medando

Messaging for the Internet of Things

Andreas Schreiber <andreas.schreiber@medando.de>

EuroPython 2013, Florence, July 3, 2013

Me

Scientist, Head of department

DLR Deutsches Zentrum für Luft- und Raumfahrt
German Aerospace Center

Founder, CEO

medando

Python user since 1992

Py cologne

PyHPC2013

PyCon. DE 2013

Why?

What is my motivation for messaging and the Internet of Things?

Quantified Self – tracking myself

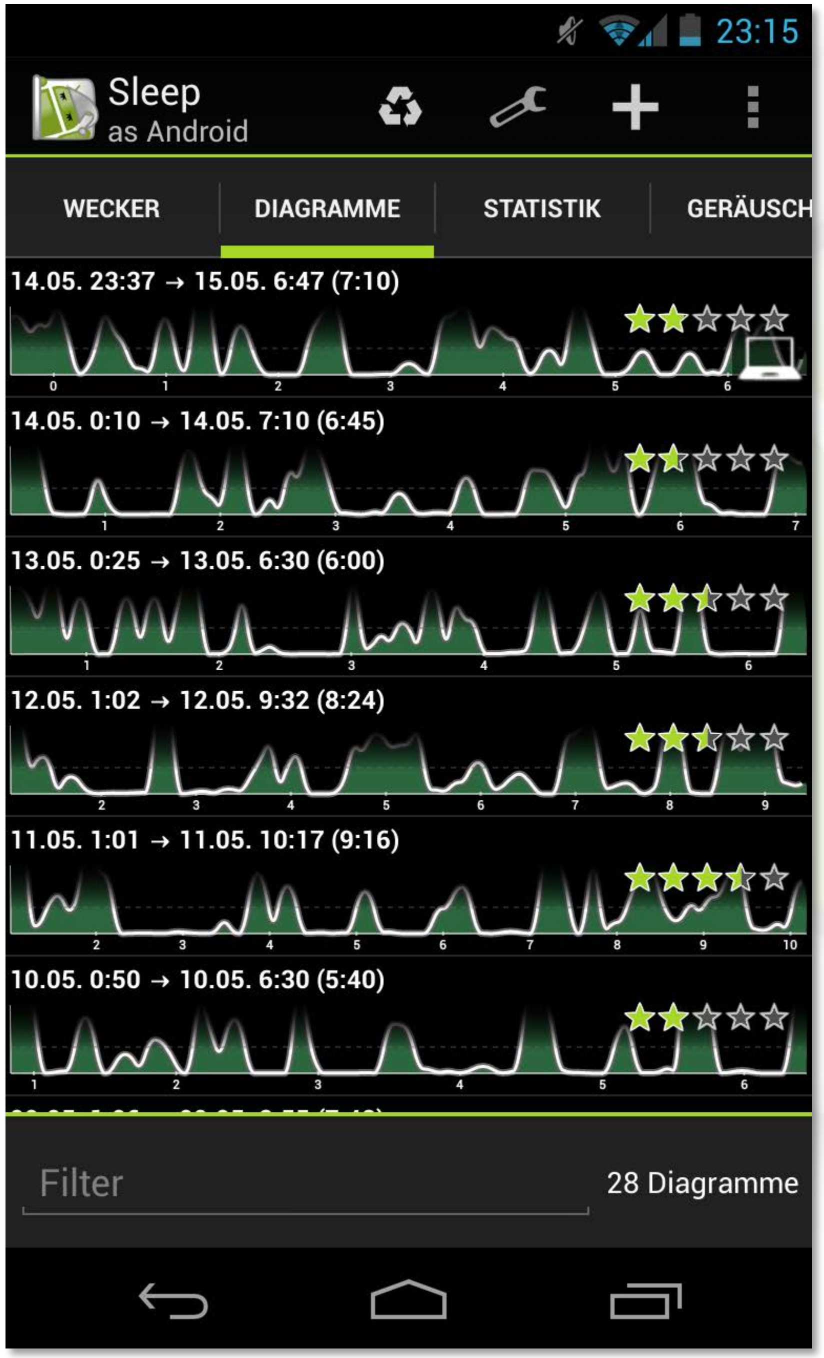
- With sensors
- With smartphone apps

Steps

Activity

Blood Pressure

Smartphone: Sleep, Coffee, Medication, Money



14.05. 23:37 → 15.05. 6:47 (7:10)

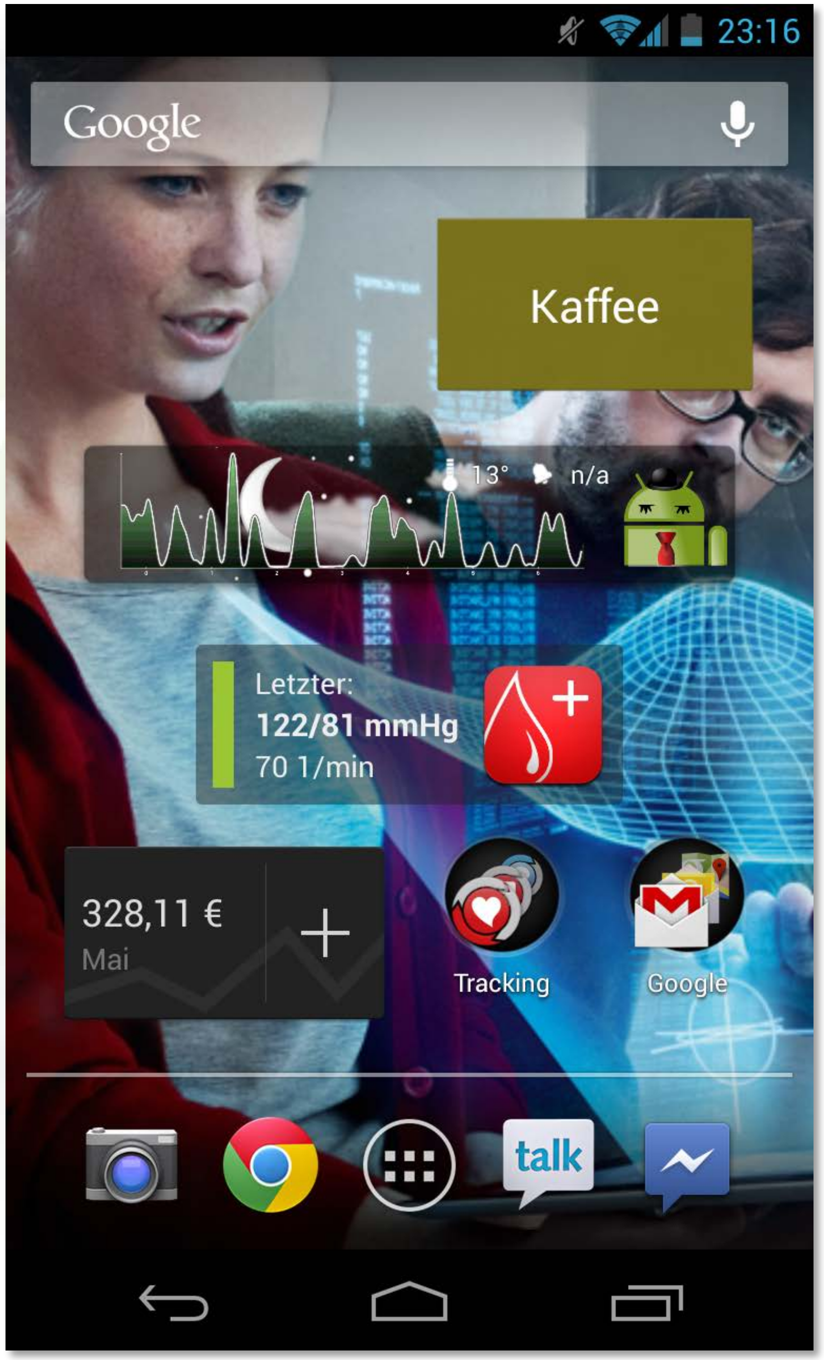
14.05. 0:10 → 14.05. 7:10 (6:45)

13.05. 0:25 → 13.05. 6:30 (6:00)

12.05. 1:02 → 12.05. 9:32 (8:24)

11.05. 1:01 → 11.05. 10:17 (9:16)

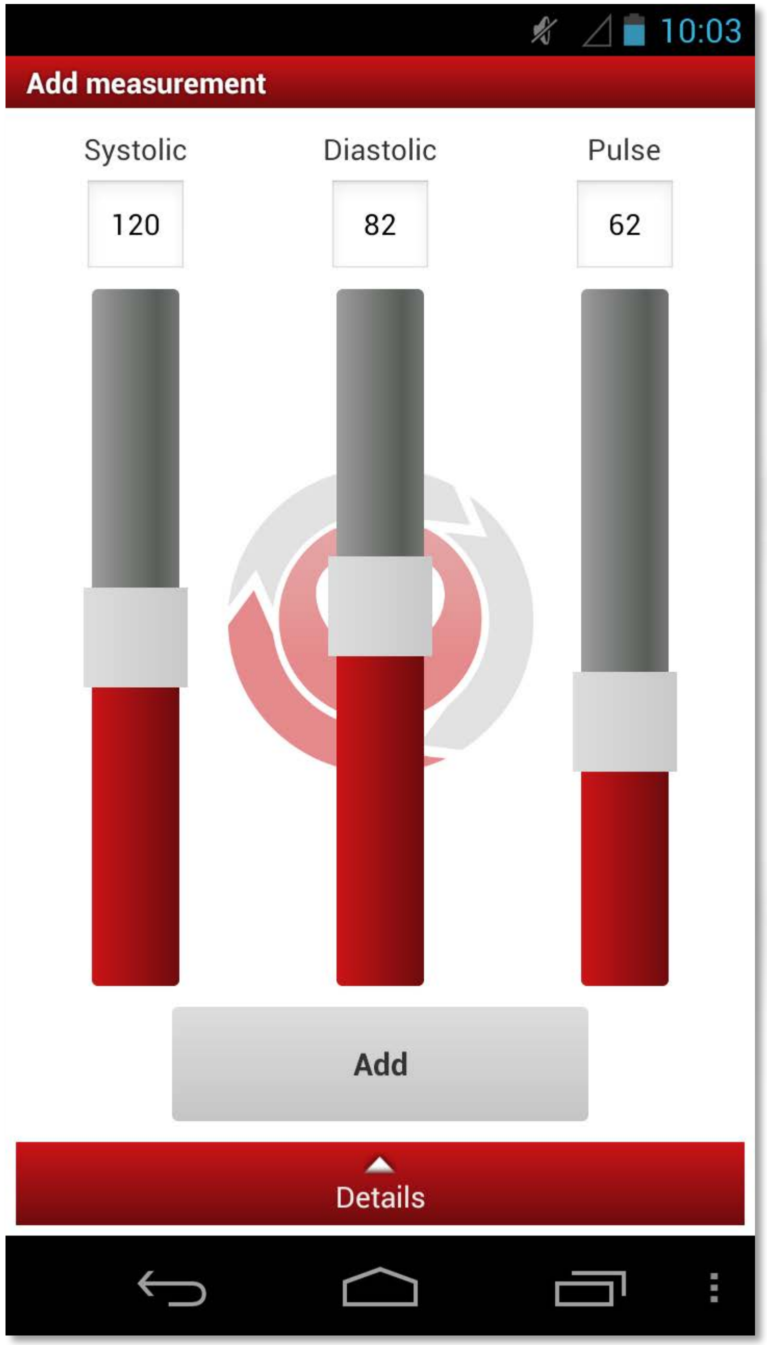
10.05. 0:50 → 10.05. 6:30 (5:40)



Letzter:
122/81 mmHg
70 1/min

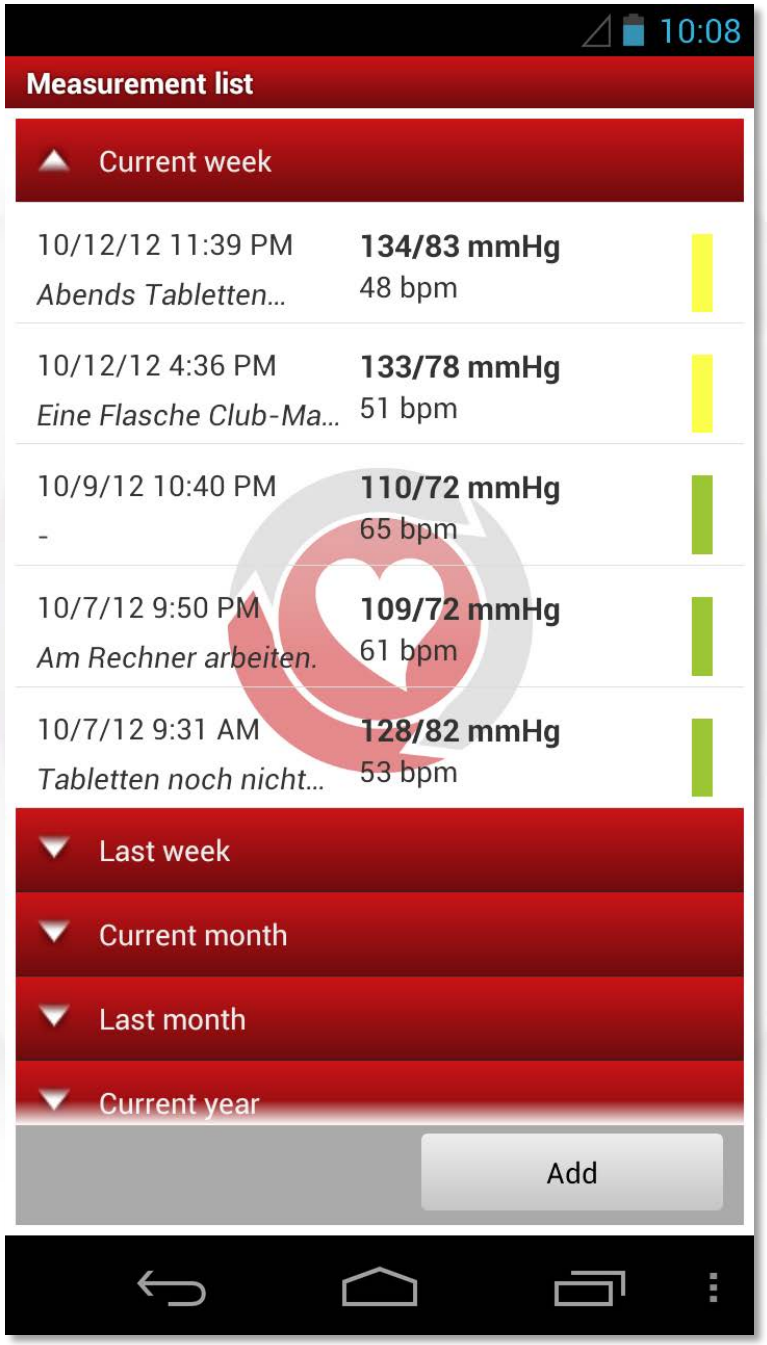
328,11 €
Mai

Medando: *BloodPressureCompanion*



Add measurement

Systolic	Diastolic	Pulse
120	82	62



Measurement list

Current week

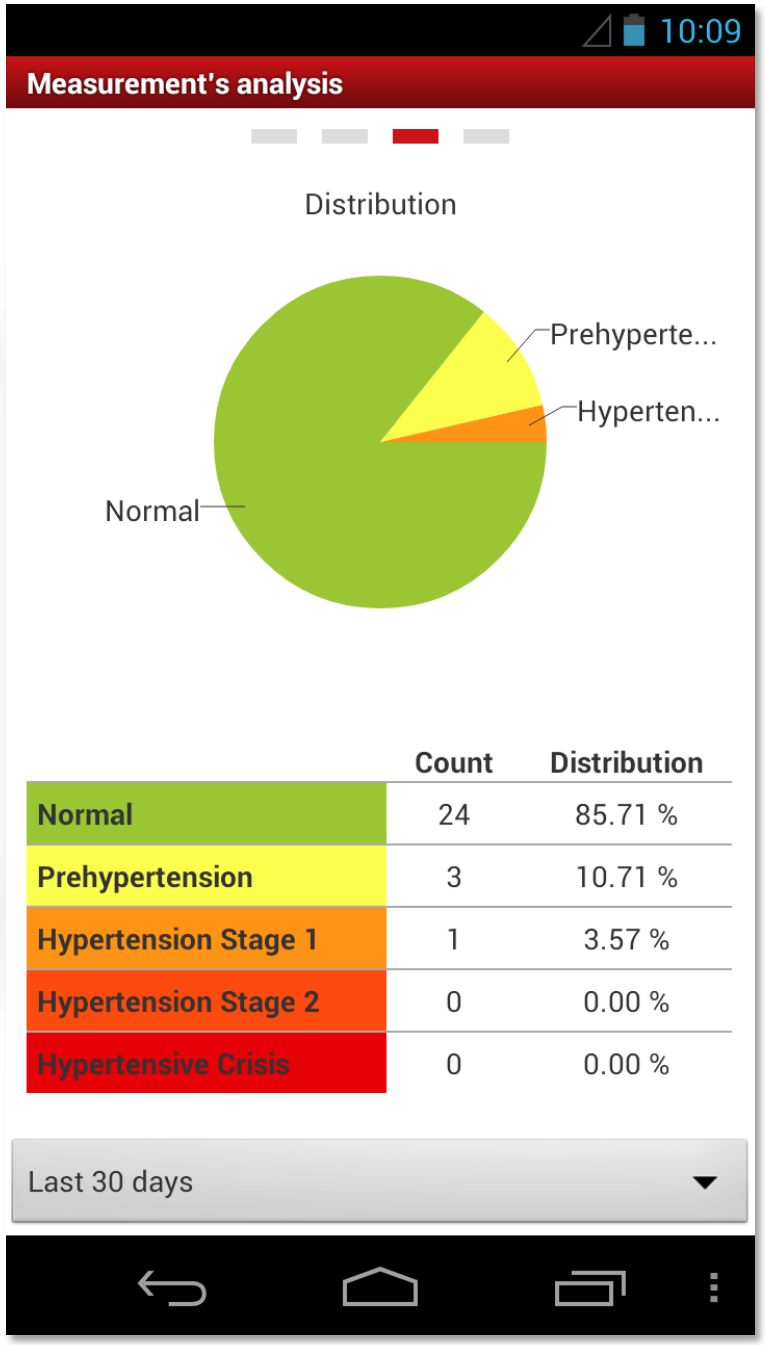
10/12/12 11:39 PM *Abends Tabletten...*	**134/83 mmHg** 48 bpm
10/12/12 4:36 PM *Eine Flasche Club-Ma...*	**133/78 mmHg** 51 bpm
10/9/12 10:40 PM -	**110/72 mmHg** 65 bpm
10/7/12 9:50 PM *Am Rechner arbeiten.*	**109/72 mmHg** 61 bpm
10/7/12 9:31 AM *Tabletten noch nicht...*	**128/82 mmHg** 53 bpm

Last week

Current month

Last month

Current year

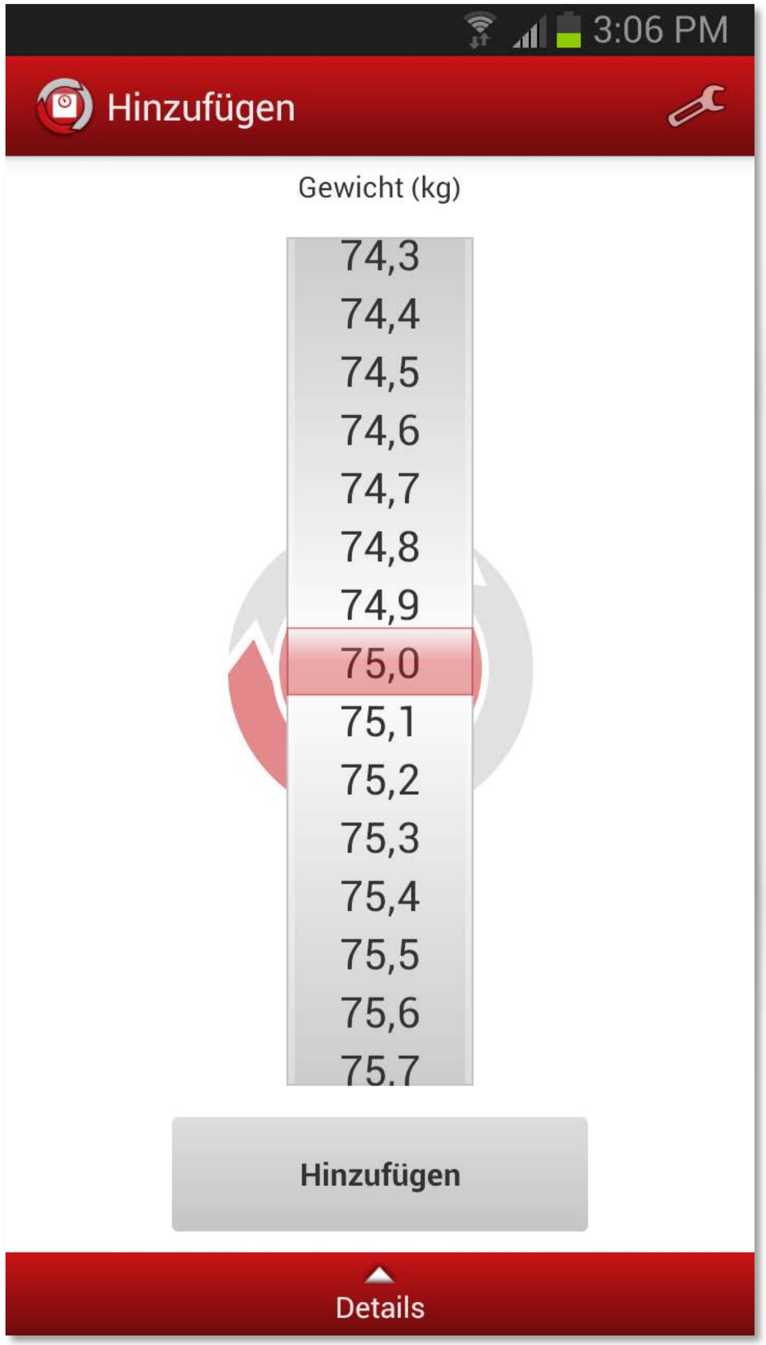


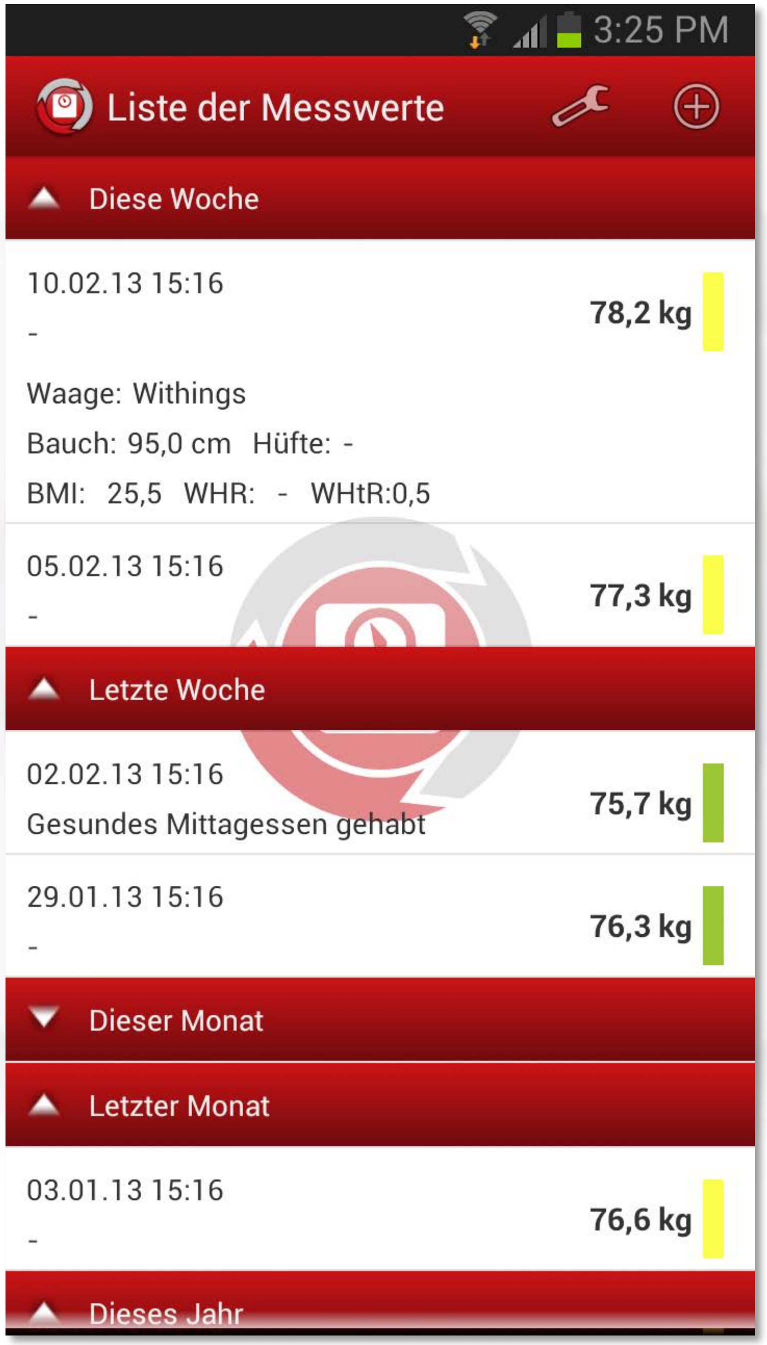
Measurement's analysis

	Count	Distribution
Normal	24	85.71 %
Prehypertension	3	10.71 %
Hypertension Stage 1	1	3.57 %
Hypertension Stage 2	0	0.00 %
Hypertensive Crisis	0	0.00 %

Last 30 days ▾

Medando: *WeightCompanion*





Diese Woche

10.02.13 15:16 78,2 kg
-
Waage: Withings
Bauch: 95,0 cm Hüfte: -
BMI: 25,5 WHR: - WHtR:0,5

05.02.13 15:16 77,3 kg
-

Letzte Woche

02.02.13 15:16 75,7 kg
Gesundes Mittagessen gehabt

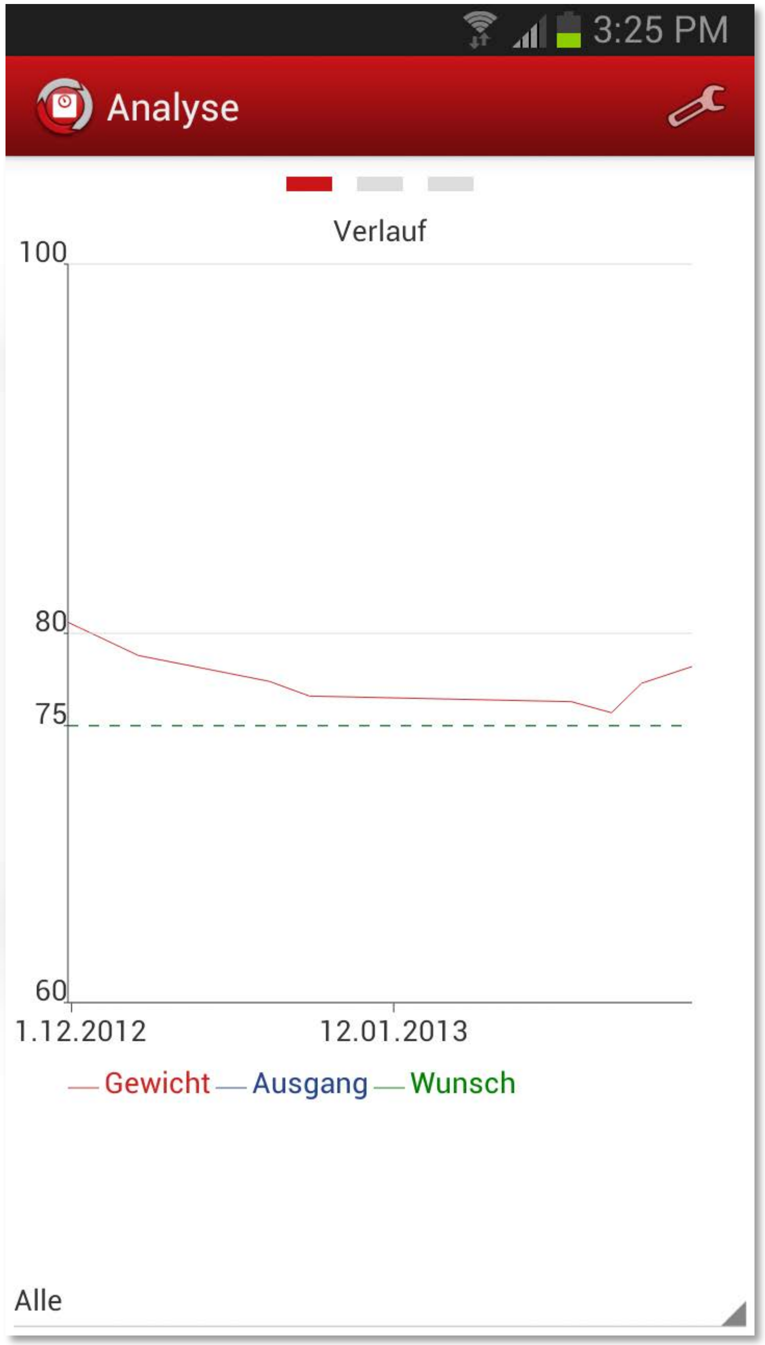
29.01.13 15:16 76,3 kg
-

Dieser Monat

Letzter Monat

03.01.13 15:16 76,6 kg
-

Dieses Jahr



Many Devices, Sensors, and Apps

Data Exchange

Internet of Things

Billions of devices, sensors, and chips

- Connected physical objects (or their virtual representation)
- Connected via the internet
- Uniquely identified
- They interact

Devices

The “Things” are

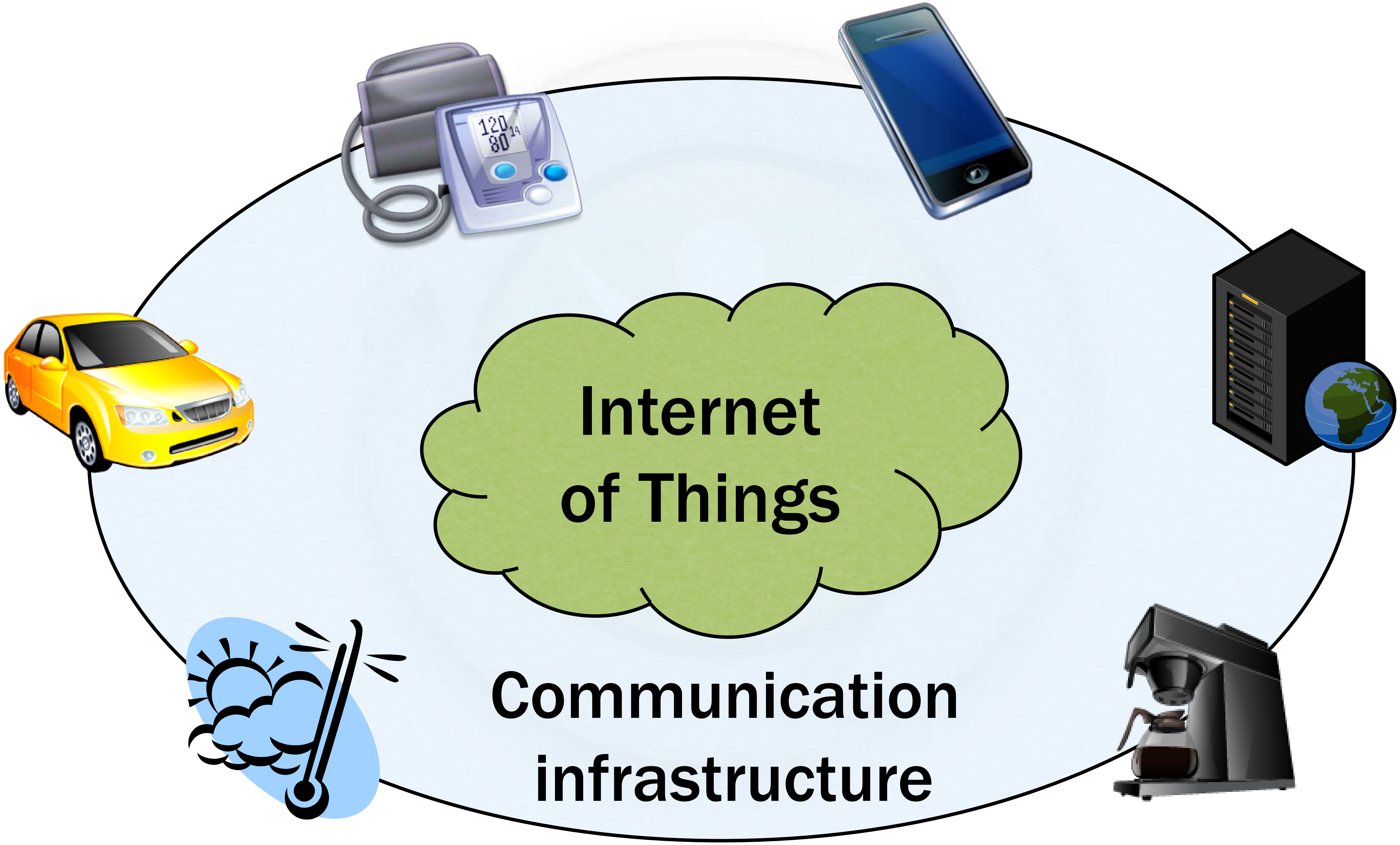
- Embedded controllers
- Sensors
- Actuators

Growth

Number of devices connected to the internet grow every day

50.000.000.000 "Things" by 2020

Communication

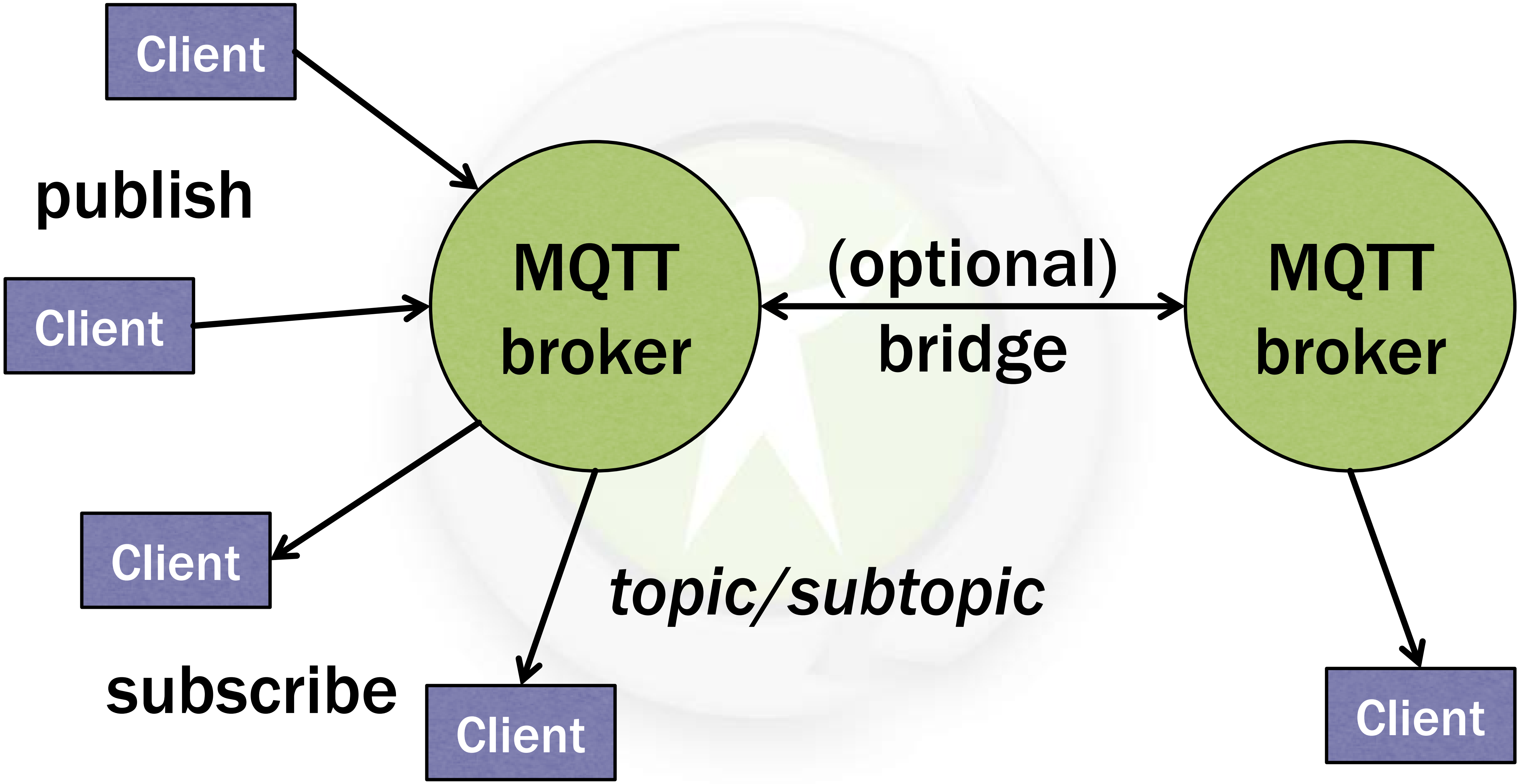


MQTT

MQ Telemetry Transport

- Machine-to-machine (M2M) connectivity protocol
- Publish/subscribe messaging
- Expect unreliable networks with low bandwidth and high latency
- Expect clients with limited processing resources
- Provides Quality of Service, if network/environment allows
- Easy to implement

Broker


Client
publish
Client
MQTT broker
(optional) bridge
MQTT broker
Client
topic/subtopic
subscribe
Client
Client

MQTT Protocol

- One-to-many message distribution over TCP/IP
- Notifies if clients disconnect abnormally
- Message format
  - Fixed 2-byte header
  - Variable header for some message type
  - Payload (e.g., the topic or small pieces of data)

Topics

- Messages in MQTT are published on topics
- No need to configure, just publish on it
- Topics are hierarchical, with “/” as separator

```
my/home/temperature/kitchen

my/home/temperature/livingroom

my/server/temperature
```

MQTT Implementations

Servers/Brokers

- IBM Websphere MQ
- RSMB
- Mosquitto
- Eclispe Paho
- MQTT.js
- Apache ActiveMQ
- RabittMQ
- HiveMQ

Libraries for

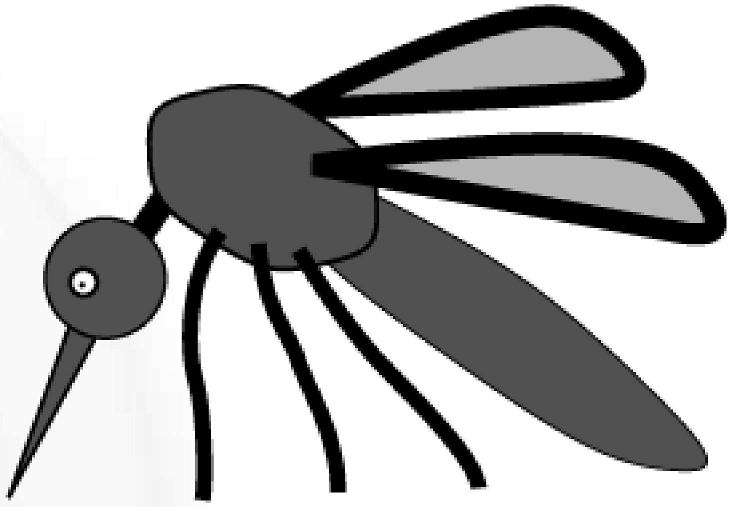
- C/C++
- Java
- Python
- Perl
- PHP
- Ruby
- …

`http://mqtt.org/wiki/software`

Mosquitto

Open Source MQTT Broker

- `http://mosquitto.org`
- Implemented in C
- Source code on bitbucket
- Many binary packages



Starting a Broker

- Install it

  -
    ```
    apt-get install mosquitto
    ```
- Just start with config file

  -
    ```
    mosquitto -c mosquitto.conf
    ```

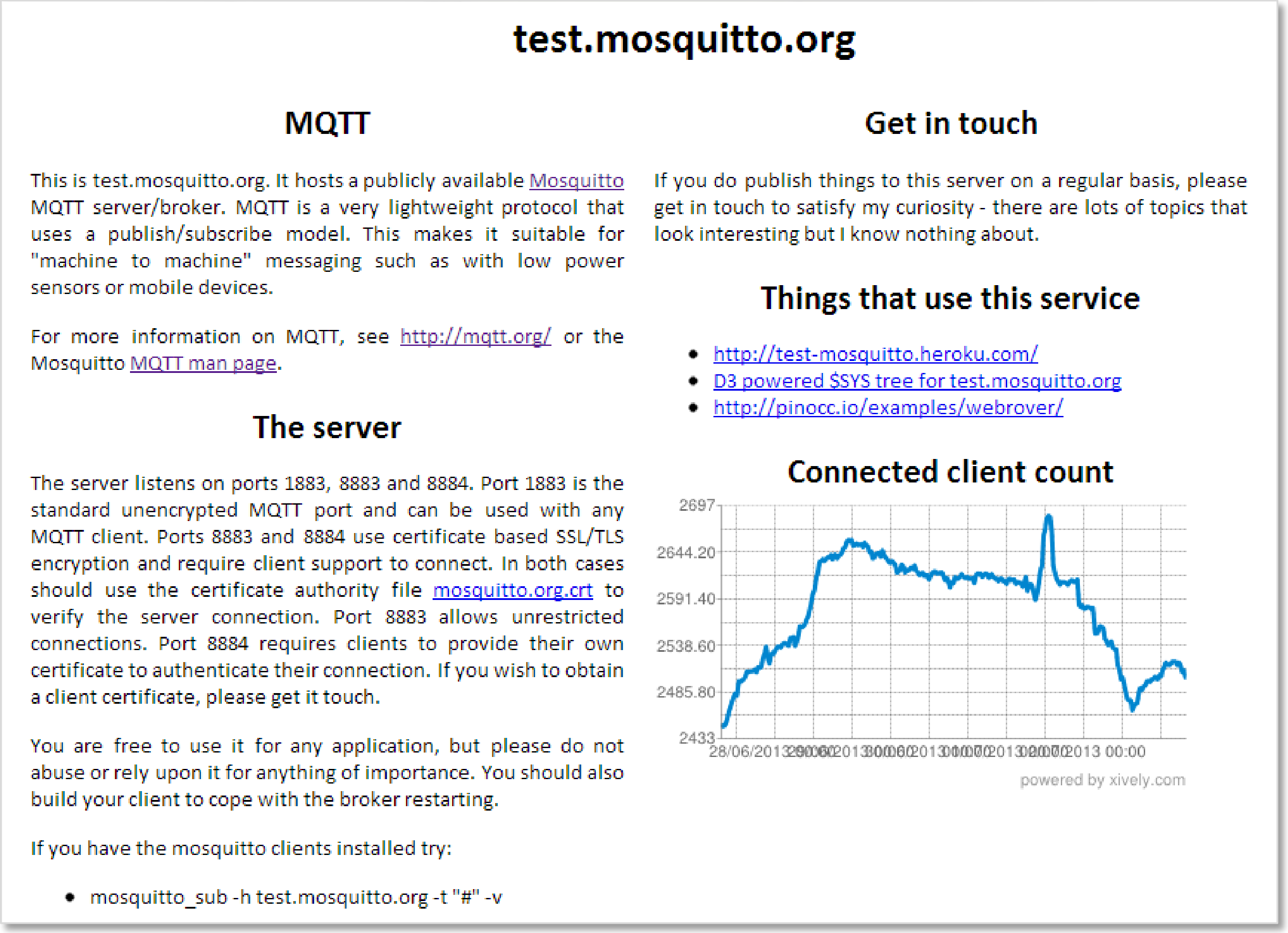
Broker Status

Mosquitto broker publishes status messages

```
$SYS/broker/messages/sent

$SYS/broker/subscriptions/count

$SYS/broker/uptime

. . .
```

test.mosquitto.org

Publicly available Mosquitto MQTT server/broker



test.mosquitto.org

MQTT

This is test.mosquitto.org. It hosts a publicly available Mosquitto MQTT server/broker. MQTT is a very lightweight protocol that uses a publish/subscribe model. This makes it suitable for "machine to machine" messaging such as with low power sensors or mobile devices.

For more information on MQTT, see http://mqtt.org/ or the Mosquitto MQTT man page.

The server

The server listens on ports 1883, 8883 and 8884. Port 1883 is the standard unencrypted MQTT port and can be used with any MQTT client. Ports 8883 and 8884 use certificate based SSL/TLS encryption and require client support to connect. In both cases should use the certificate authority file mosquitto.org.crt to verify the server connection. Port 8883 allows unrestricted connections. Port 8884 requires clients to provide their own certificate to authenticate their connection. If you wish to obtain a client certificate, please get it touch.

You are free to use it for any application, but please do not abuse or rely upon it for anything of importance. You should also build your client to cope with the broker restarting.

If you have the mosquitto clients installed try:

- mosquitto_sub -h test.mosquitto.org -t "#" -v

Get in touch

If you do publish things to this server on a regular basis, please get in touch to satisfy my curiosity - there are lots of topics that look interesting but I know nothing about.

Things that use this service

- http://test-mosquitto.heroku.com/
- D3 powered $SYS tree for test.mosquitto.org
- http://pinocc.io/examples/webrover/

Connected client count

Mosquitto with Python

Python client module

- Single file, pure Python implementation
- Publishing and receiving messages
- Callbacks
  - Connect
  - Disconnect
  - Publish
  - Message
  - Subscribe

Subscribe

```python
import mosquitto

def on_message(mosq, obj, msg):
    print(msg.topic + ' ' + str(msg.payload))

mqtt_client = mosquitto.Mosquitto()
mqtt_client.on_message = on_message

mqtt_client.connect('test.mosquitto.org')
mqtt_client.subscribe(‘#', 0) # all topics

return_code = 0
while return_code == 0:
    return_code = mqtt_client.loop()
```

Publish

```
import mosquitto

mqtt_client = mosquitto.Mosquitto()

mqtt_client.connect('test.mosquitto.org')

mqtt_client.publish('europython/demo',
                    'hello world', 1)
```

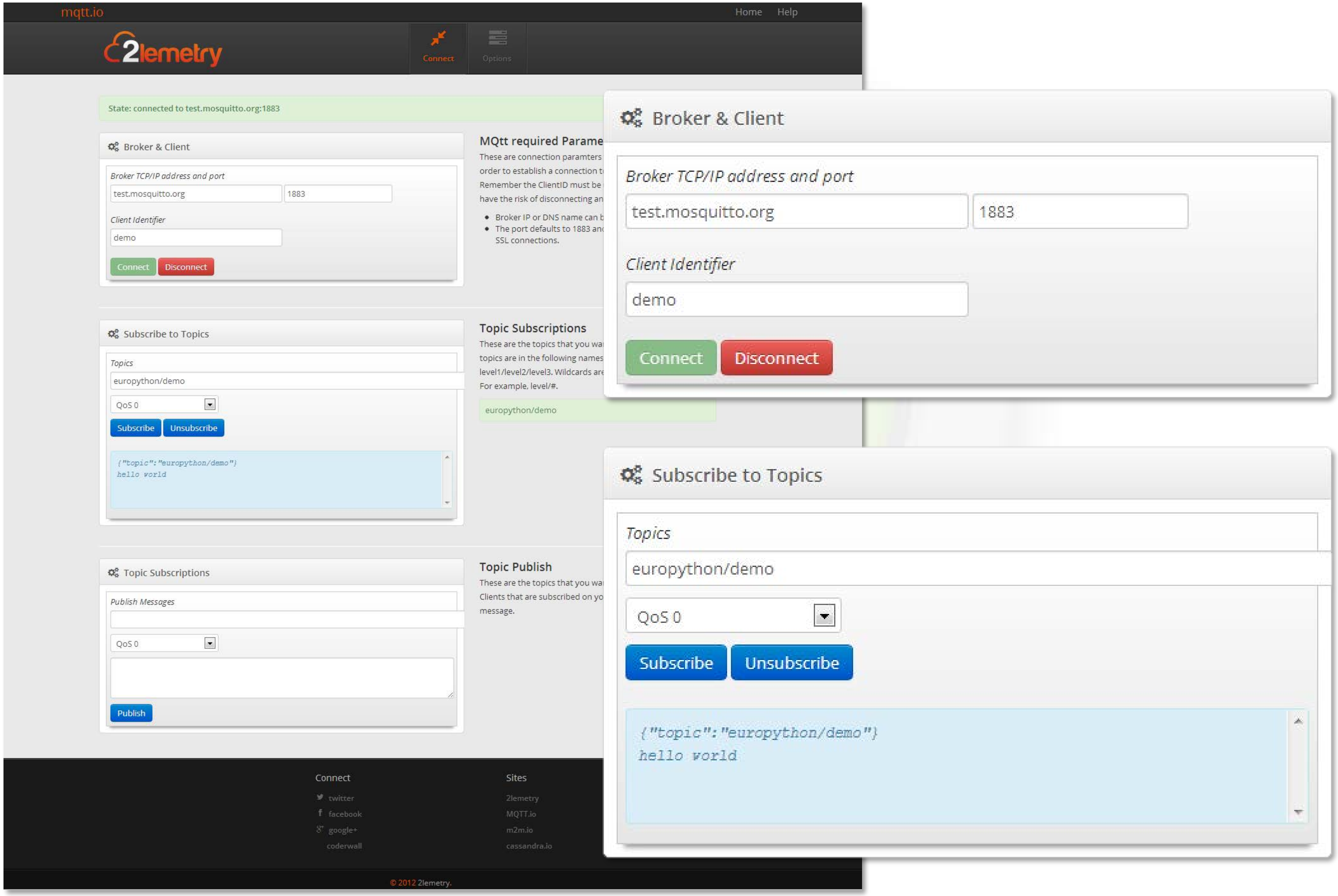
Tools

Tools for publishing and subscribing MQTT topics

- mqtt.io (Web)
- Eclipse Paho (Java library and Eclipse View)
- MQTT.app (Mac OS X)
- ...

See http://mqtt.org/wiki/software

mqtt.io



State: connected to test.mosquitto.org:1883

Broker & Client

Broker TCP/IP address and port: test.mosquitto.org 1883

Client Identifier: demo

MQtt required Parame

These are connection paramters
order to establish a connection t
Remember the ClientID must be
have the risk of disconnecting an

- Broker IP or DNS name can b
- The port defaults to 1883 an
SSL connections.

Subscribe to Topics

Topics: europython/demo

QoS 0

```
{"topic":"europython/demo"}
hello world
```

Topic Subscriptions

These are the topics that you wa
topics are in the following names
level1/level2/level3. Wildcards are
For example, level/#.

europython/demo

Topic Subscriptions

Publish Messages

QoS 0

Topic Publish

These are the topics that you wa
Clients that are subscribed on yo
message.

Broker & Client

Broker TCP/IP address and port: test.mosquitto.org 1883

Client Identifier: demo

Connect Disconnect

Subscribe to Topics

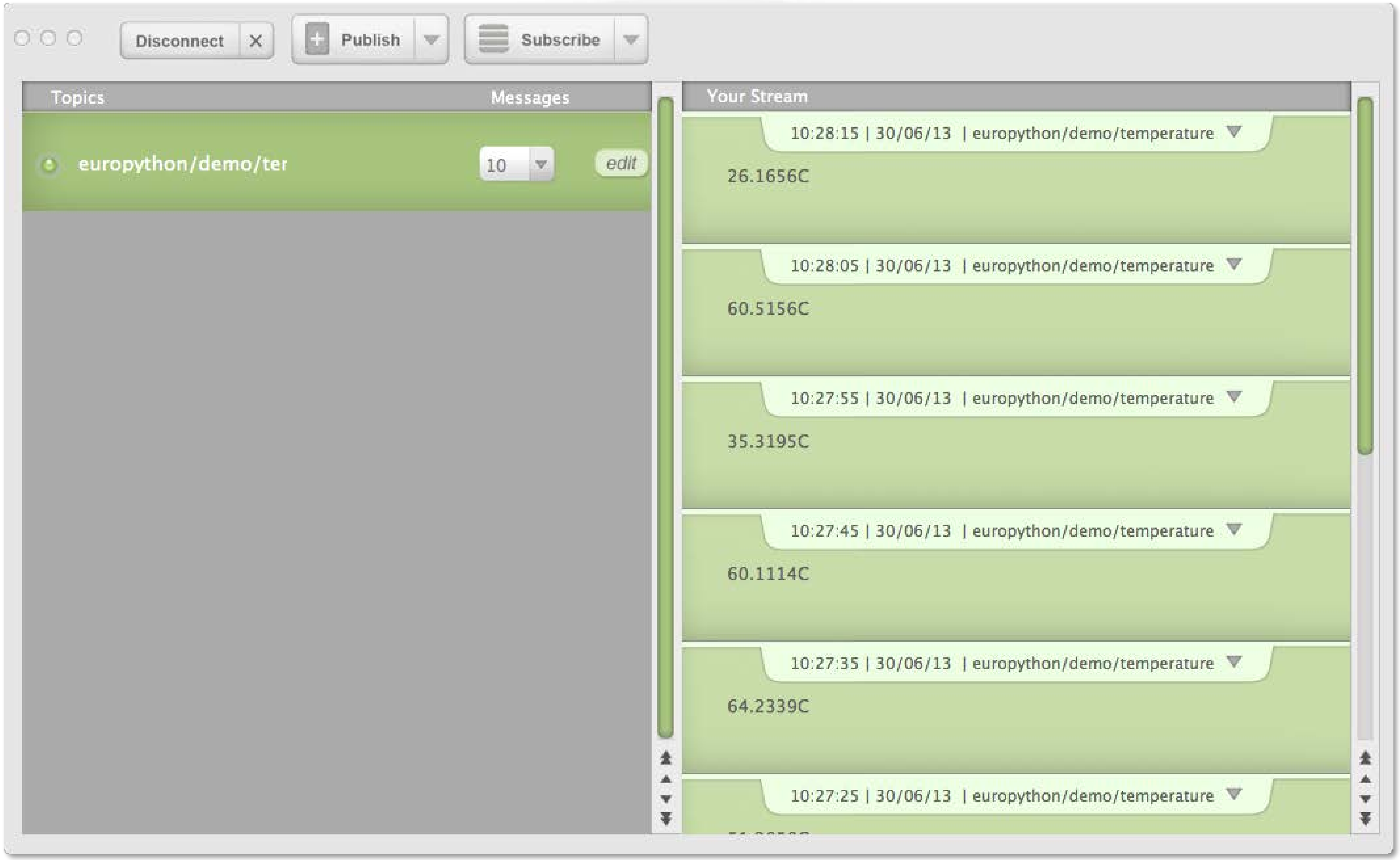
Topics: europython/demo

QoS 0

Subscribe Unsubscribe

{"topic":"europython/demo"}
hello world

MQTT.app (OS X)



europython/demo/ter 10

10:28:15 | 30/06/13 | europython/demo/temperature

26.1656C

10:28:05 | 30/06/13 | europython/demo/temperature

60.5156C

10:27:55 | 30/06/13 | europython/demo/temperature

35.3195C

10:27:45 | 30/06/13 | europython/demo/temperature

60.1114C

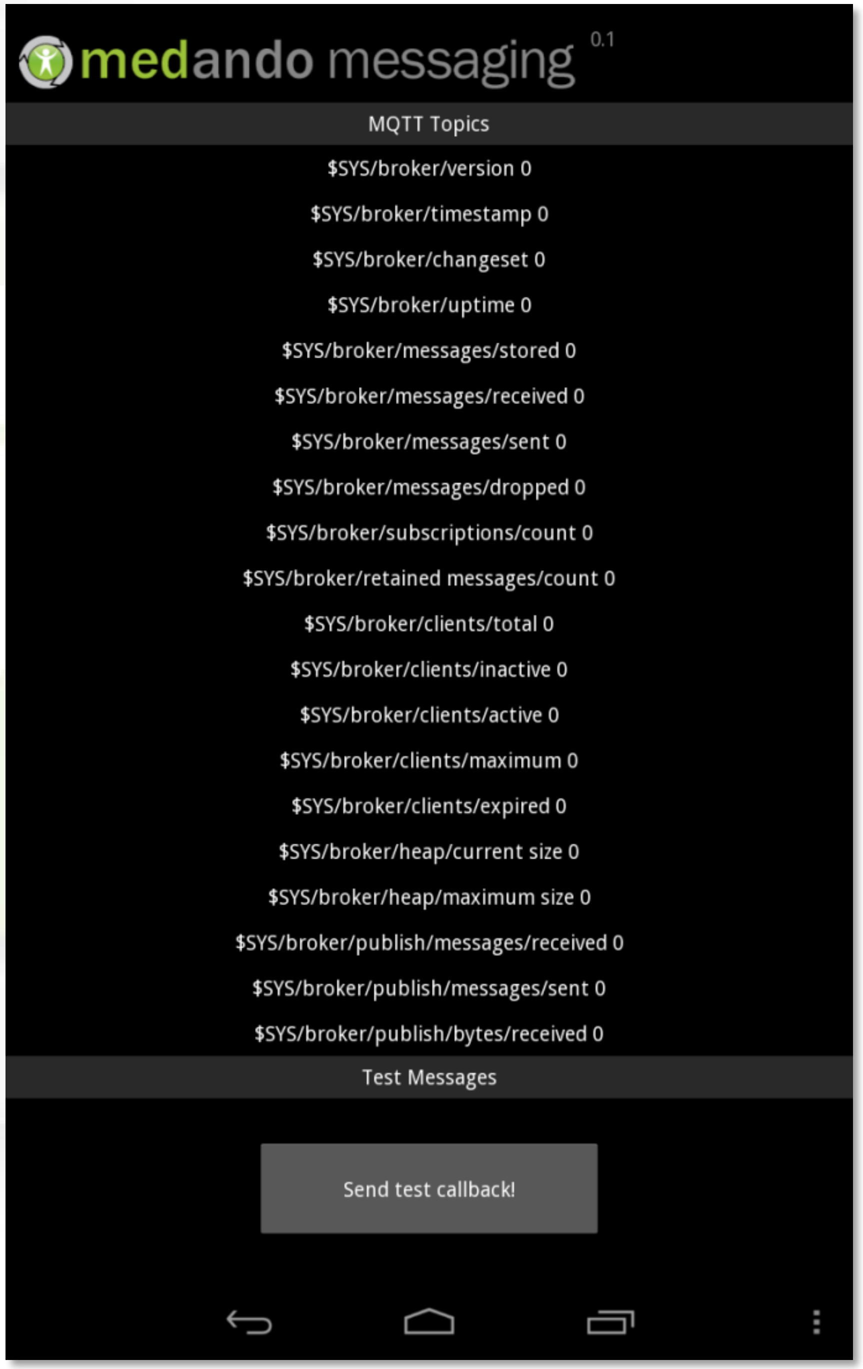
10:27:35 | 30/06/13 | europython/demo/temperature

64.2339C

10:27:25 | 30/06/13 | europython/demo/temperature

Mosquitto on Android

- The Python module works with python-for-android
- Easy to use in Kivy clients



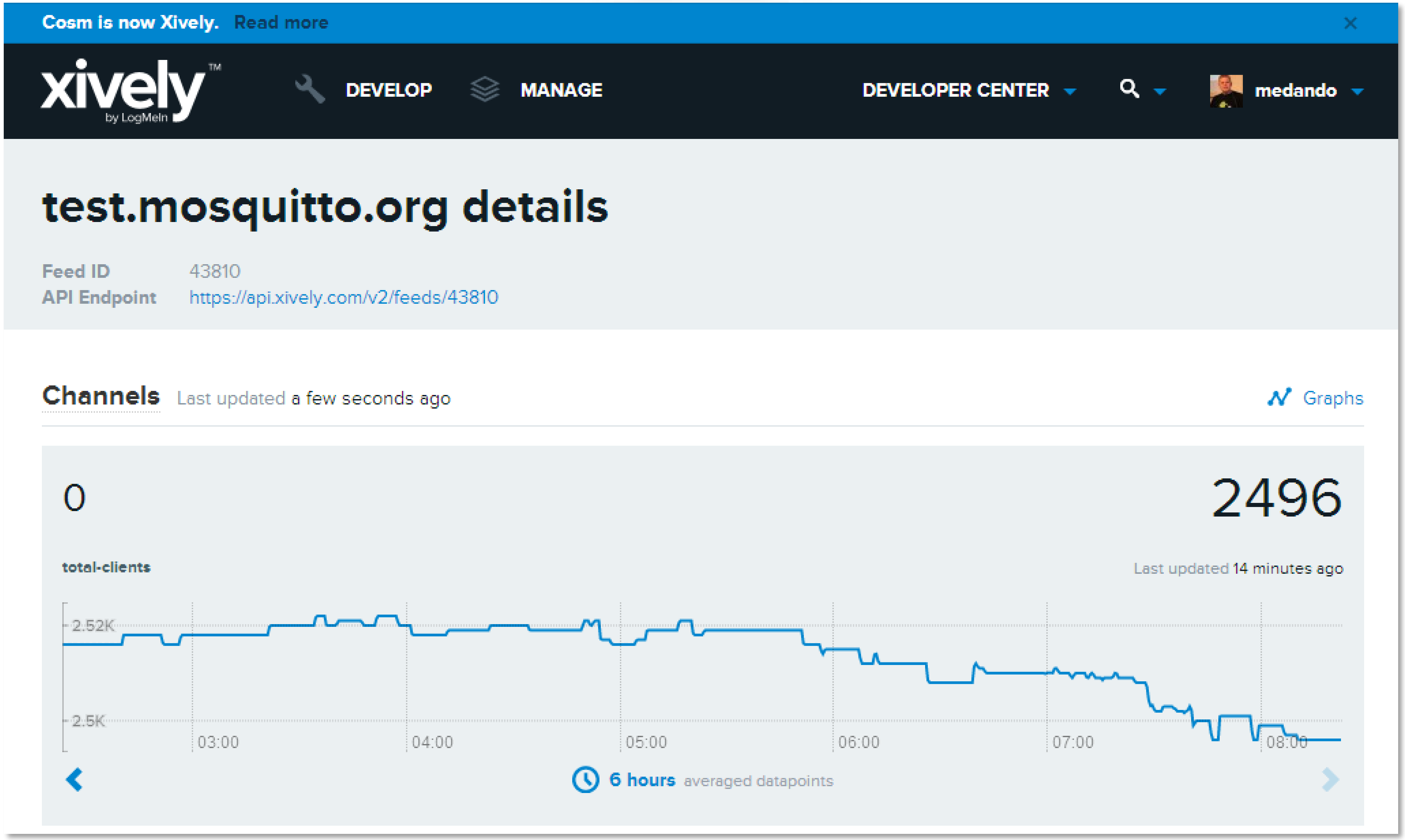
MQTT Topics

```
$SYS/broker/version 0
$SYS/broker/timestamp 0
$SYS/broker/changeset 0
$SYS/broker/uptime 0
$SYS/broker/messages/stored 0
$SYS/broker/messages/received 0
$SYS/broker/messages/sent 0
$SYS/broker/messages/dropped 0
$SYS/broker/subscriptions/count 0
$SYS/broker/retained messages/count 0
$SYS/broker/clients/total 0
$SYS/broker/clients/inactive 0
$SYS/broker/clients/active 0
$SYS/broker/clients/maximum 0
$SYS/broker/clients/expired 0
$SYS/broker/heap/current size 0
$SYS/broker/heap/maximum size 0
$SYS/broker/publish/messages/received 0
$SYS/broker/publish/messages/sent 0
$SYS/broker/publish/bytes/received 0
```

Test Messages

kivy.org

Xively – Public Cloud for the Internet of Things



Cosm is now Xively. Read more

test.mosquitto.org details

Feed ID 43810
API Endpoint https://api.xively.com/v2/feeds/43810

Channels Last updated a few seconds ago

MQTT Usage Examples

- Home automation with Raspberry Pi
- Android Push Notification

Getting sensor data with sensors connected via 1-Wire

- *1-Wire*: Single line bus system, low-speed
- Sensors for temperature, voltage, light, humidity, …
- Connected via 1-Wire-USB adapter

Temperature Sensor

http://www.iButtonLink.com

Temperature Sensor

http://www.iButtonLink.com

Mosquitto on Raspberry Pi

Mosquitto works nicely on Raspberry Pi

- Just install
  - `apt-get install mosquitto`
- You can start the broker or clients

Getting Temperature

Getting measurements from 1-Wire devices on Linux

- Two solutions that work with Python
  - OWFS: One Wire File System (`http://owfs.org`)
  - DigiTemp and DigitemPy (`http://www.digitemp.com`)

Publishing Temperature with OWFS

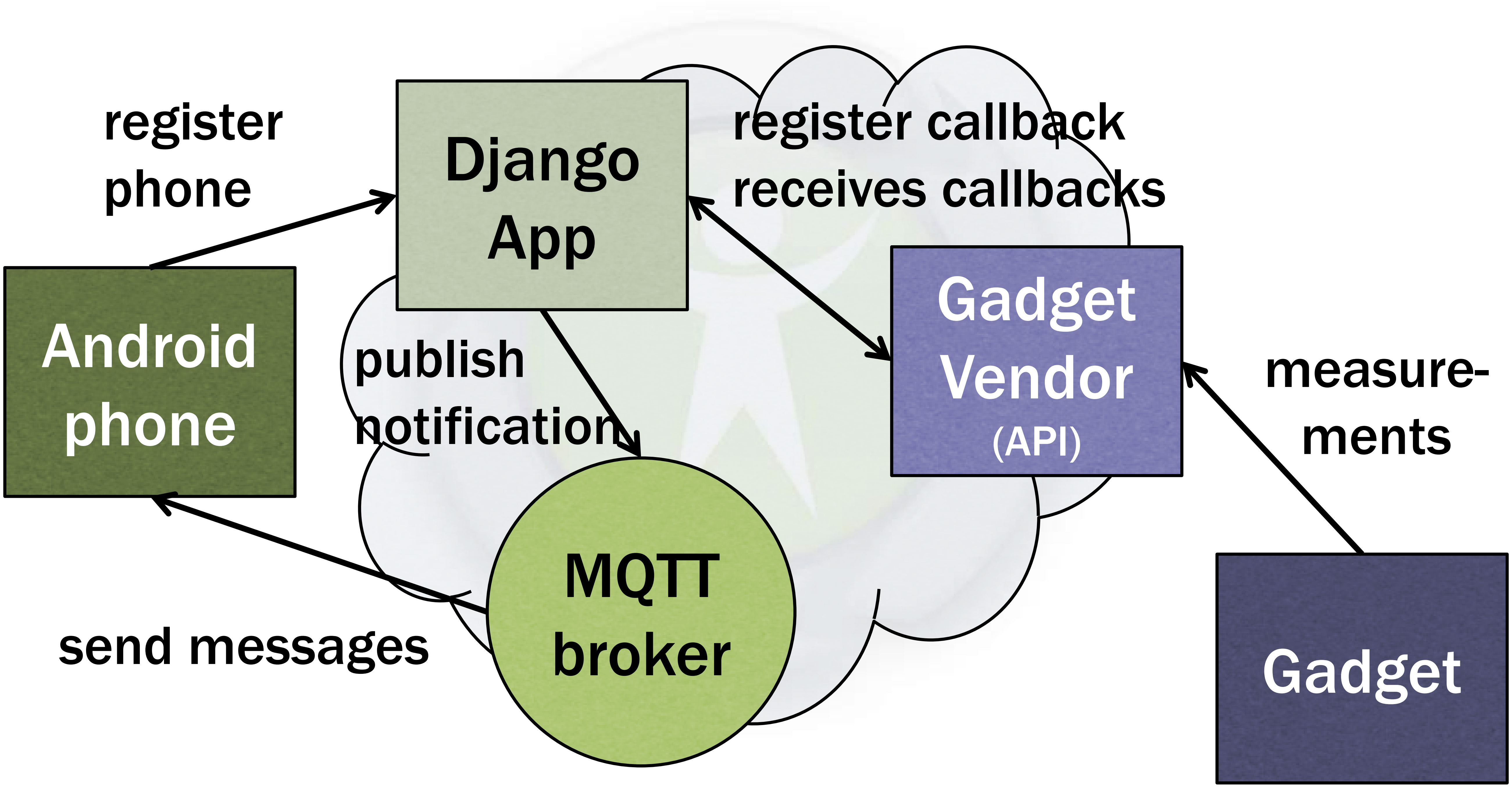
```python
import time
import os
import mosquitto

file_name = os.path.join('/', 'mnt', '1wire',
                         '10.67C6697351FF', 'temperature')

mqtt_client = mosquitto.Mosquitto('home-temperature')
mqtt_client.connect('test.mosquitto.org')

while 1:
    file_object = open(file_name, 'r')
    temperature = '%sC' % file_object.read()
    mqtt_client.publish('home/demo/temperature', temperature, 1)
    mqtt_client.loop()
    time.sleep(5)
    file_object.close()
```

Android Push Notifications

Getting data from Quantified Self gadgets to Android

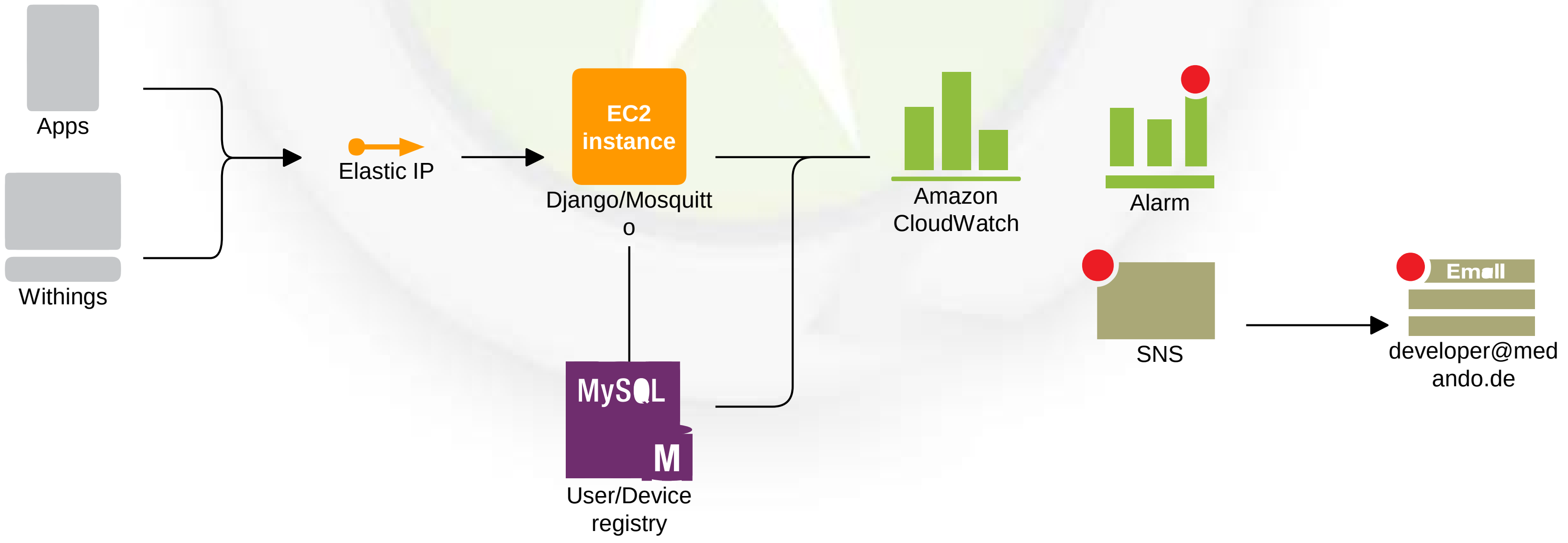
- The Gadget sends data to “somewhere” in the Cloud
  - Withings, Fitbit, and Nike provide APIs to access the data
  - Register for callbacks to get notifications
  - We use a Django app that registers as callback listener and send MQTT messages on updates
  - MQTT Java client on Android receives notifications

MQTT Push Notification Architecture



Implementation & Deployment

- Implementation includes OAuth stuff
- Most complex part was the Java code on Android (error handling etc.)
- Deployment on Amazon Web Services



Callback Implementation (Withings)

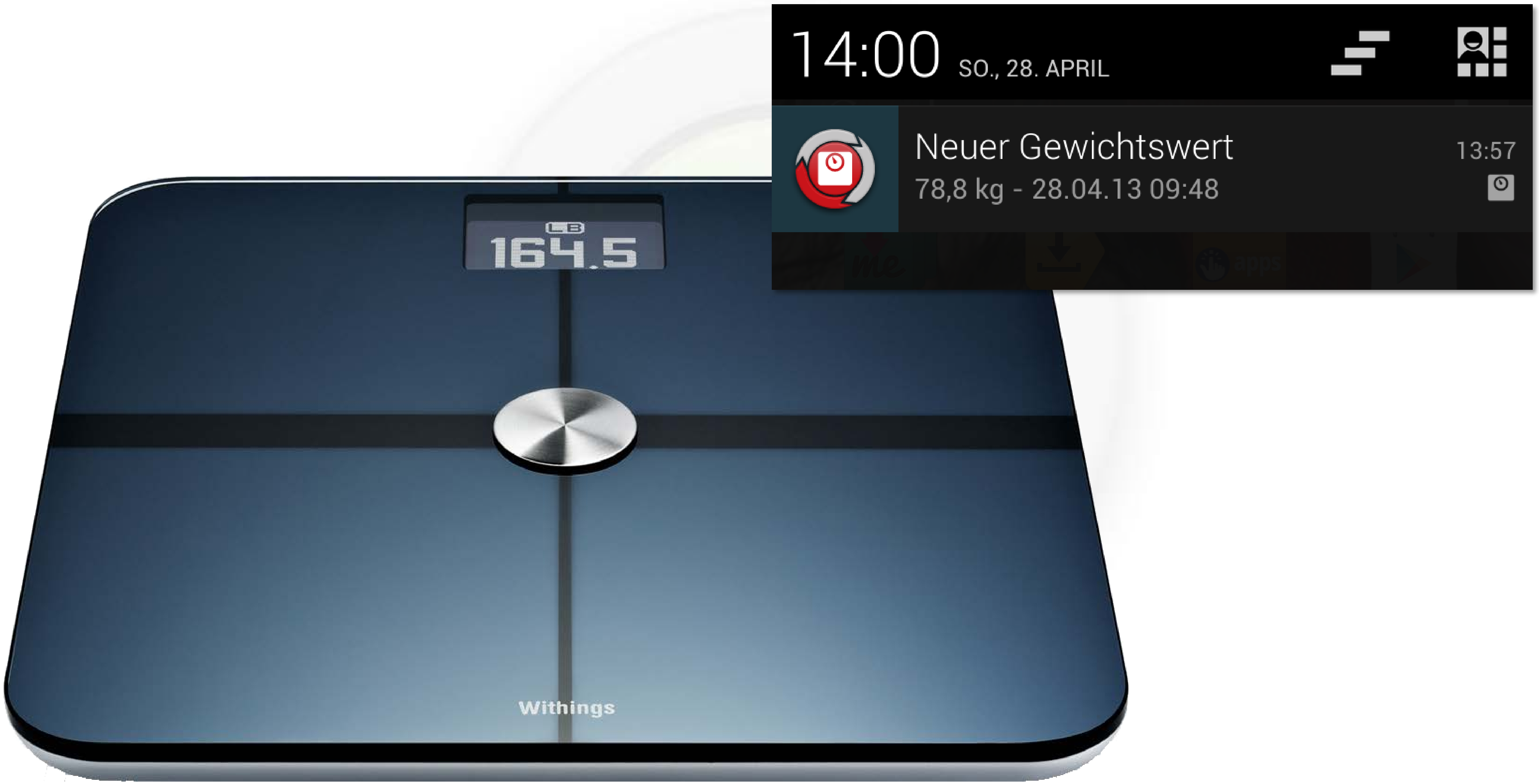
```
def callback(request):
    """ Callback function for Withings notifications. """

    . . . # request parameter handling

    devices = RegisteredWithingsUser.objects.filter(user_id=user_id)

    mqtt_client = MosquittoHandler(len(devices))

    for device in devices:
        device_id = device.device_id
        mqtt_topic = 'medando/weightcompanion/weights/%s/%s' %
                                                 (user_id, device_id)
        payload = simplejson.dumps({'startdate': startdate, 'enddate': enddate})
        mqtt_client.publish(mqtt_topic, payload, 2, True)

    mqtt_client.wait()
```

MQTT Messages

```
medando/weightcompanion/weights/1883073/34bae8cbe8dd92f3 0 {"startdate": "1371856646", "enddate": "1371856647"}
medando/weightcompanion/weights/1791607/898efc38ac5d4211 0 {"startdate": "1372742400", "enddate": "1372742401"}
medando/weightcompanion/weights/1527601/2ebcf034b8585668 0 {"startdate": "1368851117", "enddate": "1368851118"}
medando/weightcompanion/weights/16121/f2a8ca66fd067954 0 {"startdate": "1372750563", "enddate": "1372750564"}
medando/weightcompanion/weights/449599/4d701e076912648f 0 {"startdate": "1372751111", "enddate": "1372751112"}
medando/weightcompanion/weights/642578/b33356881163a389 0 {"startdate": "1370585275", "enddate": "1370585276"}
medando/weightcompanion/weights/2019258/33b1d416aeaec9ef 0 {"startdate": "1371377131", "enddate": "1371377132"}
medando/weightcompanion/weights/2019258/61bdf242b37d8a29 0 {"startdate": "1371377131", "enddate": "1371377132"}
```

```
medando/weightcompanion/weights/2019258/61bdf242b37d8a29 0
    {"startdate": "1371377131", "enddate": "1371377132"}
```

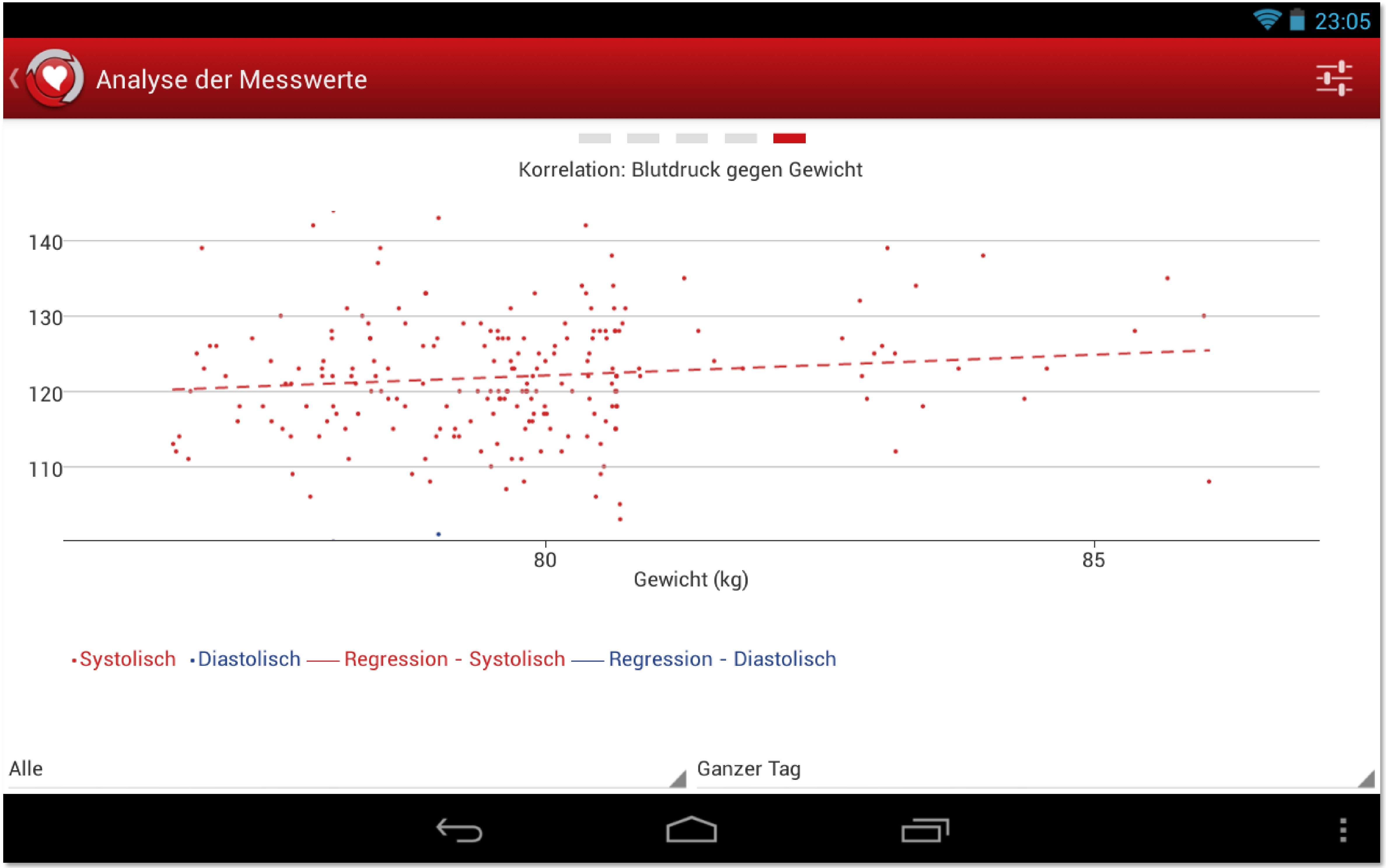
Notification on Android



14:00 SO., 28. APRIL

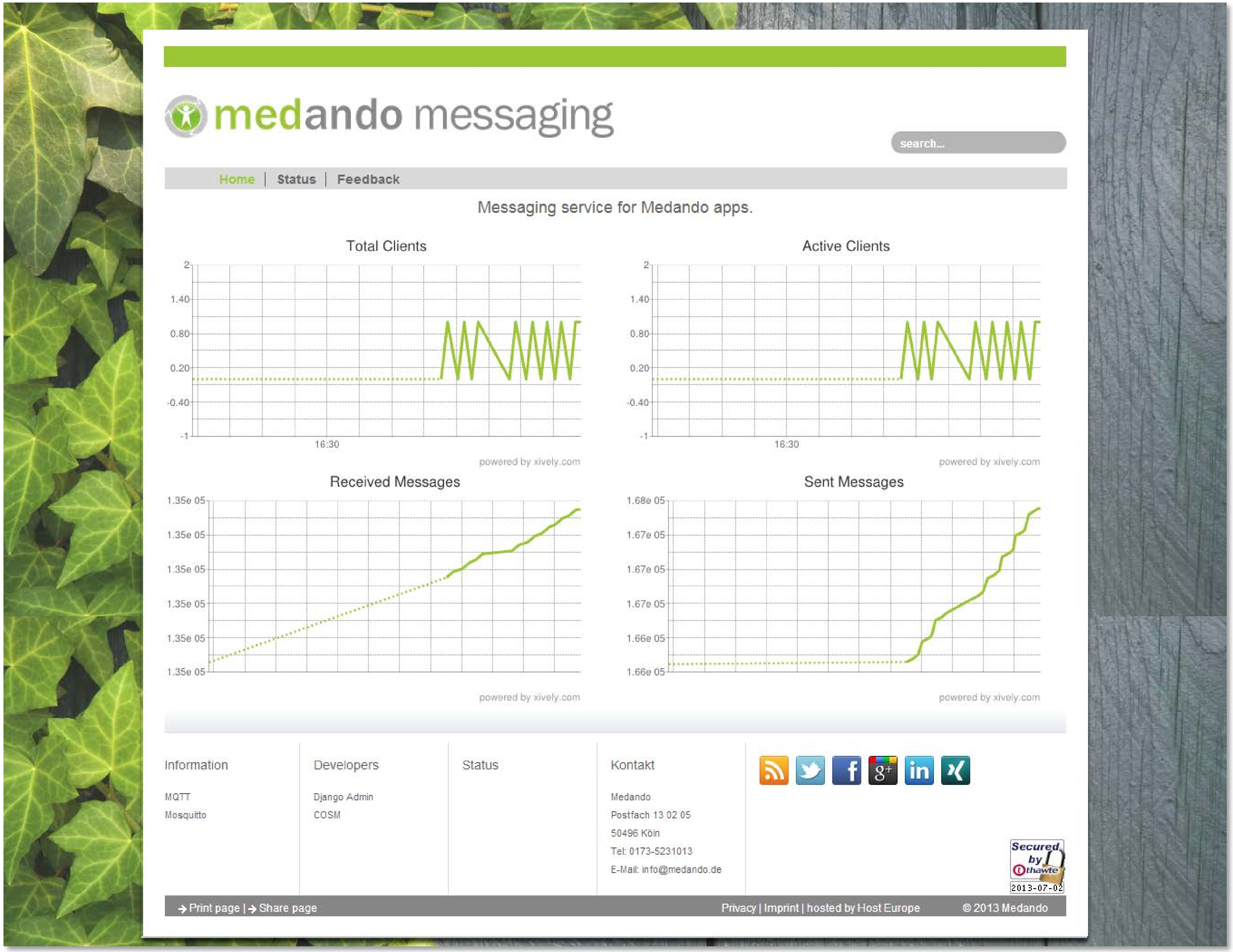
Neuer Gewichtswert 13:57
78,8 kg - 28.04.13 09:48

Blutdruck vs. Gewicht



23:05
Analyse der Messwerte
Korrelation: Blutdruck gegen Gewicht
140
130
120
110
80
85
Gewicht (kg)
•Systolisch •Diastolisch —— Regression - Systolisch —— Regression - Diastolisch
Alle
Ganzer Tag

Status Page



Messaging service for Medando apps.

Information

MQTT
Mosquitto

Developers

Django Admin
COSM

Status

Kontakt

Medando
Postfach 13 02 05
50496 Köln
Tel: 0173-5231013
E-Mail: info@medando.de

→ Print page | → Share page

Privacy | Imprint | hosted by Host Europe

© 2013 Medando

Conclusions

- There are other message broker
- There are other push notification services
- MQTT is very lightweight
- Mosquitto is easy to use from Python

Questions?

Andreas.Schreiber@medando.de

@MedandoEN | @onyame