So, you have a data source. And now?

A simple approach to deal with own (energy management) data

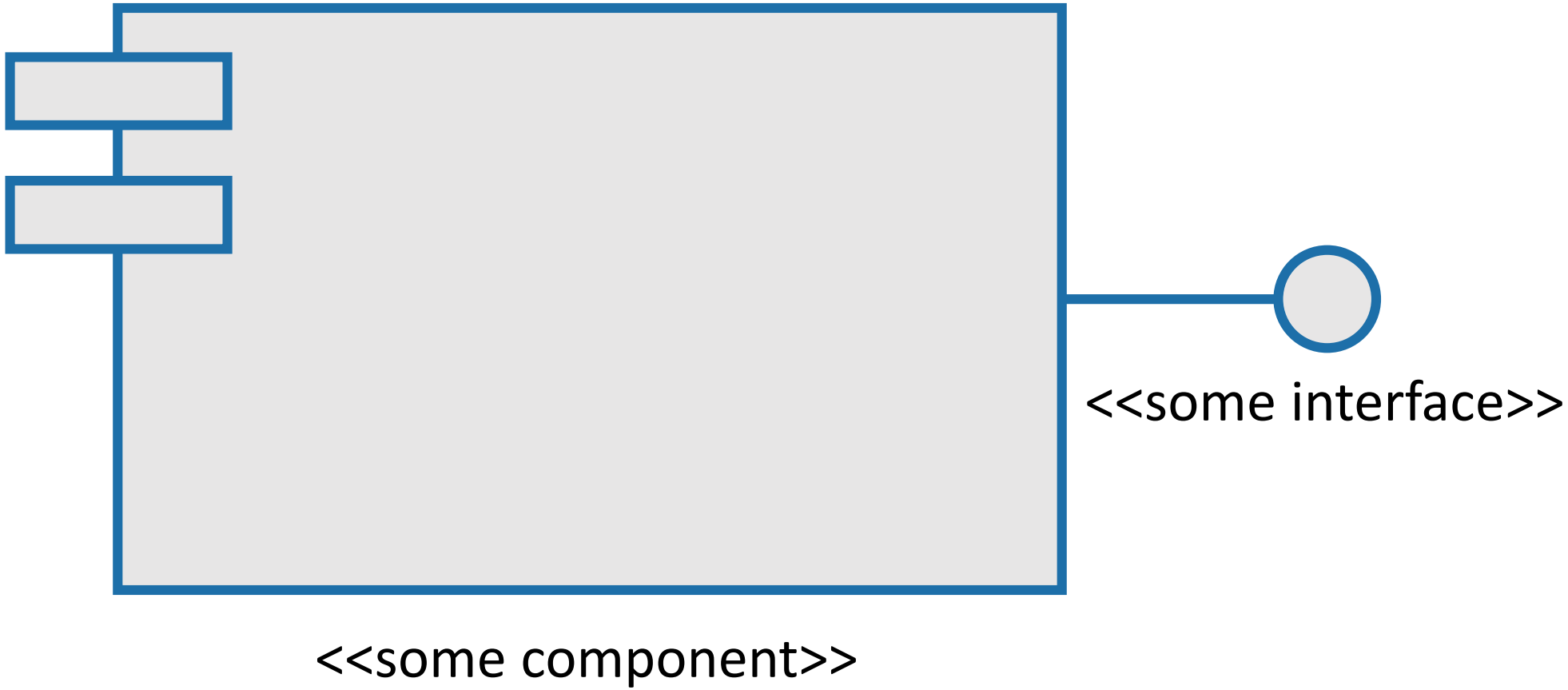
Markus Gebhard, Karlsruhe, 2016
http://github.com/gebhardm

grid:camp(<<2016.04>>)

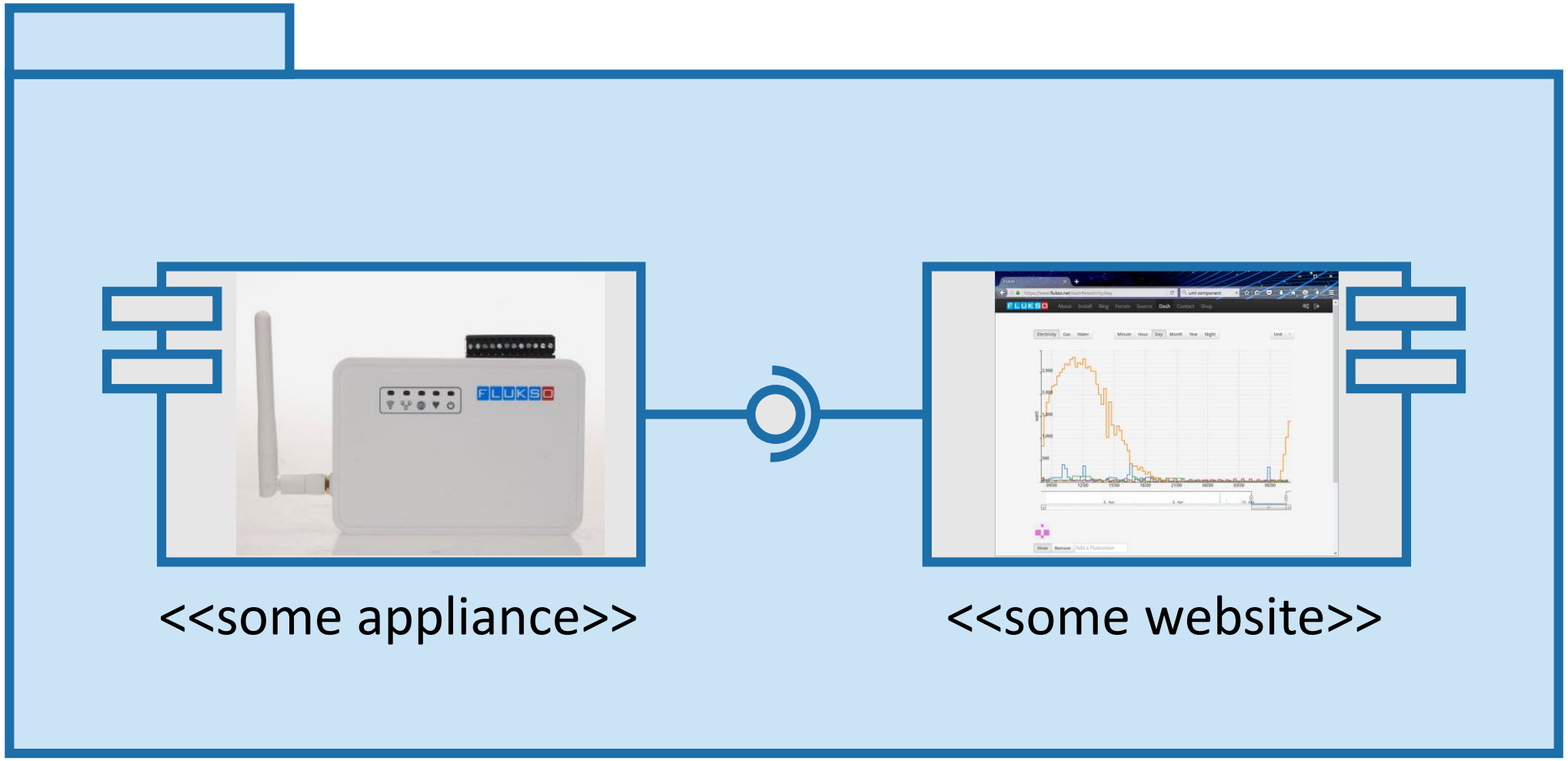
About

- Data Source
- Protocols
- Payloads
- Persistence
- Visualization

Data Source



Data Source



images (c) flukso.net

Protocols

- REST
  - REpresentational State Transfer
  - Architecture Style of the WWW
  - RESTful API – A system's interface conforming to the REST constraints*
  - Resource oriented
  - Based on HTTP methods - verbs GET, PUT, POST, DELETE
- Nota bene: STYLE, not protocol
  - A RESTful protocol may be OData (Open Data Protocol)

*c/s, stateless, cacheable, layered, uniform interface

Protocols

- MQTT
  - Message Queue Telemetry Transport
  - Open message protocol (mainly) for machine-to-machine communication
  - Standardized as protocol (v3.1.1) for the Internet of Things*
  - Broker based publish and subscribe
  - Minimal overhead

- Libraries available “down to” Arduino (works on less than 32kB memory)

*beside CoAP, UDP, XMPP, AMQP, LLAP, ...

Payloads

- From bit via single value to semantically rich document
- RAW – value\0value\0value\0
- “key value” – key=value&
- XML - <tag>value</tag>
- JSON – { key:content }

- How to distinguish commands from actual content?
  - REST: Address resource and use verb
  - MQTT: Utilize topic on messages exchanged

Persistence

- Use a little bit of Python or JavaScript
  - Connect to data source
  - Poll or subscribe to content
  - Write content into a file or database
- There may be more sophisticated storage formats... (e.g. TMPO)
- Runs on a minimal hardware (Raspberry Pi, NAS, even appliance itself)
- Retrieve data to visualize (even in Excel via csv)

```
// database access
var sqlite = require("sqlite3").verbose();
// prepare the database access
var db = new sqlite.Database("./flm.db");
// define the data persistence if it does not exist
createTabStr = "CREATE TABLE IF NOT EXISTS <<table>>" +
               " (sensor CHAR(32), timestamp CHAR(10)," +
               " value CHAR(5), unit CHAR(5));";
// and send the create command to the database
db.run(createTabStr, function (err) {
  if (err) {
    db.close();
    throw err;
  }
});
// use mqtt for client, socket.io for push,
var mqtt = require("mqtt");
var mqttclient = mqtt.connect({
    port :  << brokerport >> ,
    host :  << brokeraddress >>
  });

mqttclient.on("connect", function () {
  mqttclient.subscribe("/sensor/+/gauge");
  // handle mqtt messages
  mqttclient.on("message", function (topic, message) {
    var topicArray = topic.split("/");
    var payload = message.toString();
    var insertStr = "INSERT INTO <<table>>" +
      " (sensor, timestamp, value, unit)" +
      ' VALUES ("' + topicArray[2] + '",' +
      ' "' + payload[0] + '",' +
      ' "' + payload[1] + '",' +
      ' "' + payload[2] + '");';
    db.run(insertStr, function (err, res) {
      if (err) {
        db.close();
        throw err;
      }
    });
  });
});
```

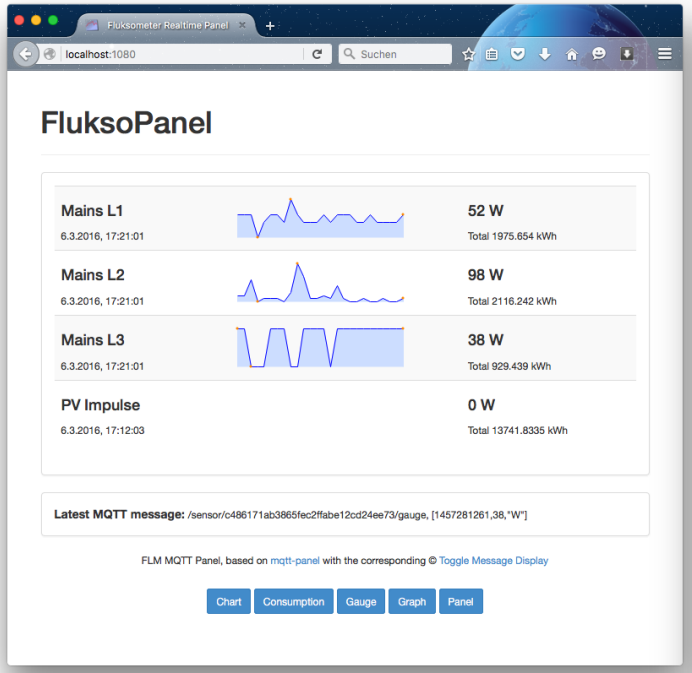
Visualization

- Again, use a little bit of Python or JavaScript
  - Connect to data source or persistence
  - Poll/subscribe to or query content
  - Pipe content into visualization application
- Use a little webserver for publication
  - And again, just minimal hardware required
    (if not highly sophisticated math shall be applied)

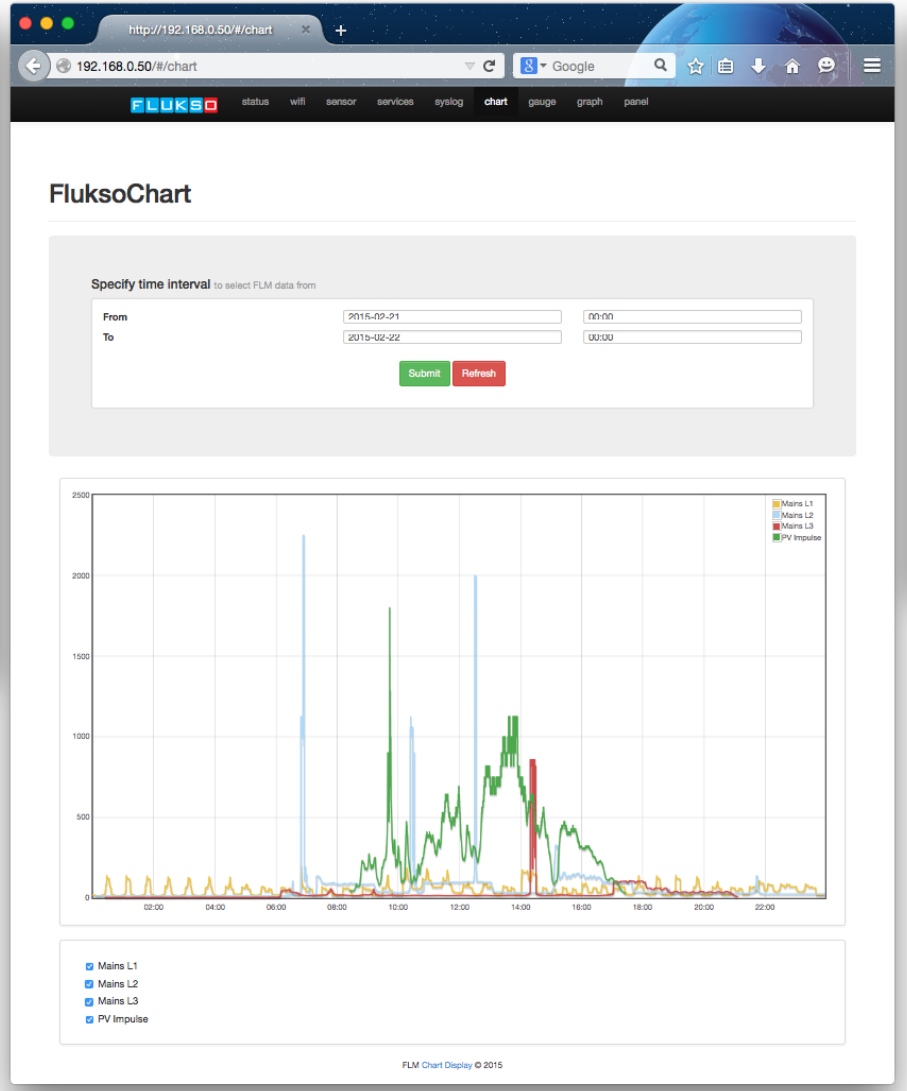
```
var socket = io.connect(location.host);

socket.on("connect", function() {
    socket.on("mqtt", function(msg) {
        // determine topic and payload
        var topic = msg.topic.split("/");
        var payload = msg.payload;
        switch (topic[1]) {
          case "device":
            handle_device(topic, payload);
            break;

          case "sensor":
            handle_sensor(topic, payload);
            // pass the message topic and content to the html part
            $("#message").html(msg.topic + ", " + msg.payload);
            break;

          default:
            break;
        }
    });
    // handle the device information
    function handle_device(topic, payload) {
        var deviceID = topic[2];
        if (topic[3] == "config") {
            var config = JSON.parse(payload);
            for (var obj in config) {
                var cfg = config[obj];
                if (cfg.enable == "1") {
                    var sensorId = cfg.id;
                    if (sensors[sensorId] == null) {
                        sensors[sensorId] = new Object();
                        sensors[sensorId].id = cfg.id;
                        sensors[sensorId].name = cfg.function;
                    } else sensors[sensorId].name = cfg.function;
                }
            }
        }
    }
    // handle the sensor information
    function handle_sensor(topic, payload) {
        var sensor = {};
        var msgType = topic[3];
        var sensorId = topic[2];
```

Beauty is in the eye of the beholder

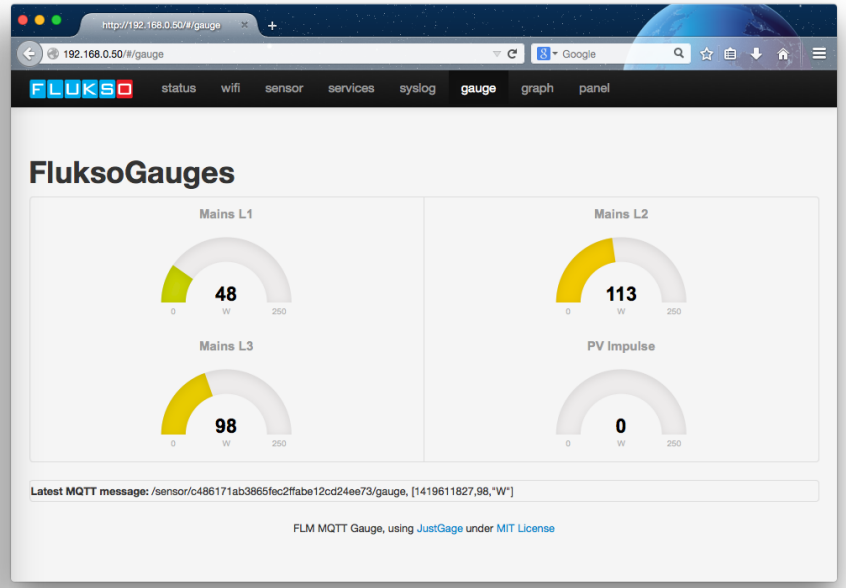
Visualization



using jQuery Sparkline

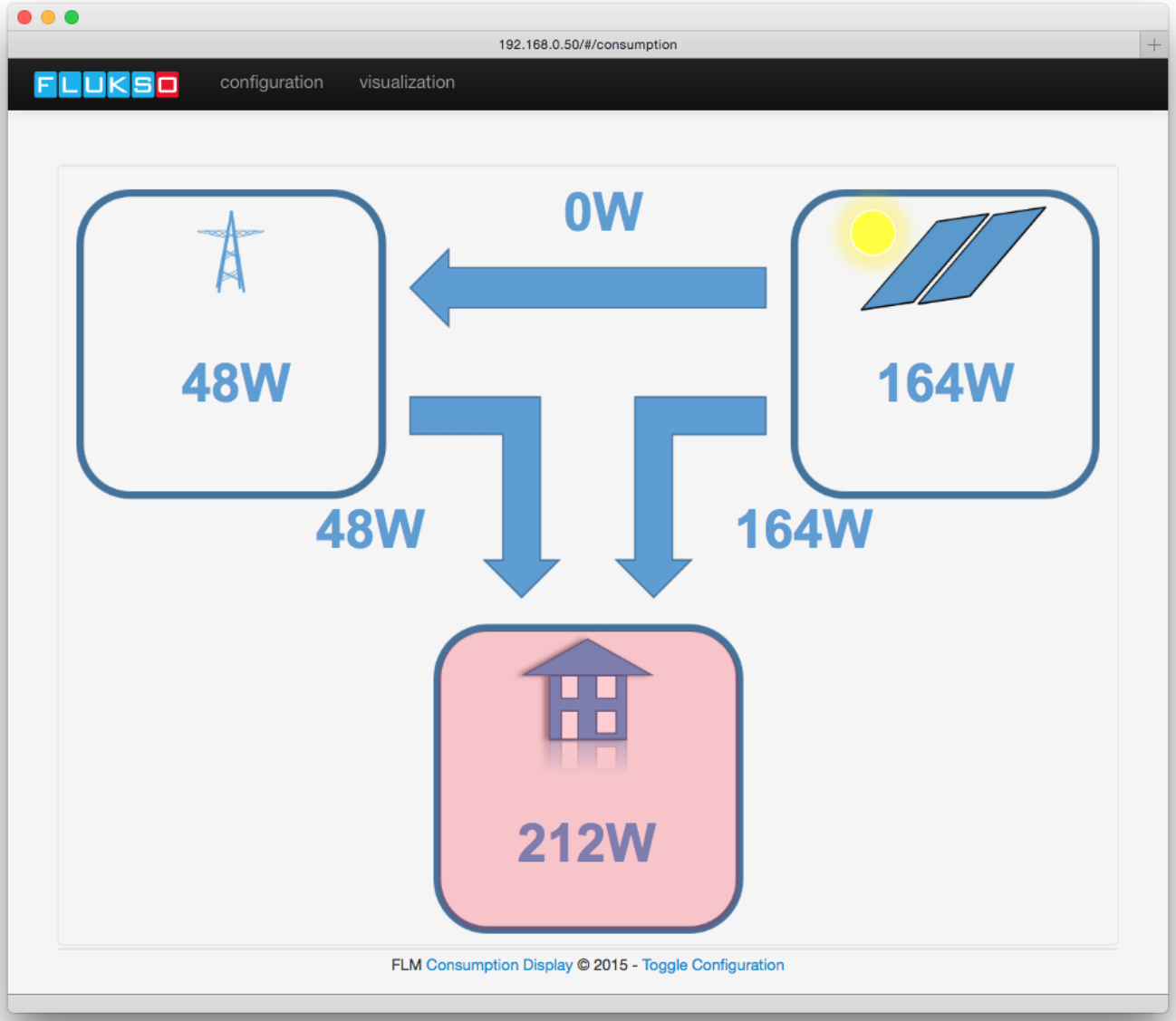


using Flot Charts



using JustGage

Visualization



Resources

- Github: http://github.com/gebhardm
  - flmdisplay – an easy to adapt MQTT visualization
  - flmlocal – a Fluksometer-onboard visualization
  - energyhacks – different projects

- Wordpress: https://energyhacks.wordpress.com/