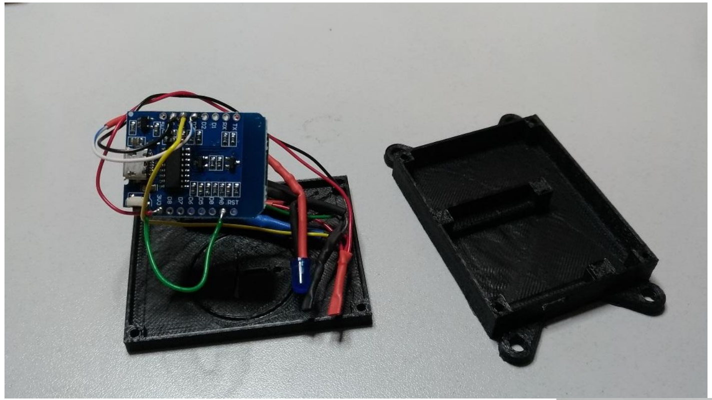
Anatomy of a Simple Device

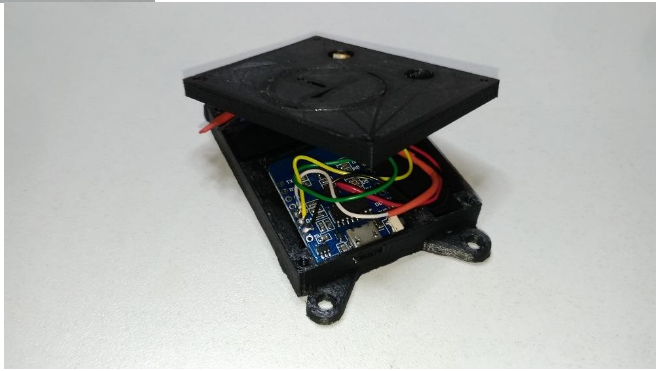
- Maybe your business grows plants?
- **Temperature, Humidity** and **Light** intensity through the day might be interesting?

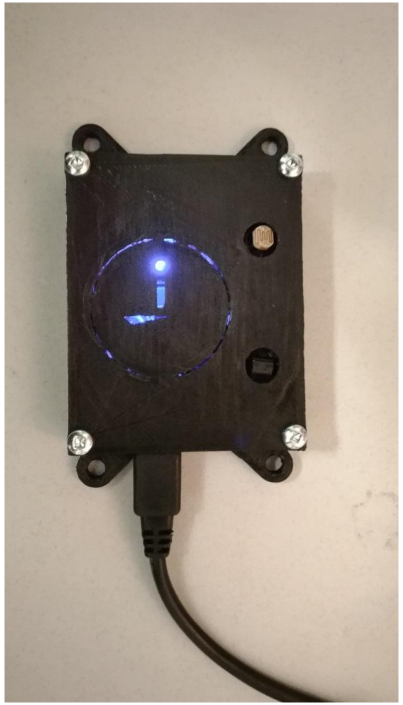




Anatomy of a Simple Device - Temp and Light Sensor

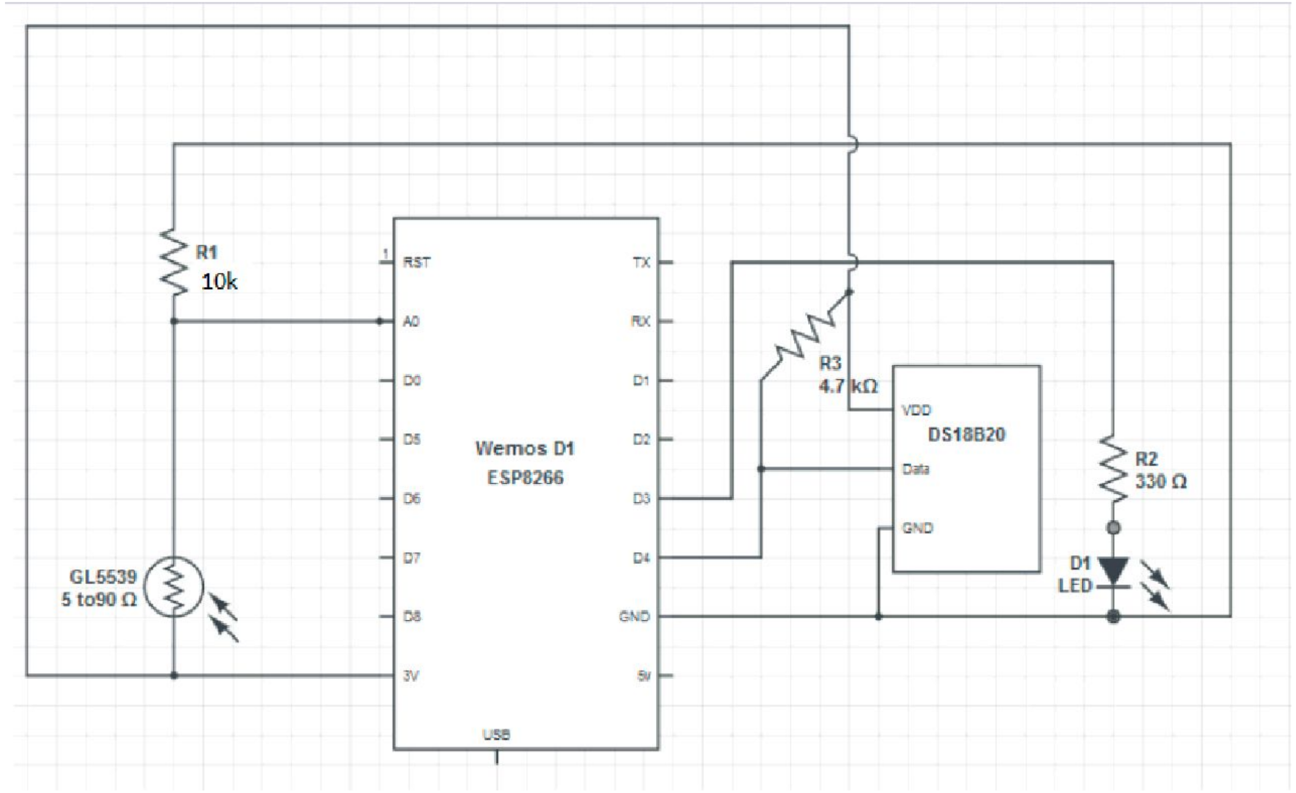




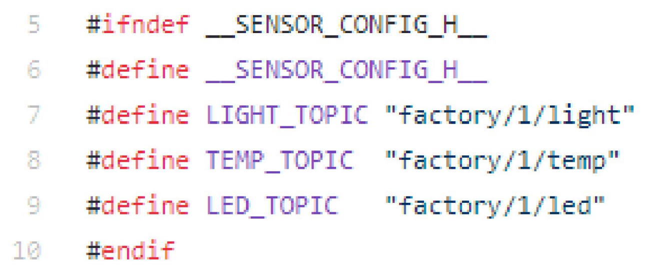


https://github.com/mhdawson/arduino-esp8266/tree/master/TempAndLightSensor

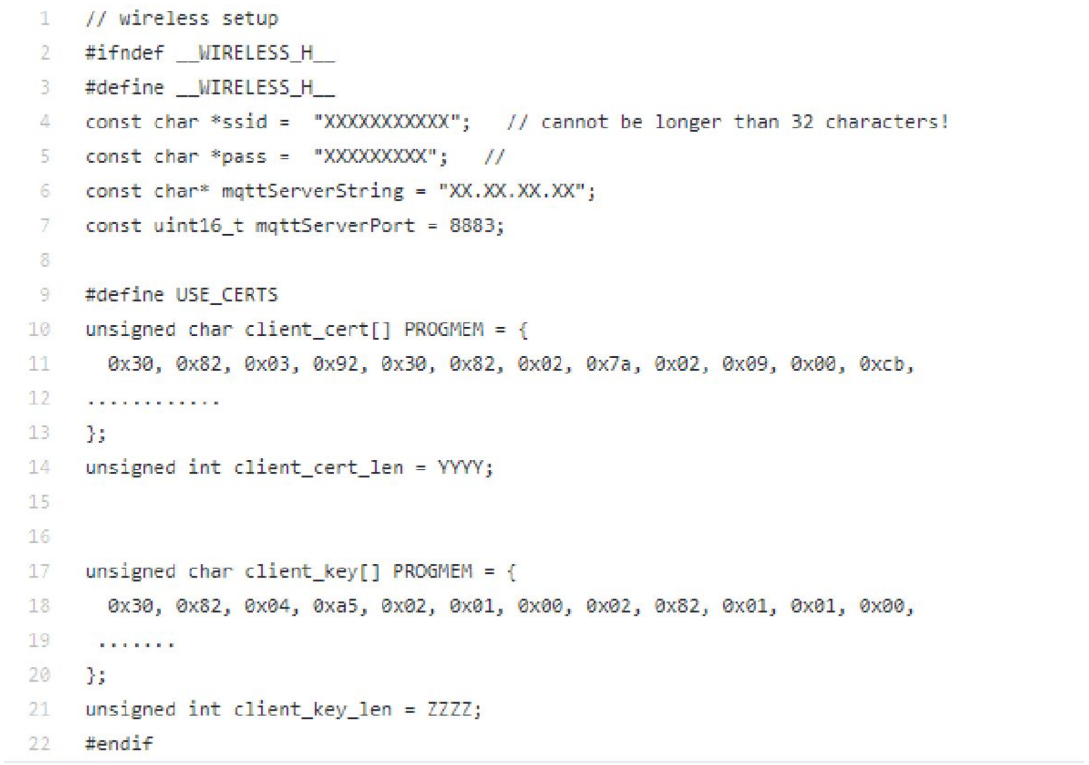
Anatomy of a Simple Device



Anatomy of a Simple Device

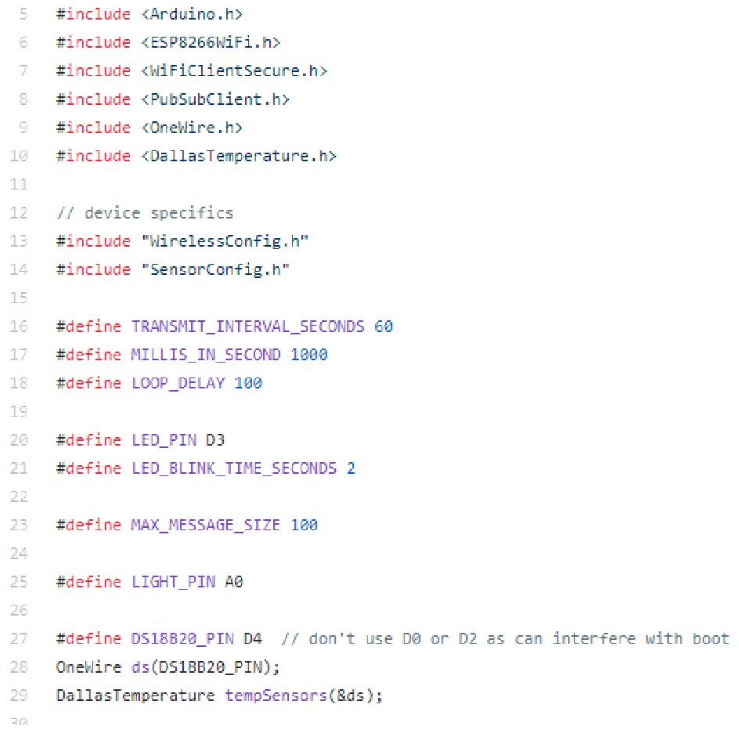


```
#ifndef __SENSOR_CONFIG_H__
#define __SENSOR_CONFIG_H__
#define LIGHT_TOPIC "factory/1/light"
#define TEMP_TOPIC  "factory/1/temp"
#define LED_TOPIC   "factory/1/led"
#endif
```



```
// wireless setup
#ifndef __WIRELESS_H__
#define __WIRELESS_H__
const char *ssid =  "XXXXXXXXXXX";   // cannot be longer than 32 characters!
const char *pass =  "XXXXXXXXX";   //
const char* mqttServerString = "XX.XX.XX.XX";
const uint16_t mqttServerPort = 8883;

#define USE_CERTS
unsigned char client_cert[] PROGMEM = {
  0x30, 0x82, 0x03, 0x92, 0x30, 0x82, 0x02, 0x7a, 0x02, 0x09, 0x00, 0xcb,
............
};
unsigned int client_cert_len = YYYY;


unsigned char client_key[] PROGMEM = {
  0x30, 0x82, 0x04, 0xa5, 0x02, 0x01, 0x00, 0x02, 0x82, 0x01, 0x01, 0x00,
 .......
};
unsigned int client_key_len = ZZZZ;
#endif
```

Anatomy of a Simple Device

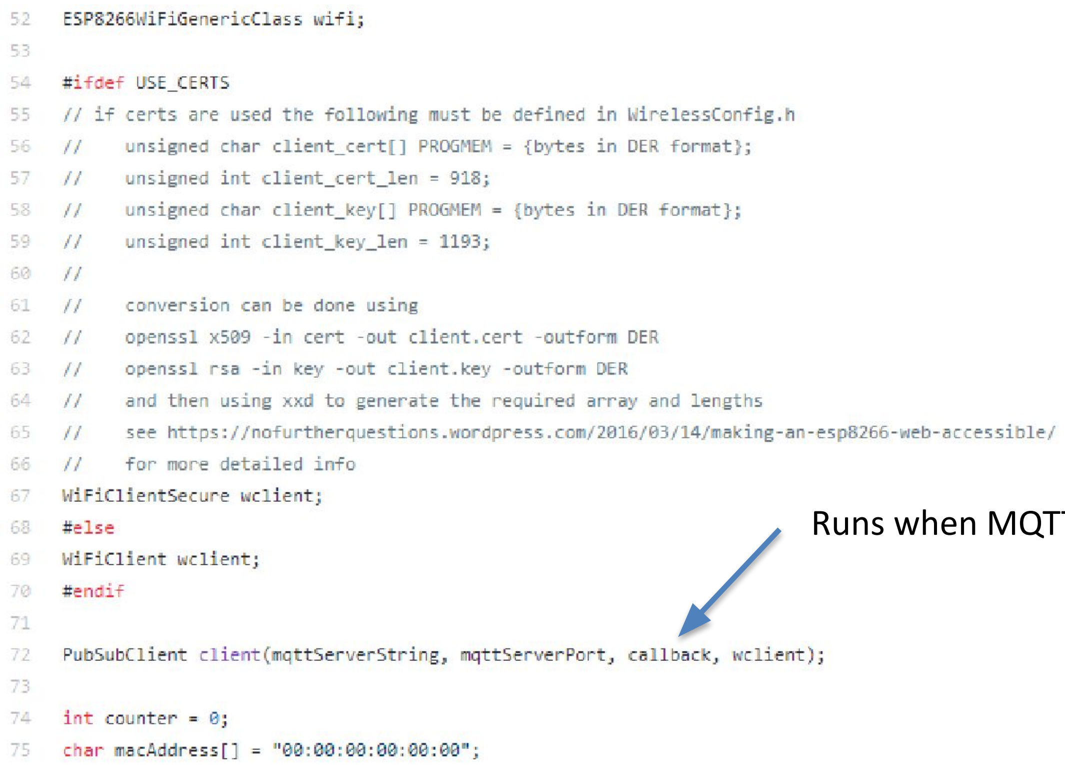


```
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>
#include <PubSubClient.h>
#include <OneWire.h>
#include <DallasTemperature.h>

// device specifics
#include "WirelessConfig.h"
#include "SensorConfig.h"

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

#define LED_PIN D3
#define LED_BLINK_TIME_SECONDS 2

#define MAX_MESSAGE_SIZE 100

#define LIGHT_PIN A0

#define DS18B20_PIN D4  // don't use D0 or D2 as can interfere with boot
OneWire ds(DS18B20_PIN);
DallasTemperature tempSensors(&ds);
```

Anatomy of a Simple Device

```
bool ledOn = true;
void toggleLED() {
  if (ledOn) {
    ledOn = false;
    digitalWrite(LED_PIN, LOW);
  } else {
    ledOn  = true;
    digitalWrite(LED_PIN, HIGH);
  }
}

void callback(char* topic, uint8_t* message, unsigned int length) {
  if (strncmp((const char*)message,"on", strlen("on")) == 0) {
    digitalWrite(LED_PIN, HIGH);
    ledOn = true;
  } else {
    digitalWrite(LED_PIN, LOW);
    ledOn = false;
  }
};
```

Anatomy of a Simple Device



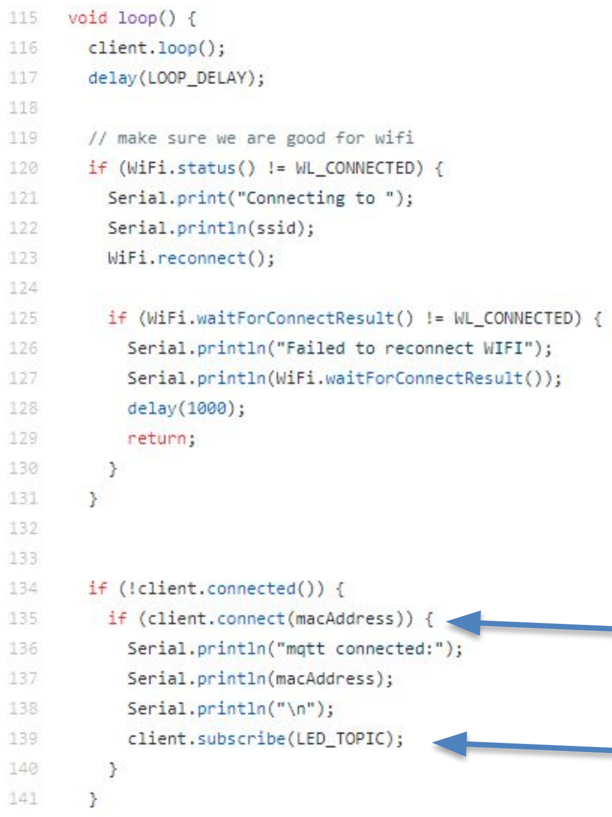
```
ESP8266WiFiGenericClass wifi;

#ifdef USE_CERTS
// if certs are used the following must be defined in WirelessConfig.h
//    unsigned char client_cert[] PROGMEM = {bytes in DER format};
//    unsigned int client_cert_len = 918;
//    unsigned char client_key[] PROGMEM = {bytes in DER format};
//    unsigned int client_key_len = 1193;
//
//    conversion can be done using
//    openssl x509 -in cert -out client.cert -outform DER
//    openssl rsa -in key -out client.key -outform DER
//    and then using xxd to generate the required array and lengths
//    see https://nofurtherquestions.wordpress.com/2016/03/14/making-an-esp8266-web-accessible/
//    for more detailed info
WiFiClientSecure wclient;
#else
WiFiClient wclient;
#endif

PubSubClient client(mqttServerString, mqttServerPort, callback, wclient);

int counter = 0;
char macAddress[] = "00:00:00:00:00:00";
```

Runs when MQTT message received

Anatomy of a Simple Device

```
void setup() {
  delay(1000);

  // Setup console
  Serial.begin(115200);
  delay(10);
  Serial.println();
  Serial.println("Started");

  pinMode(LED_PIN, OUTPUT);
  digitalWrite(LED_PIN, HIGH);

#ifdef USE_CERTS
  wclient.setCertificate_P(client_cert, client_cert_len);
  wclient.setPrivateKey_P(client_key, client_key_len);
#endif

  // turn of the Access Point as we are not using it
  wifi.mode(WIFI_STA);
  WiFi.begin(ssid, pass);

  // first reading always seems to be wrong, read it early and
  // throw it away
  tempSensors.requestTemperatures();

   // get the mac address to be used as a unique id for connecting to the mqtt server
  byte macRaw[6];
  WiFi.macAddress(macRaw);
  sprintf(macAddress,
          "%02.2X:%02.2X:%02.2X:%02.2X:%02.2X:%02.2X",
          macRaw[0],
          macRaw[1],
          macRaw[2],
          macRaw[3],
          macRaw[4],
          macRaw[5]);
}
```

Anatomy of a Simple Device

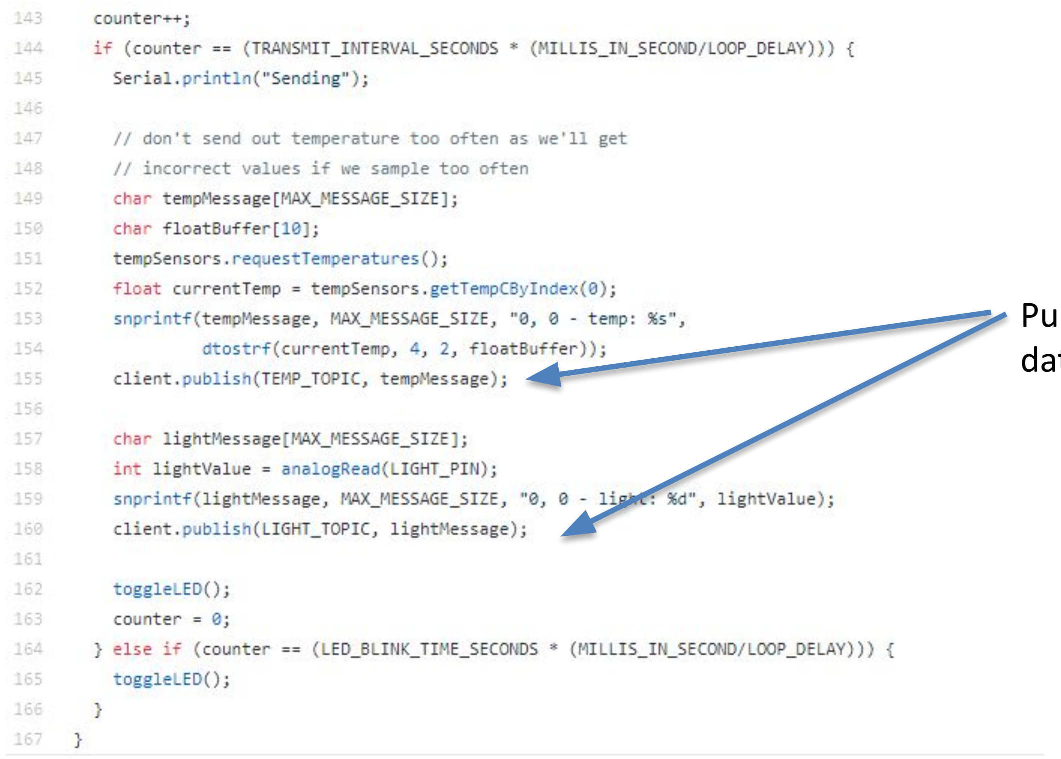


```
void loop() {
  client.loop();
  delay(LOOP_DELAY);

  // make sure we are good for wifi
  if (WiFi.status() != WL_CONNECTED) {
    Serial.print("Connecting to ");
    Serial.println(ssid);
    WiFi.reconnect();

    if (WiFi.waitForConnectResult() != WL_CONNECTED) {
      Serial.println("Failed to reconnect WIFI");
      Serial.println(WiFi.waitForConnectResult());
      delay(1000);
      return;
    }
  }


  if (!client.connected()) {
    if (client.connect(macAddress)) {
      Serial.println("mqtt connected:");
      Serial.println(macAddress);
      Serial.println("\n");
      client.subscribe(LED_TOPIC);
    }
  }
```

Make sure ID is Unique !

Subscribe to topics of interest

Anatomy of a Simple Device



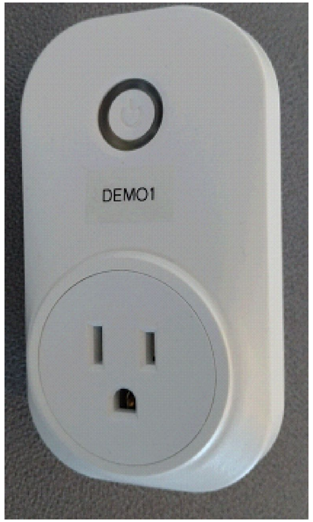
```
  counter++;
  if (counter == (TRANSMIT_INTERVAL_SECONDS * (MILLIS_IN_SECOND/LOOP_DELAY))) {
    Serial.println("Sending");

    // don't send out temperature too often as we'll get
    // incorrect values if we sample too often
    char tempMessage[MAX_MESSAGE_SIZE];
    char floatBuffer[10];
    tempSensors.requestTemperatures();
    float currentTemp = tempSensors.getTempCByIndex(0);
    snprintf(tempMessage, MAX_MESSAGE_SIZE, "0, 0 - temp: %s",
             dtostrf(currentTemp, 4, 2, floatBuffer));
    client.publish(TEMP_TOPIC, tempMessage);

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

    toggleLED();
    counter = 0;
  } else if (counter == (LED_BLINK_TIME_SECONDS * (MILLIS_IN_SECOND/LOOP_DELAY))) {
    toggleLED();
  }
}
```

Publish data

Anatomy of a Simple Device

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

```
client.on('publish', function(message) {
  console.log(message);
  if (message.topic === (devicePrefix + '/power')) {
    if (message.message === 'on') {
      powerState = 1;
    } else if (message.message === 'off') {
      powerState = 0;
    }
    digitalWrite(powerPin, powerState);
    console.log('Power state:' + powerState);
  } else if (message.topic === (devicePrefix + '/led')) {
    clearLedFlashTimer();
    if (message.message === 'on') {
      ledState = 1;
    } else if (message.message === 'off') {
      ledState = 0;
    } else if (message.message.substr(0, 'flash'.length) === 'flash') {
      try {
        timeout = message.message.split(':')[1];
        startFlashTimer(timeout);
      } catch (err) {
        console.log(err);
      }
    }
    digitalWrite(ledPin, (ledState + 1) % 2);
    console.log('Led state:' + ledState);
  } else if (message.topic === (devicePrefix + '/query_state')) {
    client.publish(devicePrefix + '/state/power', powerState);
    client.publish(devicePrefix + '/state/led', ledState);
  }
});
```



MQTT and Devices In Action

Demo Time!

Demo Consuming Data

```
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() + ')');
    });
});
```

Demo Consuming Data - Store to Database

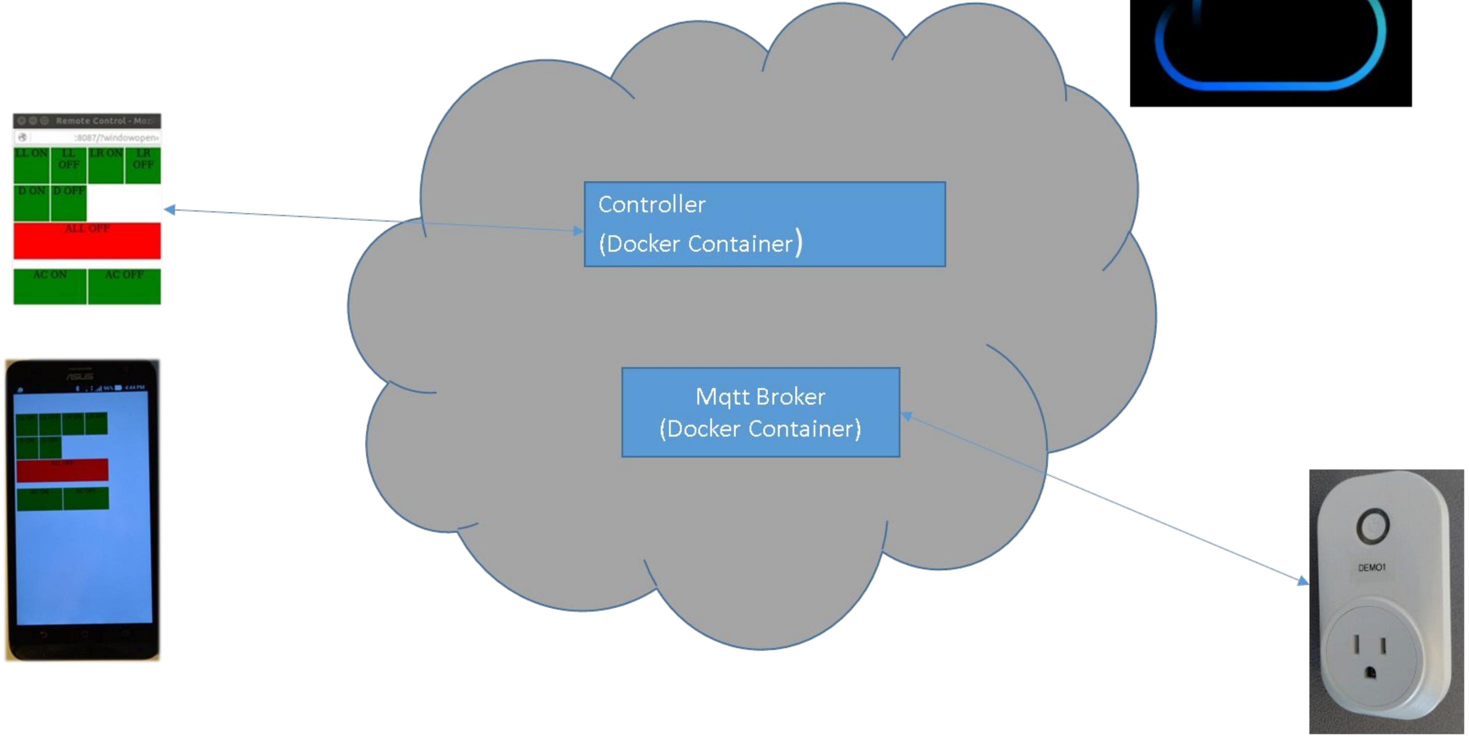
```
const { DBPool } = require('idb-pconnector');
const pool = new DBPool();

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!');
}
```

Demo Consuming Data - Store to Database

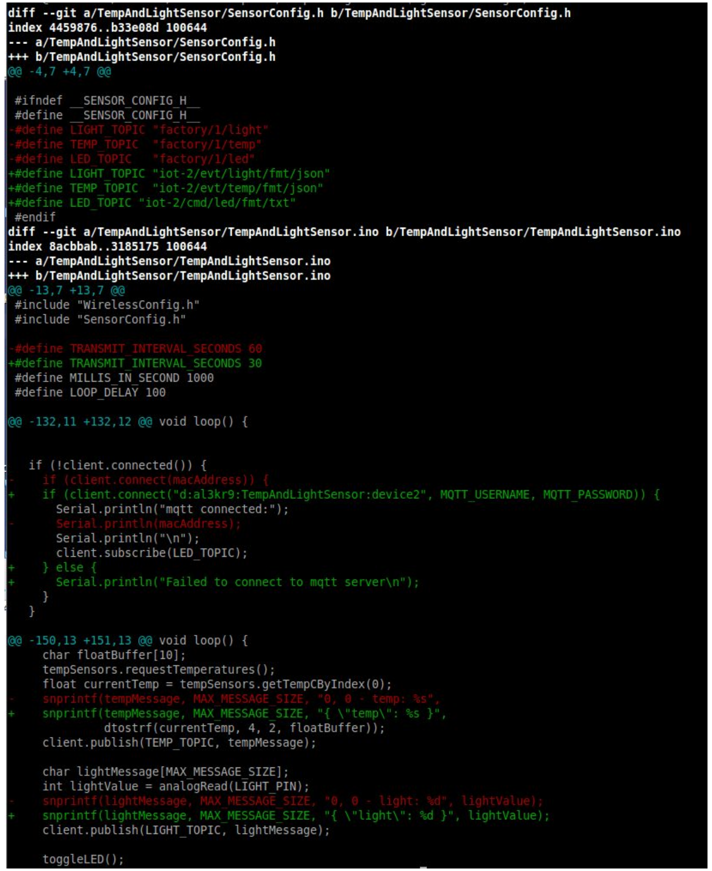
```
const mqttClient = mqtt.connect('mqtts:common1.iinthecloud.com:8883', mqttOptions); mqttClient.on('connect', () => {
    console.log('connected');
    mqttClient.subscribe('factory/1/light');
    mqttClient.subscribe('factory/1/temp');
    mqttClient.on('message', (topic, message) => {
      let value = message.toString().replace(/.*:/g,'').replace(/[^0-9.]+/g,'');
      pool.prepareExecute('insert into JESSEGIOT.IOT_RECORDS(device, sensorvalue) values(?, ?)', [topic, value]);
      console.log('message received topic (' + topic + ') message (' + message.toString() + ')');
    });
});
```

Leveraging the Cloud

- Could do this:



Leveraging the Cloud - IoT Platforms



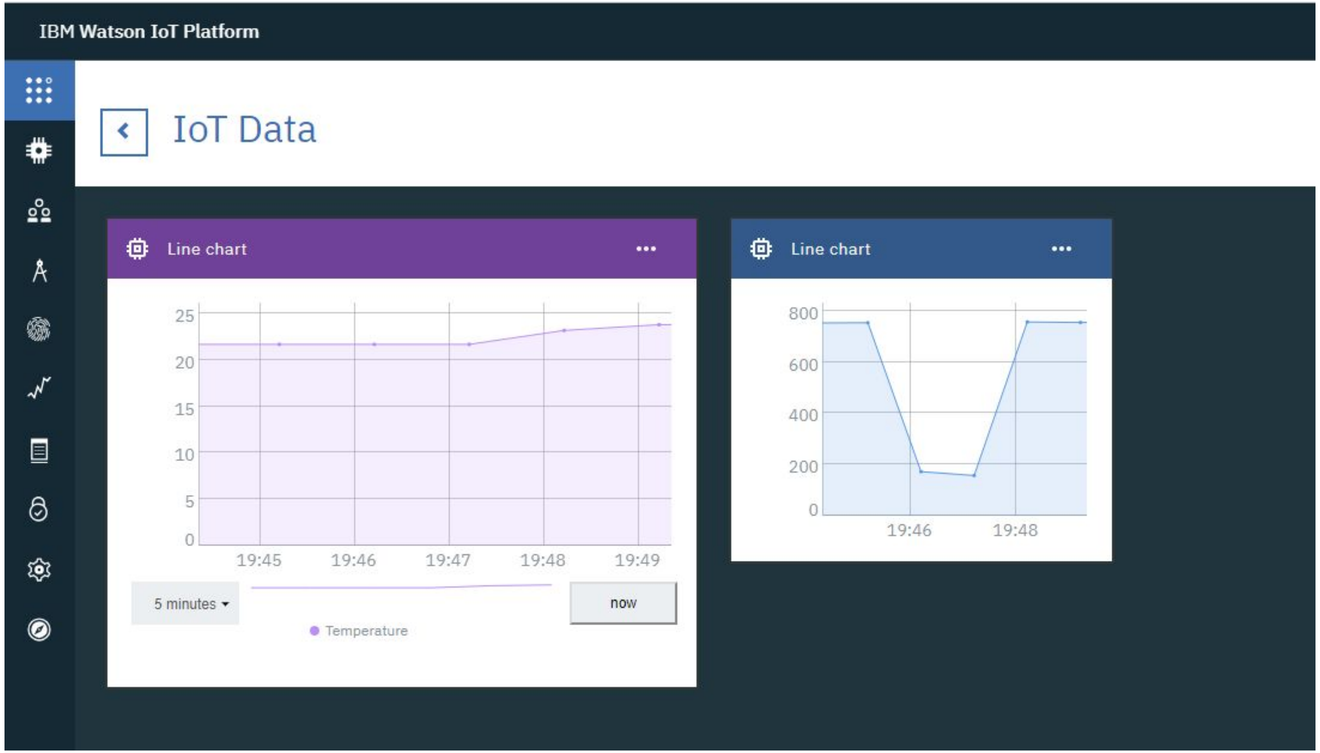
```
diff --git a/TempAndLightSensor/SensorConfig.h b/TempAndLightSensor/SensorConfig.h
index 4459876..b33e08d 100644
--- a/TempAndLightSensor/SensorConfig.h
+++ b/TempAndLightSensor/SensorConfig.h
@@ -4,7 +4,7 @@

 #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("d:al3kr9: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);
-    snprintf(tempMessage, MAX_MESSAGE_SIZE, "0, 0 - temp: %s",
+    snprintf(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);
-    snprintf(lightMessage, MAX_MESSAGE_SIZE, "0, 0 - light: %d", lightValue);
+    snprintf(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"
```

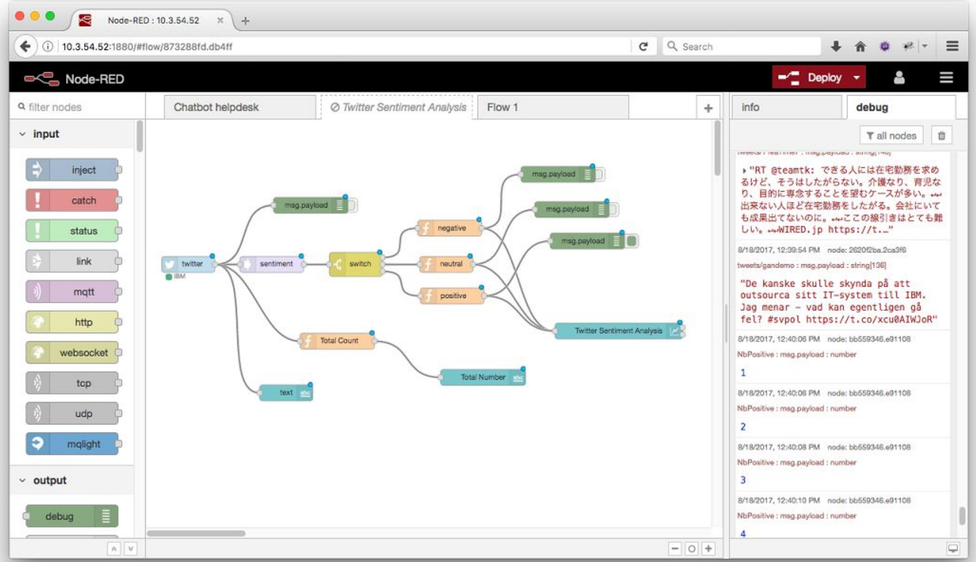
```
-    snprintf(lightMessage, MAX_MESSAGE_SIZE,
"0, 0 - light: %d", lightValue);
+    snprintf(lightMessage, MAX_MESSAGE_SIZE,
"{ \"light\": %d }", lightValue);
```

Leveraging the Cloud - IoT Platforms



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



```
"RT @teamtk: できる人には在宅勤務を求めるけど、そうはしたがらない。介護なり、育児なり、目的に専念することを望むケースが多い。↵↵出来ない人ほど在宅勤務をしたがる。会社にいても成果出てないのに。↵↵ここの線引きはとても難しい。↵↵WIRED.jp https://t.…"

8/18/2017, 12:39:54 PM   node: 2620f2ba.2ca3f6
tweets/gandemo : msg.payload : string[136]
"De kanske skulle skynda på att outsourca sitt IT-system till IBM. Jag menar - vad kan egentligen gå fel? #svpol https://t.co/xcu0AIWJoR"

8/18/2017, 12:40:06 PM   node: bb559346.e91108
NbPositive : msg.payload : number
1

8/18/2017, 12:40:06 PM   node: bb559346.e91108
NbPositive : msg.payload : number
2

8/18/2017, 12:40:08 PM   node: bb559346.e91108
NbPositive : msg.payload : number
3

8/18/2017, 12:40:10 PM   node: bb559346.e91108
NbPositive : msg.payload : number
4
```

Reactive Applications/Kafka - Up and coming

- Reactive Systems gaining popularity in the Enterprise - https://www.reactivemanifesto.org/
- Kafka becoming the messaging component of choice
- Good fit with IoT System so plan on how to bridge MQTT data in your systems
- Apache Camel is one good option.

Summary

- Intro to IoT and MQTT
- Using MQTT with Node.js
- Let's Look at some devices
- Anatomy of a simple MQTT Light and Temperature Sensor
- MQTT and Devices in Action!
- Leveraging the Cloud
- Reactive Applications/Kafka

Copyright and Trademarks

The END!