MQTT for system administrators (and for the IoT)

Jan-Piet Mens
BSDCan, May 2019
@jpmens

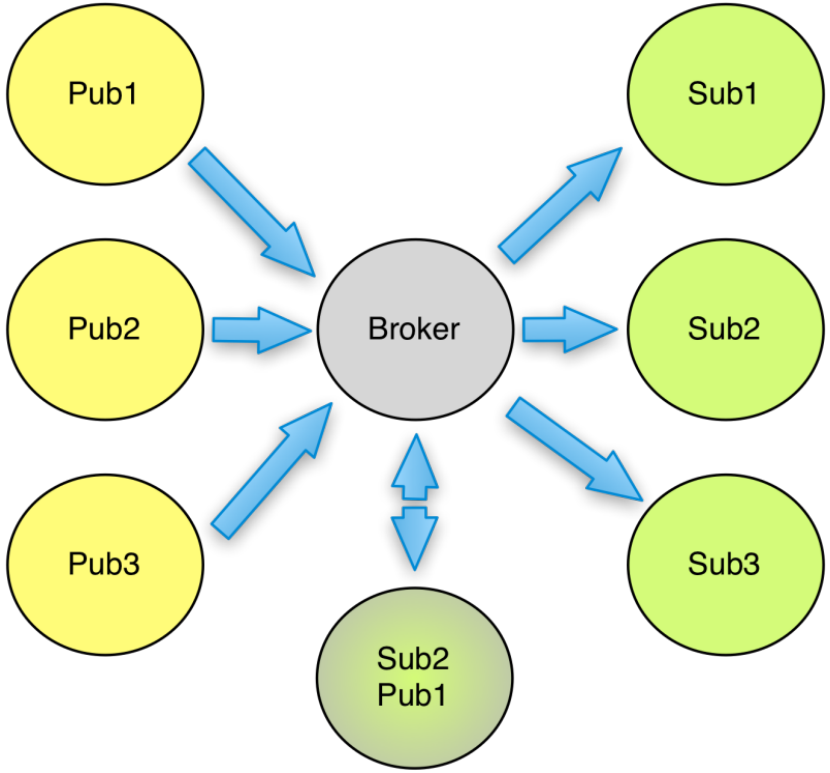
`@jpmens`: consultant, part-time admin, **trainer**, small-scale fiddler, loves plain text, and things which work. Contributes to **Ansible**, dreamed up **OwnTracks**, and chases bugs in open source **DNS** servers.

Have you heard of
MQTT?

MQTT

MQTT is a standard, a TCP-based transport, for **PUB/SUB** messaging, designed for unreliable networks, **binary** payloads up to 256MB, (+2 bytes), fast, **lightweight**, ideal for low-bandwith, high-latency networks, **TLS**, authentication, **ACLs**, TLS-PSK, (payload encryption), keepalive, **last will & testament**, UTF-8 hierarchical **topics**, wildcards

the **landscape**



topic names

UTF-8, hierarchical, **wildcards**

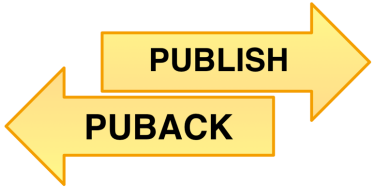
```
home/ground-floor/kitchen/kettle
finance/eur/rate
finance/+/rate
14dfa2e2-d580-4574-88ff-dcc120330482
cellar/stairlamp/cmd
cellar/stairlamp/status
owntracks/jpm/5s/event
owntracks/jpm/#
openhab/homie/5ccf7faac88e/$stats/uptime
```

PUB/SUB **cauldron**

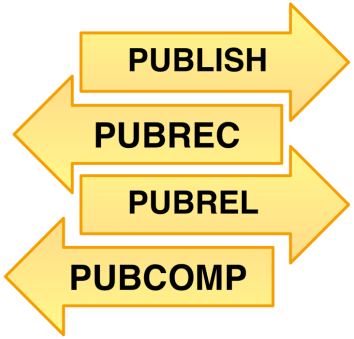
Quality of **Service**

0 At most once

1 Assured delivery

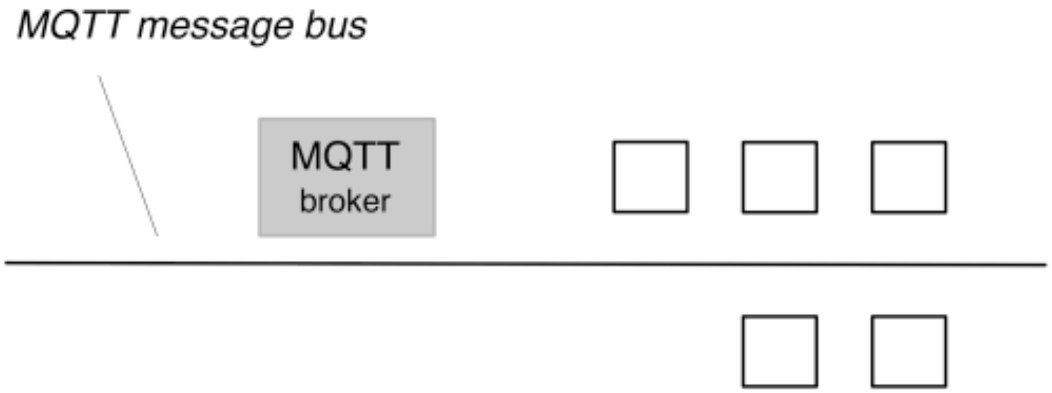


2 Once only



MQTT **brokers**

the **server** bit of MQTT



Mosquitto

C, fast, lightweight, ACLs (plugin), TLS, TLS-PSK, **bridge**, logging via **$SYS**

http://mosquitto.org

VerneMQ

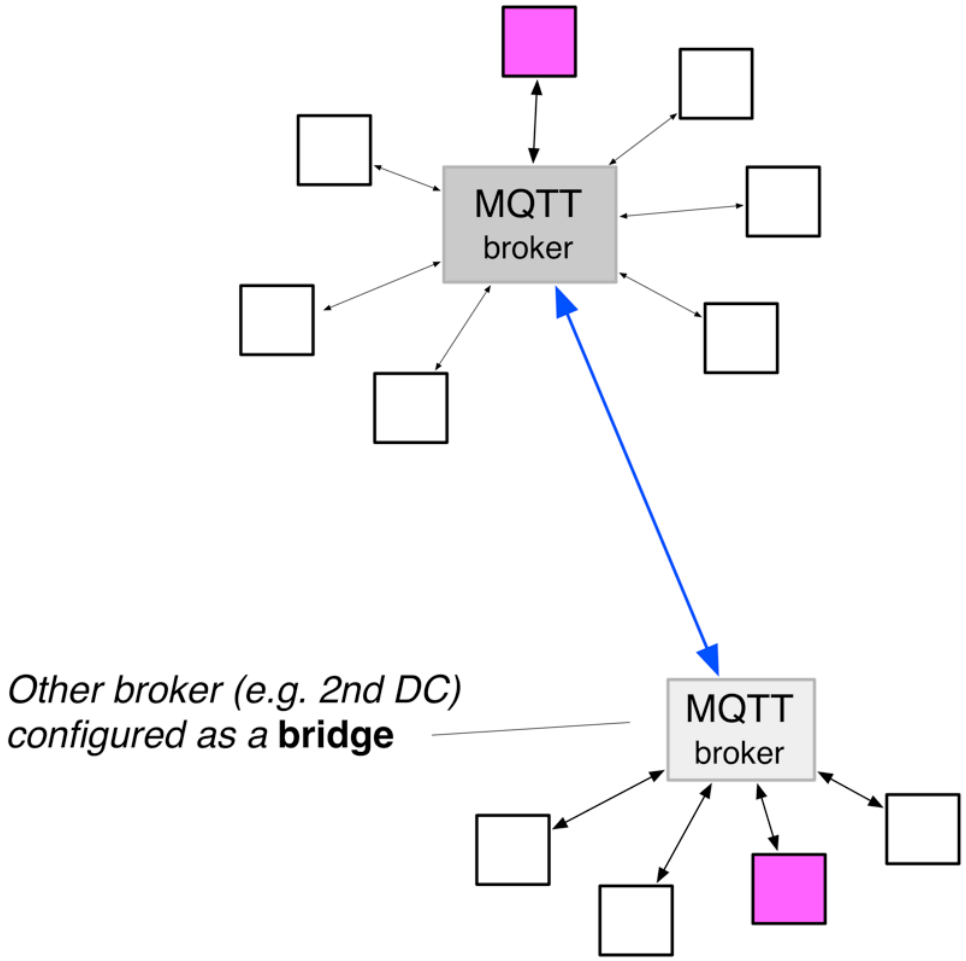
Erlang, **Websockets**, clustering, **file**, SQL & Redis
authentication, Lua plugins, **Webhooks**

http://vernemq.com

more brokers

RSMB, Mosca, Apollo, HiveMQ, (RabbitMQ)

bridging



Other broker (e.g. 2nd DC)
configured as a **bridge**

CLI utilities

```
mosquitto_sub
    [-h localhost] [-p 1883]
    [--cafile file]
    [--cert file --key file]
    [-u username [-P password]]
   -v
   -t 'topic/#'
```


```
mosquitto_pub
    ...
    [-r]
    -t topic
    -m message
```

Language **bindings**

C, C++, Clojure, Dart, Delphi, Erlang, Elixir, Go, Haskell, Java, JavaScript, LotusScript, Lua, .NET, Objective-C, OCaml, Perl, PHP, Python, REXX, Ruby, Smalltalk, Swift, Tcl, …

~~COBOL~~

Python API: **PUB**

```
#!/usr/bin/env python

import paho.mqtt.publish as mqtt

mqtt.single('conf/hello', 'Hello MQTT')
```


```
$ mosquitto_sub -h localhost -v -t 'conf/#'
conf/hello Hello MQTT
```

Python API: **SUB**

```
#!/usr/bin/env python

import paho.mqtt.client as paho

def on_connect(mosq, userdata, flags, rc):
    mqttc.subscribe("conf/+", 0)

def on_message(mosq, userdata, msg):
    print "%s %s" % (msg.topic, str(msg.payload))

mqttc = paho.Client(userdata=None)
mqttc.on_connect = on_connect
mqttc.on_message = on_message

mqttc.connect("localhost", 1883, 60)
mqttc.loop_forever()
```

Python API: **SUB**

```
$ mosquitto_pub -t 'conf/thirsty' -m 'Beer time?'
$ mosquitto_pub -t 'conf/catering' -m 'Coffee is ready'
```

```
$ ./sub.py
conf/thirsty Beer time?
conf/catering Coffee is ready
```

```c
#include <stdio.h>
#include <string.h>
#include <mosquitto.h>

#define MESSAGE "Goodbye, cruel world"

int main(int argc, char *argv[])
{
     struct mosquitto *mosq;

     mosquitto_lib_init();

     if ((mosq = mosquitto_new(NULL, true, NULL)) == NULL) {
          return fprintf(stderr, "Error: Out of memory.\n");
     }
     if (mosquitto_connect(mosq, "192.168.1.130", 1883, 60) != 0) {
          return fprintf(stderr, "Unable to connect to MQTT broker\n");
     }
     mosquitto_publish(mosq,
          NULL,               /* mid */
          "message/adieu",    /* topic */
          strlen(MESSAGE),    /* payload length */
          MESSAGE,            /* payload */
          1,                  /* qos */
          false);             /* retain */
     mosquitto_loop(mosq, -1, 1);

     mosquitto_disconnect(mosq);

     mosquitto_destroy(mosq);
     mosquitto_lib_cleanup();
     return (0);
}
```

libmosquitto

job monitor, **reporting**

```
#!/bin/sh
topic="processes/$(basename $0)"
mqtt_opts="--quiet -h 192.168.1.130 -p 1883"

mqtt() {
    mosquitto_pub ${mqtt_opts} -t "${topic}" -m "$*" || true
}

mqtt "Starting"
```

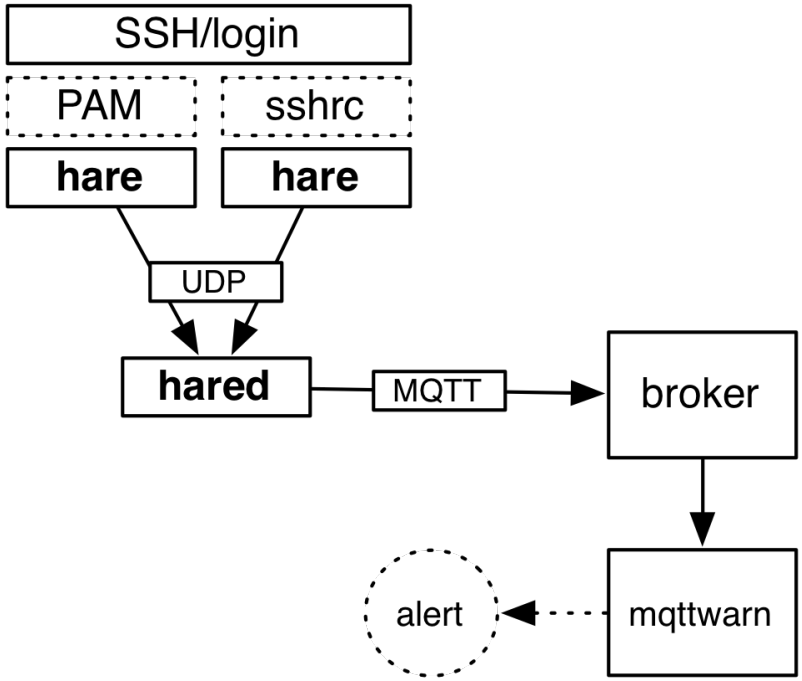
```
$ mosquitto_sub -v -t 'processes/#'
processes/run.sh Starting
processes/monitor/spec1 Starting
processes/run.sh Still going strong at Tue Oct 22 15:49:07 CEST 2013
processes/run.sh That's it, folks!
```

https://gist.github.com/jpmens/7101170

"That is what I ask you to keep in mind as you read this. Think of the possibilities."

— Dan Langille

tracking **logins** (I)



https://jpmens.net/2018/03/25/alerting-on-ssh-logins/

tracking **logins** (2)

```
#!/bin/sh

export PAM_TYPE=open_session
export PAM_USER=$LOGNAME
export PAM_SERVICE=ssh
export PAM_RHOST="$(echo $SSH_CLIENT | cut -d' ' -f1)"
export PAM_TTY=$SSH_TTY

/usr/local/bin/hare mqtt.ww.mens.de
```

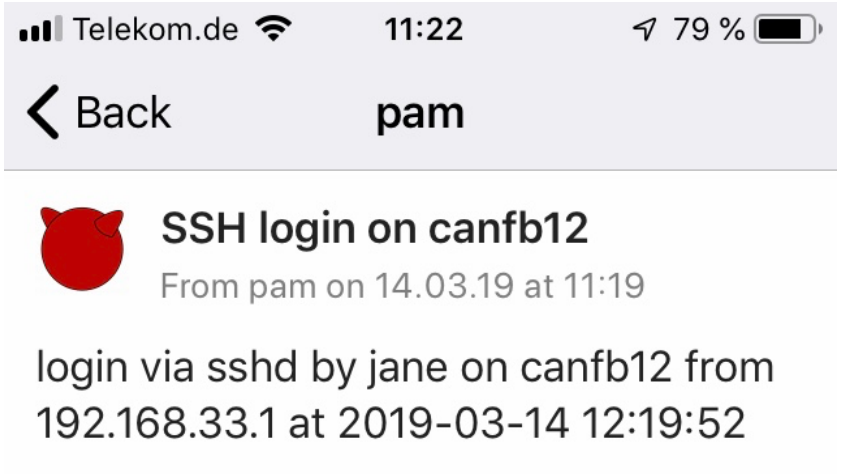
tracking **logins** (3)

```
$ mosquitto_sub -v -t 'logging/#' -F '%I %J'
2019-03-14T10:19:54+0000 {
                           "tst": 1552558794,
                           "topic": "logging/hare",
                           "qos": 0,
                           "retain": 0,
                           "payloadlen": 130,
                           "payload": {
                             "hostname": "canfb12",
                             "remote": "192.168.33.123",
                             "rhost": "192.168.33.1",
                             "service": "sshd",
                             "tst": 1552562392,
                             "tty": null,
                             "user": "jane"
                           }
                         }
```

https://jpmens.net/2018/03/25/alerting-on-ssh-logins/

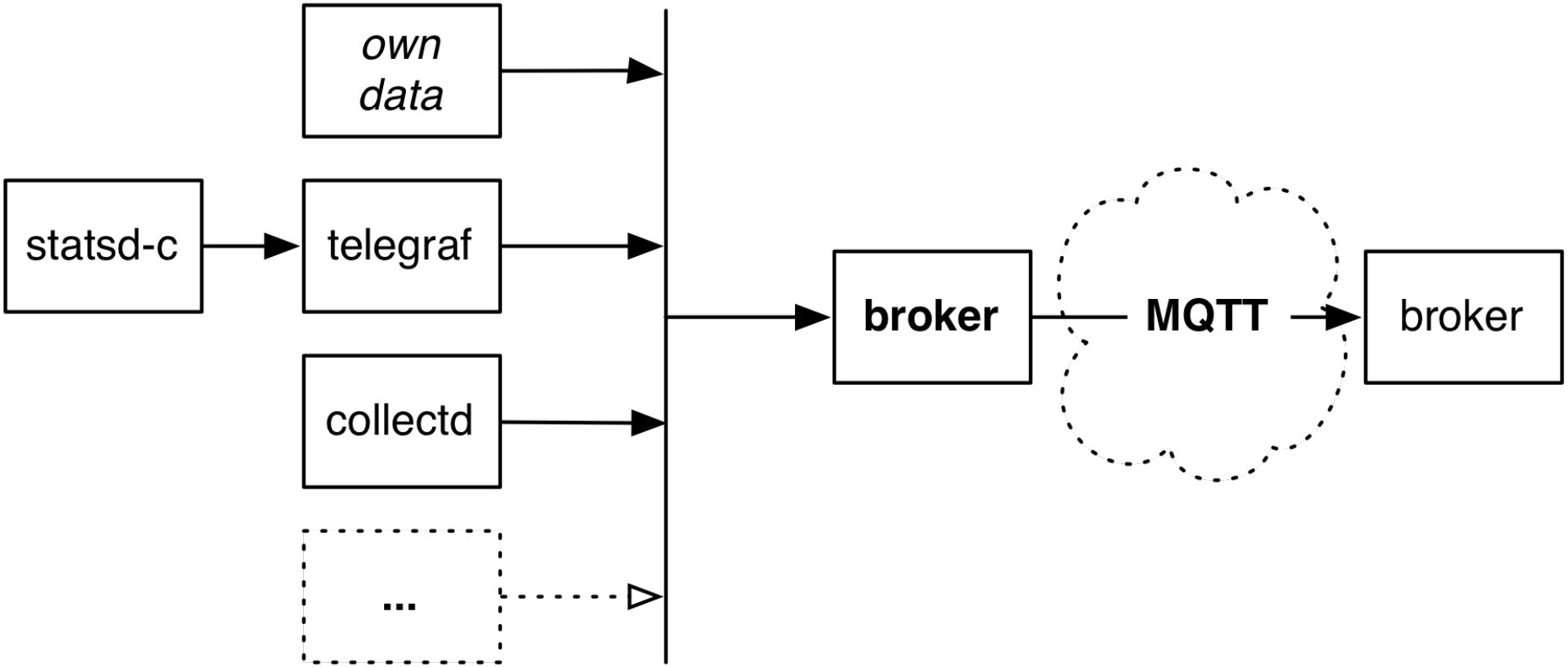
tracking **logins** (4)

```
Date: Thu, 14 Mar 2019 11:19:54 +0100
From: MQTTwarn <jpm@localhost>
Subject: SSH login on canfb12
X-Mailer: mqttwarn

login via sshd by jane on canfb12 from 192.168.33.1
at 2019-03-14 12:19:52
```



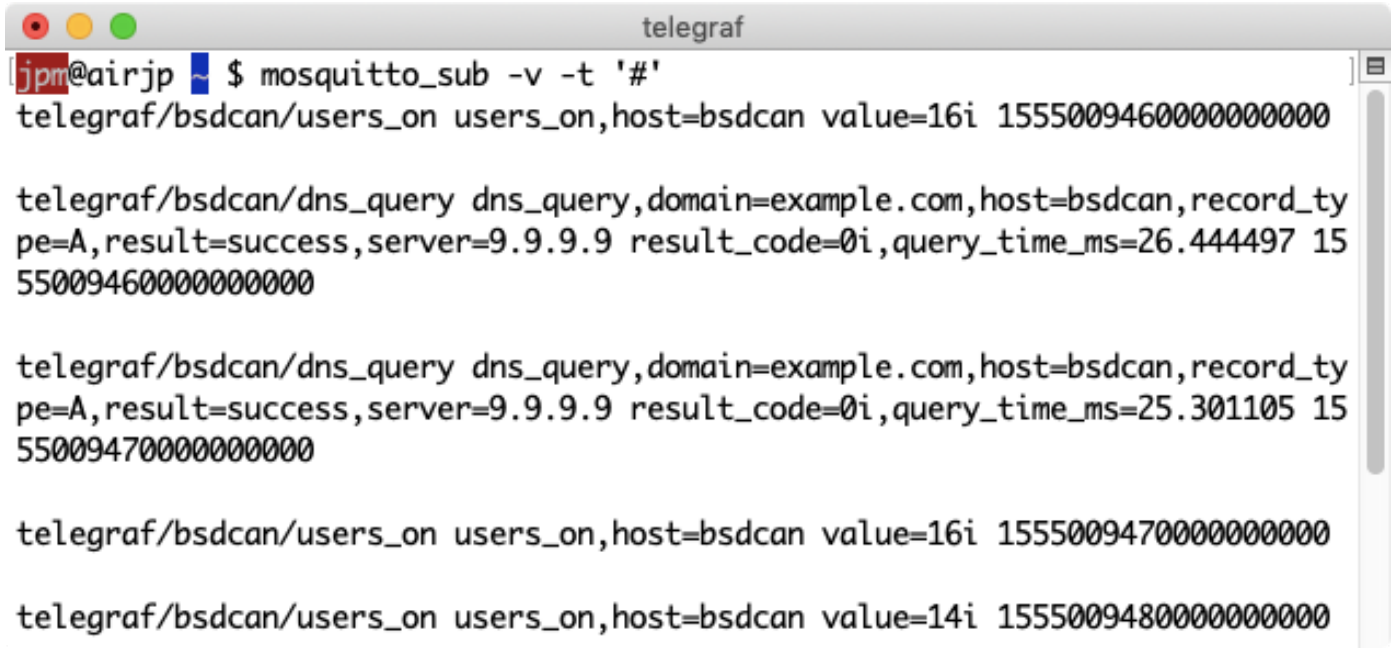
https://dan.langille.org/2018/04/15/using-mtqq-to-create-a-notification-network-mosquitto-mqttwarn-hare-and-hared/

For the **sysadmin**

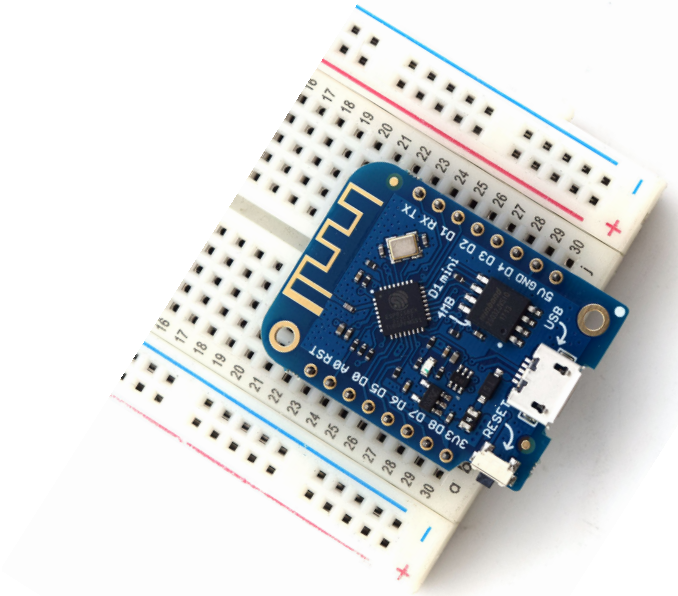


telegraf to mqtt

```
[agent]
  interval = "10s"
  hostname = "bsdcan"

[[outputs.mqtt]]
   servers = ["localhost:1883"]
   topic_prefix = "telegraf"
   batch = false
   data_format = "influx"

[[inputs.dns_query]]
  servers = ["9.9.9.9"]
  domains = ["example.com"]
  record_type = "A"

[[inputs.exec]]
  commands = ["./howmany.sh"]
  name_override = "users_on"
  data_format = "value"
  data_type = "integer"
```



Your **things** speak **MQTT**

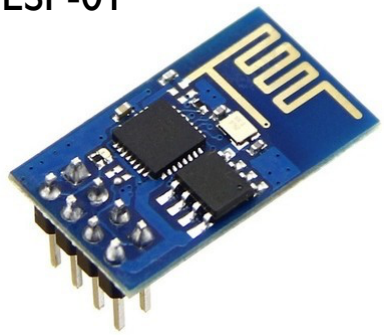


ESP8266

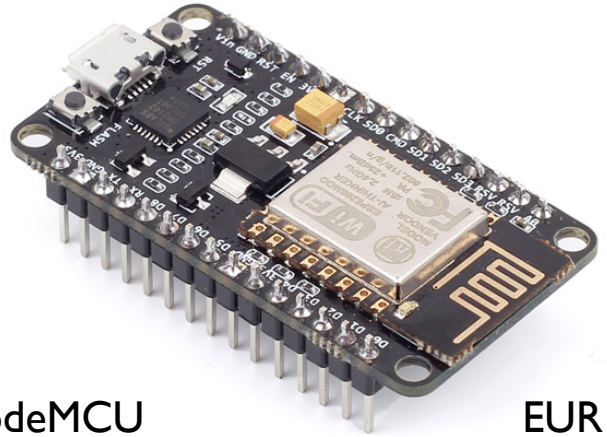
ESP-12

EUR 1.50

ESP-01



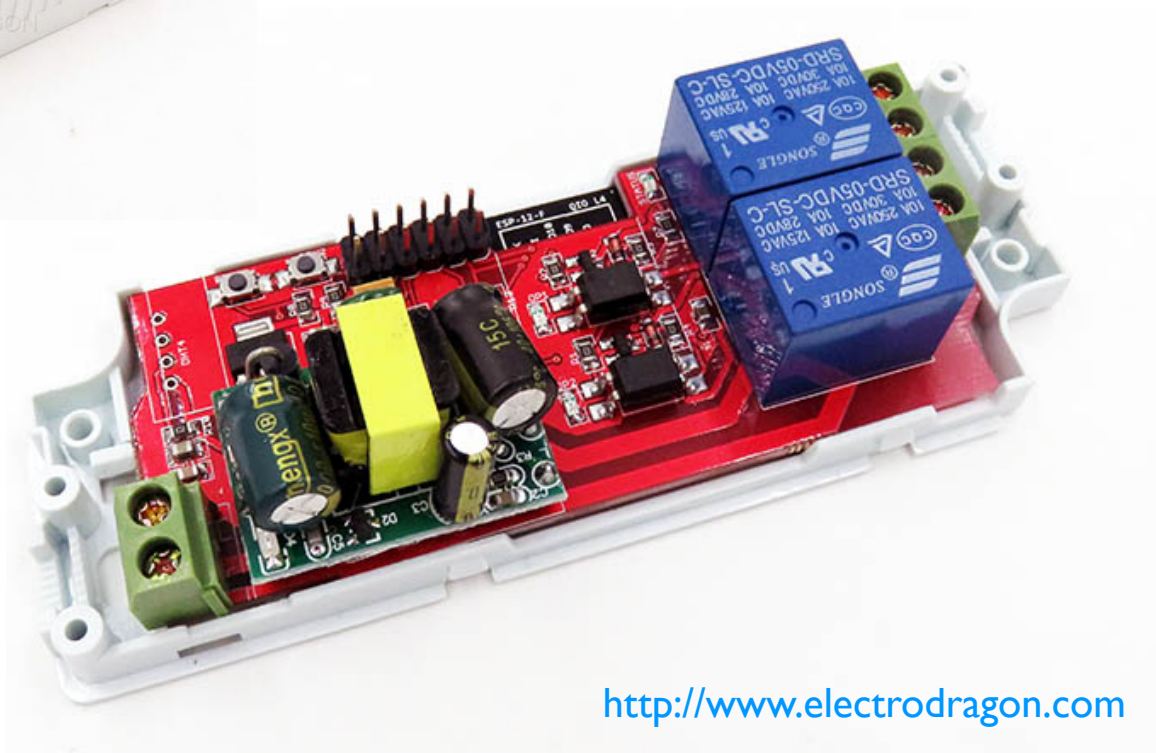
EUR 1.50



NodeMCU

EUR 2.60

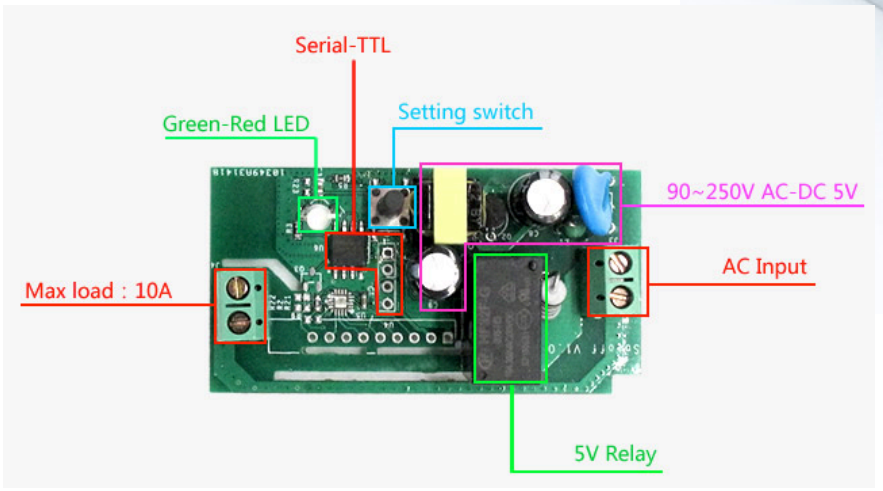
Electrodragon



EUR 5.50

http://www.electrodragon.com

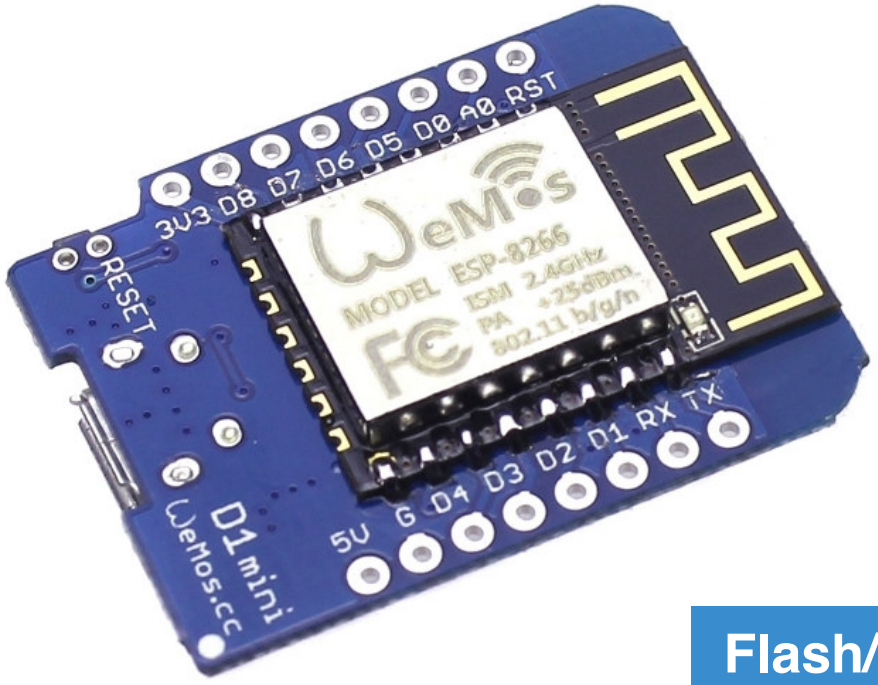
Sonoff



EUR 4.47

https://www.itead.cc/sonoff-wifi-wireless-switch.html

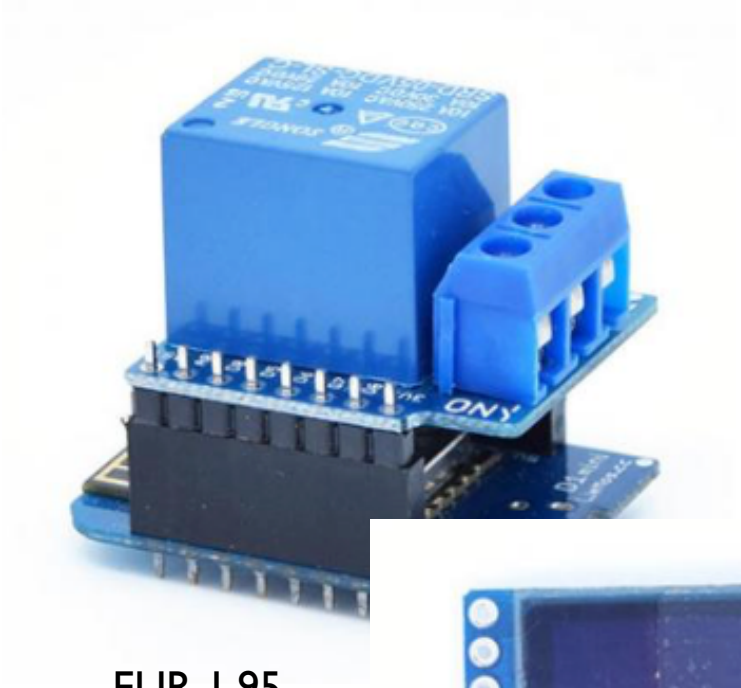
Wemos D1 mini



EUR 4.00

Flash/RAM	4MB / 64 KB
Voltage	3.3V
Digital I/O	11
Analog	1

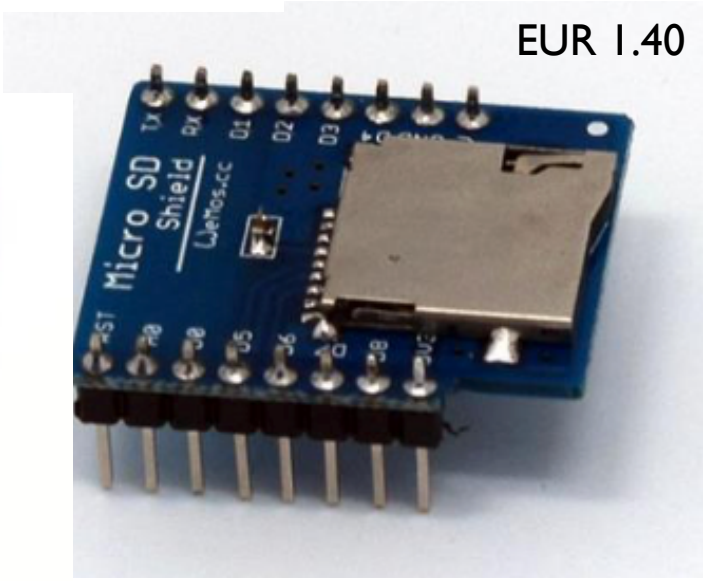
Wemos shields



EUR 1.95

EUR 2.90

EUR 4.50

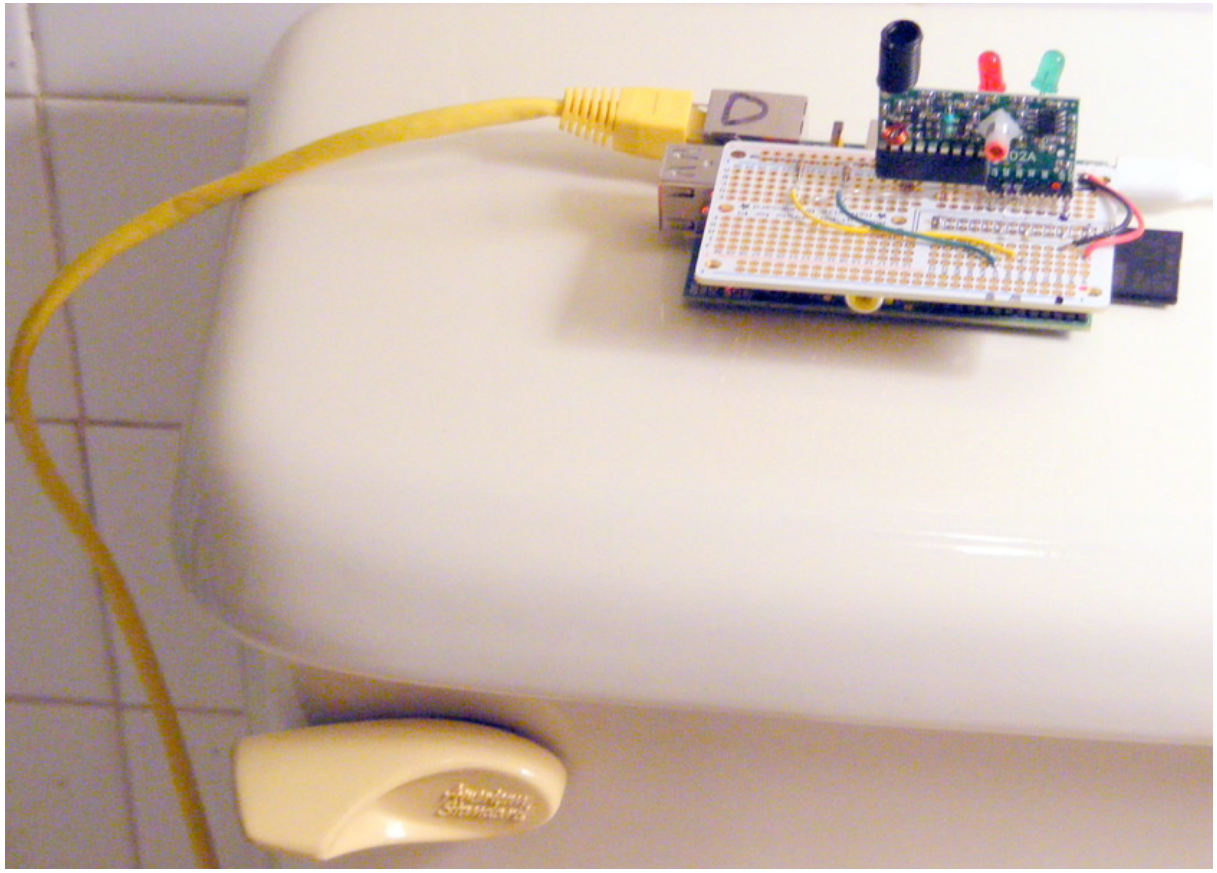


EUR 1.40

Arduino **IDE**


homie-mqtt-demo | Arduino 1.6.12
homie-mqtt-demo
/*
* button homie for
* async-mqtt-demo
*/
#include <Homie.h>
#define FW_NAME "button-homie"
#define FW_VERSION "1.0.0"
const int PIN_BUTTON = D3;
Bounce debouncer = Bounce();
int lastButtonValue = -1;
HomieNode buttonNode("button", "button");
Done compiling.
Build options changed, rebuilding all
Sketch uses 371,527 bytes (35%) of program storage space. Maximum is 1,044,464 bytes.
Global variables use 45,556 bytes (55%) of dynamic memory, leaving 36,364 bytes for local variables. Maximum is 81,920 b
8
WeMos D1 R2 & mini, 80 MHz, 921600, 4M (3M SPIFFS) on /dev/cu.SLAB_USBtoUART

IoT



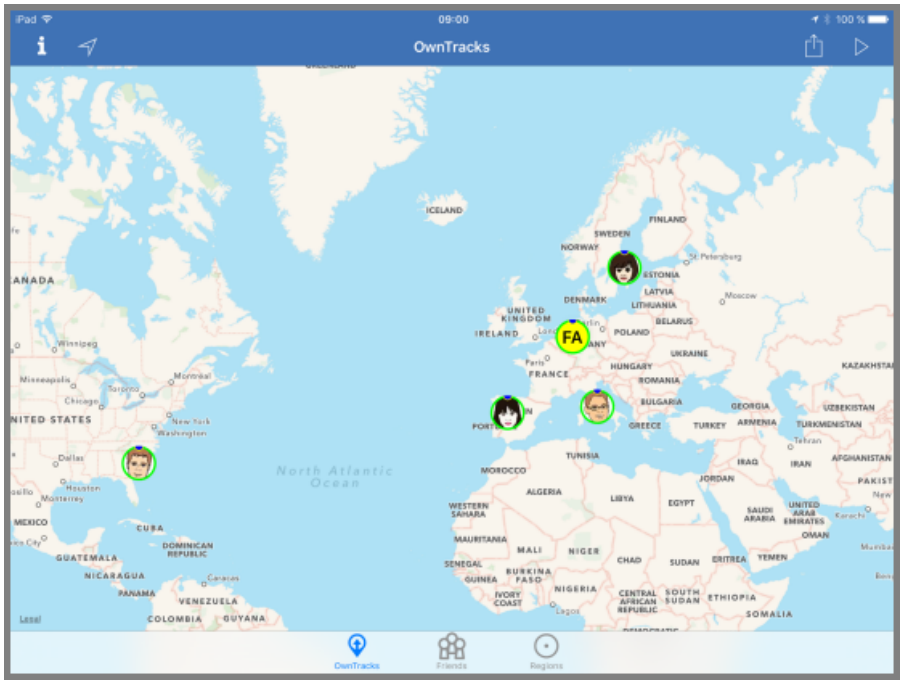
http://www.instructables.com/id/Internet-of-Things-Toilet-Uploads-Events-to-the-Cl/

Last Will & Testament

```
#!/usr/bin/env python

import paho.mqtt.subscribe as subscribe
import os

def on_message(client, userdata, m):
    print("%s %s" % (m.topic, m.payload))

lwt = {
    "topic"  : "clients/{0}".format(os.path.basename(__file__)),
    "payload": "I am no longer" }

subscribe.callback(on_message, "test/+", hostname="localhost", will=lwt)
```

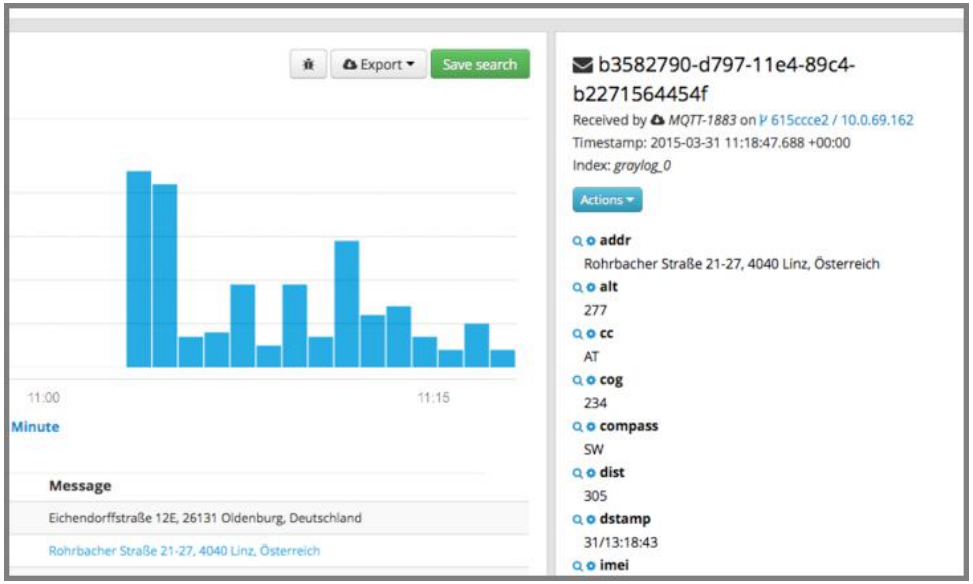
practical **solutions**

alerting, metering, **logging**, location awareness, tracking, **automation**, and **controlling**, host **monitoring**



MQTT in the wild

Graylog, beaver, Ansible, RabbitMQ, collectd, openHAB, ~~Github~~, Wireshark, Flukso, RemakeElectric, Jenkins, Diamond, OwnTracks, Telegraf



mqtt.org

@mqttorg