

# MQTT – protocol for yours IoT

Miroslav Rešetar, [mresetar@croz.net](mailto:mresetar@croz.net), @MiroslavResetar



# What the heck is that IoT anyway?

- More formal definition:
  - ... network of physical objects that contain embedded technology to communicate and sense or interact with their internal states or the external environment. (Gartner)

# IoT functionalities

- Things that **publish** data to Internet/Intranet
- Publishes the data **without knowing** about the **consumers**
- Makes **data available** to multiple systems in a way that **can be expanded** and revised as new requirements emerge
- Being able to **investigate** dependencies and causality between seemingly **unrelated data feeds**

# Why should I care?

- Business important reasons, compared to 2015 (expectations by Gartner):
  - 30% up in connected devices
  - 22% up in total services spending
  - 3,010 Billion \$ industry by 2020
- More personal
  - It's cool to hack with IoT devices 😊

# Popular IoT hacking devices

- Raspberry Pi 1-3 (A, B, Zero)
  - Price \$5-\$35 (Real world: €20 - €100)
  - Linux powered
  - Quad-core
  - Ideal for edge devices
- Arduino
  - Price €35 - €100
  - Lots of shields
  - Great for robotics
  - Good tools
- Both are quite expensive
- There are cheaper options available →

# Amazon Dash - Not really hacking device

- Amazon Dash Button is a 5\$ Wi-Fi connected device that reorders your favorite product with the press of a button.

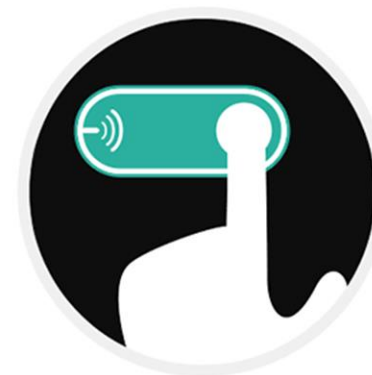


## HOW IT WORKS



### SET IT

Set-up and place  
Dash Button



### PRESS IT

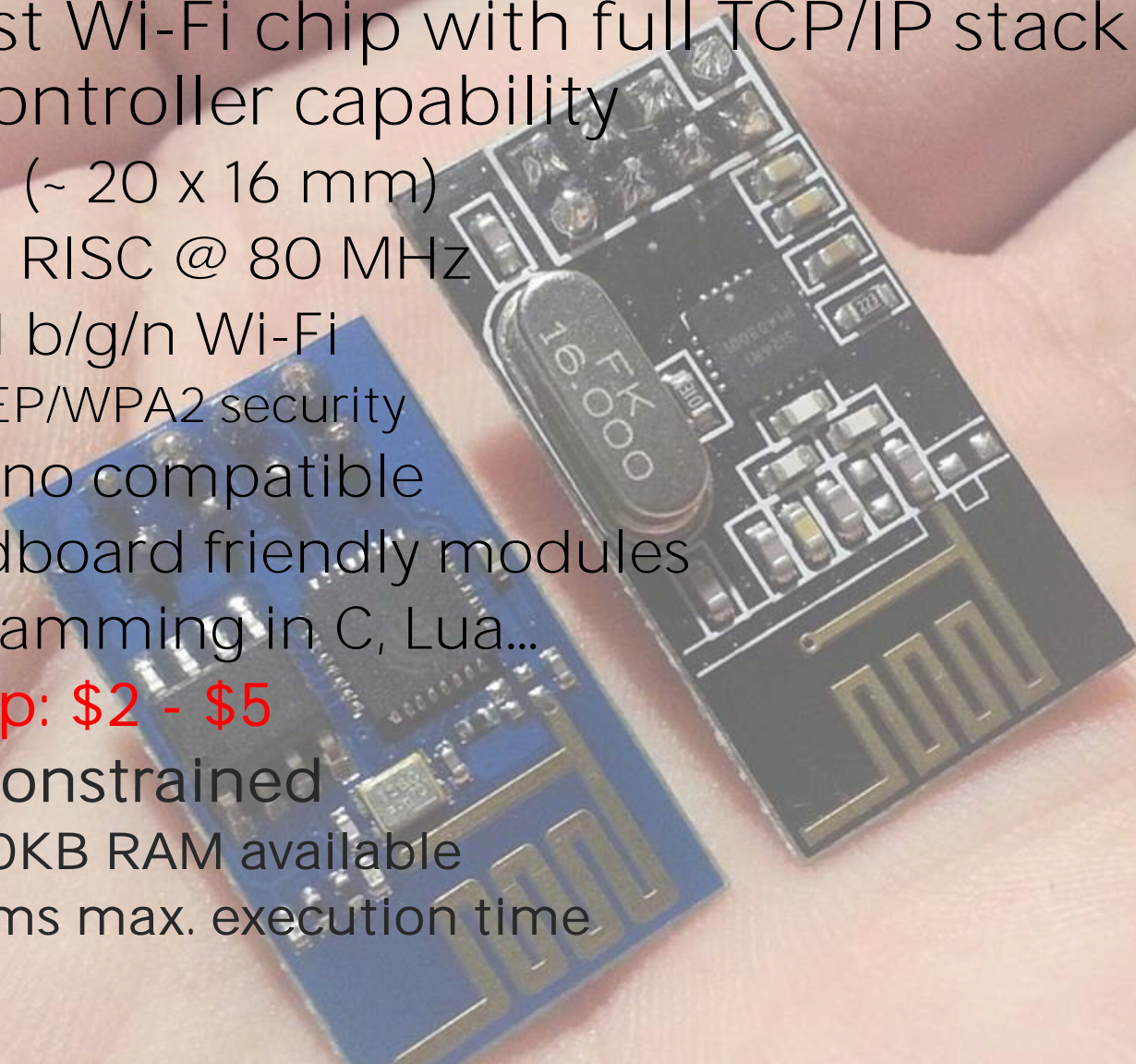
When you're running low  
on your favorite product



### GET IT

Receive your product  
before you run out

# ESP8266 - Game changer (2014)

- Low cost Wi-Fi chip with full TCP/IP stack and microcontroller capability
    - Small (~ 20 x 16 mm)
    - 32 bit RISC @ 80 MHz
    - 802.11 b/g/n Wi-Fi
      - WEP/WPA2 security
    - Arduino compatible
    - Breadboard friendly modules
    - Programming in C, Lua...
    - **Cheap: \$2 - \$5**
    - But constrained
      - ~30KB RAM available
      - 10ms max. execution time
- 

# Enter MQTT

- Developed in late 90's by IBM
  - Aimed to resolve M2M issue (device specific protocol)
  - Use case – remote monitoring in oil & gas, hence the name:
    - MQ Telemetry Support
- Had five goals
  - **Simple** to implement
  - Support for different **QoS** delivery
  - Lightweight & bandwidth **efficient**
  - Data **agnostic**
  - Continuous **session** awareness
- Now standardized by OASIS, current version **V3.1.1**



# MQTT - Overview

- Client/Server
- Publish/Subscribe
- Light weight (small protocol overhead)
- Open & Simple (14 messages)
- Easy to implement
- Ideal for
  - Constrained environments: M2M, IoT
  - Small code footprint
  - Constrained network bandwidth

# Network characteristics

- Runs over TCP/IP
  - On network layer: ordered, lossless, bi-directional
- Publish/Subscribe
  - Provides one-to-many messaging
  - Decoupling of applications



# Security

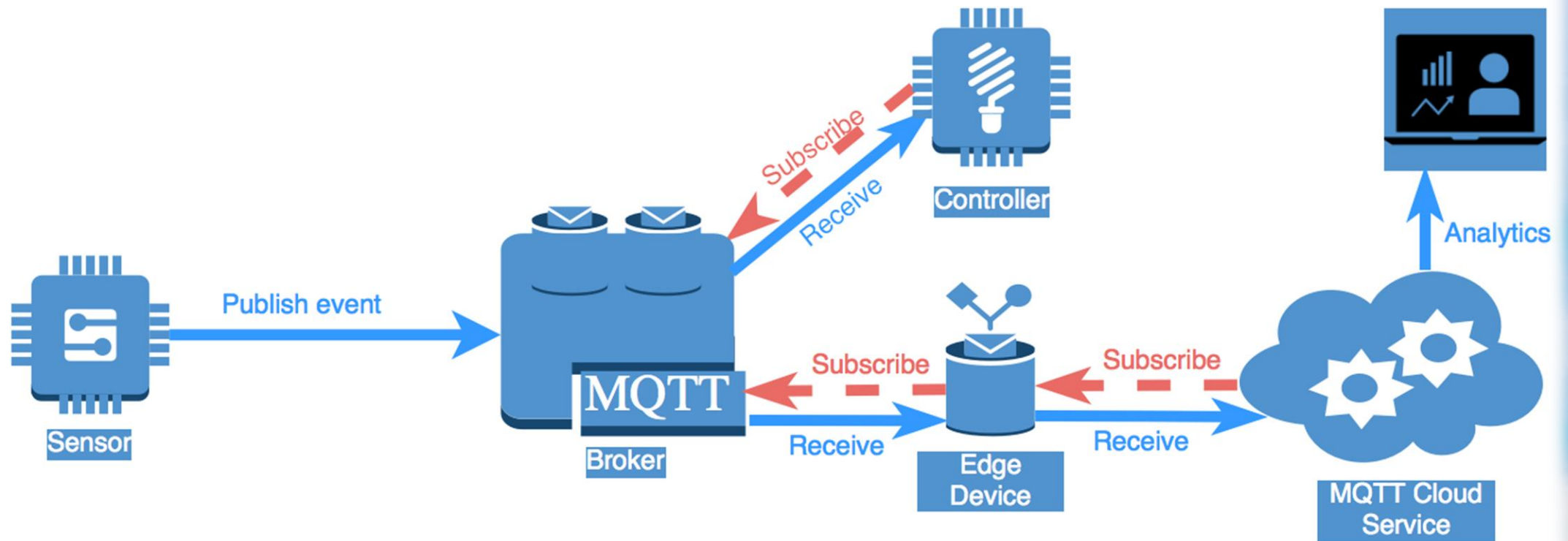
- Out of the box not secure
  - Plain text(bytes) over TCP/IP
  - Default port 1883
- Could be secured by TLS
  - Resource intensive for lightweight clients
  - More network intensive
    - Requires handshake
    - Increased packet overhead

# Decoupling

- Decoupling of
  - Time
    - via persistent session
  - Space
    - devices are not network connected
  - Synchronization
    - asynchronous message exchange



# MQTT dynamics



# QoS

- Message delivery QoS
  - At most once
  - At least once
  - Exactly once
- Small transport overhead
- Abnormal disconnection (Last Will and Testament)



# Topics

- Lightweight – just use them, no need for prior creation
- Name is UTF-8 string with / between levels
  - myLevel1/sensors/temperature/5673
  - cvrči/cvrčak/na/čvoru – also valid topic name
- To subscribe use the whole topic name or wildcards
  - + - single level
  - # - matches any number of levels, e.g.
    - myLevel1/sensors/temperature/+
    - myLevel1/sensors/+/5673
    - myLevel1/+/#
    - +/#
    - #


# Sessions

- Clean or Persistent Session
  - Clean session
    - Publisher only
  - Persistent session
    - Broker stores all QoS 1 & 2 level messages
    - Client receives them on re-connection
- Retained messages
  - Message option
  - If set, will act as last known good value and persisted
  - Only last message will be delivered to client upon connection



# Things to remember so far

- You need
  - MQTT Broker (on-premise or cloud)
  - Client library
  - Device that supports TCP/IP
- To connect
  - Broker **hostname**, port(**1883**)
  - Username/password (optional)
  - TLS (optional)
- To communicate
  - Session (clean or persistent)
  - Publish or subscribe
  - QoS on each message



# Mosquitto – MQTT Broker

- Open Source
- Eclipse project
- Written in C++
- 200 KB download
  - Windows dependencies (OpenSSL, pthreads)
  - Use Linux
  - Or Docker
    - `docker pull ansi/mosquitto`
    - `docker run -d -p 1883:1883 --name mosquitto ansi/mosquitto`

# Eclipse Paho - MQTT Java client lib

```
dependencies {  
    compile group: 'org.eclipse.paho', name: 'mqtt-client', version: '0.4.0'  
}
```

```
public static void main(String[] args) {  
    String broker = "tcp://192.168.99.100:1883";  
    MqttClientPersistence persistence = new MemoryPersistence();  
    try {  
        MqttClient sampleClient = new MqttClient(broker, "MqttExampleClient", persistence);  
        MqttConnectOptions connOpts = new MqttConnectOptions();  
        connOpts.setCleanSession(true);  
        connOpts.setWill("mresetar/paho/lwt", "Eclipse paho client is shut down!".getBytes(), 0, false);  
        sampleClient.connect(connOpts);  
  
        sampleClient.setCallback(new AlertCallback());  
        sampleClient.subscribe("mresetar/alertbox/+/alert");  
  
        String content = "Hello JavaCro16 from Eclipse Paho!";  
        MqttMessage message = new MqttMessage(content.getBytes());  
        message.setQos(0);  
        sampleClient.publish("mresetar/paho", message);  
    } catch (MqttException me) {  
        me.printStackTrace();  
    }  
}
```

```
@Override  
public void messageArrived(String topic, MqttMessage message) throws Exception {  
    System.out.println("Message on topic: " + topic + " arrived.");  
    System.out.println("Message content: " + message);  
    doSomeAction(message);  
}
```

# Eclipse Paho - MQTT JavaScript client

```
// Create a client instance
client = new Paho.MQTT.Client(location.hostname, Number(location.port), "clientId");

// set callback handlers
client.onConnectionLost = onConnectionLost;
client.onMessageArrived = onMessageArrived;

// connect the client
client.connect({onSuccess: onConnect});

// called when the client connects
function onConnect() {
    // Once a connection has been made, make a subscription and send a message.
    console.log("onConnect");
    client.subscribe("/World");
    message = new Paho.MQTT.Message("Hello");
    message.destinationName = "/World";
    client.send(message);
}

// called when the client loses its connection
function onConnectionLost(responseObject) {
    if (responseObject.errorCode !== 0) {
        console.log("onConnectionLost:" + responseObject.errorMessage);
    }
}

// called when a message arrives
function onMessageArrived(message) {
    console.log("onMessageArrived:" + message.payloadString);
}
```

# ESP8266 NodeMCU MQTT Lua client

```
-- init mqtt client with keepalive timer 120sec
m = mqtt.Client("clientid", 120, "user", "password")

-- setup Last Will and Testament (optional)
m:lwt("/lwt", "offline", 0, 0)

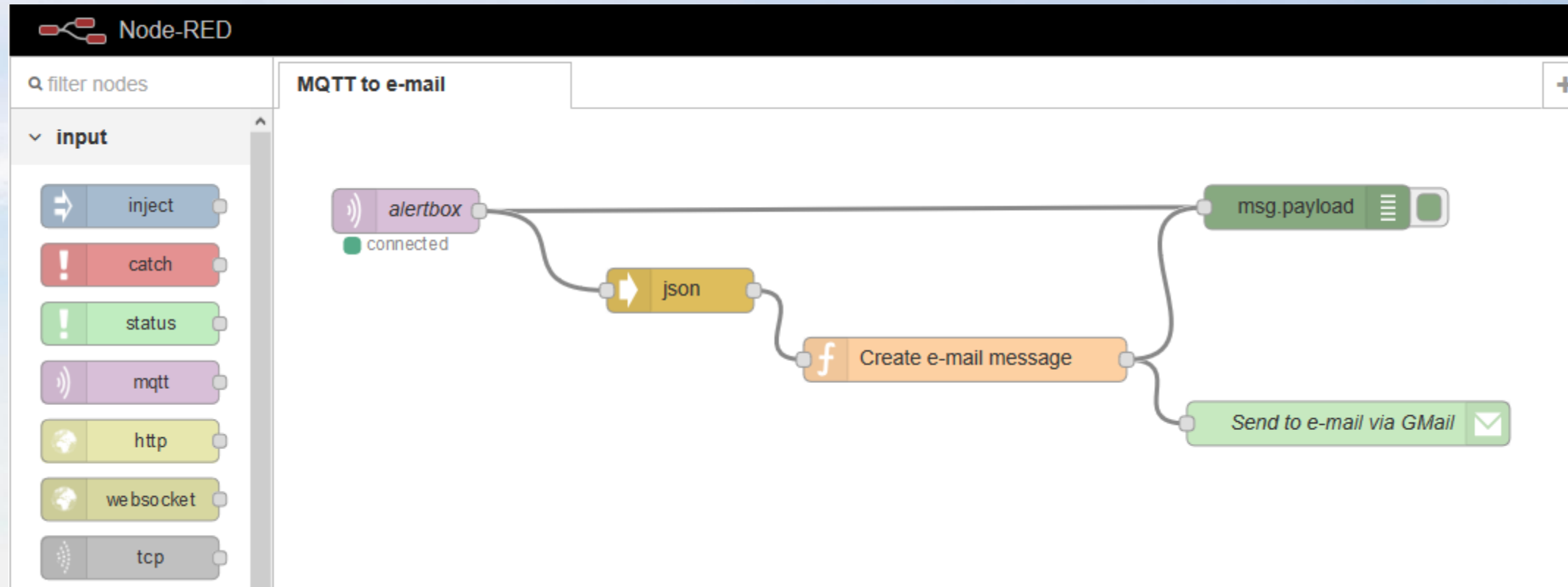
m:on("connect", function(client) print ("connected") end)
m:on("offline", function(client) print ("offline") end)

-- on publish message receive event
m:on("message", function(client, topic, data)
  print(topic .. ":" )
  if data ~= nil then
    print(data)
  end
end)

-- for TLS: m:connect("192.168.11.118", secure-port, 1)
m:connect("192.168.11.118", 1883, 0, function(client) print("connected") end,
  function(client, reason) print("failed reason: "..reason) end)
-- subscribe topic with qos = 0
m:subscribe("/topic",0, function(client) print("subscribe success") end)
-- publish a message with data = hello, QoS = 0, retain = 0
m:publish("/topic","hello",0,0, function(client) print("sent") end)

m:close();
-- you can call m:connect again
```

# Node-RED MQTT integration



# Tools – MQTT.fx

The screenshot displays the MQTT.fx - 1.1.0 application window. The interface includes a menu bar (File, Extras, Help), a toolbar with 'Connect' and 'Disconnect' buttons, and a 'Broker Status' tab. The main content area is divided into several sections:

- Broker:**
  - Version: mosquitto version 1.4.8
  - Timestamp: 2016-05-07 00:10:59+0000
  - Uptime: 20075 seconds
  - Subscriptions: 31
  - Changeset: -
- Clients:**
  - Clients Connected: 2
  - Clients Disconnected: 0
  - Clients Expired: 0
  - Clients Maximum: -
  - Clients Total: 2
- Messages:**
  - Messages Sent: 14877
  - Messages Received: 14881
  - Messages Stored: 47
  - Messages Inflight: -
  - Messages Retained: -
- Traffic:**
  - Bytes Sent: 641055
  - Bytes Received: 61186
- Load:**
  - Load Bytes Sent: -
  - Load Bytes Received: -
  - Messages Sent: -
  - Messages Received: -
  - Messages Publish Sent: 5
  - Messages Publish Received: 8
  - Messages Publish Dropped: -
  - Connections: -
  - Sockets: -

The interface also features a 'Publish' button, a 'Log' button, and a dropdown menu for selecting the broker (currently set to 'Mosquitto'). A status bar at the bottom right indicates the last update time: 18.05.2016 at 14:57:46.

- Java.FX app
- MQTT client (uses Eclipse Paho)
- Can monitor broker status (\$SYS topics)
- Supports scripting

# Demo - Alertbox

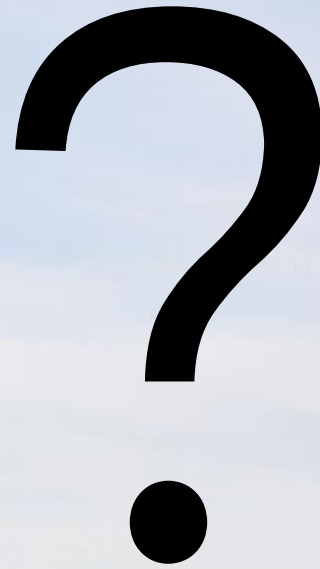




# Source code

- NodeMCU Lua Code
  - <https://github.com/mresetar/alertbox>
- Node-RED integration
  - <https://github.com/mresetar/alertbox-node-red>
- Simple Java MQTT client
  - <https://github.com/mresetar/mqtt-java-client-sample>

# Q&A



# References

- Gartner IoT forecast - <http://www.gartner.com/newsroom/id/3291817>
- Getting Started With MQTT - <https://dzone.com/refcardz/getting-started-with-mqtt>
- Amazon Dash - <http://www.amazon.com/b?node=10667898011>
- Eclipse Paho - <https://eclipse.org/paho/>
- NodeMCU MQTT - <https://nodemcu.readthedocs.io/en/dev/en/modules/mqtt/#mqttclientconnect>
- Pictures used:
  - <https://pixabay.com/en/wind-turbine-field-wind-sky-energy-1149604/>
  - <https://pixabay.com/en/network-iot-internet-of-things-782707/>
  - <https://pixabay.com/en/usb-technology-computer-microchip-1284227/>
  - <http://i.imgur.com/mcyxNnq.jpg>

